Université de Montréal

Multivariate stochastic loss reserving with common shock approaches

par

Phuong Anh Vu

Département de mathématiques et de statistique
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiae doctor (Ph.D.) en Mathématiques
avec Spécialisation en Mathématiques appliquées

janvier 2019

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée

Multivariate stochastic loss reserving with common shock approaches

présentée par

Phuong Anh Vu

a été évaluée par un jury composé des personnes suivantes:

Maciej Augustyniak
(président-rapporteur)

Benjamin Avanzi
(directeur de recherche)

Bernard Wong
(co-directeur)

Greg Taylor
(co-directeur)

Manuel Morales
(membres du jury)

Étienne Marceau
(examinateur externe)

Jean-Philippe Boucher
(examinateur externe)

Manu Paranjape
(représentant do doyen de la FES)

Sommaire

Les provisions techniques (ou « réserves ») constituent habituellement l'un des plus importants passifs au bilan d'un assureur IARD (Incendie, Accidents et Risques Divers). Conséquemment, il est crucial pour les assureurs de les estimer avec précision. En outre, un assureur IARD opère généralement dans plusieurs lignes d'affaires dont les risques ne sont pas parfaitement dépendants. Il en résulte un « bénéfice de diversification » qu'il est primordial de considérer pour son effet sur les réserves et le capital. Il est donc essentiel de prendre en compte la dépendance entre lignes d'affaires dans l'estimation des réserves.

L'objectif de cette thèse est de développer de nouvelles approches d'évaluation des réserves pour des portefeuilles d'assurance avec lignes d'affaires dépendantes. Pour ce faire, nous explorons la technique de choc commun, une méthode populaire de modélisation de la dépendance qui offre plusieurs avantages, tels qu'une structure de dépendance explicite, une facilité d'interprétation et une construction parcimonieuse des matrices de corrélation. Afin de rendre les méthodes utiles à la pratique, nous incorporons au modèle des caractéristiques réalistes et souhaitables.

Motivés par la richesse de la famille de distributions de probabilité Tweedie, laquelle recoupe les distributions Poisson, amma et bien d'autres, nous introduisons un cadre commun de choc Tweedie avec dépendance entre lignes d'affaires. Les propriétés attrayantes de ce cadre sont étudiées, y compris la flexibilité de ses marges, ses moments ayant une forme analytique et sa capacité d'inclure des masses à 0.

Pour surmonter la complexité de la structure distributionnelle Tweedie, nous utilisons une approche bayésienne dans l'estimation des paramètres du modèle, que nous appliquons à un ensemble de données réelles. Nous formulons des remarques sur les caractéristiques pratiques de notre cadre.

Les données sur les provisions techniques pour sinistres sont asymétriques par nature.

C'est-à-dire, les montants de réclamations qu'on retrouve dans différentes cases d'un même triangle et entre différents triangles peuvent varier considérablement. Ceci s'explique car, habituellement, le nombre de sinistres est plus élevé dans les premières périodes de développement. Nous tenons compte explicitement de cette caractéristique dans nos modèles de chocs communs en y incluant un ajustement parcimonieux. Des illustrations utilisant des données théoriques (simulées) et réelles sont présentées.

Enfin, dans la dernière partie de cette thèse, nous élaborons un cadre dynamique avec des facteurs évolutifs tenant compte des tendances de développement des sinistres pouvant changer avec le temps. Une dépendance entre années civiles est introduite à l'aide de chocs communs. Nous formulons également une méthode d'estimation adaptée à la structure des données de provisionnement des sinistres, que nous illustrons à l'aide de données réelles.

Mots clés: réserves stochastiques, triangles de développement, choc commun, modélisation évolutive, réservation robotisée, distribution Tweedie, estimation bayésienne, données asymétriques, filtrage de Kalman, filtre particulaire.

Summary

Outstanding claims liability is usually one of the largest liabilities on the balance sheet of a general insurer. Therefore, it is critical for insurers to accurately estimate their outstanding claims. Furthermore, a general insurer typically operates in multiple business lines whose risks are not perfectly dependent. This results in "diversification benefits", the consideration of which is crucial due to their effects on the aggregate reserves and capital. It is then essential to consider the dependence across business lines in the estimation of outstanding claims.

The goal of this thesis is to develop new approaches to assess outstanding claims for portfolios of dependent lines. We explore the common shock technique for model developments, a very popular dependence modelling technique with distinctive strengths, such as explicit dependence structure, ease of interpretation, and parsimonious construction of correlation matrices. We also aim to enhance the practicality of our approaches by incorporating realistic and desirable model features.

Motivated by the richness of the Tweedie distribution family which covers Poisson distributions, gamma distributions and many more, we introduce a common shock Tweedie framework with dependence across business lines. Desirable properties of this framework are studied, including its marginal flexibility, tractable moments, and ability to handle masses at 0.

To overcome the complex distributional structure of the Tweedie framework, we formulate a Bayesian approach for model estimation and perform a real data illustration. Remarks on practical features of the framework are drawn.

Loss reserving data possesses an unbalanced nature, that is, claims from different positions within and between loss triangles can vary widely as more claims typically develop in early development periods. We account for this feature explicitly in common shock models

with a parsimonious common shock adjustment. Theoretical and real data illustrations are performed using the multivariate Tweedie framework.

Finally, in the last part of this thesis, we develop a dynamic framework with evolutionary factors to account for claims development patterns that change over time. Calendar year dependence is introduced using common shocks. We also formulate an estimation approach that is tailored to the structure of loss reserving data and perform a real data illustration.

Keywords: stochastic reserving, loss triangles, common shock, evolutionary modelling, robotic reserving, Tweedie family of distributions, Bayesian estimation, unbalanced data, Kalman filtering, particle filtering.

Table of contents

List of tables

List of figures

List of abbreviations

AB	Accident Benefits
APRA	Australian Prudential Regulation Authority
AR	Auto-regressive
ARMA	Autoregressive-moving-average
CI	Confidence interval
DB	Diversification benefits
DI	Disability Income
EDF	Exponential dispersion family
GLM	Generalised linear model
GPS	General prudential standards
IBNER	Incurred-but-not-enough-reported
IBNR	Incurred-but-not-reported
MCMC	Markov chain Monte Carlo
MR	Metropolis (acceptance) ratio
ODP	Over-dispersed Poisson
P&C	Property and casualty
QQ	Quantile-quantile
SD	Standard deviation
VaR	Value-at-risk

Acknowledgements

My most profound gratitude goes to my supervisors, A/Prof. Benjamin Avanzi, Prof. Greg Taylor and A/Prof. Bernard Wong. You have introduced me to this exciting field of research and have consistently given me inspiration and invaluable guidance in my quest for knowledge. Despite your extremely busy schedules, you have always found time to be there when I needed help. I have learned from you how to approach research with a critical mind as well as an adaptable and passionate attitude. Your guidance has also greatly helped with my personal development and my future career. Without your tremendous support and encouragement, this thesis would not have been possible.

I would also like to thank my research panel at UNSW, Dr. Jonathan Ziveyi, Prof. Michael Sherris, Dr. Katja Ignatieva and Dr. Jinxia Zhu for their time spent to review my progress and their support in my PhD journey. My gratitude also goes to my research committee at UdeM, A/Prof. Maciej Augustyniak, A/Prof. Manuel Morales and Prof. Louis Doray for their careful advice. When everything was still very new for me on my first days at UdeM, you helped make the transition much easier.

My special thanks go to Prof. Etienne Marceau and Prof. Jean-Philippe Boucher. I am very grateful for your time spent on your review and your constructive feedback on my thesis for my submissions at both universities.

I have been a part of a very interactive and supportive environment at the School of Risk and Actuarial Studies. My special thanks go to Prof. Hazel Bateman for her tremendous support and very useful research training that I received as a student in her research class. I am grateful to A/Prof. Jae-Kyung Woo for spending her time to review my paper and giving me constructive feedback in my PhD seminar. I also appreciate all the valuable insights and feedback from staff members of the school during my PhD journey. My thanks also go to administrative staff members for always being there during the countless number of times

that I needed help with various things no matter how big or small.

During my year in Montréal I met so many academic professors and fellow PhD candidates from UdeM, UQAM, Concordia University, McGill University and Laval University. I am grateful for all the laughs and good discussions that we shared on research and life. I also truly appreciate the help from staff members at UdeM which made my transition much easier. My thanks also go to my Vietnamese friends in Montréal and Québec. Our unexpected encounter on one of my very first days in Canada has turned into a friendship that I will always cherish.

I am very thankful to my fellow PhD students, Alan, Guillaume, Hayden, Nikolay and Vincent for making my PhD experience very enjoyable. My special thanks go to Alan for organising all the gathering events that helped bring us closer.

I would like to especially thank Xinda. You are my mentor who has never gotten tired of answering all my silly questions. From you, I have received invaluable advice not only in research but also in life. Thank you for being like a big brother that I never had. I would also like to thank my close friend An for the precious friendship that we have had for years which started when we were still in Vietnam and has continued in Australia until these days.

To my mother, my father and my little brother, an acknowledgement does not do you justice. I am forever indebted for your unconditional love and endless patience that have guided me all the way to the end of this journey. You have always put much faith in me and supported me so wholeheartedly in every step of my life. I am also indebted to my boyfriend Nathan. Thank you for being such a loving and kind man who has always provided a shoulder for me to lean on. You have consistently managed to channel your positivity onto me and help me through my worst days. I am so grateful to have you in my life.

Lastly, I would like to acknowledge the generous financial support provided by UNSW Graduate Research School, UNSW Business School and UdeM which have allowed me to better commit to my studies. This research is supported under the Australian Research Council's Linkage Projects funding scheme (project number LP130100723). I would also like to acknowledge my travel funds from a grant of the Natural Science and Engineering Research Council of Canada (Grant no. RGPIN-2015-04975). The views expressed herein are not necessarily those of the supporting organisations.

CHAPTER 1

Introduction

1.1 Loss reserving background

In general insurance (also known as non-life insurance, or property and casualty (P&C) insurance), there is typically a delay between the occurrence of an insured event, its reporting and the actual payments of claims. This delay can be driven by various reasons such as delays in reporting claims, administrative delays, investigations and legal proceedings. It can also be the nature of some policies that pay losses over a long period of time such as workers compensation insurance. A typical time line of a claim is given in Figure 1.1, replicating that in Taylor (2000). A claim occurs due to an accident within the insured period. The company, however, is not aware of this claim until it is reported. After the processing period, payments are made. At some point, the insurer considers the claim complete and closes the file. Further information may arrive and the claim is reopened, which is then followed by more payments until the file is completely closed.

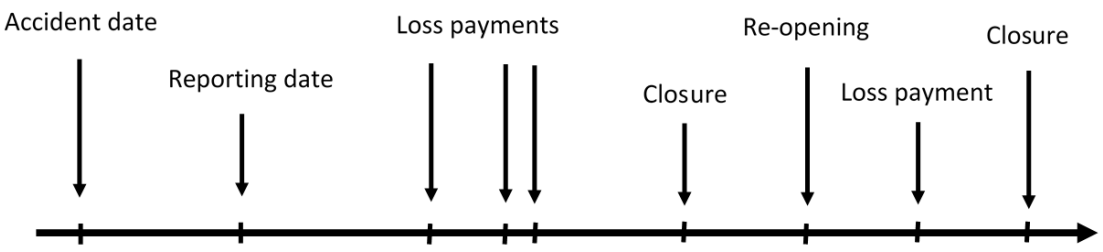

Figure 1.1: Time line of a claim

An insurer is obligated to pay outstanding claims in accordance with the terms of the policies for which they have already earned premiums. To sustain financial stability, it is then essential for them to set aside sufficient reserves for these claims. This is also often enforced by regulators to maintain the soundness of the financial system. Outstanding claims liability is typically one of the largest liabilities on the balance sheet of a general insurer, see for example, Alai and Wüthrich (2009); Heberle and Thomas (2016); Saluz and Gisler (2014). It can indeed be a number of times larger than the annual profit of the insurer (Taylor and Ashe, 1983). Thus it is essential to estimate outstanding claims accurately because any errors in the estimation of outstanding claims can have serious consequences on the emerging profit and solvency of the insurer.

Loss reserving methodologies have been available for many decades. Traditional methods include the chain ladder and the Bornhuetter-Ferguson techniques which apply a set of relatively simple algorithms to observed data to project outstanding claims. These traditional methods have gained their popularity due to their simplicity. However, outputs obtained from these algorithms are only single point (central) estimates of outstanding claims liabilities while the nature of outstanding claims is stochastic. To get a protection against downside movements, the insurer often holds a safety or risk margin related to the variability of outstanding claims liability. The uncertainty is often of more interest than the mean estimate itself in good risk management practice (Shi et al., 2012). Indeed, this is also a regulatory requirement in many countries (Gismondi et al., 2012). For example, the general prudential standards (GPS) 340 in Australia requires insurers to hold a risk margin for their outstanding claims liabilities. The risk margin is defined as the maximum of a half of the standard deviation, and the difference between $\text{VaR}_{75\%}$ and the central estimate of outstanding claims liability. Solvency II in Europe requires insurers to quantify their outstanding claims uncertainties using standard deviations of their total outstanding claims over a one-year-basis. For solvency purposes, the $\text{VaR}_{99.5\%}$ of the distribution of total outstanding claims is also an input in the calculation of risk-based capital in both of these frameworks.

Traditionally, actuaries add margins separately to the central estimate of outstanding claims obtained from deterministic methods if and when needed. These margins may be results of scenario and sensitivity testing and are not validated statistically. Over time, there has been a growing need to allow for uncertainty in a more consistent manner. This has motivated the development of various stochastic modelling techniques over the last three

decades (for excellent reviews of stochastic models, see Taylor, 2000; Wüthrich and Merz, 2008). Early models typically focus on the estimation of outstanding claims liability for a single business line or segment. Over time, more advanced models have been introduced. These include more realistic data features such as dependence across business segments, or changes in claims development patterns. These models can also include desirable features such as flexibility, ease of communication and application. These features will be discussed in detail in the following subsections.

1.1.1 Dependency across segments

An insurer typically operates in multiple business segments, the number of which can be up to 100 in some cases (Avanzi, Taylor and Wong, 2018). These segments can be business lines or subsets of these. Dependency across different business segments is an important characteristic of claims for a typical general insurer. As defined in Avanzi, Taylor and Wong (2016) using non-technical terms, it typically is the situation where the experience of one segment varies in sympathy with that of other segments. This experience can arise from many causes. At the very least, some segments share the same reporting procedure (Shi and Frees, 2011; De Jong, 2012) hence any changes in the operational system can affect these segments simultaneously. Similarly, legislative changes can also have impacts on some segments such as the same business line in different geographical locations. There can also be claim causing events such as hailstorms that give rise to claims in multiple business lines (for example, motor line and property line).

Since the 1980s, various univariate stochastic models have been developed in the literature (see Taylor, 2000; England and Verrall, 2002; Wüthrich and Merz, 2008). These models typically focus on assessing outstanding claims liability in a single business segment. The total liability on the portfolio level is then obtained by aggregating liabilities from individual segments. This is also referred to as the "silo" approach (Ajne, 1994; Shi and Frees, 2011).

With the existence of dependence across segments, the "silo" approach does not allow cross-borrowing of information across segments, which can result in sub-optimal central estimates of outstanding claims liabilities. The issue of "additivity" also arises for the "silo" approach. This refers to the problem where summary statistics on the aggregate level are not

statistically equivalent to the sums of the corresponding statistics on an individual level. This occurs to many measures of uncertainty, including quantiles and standard deviations, the two measures that are commonly used in reserving. These two measures are indeed only additive in cases of co-monotonicity, also known as perfectly positive dependence. While business segments have dependence to some extent, it is quite rare to observe cases of co-monotonicity (Kirschner et al., 2002). Due to the lack of a perfectly positive dependence structure across segments, the volatility of claims on the aggregate portfolio level is reduced compared to the aggregation of volatility on the individual segment level. This reduction often is known as a "diversification benefit" (Shi and Frees, 2011; De Jong, 2012; Côté et al., 2016;Avanzi, Taylor, Vu and Wong, 2016).

It is worth re-emphasising that the problem of "additivity" shall not be taken lightly because, as mentioned in the previous section, measures of uncertainty associated with the central estimate are often of great interest. At the very least, a simple addition of uncertainty measures will result in an over-estimation of risk margin and risk-based capital. Even if a certain degree of prudence is recommended, insurers should have as correct reserves as possible, not as large reserves as possible (Ajne, 1994). This is to ensure that capital is used parsimoniously while meeting solvency expectations (Avanzi, Taylor and Wong, 2016). Indeed, many insurance regulatory frameworks enable insurers to enjoy their diversification benefits in assessing the risk margins for their outstanding claims liabilities, as well as risk capitals for their consolidated operations (Avanzi, Taylor and Wong, 2016).

In the aggregation of outstanding claims liabilities, information regarding their dependencies can be added separately after outstanding claims estimates are obtained for individual segments using the "silo" approach. Alternatively, outstanding claims from individual segments and their dependence structure can be assessed simultaneously in a more consistent manner. This also allows a cross-borrowing of information across segments which can improve the accuracy of the overall estimation (Shi et al., 2012). Some multivariate approaches, such as common shock approaches, capture the dependence structure in a transparent and parsimonious manner. This enhances the ease of interpretation and communication of the models used.

1.1.2 Realistic and desirable model features

It is essential for insurers to estimate their outstanding claims liabilities accurately because of their significant impacts on emerging profits as well as the capital utilisation of insurers, as also discussed in Section 1.1. To accomplish this, it is then important to consider and incorporate data features, especially significant ones, in modelling. One of such features is the dependency across segments which is discussed in the previous section. It is also typical to observe claim activity (i.e. the development of claims over time, which can refer to the reporting, payments or settlements of claims) reaching a peak within a few years after the accident period, then declining as the time lag extends. We can call this the unbalanced nature of claims data. Any potential impacts of this particular behaviour of claim activity on any models should be examined. Insurers occasionally experience claims of 0's or negative values in their outstanding claims data due to, for example, salvage recoveries, or payment from third parties (De Alba, 2006; Kunkler, 2006). This particular feature can result in modelling difficulties as many distributions do not have support for non-positive values. Models construction should also be mindful of this feature. Another common feature of reserving data is changes in claims development pattern (i.e. the pattern that claims from the same accident period develop over time) across accident periods. This feature will be discussed in detail in the subsequent section.

At the same time, while it is important to consider realistic data features, it is also desirable to incorporate features that enhance the flexibility and practicality of models. It is often desirable for models to have flexible choices of marginals, also known as marginal flexibility, so that they can be applicable in more scenarios. Moments tractability may also be desirable in some cases as it allows the mean and variance of outstanding losses to be obtained in closed-form. This is particularly beneficial when their calculation is computationally expensive. It is also desirable for models to be parsimonious because loss reserving data is typically of small sample size. Other desirable features also include ease of interpretation of the dependence structure and disciplined construction of correlation matrices. As the demand for more frequent liability valuations has been increasing, it can also be desirable to have repetitive reserving jobs automated. This can be achieved using evolutionary reserving. More detail regarding this is given in the next section.

1.1.3 Changing claim activity and robotic reserving

Outstanding claims valuation is a predictive modelling activity whose purpose is to use historical data to forecast future outstanding claims. It is, however, not at all unusual to observe changes in claim activity. For example, insurers can improve the administering of claims over time to enhance efficiency, hence gradually shorten the administrative delays. A legislative change such as the recent reform for Auto Bodily Injury covers in New South Wales, Australia (State Insurance Regulatory Authority, 2018) results in faster claims resolution, hence reduces payment delays.

When changes occur, the projection of future claims is unfortunately no longer straightforward (Renshaw, 1989; Zehnwirth, 1994; Ghezzi, 2001; Taylor et al., 2003). Models with the assumption that claim activity is stable over time will experience failure. Actuaries have to make a lot of judgements to remove or reduce the effects of these changes if these models are used. These judgements can be time consuming to make and also difficult to justify (Sims, 2012). Actuaries may also want to revise the algebraic structure of the model which then results in a discontinuity in the sequence of estimates of outstanding claims liability (Taylor et al., 2003).

An elegant and plausible solution for these cases is to accommodate changes directly in the models by allowing parameters to evolve over time (De Jong and Zehnwirth, 1983; Zehnwirth, 1994; Gluck and Venter, 2009; Taylor et al., 2003). This is not the same as simply randomising parameters, but letting them evolve in a recursive manner. Filtering processes are usually used for the estimation of evolving parameters in these models. A filtering process is a real-time device that recursively updates parameters in the current period using estimates from the previous period without the need to redo all calculations or keep track of previous information (De Jong and Zehnwirth, 1983). This estimation gives more weight to more recent data, hence is more responsive to recent changes (Taylor, 2000; Alpuim and Ribeiro, 2003) and reduces reliance on arbitrary model judgements. A clear picture historical experience with any changes can then be obtained and the sequence of estimates derived from these models are smooth over time (Sims, 2012).

There has also been an increasing demand from insurers for more frequent liability valuations, such as on a quarterly or even monthly basis (Taylor and McGuire, 2008; Sims, 2014). Many insurers have large portfolios of many segments, making the valuation of

outstanding claims liabilities rather time consuming. Repetitive reserving jobs that do not require substantial actuarial judgements can then be automated using reserving robots. Evolutionary models and filtering processes can be used to construct these robots as they allow reserving estimates to be continually updated with new information.

1.2 Research motivation

The aim of our research is to develop reserving methodologies that incorporate realistic as well as desirable features. As mentioned in Section 1.1, it is important to accurately evaluate outstanding claims liabilities because of their significant impacts on the emerging profits as well as the capital of insurers. By allowing for realistic features in the models, the valuation can be improved. In addition, the practicality of models can also be enhanced by incorporating desirable features such as flexibility and tractability.

1.2.1 Multivariate reserving and common shock approaches

Insurers typically operate in multiple segments whose risks are not co-monotonic. This allows them to enjoy diversification benefits when they set their risk margins for outstanding claims liabilities, as well as risk-based capital (Section 1.1.1). The importance and various benefits of multivariate reserving have motivated the development of stochastic reserving models with dependence in the literature (see for example, Schmidt, 2006; Merz and Wüthrich, 2009a; De Jong, 2012; Zhang and Dukic, 2013; Merz et al., 2013; Shi, 2014). There are three main groups of existing parametric models: models using copulas, models using multivariate distributions with specific marginals, and models using common shock approaches.

Copulas are very popular dependence modelling tools not only in reserving but also in many other actuarial areas. Various applications of copulas in reserving can be found in Shi and Frees (2011); De Jong (2012); Zhang and Dukic (2013); Abdallah et al. (2015); Côté et al. (2016), just to name a few. The popularity of copulas comes from their modelling flexibility by allowing different types of marginals to be used with a wide range of dependence structures. Besides copulas, alternative multivariate distributional approaches have been considered for some specific choices of marginal distributions such as multivariate log-normal

models in Shi et al. (2012) and Merz et al. (2013), and the multivariate gamma model in Vu (2013). The last group of models utilises common shock approaches to capture dependence within or across segments. Some of these models use hierarchical Bayesian model structures which have randomised parameters to capture dependence across related variables. Examples include Abdallah et al. (2016); Wüthrich (2010); Salzmann and Wüthrich (2012); Shi et al. (2012). Other models are constructed using multivariate reduction techniques where the decomposition of claim observations contains a component that represents common effects. Examples of these models are Vu (2013); Avanzi, Taylor and Wong (2018).

In insurance, common shock approaches have been well known tools for dependence modelling (Lindskog and McNeil, 2003) and are also called random effect approaches in the literature. They are typically used to capture structural dependence, that is, structural co-movements that are due to known relationships which can be accounted for in a modelling framework (International Actuarial Association, 2004). As their names suggest, common shock approaches use common random factors to capture drivers of dependence across related variables. As a result, these drivers can be identified, as well as monitored if needed. The transparent dependence structures in common shock models can then be interpreted more easily. This is indeed one of the four desirable properties of multivariate distributions considered in Joe (1997, Chapter 4) which include:

- interpretability,
- closure under the taking of marginals, meaning that the multivariate marginals belong to the same family (this is important if, in modelling, we need to first choose appropriate univariate marginals, then bivariate and sequentially to higher-order marginals),
- flexible and wide range of dependence,
- density and cumulative distribution function in closed-form (if not, they are computationally feasible to work with).

Furthermore, the construction of correlation matrices can also be put at ease. Correlation matrices are tools extensively used by practitioners to specify dependence in the aggregation of outstanding claims liabilities or risk-based capital. Explicit dependence structures captured using common shock approaches allow correlation matrices to be specified in a more disciplined and parsimonious manner (Avanzi, Taylor and Wong, 2018).

1.2.2 Areas of development with common shock approaches

Common shock approaches provide various benefits for modelling dependency across segments. However, it is also very desirable for models to have marginal flexibility to enhance their applicability. This is a feature that has contributed to the popularity of copulas in various fields. The exponential dispersion family (EDF), and its Tweedie subclass in particular, receive our attention in this thesis. The Tweedie subclass is a very rich family of distributions which covers various symmetric and non-symmetric, light-tailed and heavy-tailed distributions (Jorgensen, 1997). Some notable members of the Tweedie family include Poisson distributions and Tweedie's compound Poisson distributions. The former are frequently used in loss reserving and well known in stochastic models that underlie the traditional chain ladder algorithm. The latter have probability mass at 0 hence are applicable in many data sets which contain 0's. They are also considered golden distributions in actuarial risk theory (Kaas et al., 2008).

The many benefits of common shock approaches and the rich Tweedie family of distributions motivate the development of a common shock Tweedie framework for reserving. In this particular construction of the dependence structure, moments, including the mean and variance can be obtained in closed-form. This is a desirable property when the valuation of such quantities are computationally expensive. This framework as well as its various theoretical benefits is the focus of Chapter 3[1].

The Tweedie family of distributions, however, has quite a complex density. This issue further escalates in a multivariate framework. To overcome this issue, we formulate a Bayesian approach for model estimation in Chapter 4[1]. An illustration using real data from Schedule P in the United States (available in Zhang and Dukic, 2013) is provided, and remarks on applications of the framework are also drawn.

In most (if not all) reserving data sets, there is a significant variation in claim activity for different lengths of delay. In particular, it often reaches a peak in some early years, then dies out as the delay increases. Furthermore, claim activities across segments are not identical. Some segments such as Auto Property Damage are called short-tailed with typically shorter delays in claim activity. Other segments, for example Auto Bodily Injury

[1] An abbreviated version of results in Chapters 3 and 4 has been published in *Avanzi, B., Taylor, G., Vu, P.A., Wong, B., 2016. Stochastic loss reserving with dependence: A flexible multivariate Tweedie approach. Insurance: Mathematics and Economics 71, 63–78.*

covers, are long-tailed with longer delays. In Chapter 5[2], we analyse this feature and its impact on common shock models in detail. We then account for this feature explicitly with a parsimonious solution used in the development of a modified common shock Tweedie framework for unbalanced data. An illustration is performed using real data from a Canadian insurance company (available in Côté et al., 2016).

In addition to the many data features described above, insurers also typically observe changes in claim activity over time, as also mentioned in Chapter 1.1.3. This very common but also very important feature has motivated the development of evolutionary models which capture these changes naturally through evolving parameters. Evolutionary models also have many other benefits as also mentioned in Chapter 1.1.3. The first appearance of evolutionary models in the loss reserving literature dates back to the early 1980s with the work of De Jong and Zehnwirth (1983). This model, as well as the majority of existing evolutionary models, including Verrall (1989, 1994); Ntzoufras and Dellaportas (2002); Atherino et al. (2010); De Jong (2006) are based on the assumption of Gaussian distributed claims (usually on the log-scale). This specific assumption for claims distribution allows Kalman filtering to be used, an optimal closed-form filtering algorithm for these cases. All of these models focus on a single segment of business. Shi et al. (2012) briefly touched on evolutionary modelling by allowing only calendar factor to evolve in a multivariate log-normal model for multiple business segments. This factor is not updated sequentially but in a traditional hierarchical Bayesian structure.

As also mentioned previously, the EDF has been a very popular family of distributions used in outstanding claims modelling. The applications of this family are usually performed in the framework of generalised linear models (GLMs). Indeed, applications of GLMs in loss reserving have appeared since the early 1990s and have gained great popularity ever since (Taylor and Sullivan, 2016). Their popularity, in both theory and practice, comes from their ability to allow the exploration and estimation of multiple trends within the data without many subjective judgements. Traditional GLMs in reserving are static with deterministic models which assume stable claims experience over time. A natural step toward evolutionary reserving is to let these parameters evolve. This is what we call evolutionary GLM approaches. Recognising the popularity of GLMs and the EDF in outstanding claims modelling, Taylor and McGuire (2009) developed a univariate evolutionary GLM framework for a single segment

[2]An abbreviated version of results in Chapter 5 has been submitted and is under review in *Avanzi, B., Taylor, G., Vu, P.A., Wong, B., 2018. On unbalanced data and common shock models in stochastic loss reserving.*

and use an adaptive filter for random factors estimation. This model relaxes the Gaussian assumption for claims in the models mentioned in the previous paragraph. The adaptive filter used is tractable in special cases of the EDF with conjugate prior distributions. Sims (2011) then considered a particle filter, a simulation-based filter, for the same framework.

In Chapter 6, we will further explore the evolutionary reserving area. We aim to incorporate dependence across segments using common shocks while utilising the rich GLM structure to develop a multivariate evolutionary GLM framework. Our focus is also placed on the formulation of a particle filtering approach that provides real-time updates of random factors in this evolutionary framework. This filtering approach is specifically tailored to the structure of loss reserving data. An illustration is performed using real data from a Canadian insurance company (available in Côté et al., 2016). This framework offers many benefits from using common shocks for dependence modelling, rich GLM structure for marginal modelling, as well as various benefits of an evolutionary approach. It is also worth noting that it enables the development of reserving robots which automate repetitive reserving jobs. This is particularly useful in various practical cases where outstanding claims valuation is required on a regular basis for large portfolios of many segments.

1.3 Thesis outline

The rest of this thesis is organised as follows. Chapter 2 provides a review of the existing literature in loss reserving and other relevant areas. This is followed by an introduction of a multivariate Tweedie framework in Chapter 3. An analysis of the theoretical properties of this framework is also provided in this chapter. The estimation and applications of this framework on simulated and real data sets are given in Chapter 4. In Chapter 5 we consider the unbalanced nature of loss reserving data, and account for this feature explicitly in common shock models in a parsimonious manner. Chapter 6 introduces a multivariate evolutionary GLM framework to account for claims development patterns that change over time, and formulates an estimation approach that is tailored to the structure of loss reserving data. Finally, Chapter 7 concludes with a summary of contributions and areas for future development.

CHAPTER 2

Literature review

In this chapter, we review the existing literature in loss reserving and other relevant areas. The outstanding claims data representation, which is in the famous format of loss triangles, and notations used are introduced in Section 2.1. Traditional reserving methods are reviewed in Section 2.2. These are deterministic techniques that produce a single mean (central) estimate of outstanding claims liability. While deterministic methods are simple and computationally inexpensive, they do not provide indicators of uncertainty associated with the single mean estimate of outstanding claims. Stochastic models aim to overcome this limitation and allow insurers to better monitor their solvency and fulfil regulatory requirements. These models are reviewed in Section 2.3 with a focus on an important and also very popular type of models that use the EDF and its Tweedie sub-family. Section 2.4 focuses on multivariate loss reserving models for multiple business segments. These models aim to consider the dependence structure across segments in the valuation, hence allow for diversification benefits in the aggregation of outstanding claims liabilities. Common shock approaches are one of the dependence modelling tools used in these models. They have many interesting and desirable properties, including explicit dependence structure, ease of interpretation, parsimonious and disciplined construction of correlation matrices. Common shock models are also reviewed in this section. General insurers typically observe changes in their claims development patterns over time. This feature can be captured naturally using evolutionary models with evolving parameters. These models are reviewed in Section 2.5. Section 2.6 summarises the literature review and identifies areas for development.

2.1 Data and notations

In aggregate loss reserving, outstanding claims are recorded in a triangular format, which is called a loss triangle. The general format of a loss triangle is given in Figure 2.1. The index i, $i \in \{1, ..., I\}$, on the vertical axis of the triangle represents accident period i of the claims, i.e. the period when insured events occur. The index j, $j \in \{1, ..., J\}$ (with $I = J$) on the horizontal axis represents development period j, i.e. the period when claims are developed, where the development can refer to the reporting, payments, or settlements of claims depending on the information that the loss triangle represents (see also below). A company typically has multiple business lines/segments, each with a corresponding loss triangle. A loss triangle can also be generalised to a loss trapezium with $I \neq J$. For the sake of notation simplification, we use the loss triangle format for model developments in this thesis. However, all techniques applied to loss triangles can also be applied to loss trapeziums.

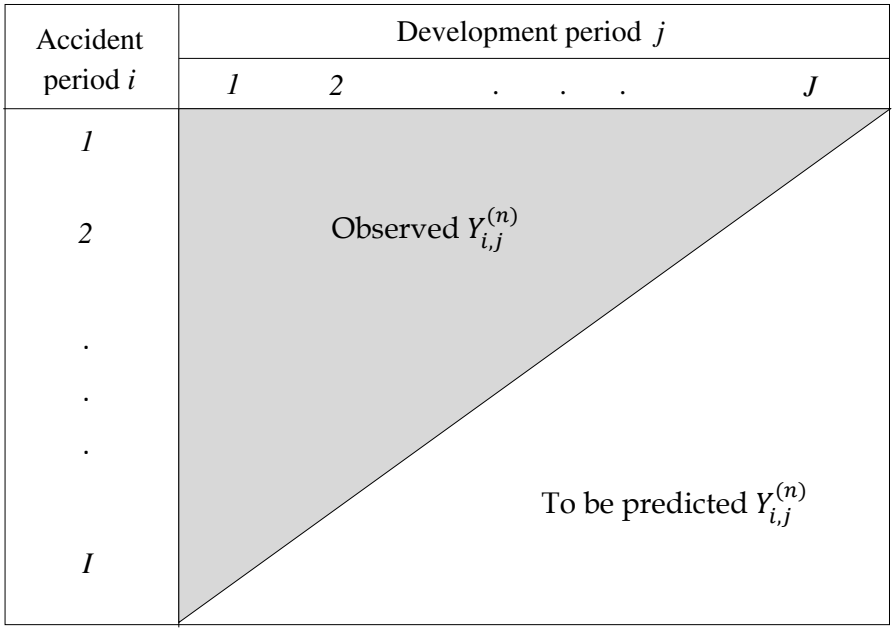

Figure 2.1: Loss triangle representation of data

Incremental claims, denoted by Y, are the total amount (or number) of newly developed claims. Notation $Y_{i,j}^{(n)}$, $n \in \{1, ..., N\}$ represents the total incremental claims that correspond to accidents in accident period i, developed in development period j and in business segment n. These claims are hence made in calendar period $t = i + j - 1$, $t \in \{1, ..., I + J - 1\}$. Correspondingly, all claims that are on the same diagonal of a loss triangle are made within the same calendar period. The shaded triangle of observed claims in Figure 2.1 is referred

to as the upper triangle and the non-shaded triangle of claims to be predicted in the same figure is referred to as the lower triangle.

Loss triangles can be used to record different types of information. Recall the time line of a claim in Chapter 1. When loss triangles record claims between their accident dates and reporting dates, we have incurred-but-not-reported (IBNR) triangles. Triangles can also record incurred-but-not-enough-reported (IBNER) claims which are reported but not settled. Claims between their accident/reporting dates and payment dates can also be recorded using paid loss triangles.

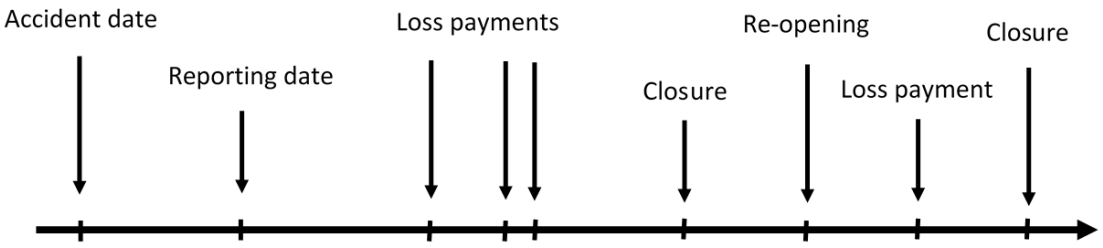

Figure 1.1: Time line of a claim

In the latest calendar period $t = I$, all incremental claims in the upper triangles are observed. The set of these observed claims is denoted by $\boldsymbol{Y}^U$,

$$\boldsymbol{Y}^U = \left\{ Y_{i,j}^{(n)};\, i \in \{1, ..., I\},\, j \in \{1, ..., I - i + 1\},\, n \in \{1, ..., N\} \right\}. \tag{2.1}$$

Incremental claims in the lower triangles are outstanding claims to be predicted at the latest calendar period I, denoted by $\boldsymbol{Y}^L$,

$$\boldsymbol{Y}^L = \left\{ Y_{i,j}^{(n)};\, i \in \{1, ..., I\},\, j \in \{I - i + 2, ..., J\},\, n \in \{1, ..., N\} \right\}. \tag{2.2}$$

A loss triangle can also record cumulative claims instead of incremental claims. Cumulative claims $X_{i,j}^{(n)}$ are defined as the total claims for accident period i aggregated to development period j in segment n. That is,

$$X_{i,j}^{(n)} = \sum_{m=1}^{j} Y_{i,m}^{(n)}. \tag{2.3}$$

Notations $\boldsymbol{X}^U$ and $\boldsymbol{X}^L$ then denote the observed cumulative claims in upper triangles and

outstanding cumulative claims to be predicted in lower triangle, respectively.

The primary goal of a loss reserving model is to use historical claim information in upper triangle(s) $\boldsymbol{Y}^U$ (or $\boldsymbol{X}^U$) to predict outstanding claims in lower triangle(s) $\boldsymbol{Y}^L$ (or $\boldsymbol{X}^L$).

2.2 Deterministic reserving techniques

Traditionally forecasts of outstanding claims are obtained using deterministic loss reserving models. These are conventional methods that provide simplicity to outstanding claims modelling and have also been used as the bases for the development of many stochastic models. In these classical models, deterministic algorithms are applied to obtain a single central estimate of the outstanding claims liability. They do not consider the stochastic nature of the liability. For an excellent review of deterministic loss reserving techniques, see Taylor (2000). In this thesis, we review three popular deterministic methods: the chain ladder method in Section 2.2.1, the Bornhuetter-Ferguson method in Section 2.2.2 and the Berquist-Sherman method in Section 2.2.3. Note that this section focuses on reserving techniques for one single business segment (i.e. one loss triangle), hence the superscript (n) is dropped from the associated notations.

2.2.1 Chain ladder algorithm

The chain ladder algorithm is the most popular deterministic technique used to predict outstanding claims (Wüthrich and Merz, 2008, Chapter 2). It has been used for many years as a self-explaining algorithm which was not derived from a stochastic model (Mack, 1993).

The chain ladder algorithm is based on the modelling of development factors (also known as age-to-age factors) of cumulative claims. Development factors, denoted by d_j, $j \in \{1, ..., J-1\}$, are defined such that

$$d_j = \frac{\sum_{i=1}^{I-j} X_{i,j+1}}{\sum_{i=1}^{I-j} X_{i,j}}. \tag{2.4}$$

The outstanding cumulative claims $X_{i,j}$ with $i + j > I + 1$ can then be calculated by

$$X_{i,j} = X_{i,I-i+1} d_{I-i+1} ... d_{j-1}, \tag{2.5}$$

and the ultimate claims $X_{i,J}$, defined as the total claims from accident period i to be developed by the end of the last development period J, by

$$X_{i,J} = X_{i,I-i+1} \prod_{j=I-i+1}^{J-1} d_j, \qquad 2 \leq i \leq I. \tag{2.6}$$

Total outstanding claims for accident period i is then given by

$$X_{i,I-i+1} \left(\prod_{j=I-i+1}^{J-1} d_j - 1 \right). \tag{2.7}$$

The popularity of the chain ladder algorithm comes from its simplicity and it is known as a heuristic algorithm that is distribution-free, meaning that it works with almost no assumptions (Taylor, 2011; Miranda et al., 2012). This algorithm, however, comes with a number of drawbacks. As mentioned in Mack (1993), variations in immature accident periods may result in misleading outstanding claims estimates. The algorithm is also based on the assumption that claims development patterns are similar across accident periods.

Despite its limitations, the chain ladder algorithm remains a very popular deterministic algorithm. Many stochastic models have been studied to provide theoretical justifications to this algorithm, the first and very popular of which include the Poisson model in Hachemeister and Stanard (1975) and the distribution free model in Mack (1993). Other studies of stochastic models underlying the chain ladder also include Renshaw and Verrall (1998); Verrall (2000); Hess and Schmidt (2002); Taylor (2011).

2.2.2 Bornhuetter-Ferguson algorithm

The Bornhuetter-Ferguson algorithm was developed by Bornhuetter and Ferguson (1972). This algorithm is also one of the most commonly used techniques in practice for highly leveraged lines (Alai et al., 2009). While the chain ladder algorithm uses development factors to forecast ultimate claims, the Bornhuetter-Ferguson algorithm uses the expected ultimate claims provided by experts to forecast the total outstanding claims (Schmidt and

Zocher, 2008).

In the Bornhuetter-Ferguson algorithm, the cumulative outstanding claims $X_{i,j+m}$ with $1 \leq j \leq J-1$, $1 \leq m \leq J-j$ are calculated by

$$X_{i,j+m} = X_{i,j} + (\tilde{d}_{j+m} - \tilde{d}_j)\alpha_i, \tag{2.8}$$

where $\tilde{d}_j$, $j \in \{1, ..., J\}$, $\tilde{d}_J = 1$ is the cumulative proportion of the expected ultimate claims α_i that is developed up to development period j. Factors $\tilde{d}_j$ can be estimated using a number of techniques including the chain ladder algorithm (Wüthrich and Merz, 2008, Chapter 2). This then results in the total outstanding claims for accident period i

$$(1 - \tilde{d}_{I-i+1})\alpha_i. \tag{2.9}$$

While the chain ladder estimates are completely driven by data, the Bornhuetter-Ferguson estimates incorporate both observations and expert knowledge. It is considered more robust than the chain ladder method, especially against instability in the proportions of ultimate claims paid in early development periods (Alai et al., 2009). This is particularly the case when only a small proportion of losses are developed in early years, which is a common observation in long tailed classes of business.

2.2.3 Berquist-Sherman technique

The impacts of structural changes in claim activities on the valuation of outstanding claims liabilities have been recognised since the 1970s. These changes invalidate the assumption of consistent claims development patterns across accident periods in popular methods such as the chain ladder algorithm. Berquist and Sherman (1977) developed a procedure to address these issues in the estimation of reserves, often known as the Berquist-Sherman technique. A comprehensive explanation of this technique can be found in Friedland (2010).

In the Berquist-Sherman technique, the first step is to gather data and search for problematic areas. Two possible treatments can then be performed on these areas: data selection and rearrangement, and data adjustment.

The data selection and rearrangement treatment aims to obtain data that is relatively unaffected by a given problem. This can be done using substitute types or forms of data. For example, an insurer can use quarterly accident data to substitute yearly accident data when the growth rate of earned exposures changes rapidly and causes distortions in development factors. Alternatively, relatively unaffected data can be obtained by subdividing the loss experience into more homogeneous groups of exposures and/or types of claims. This is particularly useful for cases where major changes have occurred in the composition of business.

If the data selection and arrangement approach is not successful, data adjustment can be used. Two types of adjustments can be considered. The first adjustment applies to the triangle of reported claims where the trend in claims severity is adjusted based on judgement. This is then used together with claims count to adjust the ultimate claims. The second adjustment applies to the triangle of paid claims. It involves analysing changes in the claims development pattern and using this pattern to adjust the paid claims triangle. These adjustments aim to achieve some level of consistency in the data. Standard development methods can then be applied to adjusted triangles project future claims.

The Berquist-Sherman technique is known as the first classical approach that allows for updates in claim activities over time. Because many adjustments and assumptions are involved in this approach, it should be done with an appropriate degree of caution (Ghezzi, 2001).

2.3 Univariate stochastic reserving models

This section reviews stochastic reserving models for a single business segment. These are models which have stochastic assumptions for claims in loss triangles. The motivation for stochastic loss reserving methods is explained in Section 2.3.1. An important class of stochastic models is GLMs which is built on the EDF, and in many cases, its Tweedie sub-family. Some of the theory of the EDF and its Tweedie sub-family is provided in Section 2.3.2. The GLM reserving framework is reviewed in Section 2.3.3. Models using the Tweedie family are reviewed in Section 2.3.4. This section focuses on reserving techniques for one single business segment, hence the superscript (n) is dropped from the associated notations.

2.3.1 Motivation for stochastic reserving

The popularity of deterministic loss reserving techniques comes from their simplicity. However, it is also essential for an insurer to assess the uncertainty associated with the single point estimate of their outstanding claims liability.

The need for stochastic loss reserving methods has been recognised since the very early 1980s, see for example, Taylor and Ashe (1983) and De Jong and Zehnwirth (1983). Because the outstanding claims liability is one of the largest liabilities on the balance sheet of an insurer (Alai and Wüthrich, 2009; Heberle and Thomas, 2016; Saluz and Gisler, 2014; Abdallah et al., 2015), a failure to consider its stochastic nature can lead to serious consequences on profits as well as insolvency issues (Taylor and Ashe, 1983). Quantifying the variability of total outstanding claims is also a compulsory regulatory requirement in many countries. For example, the GPS 340 of the Australian Prudential Regulation Authority (APRA) in Australia requires insurers to hold risk margins for their outstanding claims besides the central estimates. A risk margin is defined as the maximum of a half of the standard deviation, and the difference between $\text{VaR}_{75\%}$ and the central estimate of outstanding claims, i.e.

$$\text{Risk margin}[Y] = \max\left\{\text{VaR}_{75\%}[Y] - E[Y];\, \frac{1}{2}SD[Y]\right\}, \tag{2.10}$$

where Y is a random variable which represents the total outstanding claims in this equation. Solvency II in Europe also requires insurers to obtain the standard deviations of their projected total outstanding claims over a one-year horizon. These requirements aim to enhance the ability of insurers to meet their liabilities.

When deterministic approaches are used, risk margins can be added separately to the central estimates when needed. The development of stochastic modelling techniques has increased over the last three decades to fulfil the growing need to allow for uncertainty in a more consistent manner. Excellent reviews of various stochastic models can be found in Taylor (2000) and Wüthrich and Merz (2008). These models take into account the stochastic behaviour of outstanding claims, allowing both the mean estimate and the prediction uncertainty of a total outstanding claims liability to be obtained. Insurers not only can fulfil regulatory requirements, but also develop a good risk management practice through having a complete picture of the volatility of their outstanding claims liability (Shi et al., 2012).

There are two main strands of research in stochastic loss reserving: non-parametric modelling approaches and parametric modelling approaches (Shi et al., 2012; Abdallah et al., 2015). Non-parametric modelling approaches use distribution-free set-ups to estimate outstanding claims and associated prediction uncertainty. Parametric modelling approaches, on the other hand, use distributional assumptions. As a result, different quantities of interest can be inferred from the predictive distribution of outstanding claims, for example, the mean, the variance, and various quantiles. In this thesis, we focus on parametric modelling approaches to utilise this benefit. However, it is also worth noting that parametric models can be subject to over-fitting, and one needs to be aware of parameter uncertainty when working with these models.

2.3.2 Theory of exponential dispersion family and Tweedie family

GLMs are a rich class of models populated by McCullagh and Nelder (1989). It can be considered as a generalisation of traditional linear models, with various applications in many areas of insurance (De Jong et al., 2008; Frees et al., 2014, 2016). One of these areas is in loss reserving (De Jong et al., 2008; Taylor and McGuire, 2016). A typical GLM framework has three components: a systematic component, a random component and the link between these components. The systematic component is a linear predictor $\boldsymbol{A\gamma}$. The stochastic component then specifies the dispersion or variance around the mean of the distribution. Finally, the link component is a function that relates the linear predictor in the systematic component with the mean of the distribution.

GLMs are mainly built on distributional assumptions of the EDF, a very rich family of distributions. We provide a review of the theory of the EDF in Section 2.3.2.1. The Tweedie family, a major and particularly attractive subclass of the EDF is further reviewed in Section 2.3.2.2.

2.3.2.1 Exponential dispersion family

This section follows the comprehensive review of the EDF in Jorgensen (1997). As mentioned in Jorgensen (1997, Chapter 3), there are two representations of an EDF density: the additive form and the reproductive form. In this section, we provide a summary of these two forms.

1. **Definition**

Assume that we have a variable $\tilde{Y}$ that has an additive exponential dispersion distribution $\tilde{Y} \sim \text{EDF}^*(\theta, \vartheta)$. The parameter ϑ is called the index parameter. The parameter θ is called the canonical parameter which belongs to the canonical domain

$$\{\theta \in \mathbb{R} : \kappa(\theta) < \infty\}, \tag{2.11}$$

where $\kappa(.)$ is the unit cumulant function of the distribution defined such that

$$\kappa(\theta) = \int e^{\theta\tilde{y}} dF_{\tilde{y}}(\tilde{y}). \tag{2.12}$$

The corresponding variable $Y = \tilde{Y}/\vartheta \sim \text{EDF}(\mu, \phi)$ is called a reproductive exponential dispersion variable. The parameter μ is called the location parameter, or the mean parameter with

$$\mu = \kappa'(\theta), \tag{2.13}$$

where $\kappa'(.)$ is the first derivative of the unit cumulant function $\kappa(.)$. The parameter ϕ is the dispersion parameter and is related to the index parameter by

$$\phi = \frac{1}{\vartheta}. \tag{2.14}$$

Overall we have a transformation that provides a duality between the two forms

$$\tilde{Y} \sim \text{EDF}^*(\theta, \vartheta) \Leftrightarrow Y \sim \text{EDF}(\mu, \phi). \tag{2.15}$$

2. **Densities**

The density of a variable in the additive form $\tilde{Y} \sim \text{EDF}^*(\theta, \vartheta)$, if defined, is

$$f_{\tilde{Y}}(\tilde{Y}; \theta, \vartheta) = v^*(\tilde{Y}; \vartheta) \exp\{\theta\tilde{Y} - \vartheta\kappa(\theta)\}, \tag{2.16}$$

where $v^*(\tilde{Y}; \vartheta)$ is the density of the measure that defines the distribution.

Similarly we have the density for the corresponding variable in the reproductive form

$Y \sim \text{EDF}^*(\mu, \phi)$ as

$$f_Y(y; \theta, \vartheta) = v(y; \vartheta) \exp\{\vartheta(\theta y - \kappa(\theta))\}, \tag{2.17}$$

where $v(\tilde{Y}; \vartheta)$ is the density of the corresponding measure that defines the distribution.

3. **Cumulant generating function, mean and variance**

The cumulant generating function of a variable in the additive form $\tilde{Y} \sim \text{EDF}^*(\theta, \vartheta)$ is

$$K^*_{\tilde{Y}}(l; \theta, \vartheta) = \vartheta\{\kappa(\theta + l) - \kappa(\theta)\}. \tag{2.18}$$

The mean and variance of $\tilde{Y}$ are given by

$$E[\tilde{Y}] = \vartheta\kappa'(\theta), \tag{2.19}$$

$$Var[\tilde{Y}] = \vartheta V(\kappa'(\theta)), \tag{2.20}$$

where $V(.)$ is the unit variance function defined by

$$V(\mu) = \kappa''\left((\kappa')^{-1}(\mu)\right) = \kappa''(\theta), \tag{2.21}$$

with $\kappa''(.)$ being the second derivative and $(\kappa')^{-1}(.)$ being the inverse of the first derivative of the unit cumulant function $\kappa(.)$.

The cumulant generating function of the corresponding variable in the reproductive form $Y = \tilde{Y}/\vartheta \sim \text{EDF}(\mu, \phi)$ is

$$K_Y(l; \theta, \vartheta) = \vartheta\{\kappa(\theta + l/\vartheta) - \kappa(\theta)\}. \tag{2.22}$$

The expressions of the mean and variance of Y are somewhat more straightforward with

$$E[Y] = \mu, \tag{2.23}$$

$$Var[Y] = \phi V(\mu). \tag{2.24}$$

4. **Convolution formula**

The additive representation of the EDF has a convenient convolution formula. Consider independent random variables $\tilde{Y}_1, ..., \tilde{Y}_n$ where

$$\tilde{Y}_m \sim \text{EDF}^*(\theta, \vartheta_m), \tag{2.25}$$

then

$$\tilde{Y}_1 + ... + \tilde{Y}_n \sim \text{EDF}^*(\theta, \vartheta_1 + ... + \vartheta_n). \tag{2.26}$$

Using the duality transformation, the corresponding convolution formula can be obtained for the reproductive representation. Consider independent random variables $Y_1, ..., Y_n$ where

$$Y_m \sim \text{EDF}\left(\mu, \frac{\phi}{w_m}\right), \tag{2.27}$$

then

$$\frac{1}{w_1 + ... + w_n} \sum_{m=1}^{n} w_m.Y_m \sim \text{EDF}\left(\mu, \frac{\phi}{w_1 + ... + w_n}\right). \tag{2.28}$$

2.3.2.2 Tweedie family

The Tweedie family of distributions is a major subclass of the EDF. It consists of various symmetric and non-symmetric, light-tailed and heavy-tailed distributions (Alai et al., 2016; Jorgensen, 1997). This family is distinctively defined with a special relationship between the univariate variance function and the mean function

$$V(\mu) = \mu^p, \qquad p \in (-\infty, 0] \cap [1, \infty), \tag{2.29}$$

where p is the power parameter (Jorgensen, 1997, Chapter 4). The value of the power parameter p identifies the corresponding distribution of the Tweedie family. For example, $p = 0$ corresponds to normal distributions, $p = 1$ corresponds to Poisson distributions, $p = 2$ corresponds to gamma distributions, $1 < p < 2$ corresponds to compound Poisson-gamma distributions (i.e. a Poisson sum of gamma random variables), and $p = 3$ corresponds to inverse Gaussian distributions. These members of the Tweedie family are very commonly used distributions of the EDF. A full list of distributions with their corresponding p parameters can be found in Jorgensen (1997, Chapter 4).

As a subclass of the EDF, the Tweedie family also has two representations and general

properties of the EDF listed in Section 2.3.2.1. We denote by $\text{Tweedie}_p^*(\theta, \vartheta)$ the additive representation and $\text{Tweedie}_p(\mu, \phi)$ the reproductive representation. The canonical parameter θ of the additive Tweedie form belongs to the domain

$$\begin{cases} \mathbb{R}, & p = 0, 1, \\ [0, \infty), & p < 0, \\ (-\infty, 0), & 1 < p \leq 2, \\ (-\infty, 0], & 2 < p < \infty. \end{cases} \tag{2.30}$$

The unit cumulant function of the Tweedie family, as expressed in Jorgensen (1997), is

$$\kappa(\theta) = \begin{cases} \exp(\theta), & p = 1, \\ -\log(-\theta), & p = 2, \\ \dfrac{1}{2-p}\left[\theta(1-p)\right]^{\frac{p-2}{p-1}}, & p \notin (0, 1] \cup [2]. \end{cases} \tag{2.31}$$

The relationship between the additive form and the reproductive form of the Tweedie family, as also provided in Jorgensen (1997, Chapter 4), is

$$\text{Tweedie}_p^*(\theta, \vartheta) = \text{Tweedie}_p\left(\vartheta\kappa'(\theta), \vartheta^{1-p}\right) = \text{Tweedie}_p\left(\mu, \phi\right). \tag{2.32}$$

This relationship can be used to convert one Tweedie representation to another and specify relationships between parameters of these two forms. This duality is driven by the duality transformation of the EDF, as well as the closure under the scale transformation property of the Tweedie family. This property is shown in Jorgensen (1997, Chapter 4) as

$$\xi.\text{Tweedie}_p(\mu, \phi) = \text{Tweedie}_p(\xi\mu, \xi^{2-p}\phi), \tag{2.33}$$

and is unique to the Tweedie sub-class but not the entire EDF.

2.3.3 GLM framework

The first applications of GLMs in loss reserving can be found in Renshaw (1994); Renshaw and Verrall (1998). Since then, GLMs have gained their popularity in the literature as well as in practice. As explained in Taylor and Sullivan (2016), GLMs allow the exploration

and estimation of multiple trends within the data without many subjective judgements. Consequently, cell-specific effects can be estimated for all cells in loss triangles. Missing values can also be accommodated in a robust manner. Overall, great model generality and flexibility can be achieved with the use of GLMs. Many comprehensive reviews of GLMs in loss reserving can be found in Taylor (2000); England and Verrall (2006); Wüthrich and Merz (2008); Taylor and McGuire (2016).

There are two main types of GLM frameworks for loss reserving, known as recursive models and non-recursive models (Taylor, 2011; Taylor and McGuire, 2016). These two types of models are studied in great detail in Taylor (2011). The results show that they are very different, especially in terms of stochastic independence. Similar results specific to over-dispersed Poisson (ODP) models within the GLM framework are also shown in Mack and Venter (2000).

Recursive models, also known as EDF Mack models, are parametric versions of the Mack's stochastic chain ladder model in Mack (1993). In these models, claims are assumed to be independent across accident periods, and defined recursively within a single accident period as

$$Y_{i,j+1}|X_{i,j} \sim \text{EDF}(\theta_{i,j}, \phi_{i,j}). \tag{2.34}$$

We also assume

$$E[X_{i,j+1}|X_{i,j}] = d_j X_{i,j}, \tag{2.35}$$

which is built on the chain ladder algorithm in Section 2.2.1. Taylor (2011) showed that estimates from the chain ladder algorithm in Section 2.2.1 are indeed the maximum likelihood and minimum variance estimates in a wide range of EDF Mack models.

The second type of GLMs are non-recursive models, also called EDF cross-classified models. These are the focus of our literature review and developments in later chapters. These models typically assume that incremental claims $Y_{i,j}$ are independent and

$$Y_{i,j} \sim \text{EDF}(\theta_{i,j}, \phi_{i,j}). \tag{2.36}$$

Subsequently they have densities

$$f_{Y_{i,j}}(y_{i,j}; \theta_{i,j}, \phi_{i,j}) = v\left(y_{i,j}, \phi_{i,j}\right) \exp\left\{\frac{y_{i,j}\theta_{i,j} - \kappa(\theta_{i,j})}{\phi_{i,j}}\right\}, \tag{2.37}$$

which is the reproductive representation of the EDF in Equation (2.17) with the index parameter ϑ replaced by the dispersion parameter ϕ.

A common mean structure used in cross-classified models is

$$E[Y_{i,j}] = \mu_{i,j} = \kappa'(\theta_{i,j}) = \alpha_i \beta_j. \tag{2.38}$$

This is an example of a multiplicative mean structure which is a product of accident period effect α_i and development period effect β_j. A log-link function can be used to have a linear predictor on the log scale. For example, we have for Equation (2.38),

$$\log(\mu_{i,j}) = a_i + b_j, \tag{2.39}$$

where $a_i = \log(\alpha_i)$ and $b_j = \log(\beta_j)$. This specific mean structure is also called a chain-ladder structure in the literature (England and Verrall, 2002) as it has one parameter for each row and one parameter for each column. In this mean structure, the patterns of claim activities are specified by the development factors β_j (or b_j). As these factors are not row-specific, claim patterns are assumed to be similar across accident periods.

An alternative mean structure to the above is

$$E[Y_{i,j}] = \eta_{t=i+j-1} \beta_j, \tag{2.40}$$

where η_t is the effect of calendar period t. This is known as the separation method and was developed in Taylor (1977).

Another type of mean structure that has been used in loss reserving is the gamma curve, also known as the Hoerl curve. A detailed review of the Hoerl curve can be found in Zehnwirth (1989). It is based on the observation that claim activity typically reaches a peak in early development periods then dies out monotonically and eventually exponentially as the delay j increases. This shape is similar to the density curve of a gamma distribution (Zehnwirth, 1989; England and Verrall, 2002). The Hoerl curve mean structure on the log-link can be presented as

$$\log(\mu_{i,j}) = a_i + r_i \log(j) + s_i j, \tag{2.41}$$

where a_i, r_i and s_i are parameters for accident period i. The development curve is captured by $r_i \log(j) + s_i j$. Assuming similar claims development patterns across all accident periods,

the above mean structure is simplified to

$$\log(\mu_{i,j}) = a_i + r\log(j) + sj. \tag{2.42}$$

Instead of having one parameter for each development period, the Hoerl curve only utilises two parameters r and s to generate the claims development pattern. In this curve development period j is treated as a continuous covariate. This improves model parsimony as less parameters are used. It also smooths out fluctuations in observed data, hence makes the models more robust (Zehnwirth, 1989). In addition, the Hoerl curve enables extrapolation beyond the range of observed development period (England and Verrall, 2002). This is typically useful for long-tailed or new business segments with insufficient data in late development lags. Various applications of the Hoerl curve in loss reserving models can be found in, for example, De Jong and Zehnwirth (1983), Wright (1990), England and Verrall (2001), Taylor and McGuire (2009) and Sims (2011).

So far we have only mentioned the log-link function used for various types of mean structures of GLMs. This shall not be a constraint in model calibration and different link functions can be chosen subject to specific data features and other factors of consideration such as tractability and goodness-of-fit (Taylor and Sullivan, 2016).

Utilising the theory of the EDF in Section 2.3.2.1, we also have the variance of $Y_{i,j}$ in Equation (2.36) specified as

$$Var[Y_{i,j}] = \phi_{i,j}\kappa''(\theta_{i,j}) = \phi_{i,j}V(\mu_{i,j}). \tag{2.43}$$

2.3.4 Models using the Tweedie family

The Tweedie family and its various specific members have been used in many loss reserving models. These include ODP distributions, normal distributions (which are often used for log-transformed data), Tweedie's compound Poisson distributions and gamma distributions. These models are reviewed in this section.

2.3.4.1 Over-dispersed Poisson models

ODP models have been very popular stochastic models in the loss reserving literature. It is well-known that the maximum likelihood estimates of outstanding claims from these models recover the estimates of the chain ladder algorithm, see for example, Hachemeister and Stanard (1975); Mack (1991); Renshaw and Verrall (1998); Mack and Venter (2000); England and Verrall (2002); Schmidt (2002); Taylor (2009), and many more.

The first Poisson model was introduced in Hachemeister and Stanard (1975), full details of which can be found in for example, Wüthrich and Merz (2008, Chapter 2). Renshaw and Verrall (1998) then provided an extension of this Poisson model to account for over-dispersion using the GLM framework. ODP models relax the mean-variance restriction in standard Poisson models, allowing them to be more suitable for insurance claims which typically have large dispersions.

In an ODP model with a multiplicative mean structure, incremental claims $Y_{i,j}$ are assumed to be independent, the dispersion parameter is non cell-specific (i.e. $\phi_{i,j} = \phi$), and

$$\frac{Y_{iof,j}}{\phi} \sim \text{Poisson}\left(\frac{\alpha_i \beta_j}{\phi}\right), \tag{2.44}$$

which then implies

$$E[Y_{i,j}] = \alpha_i \beta_j, \tag{2.45}$$

$$Var[Y_{i,j}] = \phi \alpha_i \beta_j. \tag{2.46}$$

Poisson models are recovered by setting $\phi = 1$.

The relationship between the above ODP model and the traditional chain-ladder algorithm has been studied extensively in Renshaw and Verrall (1998); Mack and Venter (2000); Verrall (2000); Schmidt (2002); Taylor (2009, 2011), just to name a few. Taylor (2011) showed that the maximum likelihood estimates from ODP models are not only estimates of the chain ladder algorithm, but also minimum-variance unbiased estimates when the dispersion parameter ϕ is non cell-specific as specified in the above model. However, Verrall (2000) emphasised that it is not necessary to view chain-ladder estimates as estimates from ODP models as there certainly exists other formulations which can provide the same estimates.

In a Bayesian setting with some prior distributions for unknown parameters α_i, β_j, ϕ of ODP models, England et al. (2012) showed that the estimates of reserves are also the same as those from the chain ladder algorithm when the prior distributions are uniform and log-link functions are used. In cases where gamma prior distributions are used, the estimates of reserves are somewhat similar to estimates obtained from the Bornhuetter-Ferguson algorithm. In particular, these estimates are calculated using a combination of some prior knowledge of the total ultimate claims α_i, and development factors β_j calculated using claims experience. Similar results on Bayesian ODP models are also provided in England and Verrall (2002) and Wüthrich and Merz (2008, Chapter 4).

2.3.4.2 Gamma models

Gamma distributions, members of the Tweedie family with $p = 2$, have also been occasionally visited in the loss reserving literature. The first gamma model was introduced in Mack (1991). This model specifies each claim cell using an individual risk model

$$Y_{i,j} = \sum_{m=0}^{w_{i,j}} {}_m\dot{Y}_{i,j}, \tag{2.47}$$

where $w_{i,j}$ is the deterministic number of claims in loss cell (i, j) and ${}_m\dot{Y}_{i,j}$ is the severity of the m^{th} individual claim in this cell. Individual severities ${}_m\dot{Y}_{i,j}$ are independent and identical gamma variables. The number of claims $w_{i,j}$ can act as a unique exposure weight for cell (i, j). It then follows from properties of gamma distributions that $Y_{i,j}$ has a gamma distribution with parameters being functions of $w_{i,j}$ and parameters of ${}_m\dot{Y}_{i,j}$.

The model in Mack (1991) can also be represented using the GLM framework in Section 2.3.3 (Renshaw and Verrall, 1998; England and Verrall, 1999). In this GLM structure, we have

$$E[Y_{i,j}] = \alpha_i\beta_j, \tag{2.48}$$

$$Var[Y_{i,j}] = \phi(\alpha_i\beta_j)^2, \tag{2.49}$$

with a log-link. Discussions on gamma models can also be found in England and Verrall (2001, 2002).

2.3.4.3 Tweedie's compound Poisson models

Other distributions of the Tweedie family that have attracted attention in the reserving literature are Tweedie's compound Poisson distributions with $1 < p < 2$. These distributions are typically interpreted as compound Poisson distributions with gamma distributed severities. This formulation represents a collective risk model specification for total claims in a single loss cell. This makes them interesting because it is similar to the formulation used in pricing (England and Verrall, 2002). In addition, Tweedie's compound Poisson distributions can handle masses at 0 while many other distributions cannot. Incremental claims of size 0 can be encountered in loss reserving data due to various reasons, for example, repayments from reinsurers, or total cancellation of outstanding claims. Tweedie's compound Poisson models can be particularly useful in such cases.

The first Tweedie's compound Poisson reserving model was introduced in Wüthrich (2003). It specifies

$$Y_{i,j} = \sum_{m=0}^{W_{i,j}} {}_m\dot{Y}_{i,j}, \tag{2.50}$$

where $W_{i,j}$ is a Poisson distributed claim count and ${}_m\dot{Y}_{i,j}$ is the gamma distributed severity of the m^{th} individual claim. The gamma model by Mack (1991) reviewed in Section 2.3.4.2 is a special case of this model with deterministic claim counts. This specification can be reformulated using the GLM framework (Section 2.3.3) with

$$E[Y_{i,j}] = \alpha_i \beta_j, \tag{2.51}$$

$$Var[Y_{i,j}] = \phi_i \left(\alpha_i \beta_j\right)^p, \qquad 1 < p < 2. \tag{2.52}$$

Parameters of the models, including the power parameter p, are estimated using maximum likelihood estimation.

Peters et al. (2009) simplified $\phi_i = \phi$ in the above model and used Bayesian inference for parameter estimation. They showed that the model error inherited from fixing p can have significant impacts on the prediction error of reserve estimates.

Boucher and Davidov (2011) used a double generalised linear model with Tweedie's compound Poisson distributions to model the mean as well as the dispersion of outstanding

claims. In this model, dispersions $\phi_{i,j}$ are cell-specific and have linear structures on the log-scale. Restricted maximum likelihood estimation is used with an approximation applied to linear parameters in the specification of dispersions $\phi_{i,j}$.

Another popular model is Wright (1990) which follows the specification in Equation (2.50). Claim frequencies $W_{i,j}$ are Poisson distributed while claim severities ${}_m\dot{Y}_{i,j}$ have the means and variances of gamma distributions but are not gamma distributed. Renshaw (1996) and England and Verrall (2002) showed that this model can be simplified and represented using the GLM framework. In this model, the mean and variance are specified such that

$$E[Y_{i,j}] = \exp\left(\ddot{y}_{i,j} + a_i + r_i \log(j) + s_i j + h(i+j)\right), \tag{2.53}$$

$$Var[Y_{i,j}] = \phi_{i,j} E[Y_{i,j}]. \tag{2.54}$$

The above mean structure is similar to the Hoerl curve in Equation (2.41) with an additional cell-specific term $\ddot{y}_{i,j}$ and an inflation term $h(i+j)$. It is worth noting that the above variance structure is more similar to that of ODP models than of Tweedie models.

2.3.4.4 (Log-)Normal models

There have also been a number of appearances of (log-)normal distributions in loss reserving. see, for example De Alba (2006); Wüthrich and Merz (2008). Even though log-normal distributions are not members of the Tweedie family, normal distributions are (with $p = 0$). Log-transformed claims can be assumed to have normal distributions with

$$E[\log(Y_{i,j})] = a_i + b_j, \tag{2.55}$$

$$Var[\log(Y_{i,j})] = \phi_{i,j}. \tag{2.56}$$

De Alba (2006) developed a translated log-normal model with a constant translation term to handle masses at negative values. Bayesian inference was used to estimate the translation term as well as other parameters of this model.

2.3.4.5 Tweedie family models

While specific members of the Tweedie family of distributions have been visited frequently in the literature (see also Sections 2.3.4.1–2.3.4.4), the whole Tweedie family has made some occasional appearances. The first research that considered this whole family in detail is perhaps Alai and Wüthrich (2009). In this paper, incremental claims $Y_{i,j}$ are assumed to have distributions from the Tweedie family with the same power parameter p and dispersion parameters simplified as

$$\phi_{i,j} = \phi, \tag{2.57}$$

for the sake of simplifying the analysis. A multiplicative mean structure is used

$$E[Y_{i,j}] = \kappa'(\theta_{i,j}) = \alpha_i \beta_j, \tag{2.58}$$

with a canonical link based on Equation (2.31) so that

$$\theta_{i,j} = \begin{cases} \log(\alpha_i \beta_j), & p = 1, \\ \dfrac{(\alpha_i \beta_j)^{1-p}}{1-p}, & p \neq 1. \end{cases} \tag{2.59}$$

The variance specification then follows that of the Tweedie family with

$$Var[Y_{i,j}] = \phi \left(\mu_{i,j}\right)^p. \tag{2.60}$$

Maximum likelihood estimation is used to obtain estimators of α_i and β_j. Pearson residuals are then used to estimate the dispersion parameter

$$\widehat{\phi} = \left(\frac{(I+1)(I+2)}{2} - 2I - 1 \right) \sum_{i+j \leq I} \frac{(Y_{i,j} - \widehat{\mu}_{i,j})^2}{V(\widehat{\mu}_{i,j})}, \tag{2.61}$$

where $\widehat{\mu}_{i,j}$ are fitted mean values calculated using parameter estimates $\widehat{\alpha}_i$ and $\widehat{\beta}_j$

Alai and Wüthrich (2009) assessed the sensitivity of reserves estimates and their mean square error of predictions with respect to p. This sensitivity test was carried out by using Taylor approximations to express these quantities as functions of p. They showed that reserves estimates are relatively insensitive to p while their mean square error of predictions vary quite

significantly.

Tweedie models with multiplicative mean structures specified as above are also referred to as cross-classified Tweedie models in Taylor (2009). Taylor (2009) showed that maximum likelihood estimates from cross-classified Tweedie models are generally not equivalent to the chain ladder estimates except for ODP models.

Negative claims can be occasionally observed in loss triangles due to, for example, salvage recoveries, or payment from third parties. Many models for incremental claims, including the many members of the Tweedie family of distributions mentioned so far, are unable to handle negative observations due to their lack of support for negative masses. Exceptions are Gaussian distributions with $p = 0$. A remarkably small area of literature has been devoted for the treatment of negative payments a single business line, not necessarily on the Tweedie family. The existing methods include a three-parameter-log-normal model in De Alba (2006) and a mixture model in Kunkler (2006).

2.4 Multivariate reserving models

In this section, we review reserving models for multiple business segments with dependence. Section 2.4.1 explains the motivation for dependence modelling in reserving models. Three types of multivariate models are then reviewed: copula models (Section 2.4.2), multivariate models with specific marginals (Section 2.4.3), and common shock models (Section 2.4.4).

2.4.1 Motivation for multivariate reserving

A general insurance company typically operates in multiple business lines/segments whose risks are dependent to some extent. For example, a temporary change in the internal operations of the company can speed up the processing of claims in all segments, or an adverse weather event can result in simultaneous claims for different lines or segments. In this section we will examine the impacts of dependency across business segments on the estimation of reserves and their associated predictive variabilities.

A common approach for outstanding claims valuation is called the "silo" approach (Ajne, 1994). In this approach, reserves and their predictive variabilities are assessed for each individual business line or segment (also called silo). The reserves estimate and its predictive variability on the portfolio level are then obtained by aggregating the corresponding estimates from the silos with some adjustments if required.

A quantity of great interest is the central estimate of outstanding claims on the portfolio level, also known as the reserves estimate. Theoretically, the calculation of the aggregate central estimate using the "silo" approach is unaffected by the dependencies across business segments given the estimates are optimal. This is due to the additivity property of expected values where

$$E[Y_1 + Y_2] = E[Y_1] + E[Y_2], \tag{2.62}$$

with Y_1, Y_2 being any two random variables. However, as mentioned in Shi et al. (2012), jointly modelling claims from all individual segments allows a cross-borrowing of information which can improve the accuracy of outstanding claims valuation. In other words, the optimal total central estimate obtained from the joint modelling of claims from multiple segments may not be the same as the sum of the optimal central estimates in the "silo" approach.

Insurers are also interested in some measures of the variabilities associated with the central estimates of their outstanding claims liabilities for risk management and regulatory purposes (see also Section 2.3.1). Two common measures are (i) standard deviations, and, (ii) quantiles, which are formally required by regulators. Solvency II in Europe requires insurers to provide the standard deviations of their total outstanding claims liabilities over a one-year horizon. The APRA's GPS 340 in Australia requires insurers to hold risk margins specified using central estimates, $\text{VaR}_{75\%}$, and standard deviations of their outstanding claims (Equation (2.10)). The $\text{VaR}_{99.5\%}$ of total outstanding claims are inputs in the assessment of solvency capital under Solvency II as well as APRA's GPS 110. Standard deviations are known to be sub-additive (see for example, Embrechts et al., 2005, Joshi and Paterson, 2013, Miller, 2013), i.e.

$$SD[Y_1 + Y_2] \leq SD[Y_1] + SD[Y_2]. \tag{2.63}$$

On the other hand, quantiles are known to be non-sub-additive. When the "silo"

approach is applied, the standard deviation or quantiles on the portfolio level are obtained by simply aggregating the corresponding measures from each silo. These results are not likely to be the same as the measures obtained directly on the aggregate portfolio due to the non-additivity feature of these measures. Hence, they may not reflect truly on the risk level of the portfolio.

A well-known benefit of considering the dependency across business segments in the valuation of the aggregate outstanding claims liability are diversification benefits, see for example, Shi and Frees (2011); De Jong (2012); Côté et al. (2016);Avanzi, Taylor and Wong (2016). Diversification benefits, as defined in Avanzi, Taylor and Wong (2016), are the benefits arising when the risk associated with a collection of segments is less than the sum of their individual risks. By considering the accurate dependence structure across business segments and allowing for diversification benefits, the predictive variabilities of reserves can be correctly assessed. This can lead to a reduction in risk margins and capital that insurers have to hold. As mentioned in Ajne (1994), even if a certain degree of prudence is recommended, insurers should have as correct reserves as possible, not as large reserves as possible. This is to ensure that capital is used parsimoniously while meeting solvency expectations (Avanzi, Taylor and Wong, 2016).

The benefits mentioned above have motivated the development of stochastic loss reserving methods for multiple segments with dependence, see for example, Schmidt (2006); Merz and Wüthrich (2009b); Zhang and Dukic (2013); Abdallah et al. (2015); Shi (2014), and many more. Methodologies used in this area of the literature can be classified into two main types: parametric models and non-parametric models (Shi et al., 2012) which are multivariate extensions in the two strands of research mentioned in Section 2.3.1. Some well known multivariate non-parametric models include multivariate chain ladder models (Braun, 2004; Merz and Wüthrich, 2008; Zhang, 2010), multivariate additive models (Schmidt, 2006; Hess et al., 2006; Merz and Wüthrich, 2009b) and a multivariate model which combines multivariate chain ladder and multivariate additive loss reserving models (Merz and Wüthrich, 2009a). In this thesis, we focus on parametric models. A review of relevant existing models is provided in subsections below.

2.4.2 Copula models

Copulas are very well-known multivariate modelling tools in the literature as well as practice due to their modelling flexibility. Their applications can be found in various fields including statistics, finance and insurance. Some of the theory of copulas is provided in Section 2.4.2.1 covering the definition, as well as a description of two popular copula subclasses: elliptical copulas and Archimedean copulas. Sections 2.4.2.2 and 2.4.2.3 review the applications of elliptical copulas and Archimedean copulas in reserving, respectively. A general copula reserving framework is reviewed in Section 2.4.2.4.

2.4.2.1 Theory of copulas

The foundation for many applications of copulas is the Sklar's theorem. This theorem (see for example, Embrechts et al., 2005; Zhang and Dukic, 2013) establishes the existence of a unique copula function $C : [0,1]^N \to [0,1]$ for a random variable vector $\boldsymbol{Y} = (Y_1, ..., Y_N)'$ such that

$$F_{\boldsymbol{Y}}(\boldsymbol{y}) = C(F_{Y_1}(y_1), \ldots, F_{Y_N}(y_N)), \tag{2.64}$$

where $F_{\boldsymbol{Y}}(.)$ is the joint distribution of $(Y_1, ..., Y_N)'$

$$F_{\boldsymbol{Y}}(\boldsymbol{y}) = \Pr(Y_1 \leq y_1, ..., Y_N \leq y_N). \tag{2.65}$$

In brief, a copula translates the dependence between variables into the dependence on the transformation of these random variables into the uniform scale, also called the cumulative distribution transformation. Under standard smoothness conditions, Equation (2.64) can be differentiated to give

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = c(F_{Y_1}(y_1), ..., F_{Y_N}(y_N)) \prod_{n=1}^{N} f_{Y_n}(y_n), \tag{2.66}$$

where

$$c(u_1, ..., u_N) = \frac{\partial^N C(u_1, ..., u_N)}{\partial u_1 ... \partial u_N}, \tag{2.67}$$

with $u_n = F_{Y_n}(y_n)$. This means that marginal densities and the dependence structure can be separated. Using a copula, we can combine various marginal distributions with a variety of copulas with different dependence structures. This offers great flexibility for modelling.

There are two popular families of copulas which are the elliptical copula family and the Archimedean family. Elliptical copulas are copulas of elliptical distributions. The elliptical family of distributions is a very rich family of symmetric distributions, including normal distributions and t distributions. Following the definition in Shi and Frees (2011), a random variable vector $\boldsymbol{Y}$ has a multivariate elliptical distribution with the location parameter $\mathbf{0}$, the dispersion matrix $\boldsymbol{\Sigma}_Y$ and the density

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = \frac{1}{|\boldsymbol{\Sigma}_Y|^{1/2}} \dot{f}\left(\frac{1}{2}\boldsymbol{y}'\boldsymbol{\Sigma}_Y^{-1}\boldsymbol{y}\right), \tag{2.68}$$

where $\dot{f}(.)$ is called the density generator function. The dispersion matrix $\boldsymbol{\Sigma}_Y$ captures the association among individual variables $Y_1, ..., Y_N$, all of which follow univariate elliptical distributions. The corresponding elliptical copula is then defined using the Sklar's theorem in Equation (2.64)

$$C(u_1, ..., u_N) = F_{\boldsymbol{Y}}\left(F_{Y_1}^{-1}(u_1), ..., F_{Y_N}^{-1}(u_N)\right) \tag{2.69}$$

with the density

$$c(u_1, ..., u_N) = f_{\boldsymbol{Y}}\left(F_{Y_1}^{-1}(u_1), ..., F_{Y_N}^{-1}(u_N)\right) \prod_{n=1}^{N} \frac{1}{f_{Y_n}\left(F_{Y_n}^{-1}(u_n)\right)}. \tag{2.70}$$

A comprehensive review of elliptical distributions, their properties and elliptical copulas can be found in Embrechts et al. (2003, 2005). A review of applications of elliptical distributions in insurance is provided in Landsman and Valdez (2003).

An Archimedean copula, as defined in, for example Embrechts et al. (2005); Zhang and Dukic (2013); Abdallah et al. (2015), is specified such that

$$C(u_1, u_2, ..., u_N) = \varphi^{-1}\left(\varphi(u_1) + ... + \varphi(u_N)\right), \tag{2.71}$$

where function $\varphi(.)$ is the generator of the copula. This generator function is convex, non-decreasing with the domain $(0, 1]$, the range $[0, \infty)$ and $\varphi(1) = 0$. The copula density is then given by

$$c(u_1, ..., u_N) = \varphi^{(N)}\left(\varphi^{-1}(u_1) + ... + \varphi^{-1}(u_N)\right) \prod_{n=1}^{N} \left(\varphi^{-1}\right)'(u_n), \tag{2.72}$$

where $\varphi^{(N)}(.)$ is the N^{th} derivative, and $\varphi^{-1}(.)$ is the inverse function of the generator function. The density $c(.)$ exists if and only if $\varphi^{(N-1)}(.)$ exists and is absolutely continuous on $(0, \infty)$ (Abdallah et al., 2015).

Another important concept of Archimedean copulas is nested Archimedean copulas. Following the definition in Côté et al. (2016), we say that a $(N+1)$- variate copula C_N is fully nested with generators $\varphi_1, ..., \varphi_N$ if it is recursively defined for all $(u_1, ..., u_N)$ by

$$C_1(u_1, u_2) = \varphi_1^{-1}\left[\varphi_1(u_1) + \varphi_1(u_2)\right], \tag{2.73}$$

$$C_2(u_1, u_2, u_3) = \varphi_2^{-1}\left[\varphi_2(u_3) + \varphi_2\left(C^{(1)}(u_1, u_2)\right)\right], \tag{2.74}$$

$$\vdots \qquad = \qquad \vdots$$

$$C_N(u_1, ..., u_{N+1}) = \varphi_N^{-1}\left[\varphi_N(u_{N+1}) + \varphi_N\left(C^{(N-1)}(u_1, ..., u_N)\right)\right], \tag{2.75}$$

where $\varphi_1^{-1}, ..., \varphi_N^{-1}$ are strictly monotone and $\varphi_{n+1}(\varphi_n^{-1}(.))$ has strictly monotone derivatives for all $n \in \{1, ..., N-1\}$. Nested Archimedean copula structures allow different dependence structures to be incorporated into the modelling of multi-dimensional data.

Some popular members of the Archimedean family include Clayton copulas, Gumbel copulas and Frank copulas. Reviews of Archimedean copulas can be found in Nelsen (1999); Embrechts et al. (2003, 2005) and Durante and Sempi (2010).

2.4.2.2 Elliptical copula models

In this section we review multivariate reserving models that use elliptical copulas. A special case of elliptical copulas, namely Gaussian copulas, have received some attention due to their tractability and their ease of interpretation of dependence structures.

Shi and Frees (2011) provided a copula regression model for cell-wise dependence, i.e. the dependence between loss cells that are in the same position across business segments. In this model, all cells in the same position are first collected into a vector

$$\boldsymbol{Y}_{i,j} = \begin{pmatrix} Y_{i,j}^{(1)} \\ \vdots \\ Y_{i,j}^{(N)} \end{pmatrix}. \tag{2.76}$$

Individual claim cells $Y_{i,j}^{(n)}$ are modelled using the GLM framework with the chain ladder mean structure as described in Section 2.3.3. In particular, the means of loss cells $E[Y_{i,j}^{(n)}]$ are specified using some link functions and the linear predictor

$$a_i + b_j. \tag{2.77}$$

The two distributions chosen for demonstration are log-normal distributions (with a log-link) and gamma distributions (with an inverse link). The copula chosen for demonstration in this paper is an elliptical copula. The copula specification is similar to that described in Section 2.4.2.1 with the dispersion matrix specified as

$$\boldsymbol{\Sigma}_{\boldsymbol{Y}_{i,j}} = \begin{pmatrix} \sigma^{(1,1)} & \sigma^{(1,2)} & \dots & \sigma^{(1,N)} \\ \sigma^{(2,1)} & \sigma^{(2,2)} & \dots & \sigma^{(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{(N,1)} & \sigma^{(N,2)} & \dots & \sigma^{(N,N)} \end{pmatrix}, \tag{2.78}$$

where $\sigma^{(m,n)}$ specifies the association between individual cells $Y_{i,j}^{(m)}$ and $Y_{i,j}^{(n)}$ for all i and j. Maximum likelihood estimation is then used to estimate parameters of the marginal distributions and the copula used in the model.

Two examples of Gaussian copula models in loss reserving are De Jong (2012) and Shi (2014). The model in De Jong (2012) captures calendar period dependence within and across business segments while Shi (2014) generalised this model to also allow for cell-wise dependence. In these models, claims in the same calendar period from all loss triangles are first collected into a vector

$$\boldsymbol{Y}_t = \begin{pmatrix} Y_{1,t}^{(1)} \\ Y_{2,t-1}^{(1)} \\ \vdots \\ Y_{t,1}^{(N)} \end{pmatrix}. \tag{2.79}$$

Using the Sklar's theorem stated in Equation (2.64) with a multivariate Gaussian dependence structure, these models can be represented as

$$F_{\boldsymbol{Y}_t}(\boldsymbol{Y}_t) = C\left((\boldsymbol{I}_N \otimes \boldsymbol{1}_t)\boldsymbol{U}_t + (\boldsymbol{I}_N \otimes \boldsymbol{I}_t)\boldsymbol{Z}_t\right), \tag{2.80}$$

where $\boldsymbol{1}_t$ is a vector of 1's of length t, $\boldsymbol{I}_t$, $\boldsymbol{I}_N$ are identity matrices of dimension $t \times t$ and

$N \times N$, respectively. A tensor product of any matrices U and Z, denoted by $U \otimes Z$ is defined, see, for example in Taylor (2018), such that

$$U \otimes Z = \begin{pmatrix} u_{11}Z & u_{12}Z & \dots & u_{1n}Z \\ u_{21}Z & u_{22}Z & \dots & u_{2n}Z \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1}Z & u_{m2}Z & \dots & u_{mn}Z \end{pmatrix} \tag{2.81}$$

where u_{ij} denotes the (i, j) element of U.

We have the calendar period dependence deduced by

$$\boldsymbol{U}_t = \begin{pmatrix} U_t^{(1)} \\ \vdots \\ U_t^{(N)} \end{pmatrix} \sim \text{Normal}\left(\mathbf{0}, \boldsymbol{\Sigma}_U\right), \tag{2.82}$$

and cell-wise dependence deduced by

$$\boldsymbol{Z}_{i,j} = \begin{pmatrix} Z_{i,j}^{(1)} \\ \vdots \\ Z_{i,j}^{(N)} \end{pmatrix} \sim \text{Normal}\left(\mathbf{0}, \boldsymbol{\Sigma}_Z\right). \tag{2.83}$$

Dispersion matrices $\boldsymbol{\Sigma}_U$ and $\boldsymbol{\Sigma}_Z$ are covariance matrices of $\boldsymbol{U}_t$ and $\boldsymbol{Z}$, respectively. They are specified such that standard normal marginals are obtained for $(\boldsymbol{I}_N \otimes \mathbf{1}_t)\boldsymbol{U}_t + (\boldsymbol{I}_N \otimes \boldsymbol{I_t})\boldsymbol{Z}_t$. With $\boldsymbol{\Sigma}_Z = \boldsymbol{I}_N$ we obtain the model in De Jong (2012) where no cell-wise dependence is assumed. In brief, the percentile of $\boldsymbol{Y}_t$ is mapped to the percentile of the standard normal variables vector $(\boldsymbol{I}_N \otimes \mathbf{1}_t)\boldsymbol{U}_t + (\boldsymbol{I}_N \otimes \boldsymbol{I_t})\boldsymbol{Z}_t$ in these models.

Shi (2014) used Tweedie's compound Poisson distributions with the chain-ladder mean structure for marginal modelling. This is similar to the univariate Tweedie model reviewed in Section 2.3.4.5. Maximum likelihood estimation is used in both models for parameters estimation. While Tweedie's compound Poisson distributions are chosen for demonstration, other marginal distributions can also be used.

2.4.2.3 Archimedean copula models

Archimedean copula models in the literature include Abdallah et al. (2015) and Côté et al. (2016). The former model considers two business segments with a hierarchical structure while the later extends it to more dimensions using a nested Archimedean structure.

The hierarchical Archimedean copula model in Abdallah et al. (2015) aims to capture calendar year dependence within and between two business segments. Appropriate marginal distributions are first used to fit the data. Cumulative distribution values of claims in the same calendar period t, i.e. $F_{Y^{(n)}_{i,t-i+1}}(y^{(n)}_{i,t-i+1}) = u^{(n)}_{i,t-i+1}$, are then collected into a vector

$$\begin{pmatrix} u^{(1)}_{1,t} \\ \vdots \\ u^{(1)}_{t,1} \\ u^{(2)}_{1,t} \\ \vdots \\ u^{(2)}_{t,1} \end{pmatrix}. \tag{2.84}$$

Two levels of Archimedean copulas are then used on this vector

$$C_2 = C_2\left(u^{(1)}_{1,t}, ..., u^{(1)}_{t,1}, u^{(2)}_{1,t}, ..., u^{(2)}_{t,1}\right) \tag{2.85}$$

$$= C_2\left(C_{1,1}\left(u^{(1)}_{1,t}, ..., u^{(1)}_{t,1}\right), C_{1,2}\left(u^{(2)}_{1,t}, ..., u^{(2)}_{t,1}\right)\right) \tag{2.86}$$

$$= \varphi_2^{-1}\left(\varphi_2\left(\varphi_{1,1}^{-1}\left[\varphi_{1,1}\left(u^{(1)}_{1,t}\right) + ... + \varphi_{1,1}\left(u^{(1)}_{t,1}\right)\right]\right) + \varphi_2\left(\varphi_{1,2}^{-1}\left[\varphi_{1,2}\left(u^{(2)}_{1,t}\right) + ... + \varphi_{1,2}\left(u^{(2)}_{t,1}\right)\right]\right)\right), \tag{2.87}$$

where $C_{1,1}, C_{1,2}$ and $\varphi_{1,1}, \varphi_{1,2}$ are Archimedean copulas and their corresponding generators applied subset (1) and (2) respectively on the first level of dependence, C_2 is an Archimedean copula and φ_2 is its generator applied to the second level of dependence.

This model applies the nested Archimedean copulas concept described in Section 2.4.2.1 to a two-level dependence structure. In Abdallah et al. (2015), the first level of dependence, level 1, is the dependence between claims within a single calendar period in a single business segment. It is assumed that the dependence within a calendar period is identical across all calendar periods with identical copulas $C_{1,n}$, $n = 1, 2$. The second level of dependence, level 2, is the dependence across business segments. A copula C_2 with its generator φ_2 is applied

on top of the first level of dependence, as shown in Equation (2.87).

Côté et al. (2016) extended the above structure to a multi-dimensional dependence structure. Their model was applied on a portfolio of six business segments with cell-wise dependence. Rank based methods were used for copulas fitting. These included analysing the dependence structure using standardised ranks of residuals obtained from marginal GLM fitting and using empirical copulas on these standardised ranks to select, fit and validate copulas used.

2.4.2.4 General copula framework

The previous two sections review models that use specific families of copulas. In this section, we describe a general copula framework that does not have any specification on the copulas used. This framework is described in Shi and Frees (2011); Zhang and Dukic (2013) and Shi (2017). While the first two references look at the dependence across segments within a single company, the last one assesses the dependence across loss triangles from various companies.

In the general copula framework, various choices can be used for marginal modelling and dependence modelling. Common choices of marginal distributions typically include GLMs (a review of which is provided in Section 2.3.3). Other choices also include non-linear growth models Zhang and Dukic (2013) and semi-parametric smoothing models Zhang and Dukic (2013); Shi (2017). The dependence structure is flexible as any copulas can be used. The application of copulas on the marginals follows the Sklar's theorem in Equation (2.64).

Unlike other copula models described in Sections 2.4.2.2 and 2.4.2.3, Bayesian inference are used in Zhang and Dukic (2013) and Shi (2017) for parameter estimation. These applications are motivated by rapid advancements in Markov chain Monte Carlo (MCMC) techniques and convenience in the generation of full distributions of outstanding claims.

2.4.3 Multivariate models with specific marginals

Besides copulas, multivariate distributions with specific marginals have also been used occasionally in multivariate reserving models. We review this type of models within this section.

2.4.3.1 Multivariate log-normal models

There are two well-known multivariate log-normal models in the literature developed by Shi et al. (2012) and Merz et al. (2013). As their names suggest, individual claims are assumed to follow log-normal distributions. These two models, however, are significantly different in terms of their structures as well as their dependence properties.

The multivariate log-normal model in Shi et al. (2012) focuses on modelling incremental claims $Y_{i,j}^{(n)}$ with cell-wise dependence and calendar period dependence. This model uses a multivariate log-normal assumption on a vector of incremental claims from the same position across triangles

$$\boldsymbol{Y}_{i,j} = \begin{pmatrix} Y_{i,j}^{(1)} \\ \vdots \\ Y_{i,j}^{(N)} \end{pmatrix}, \tag{2.88}$$

with the multivariate density

$$\begin{aligned} & f_{\boldsymbol{Y}_{i,j}}\left(\boldsymbol{y}_{i,j}\right) \\ & = \frac{1}{(2\pi)^{N/2}|\tilde{\boldsymbol{\Sigma}}|^{1/2}\left(\prod_{n=1}^{N} y_{i,j}^{(n)}\right)} \exp\left(\frac{1}{2}(\log \boldsymbol{y}_{i,j} - \boldsymbol{\theta}_{i,j})'\tilde{\boldsymbol{\Sigma}}^{-1}(\log \boldsymbol{y}_{i,j} - \boldsymbol{\theta}_{i,j})\right), \end{aligned} \tag{2.89}$$

where

$$\boldsymbol{\theta}_{i,j} = \begin{pmatrix} \theta_{i,j}^{(1)} \\ \vdots \\ \theta_{i,j}^{(N)} \end{pmatrix}, \qquad \tilde{\boldsymbol{\Sigma}} = \begin{pmatrix} \tilde{\sigma}^{(1,1)} \dots \tilde{\sigma}^{(1,N)} \\ \vdots \quad \ddots \quad \vdots \\ \tilde{\sigma}^{(N,1)} \dots \tilde{\sigma}^{(N,N)} \end{pmatrix}, \tag{2.90}$$

specify the parameters of the multivariate distribution. The covariances between $\log(Y_{i,j}^{(n)})$ from any pair of loss triangles m, n is denoted as $\tilde{\sigma}^{(m,n)}$.

The mean structure in this model is a chain-ladder structure with an additional term for calendar period effect

$$\theta_{i,j}^{(n)} = a_i^{(n)} + b_j^{(n)} + h_{t=i+j-1}, \tag{2.91}$$

where h_t is the random calendar period effect common to all cells in the same calendar period t across all business segments. This random effect can have a time series specification such

as a random walk

$$h_t = h_{t-1} + {}_h\epsilon_t, \qquad {}_h\epsilon_t \sim \text{Normal}(0, \sigma^2_{{}_h\epsilon}), \tag{2.92}$$

or an Auto-regressive (AR) process

$$h_t = {}_h\tau.h_{t-1} + {}_h\epsilon_t, \qquad {}_h\epsilon_t \sim \text{Normal}(0, \sigma^2_{{}_h\epsilon}), \tag{2.93}$$

where ${}_h\tau$ is a coefficient of the AR(1) process. This structure allows an evolution of the factor h_t over time. Bayesian inference is used for model estimation.

In Merz et al. (2013), the multivariate normal assumption is used on log-link ratios of cumulative claims defined such that

$$D^{(n)}_{i,j} = \log\left(\frac{X^{(n)}_{i,j}}{X^{(n)}_{i,j-1}}\right). \tag{2.94}$$

This in turn implies a log-normal distributional assumption on cumulative claims $X^{(n)}_{i,j}$. The recursive specification structure of this model is a chain ladder type (Mack, 1993). The dependence from all possible dimensions and sources within and between loss triangles is considered in this model. All log-link ratio variables are first collected into a vector

$$\boldsymbol{D}_{i,j} = \begin{pmatrix} D^{(1)}_{i,j} \\ \vdots \\ D^{(N)}_{i,j} \end{pmatrix}, \qquad \boldsymbol{D}_i = \begin{pmatrix} \boldsymbol{D}_{i,1} \\ \vdots \\ \boldsymbol{D}_{i,J} \end{pmatrix}, \qquad \boldsymbol{D} = \begin{pmatrix} \boldsymbol{D}_1 \\ \vdots \\ \boldsymbol{D}_I \end{pmatrix}. \tag{2.95}$$

The vector $\boldsymbol{D}$ is then assumed to have a multivariate normal distribution with mean $\boldsymbol{\delta}$ and variance $\boldsymbol{\Sigma}_D$. Bayesian inference is used with a multivariate normal specification for the prior distribution of $\boldsymbol{\delta}$. In this specification, the mean of $\boldsymbol{\delta}$ is $\boldsymbol{\Delta}$ and the variance is $\boldsymbol{\Sigma}_{\boldsymbol{\delta}}$. The posterior density is then given by

$$\begin{aligned} f_{\boldsymbol{D}}(\boldsymbol{D}) = \quad & \frac{1}{(2\pi)^{\tilde{N}/2}|\boldsymbol{\Sigma}_D|^{1/2}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{D}-\boldsymbol{\delta}\right)'\boldsymbol{\Sigma}_D^{-1}\left(\boldsymbol{D}-\boldsymbol{\delta}\right)\right\} \\ & \times \frac{1}{(2\pi)^{N_\delta/2}|\boldsymbol{\Sigma}_{\boldsymbol{\delta}}|^{1/2}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\delta}-\boldsymbol{\Delta}\right)'\boldsymbol{\Sigma}_{\boldsymbol{\delta}}^{-1}\left(\boldsymbol{\delta}-\boldsymbol{\Delta}\right)\right\}, \end{aligned} \tag{2.96}$$

where $\tilde{N}$ denotes the total number of observations in the data set, and N_δ the length of $\boldsymbol{\delta}$. Closed-form estimates of reserves and their corresponding uncertainty estimate are also provided in Merz et al. (2013).

2.4.3.2 Multivariate gamma model

Vu (2013) developed a multivariate gamma model for outstanding claims with cell-wise dependence across business segments. The multivariate gamma distribution used in this model was developed using the multivariate reduction technique in Mathai and Moschopoulos (1991) and studied further by Furman (2008). This technique is explained in detail in Section 2.4.4.2.

In this model, incremental claims are specified such that

$$Y_{i,j}^{(n)} = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{-1} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}, \tag{2.97}$$

$$U_{i,j} \sim \text{Gamma}\left(\frac{1}{\tilde{\phi}}, \frac{1}{\tilde{\alpha}\tilde{\phi}}\right), \tag{2.98}$$

$$Z_{i,j}^{(n)} \sim \text{Gamma}\left(\frac{1}{\ddot{\phi}^{(n)}}, \frac{1}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\ddot{\phi}^{(n)}}\right), \tag{2.99}$$

where the first parameters of the above distributions are called shape parameters, and the second parameters are called rate parameters. It follows that the vector $\boldsymbol{Y}_{i,j} = (Y_{i,j}^{(1)}, ..., Y_{i,j}^{(n)})'$ has a multivariate gamma distribution with marginal distributions

$$Y_{i,j}^{(n)} \sim \text{Gamma}\left(\frac{1}{\ddot{\phi}^{(n)}} + \frac{1}{\tilde{\phi}}, \frac{1}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\ddot{\phi}^{(n)}}\right). \tag{2.100}$$

In this model, the incremental claim $Y_{i,j}^{(n)}$ in each loss triangle is the sum of two components: a systematic component $U_{i,j}$ and an idiosyncratic component $Z_{i,j}^{(n)}$. The systematic component induces the cell-wise dependence across triangles. This is a typical common shock model structure (more details of which are provided in Section 2.4.4). Bayesian inference is then used for model estimation.

2.4.4 Common shock models

In insurance, common shock approaches have been quite well known dependence modelling tools (Lindskog and McNeil, 2003). According to the International Actuarial Association (2004) report, common shock approaches are typically used to capture structural dependence (i.e. structural co-movements that are due to known relationships which can be

accounted for in a modelling framework). Hence such models provide ease of interpretation for the dependence structure. Many of such models also allow correlation matrices to be constructed in a parsimonious and disciplined manner.

These approaches have been occasionally used in the literature. Two main types of common shock modelling techniques used are models using random factors, and models using multivariate reduction techniques. These two types of models are reviewed in Section 2.4.4.1 and 2.4.4.2.

2.4.4.1 Models using random factors

A group of common shock models randomise model factors to introduce dependence. They are also referred to as also Bayesian models in the literature by Wüthrich (2010); Salzmann and Wüthrich (2012). These models are hierarchical with randomised factors constituting different levels in the hierarchy. This is a typical Bayesian structure, however, in these models, random factors are selectively chosen and specified to capture specific dependence sources.

A number of random factor models have been introduced to the literature. In the literature area for one single business segment, Bayesian models were considered in Wüthrich (2010) and Salzmann and Wüthrich (2012) for calendar period dependence. De Alba and Nieto-Barajas (2008) considered a Bayesian model with a correlated latent process to capture dependence within an accident period.

Two multivariate log-normal models reviewed in Section 2.4.3.1 also use random factors to capture dependence. In Shi et al. (2012), a modified chain ladder mean structure is used with

$$\theta_{i,j}^{(n)} = a_i^{(n)} + b_j^{(n)} + h_{t=i+j-1}, \tag{2.91}$$

where h_t is an additional random effect term that deduces dependency across cells within the same calendar period within and across triangles. This factor can be specified using a time series process, such as an AR(1) process

$$h_t = {}_h\tau \cdot h_{t-1} + {}_h\epsilon_t, \qquad {}_h\epsilon_t \sim \text{Normal}(0, \sigma_{{}_h\epsilon}^2). \tag{2.93}$$

The serial correlation between h_t and h_{t-1} in this structure further captures the dependence

across calendar periods. In Merz et al. (2013), the vector of log-link development ratios $\boldsymbol{D}$ is assumed to have a multivariate normal distribution with mean $\boldsymbol{\delta}$ and variance $\boldsymbol{\Sigma}_D$. The mean vector $\boldsymbol{\delta}$ is then assumed to have a multivariate normal distribution with mean $\boldsymbol{\Delta}$ and variance $\boldsymbol{\Sigma}_{\boldsymbol{\delta}}$. This hierarchy introduces a layer of dependency in the model.

Abdallah et al. (2016) developed a bivariate Sarmanov model for two business segments with dependence. This was based on a bivariate Sarmanov distribution introduced in Ting Lee (1996). In this model, claim cells $Y_{i,j}^{(1)}$ from the first loss triangle are specified with log-normal distributions

$$Y_{i,j}^{(1)}|\eta_t^{(1)} \sim \text{Log-normal}\left(\theta_{i,j}^{(1)}\eta_t^{(1)}, \tilde{\sigma}^2\right), \tag{2.101}$$

where $\eta_t^{(1)}$ is a random calendar period factor that captures dependence across cells within the calendar period t in business segment (1), $\tilde{\sigma}^2$ is the variance parameter of the log-normal distribution, and $\theta_{i,j}^{(1)}$ is specified such that

$$\theta_{i,j}^{(1)} = a_i^{(1)} + b_j^{(1)}. \tag{2.102}$$

The random factor $\eta_t^{(1)}$ is assumed to have a normal distribution

$$\eta_t^{(1)} \sim \text{Normal}(e, k^2), \tag{2.103}$$

where e and k^2 are parameters of the distribution.

Claim cells $Y_{i,j}^{(2)}$ from the second business segment are assumed to have gamma distributions

$$Y_{i,j}^{(2)}|\eta_t^{(2)} \sim \text{Gamma}\left(\frac{1}{\phi^{(2)}}, \frac{\eta_t^{(2)}}{\exp(\theta_{i,j}^{(2)})\phi^{(2)}}\right), \tag{2.104}$$

where $\eta_t^{(2)}$ is a random calendar period factor that captures the dependence across cells within the calendar period t in business segment (2), $\frac{1}{\phi^{(2)}}$ is the shape parameter and $\frac{\eta_t^{(2)}}{\exp(\theta_{i,j}^{(2)})\phi^{(2)}}$ is the rate parameter with

$$\theta_{i,j}^{(2)} = a_i^{(2)} + b_j^{(2)}. \tag{2.105}$$

A gamma distribution is then used to model the random calendar period factor

$$\eta_t^{(2)} \sim \text{Gamma}(\tilde{e}, \tilde{k}), \tag{2.106}$$

where $\tilde{e}$ and $\tilde{k}$ are parameters of the distribution.

A distribution from the Sarmanov family of bivariate distributions is then used to model the joint distribution of $\eta_t^{(1)}$ and $\eta_t^{(2)}$, which is expressed as

$$f_{\eta_t^{(1)},\eta_t^{(2)}}\left(\eta_t^{(1)}, \eta_t^{(2)}\right) = f_{\eta_t^{(1)}}\left(\eta_t^{(1)}\right) f_{\eta_t^{(2)}}\left(\eta_t^{(2)}\right)\left[1 + \xi\tilde{\varphi}_1\left(\eta_t^{(1)}\right)\tilde{\varphi}_2\left(\eta_t^{(2)}\right)\right], \tag{2.107}$$

where ξ is a real number that satisfies the condition

$$1 + \xi\tilde{\varphi}_1\left(\eta_t^{(1)}\right)\tilde{\varphi}_2\left(\eta_t^{(2)}\right) \geq 0 \text{ for all } \theta_t^{(1)}, \theta_t^{(2)}, \tag{2.108}$$

and $\tilde{\varphi}_1(.)$ and $\tilde{\varphi}_2(.)$ are functions to be specified. Given the choices of distribution for $\theta_t^{(1)}$ and $\theta_t^{(2)}$, we have

$$\tilde{\varphi}_1\left(\eta_t^{(1)}\right) = \exp\left(\eta_t^{(1)}\right) - \exp\left(-e + \frac{k^2}{2}\right), \tag{2.109}$$

$$\tilde{\varphi}_2\left(\eta_t^{(2)}\right) = \exp\left(\eta_t^{(2)}\right) - \left(1 + \tilde{k}\right)^{-\tilde{e}}. \tag{2.110}$$

With the choice of conjugate prior distributions for random factors as specified, closed-form expected values and variances can be obtained. The above structure can be modified for accident period dependence and development period dependence.

2.4.4.2 Models using multivariate reduction technique

The multivariate reduction technique is an appealing technique for constructing multivariate distributions, see for example Johnson et al. (1997, 2002); Karlis (2003); Furman (2008); Furman and Landsman (2010). In this technique, dependent variables are created as functions of a number of common independent variables. Mathematically, we have independent random variables $\tilde{Y}_n$, $n = 0, ..., N$ with distributions $F_{\tilde{Y}_n}$, $n = 0, ..., N$. Random variables Y_n, $n = 1, ..., N$ are constructed such that

$$Y_n = \tilde{g}(\tilde{Y}_0, \tilde{Y}_n), \tag{2.111}$$

where $\tilde{g}(.)$ is a specified function. In this expression, $\tilde{Y}_0$ is the common shock, and $\tilde{Y}_n$ is the idiosyncratic component. The common shock component then deduces the dependence across Y_n, $n = 1, ..., N$. We are particularly interested in the choice of the function g where the closure under the taking of marginals property can be satisfied. These are cases where Y_n all have the same type of distribution. This is also one of the four desirable properties of multivariate distributions considered by Joe (1997, Chapter 4). These properties are also listed in Chapter 1. The joint density of Y_n, $n = 1, ..., N$ is then a multivariate density with the corresponding marginal density of Y_n. It is worth noting that there is often no unique definition of multivariate distributions for most marginal distributions, except for the multivariate normal distribution which is is clearly defined (Johnson et al., 2002). This technique has been used to construct many multivariate distributions, including a multivariate Poisson distribution (see for example, Karlis, 2003), a multivariate gamma distribution (see for example, Mathai and Moschopoulos, 1991), and a multivariate Tweedie distribution (Furman and Landsman, 2010).

Common shock models using the multivariate reduction technique have been used in various insurance contexts. These include mortality modelling (Alai et al., 2013, 2016), capital modelling (Furman and Landsman, 2010), claim counts modelling (Meyers, 2007; Shi and Valdez, 2014). A strong advantage of models constructed using this technique is the ability to provide a disciplined construction of correlation matrices using a significantly smaller number of parameters (Avanzi, Taylor and Wong, 2018). This is particularly beneficial when one works on a portfolio of a large number of sub-portfolios, which can be up to 100 in some real life cases.

Vu (2013) introduced a multivariate gamma loss reserving model using the multivariate reduction technique (see also Section 2.4.3.2). The copula models in De Jong (2012) and Shi (2014) can also considered to be of the "common shock" type, however, as shown in Equation (2.80), the additive common shock structure applies on the cumulative transformation of variables. The effects of common shock on dependence, however, are studied extensively and illustrated numerically in both papers.

A general common shock loss reserving framework is introduced in Avanzi, Taylor and Wong (2018). In this framework, incremental claim cells $Y_{i,j}^{(n)}$ are specified such that

$$Y_{i,j}^{(n)} = {}_{U}\lambda_{i,j}^{(n)} \cdot U_{\boldsymbol{\pi}_{(i,j)}} + {}_{\tilde{U}}\lambda_{i,j}^{(n)} \cdot \tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}} + {}_{Z}\lambda_{i,j}^{(n)} \cdot Z_{i,j}^{(n)} \tag{2.112}$$

where $U_{\boldsymbol{\pi}_{(i,j)}}$, $\tilde{U}_{\boldsymbol{\pi}^{(n)}_{(i,j)}}$, $Z^{(n)}_{i,j}$ are independent random variables, representing common shock for set $\boldsymbol{\pi}_{(i,j)}$ of claims from all business segments, common shock for set $\boldsymbol{\pi}^{(n)}_{(i,j)}$ of claims from business segment (n) and idiosyncratic component, respectively. Sets $\boldsymbol{\pi}_{(i,j)}$ and $\boldsymbol{\pi}^{(n)}_{(i,j)}$ are subsets of claims that common shocks have effects on. For example, $\boldsymbol{\pi}_{(i,j)} = \{Y^{(1)}_{i,1}, ..., Y^{(N)}_{i,I-i+1}\}$ gives a set of claims from accident period i from all business segments. Hence the model applied to this set captures accident period dependence through the use of the common shock term $U_{\boldsymbol{\pi}_{(i,j)}}$. Scaling factors, denoted by ${}_U\lambda^{(n)}_{i,j}$, ${}_{\tilde{U}}\lambda^{(n)}_{i,j}$, ${}_Z\lambda^{(n)}_{i,j}$, scale the effects of common shocks and idiosyncratic components so that they contribute proportionately to observed claim values. We can set ${}_Z\lambda^{(n)}_{i,j} = 1$ without loss of generality.

In most (if not all) reserving data sets, there is a large variation in claim activity for different lengths of delay. In particular, it often reaches a peak in some early years, then dies out as the delay increases. It is mentioned in Avanzi, Taylor and Wong (2018) that the construction of common shock models should be mindful of this feature. In particular, it is desirable to ensure that the common shocks contribute proportionately to the claims observed in different positions within a loss triangle.

2.5 Evolutionary reserving models

The focus of this section is on evolutionary reserving models. These are models that allow parameters to evolve over time, hence naturally incorporate changes in circumstances. Section 2.5.1 explains the motivation for evolutionary modelling, covering various benefits of these models. Two main types of models are then reviewed: Gaussian models and non-Gaussian models. Some of the theory of Gaussian models and their well-known estimation technique, Kalman filtering, is provided in Section 2.5.2. Gaussian loss reserving models are then reviewed in Section 2.5.3. Some of the theory of non-Gaussian models and particle filtering techniques used for estimation is provided in Section 2.5.4. Section 2.5.5 then reviews non-Gaussian reserving models.

To the best of our knowledge, all of the existing work on evolutionary modelling in the literature is for one single segment, except the model developed by Shi et al. (2012) in which multiple segments are considered. Hence, for ease of notation, the segment index (n) is dropped in this section. For the only case when multiple business segments are considered, the index is added back.

2.5.1 Motivation for evolutionary reserving

Insurers frequently encounter changes in their claims development patterns over time (Ghezzi, 2001; Renshaw, 1989; Gluck and Venter, 2009). This can be due to various reasons, such as legislative changes, or changes to the internal operations of insurers. For example, insurers can improve the administering of claims over time to enhance efficiency, hence gradually shorten the administrative delays. A legislative change such as the recent reform for Auto Bodily Injury covers in New South Wales (Australia) results in faster claims resolution (State Insurance Regulatory Authority, 2018), hence reduces payment delays. For another example, we provide in Figure 2.2 plots of loss ratios for ten accident years from 1988 to 1997 from the Commercial Auto line of an American insurer. Loss ratios are calculated as incremental claims standardised using total premium earned in the corresponding accident year. The data is provided in the Schedule P and used for illustration in Shi and Frees (2011). Changes in claim activity across accident periods are quite evident in this figure with variation in the development patterns across accident years. The top two values in each accident period are highlighted and presented in Figure 2.3 to identify the peaks in the development patterns. It can be observed that the peak in the claims development shifts between development years 1, 2 and 3 over time.

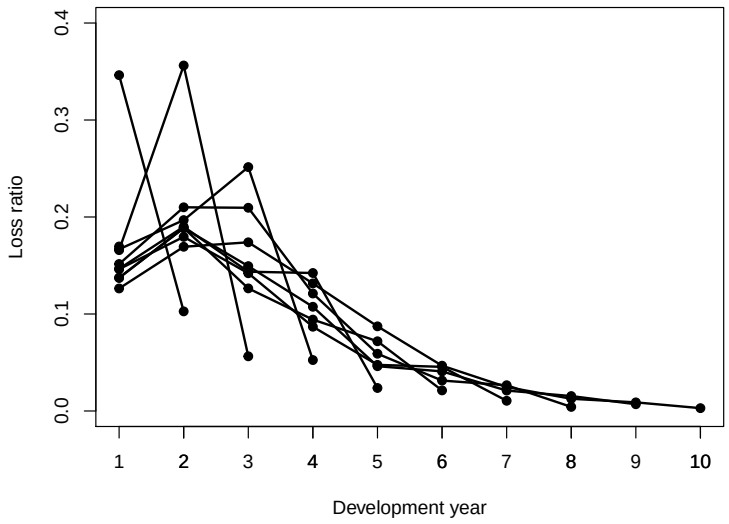

Figure 2.2: Loss ratios in the Commercial Auto line of an American insurer (Schedule P)

Year	1	2	3	4	5	6	7	8	9	10
1988	0.126	0.169	0.174	0.132	0.087	0.047	0.025	0.013	0.009	0.003
1989	0.137	0.189	0.149	0.107	0.046	0.041	0.021	0.015	0.007	
1990	0.152	0.210	0.210	0.121	0.059	0.031	0.027	0.004		
1991	0.146	0.180	0.142	0.087	0.048	0.045	0.011			
1992	0.137	0.189	0.126	0.094	0.072	0.021				
1993	0.147	0.190	0.144	0.142	0.024					
1994	0.167	0.197	0.251	0.053						
1995	0.166	0.356	0.056							

Figure 2.3: Loss triangle with top two values in each accident period highlighted (first eight accident periods)

Changes in claims development patterns can have various impacts on the analysis and prediction of outstanding claims (see for example, Fleming and Mayer, 1988; Ghezzi, 2001; Zehnwirth, 1994; Taylor and McGuire, 2009; Gluck and Venter, 2009; Alpuim and Ribeiro, 2003; Dong and Chan, 2013). As mentioned in Section 2.1, outstanding claims modelling typically involves using past information in upper triangles to predict outstanding claims in lower triangles. When changes in development patterns have taken place, it is no longer straightforward to project future trends (Zehnwirth, 1994; Ghezzi, 2001; Sims, 2012). Reserving methods that are based on the assumption of similar claim activities across accident periods, such as the traditional chain ladder algorithm, are no longer appropriate. In some cases, actuaries have to make a lot of judgements to remove or reduce the effects of these changes on traditional reserving methods. An example of this procedure is the Berquist-Sherman technique described in Section 2.2.3. These judgements are often difficult and time consuming to make, as well as to justify to management and peer reviewers (Sims, 2012). In other cases, when the model selected using earlier data no longer fits more recent data, there may be a need to revise its algebraic structure. This will result in a fundamental discontinuity in the sequence of estimates such as the central estimates of outstanding claims (Taylor et al., 2003).

A solution for this issue is to accommodate changes directly in the model structures by allowing parameters to evolve in a recursive manner (De Jong and Zehnwirth, 1983; Zehnwirth, 1994; Taylor et al., 2003). These models are known as evolutionary models. Other names for these models also include state space models (Alpuim and Ribeiro, 2003; Chukhrova and Johannssen, 2017), adaptive models (Taylor and McGuire, 2009), and robotic models (McGuire, 2007). With this type of model structure, changes are incorporated more

naturally as parameters are updated over time given the arrival of new data. More weight is also given to recent data, allowing these models to provide better projections of future claims (Zehnwirth, 1994; Taylor, 2000; Alpuim and Ribeiro, 2003). By having changes recognised gradually, a clear picture of changes in the historical experience can be obtained, and estimates derived from these models are also smooth over time (Taylor et al., 2003; Sims, 2012).

Evolutionary models are also considered an elegant and plausible solution by Zehnwirth (1994) and Gluck and Venter (2009). They not only can provide a better fit to the data, but also are more parsimonious. Instead of estimating parameters separately using scarce information for immature accident periods, these parameters can be projected recursively using the previous ones (Gluck and Venter, 2009). This also reduces the reliance on arbitrary modelling judgements. Trends in the data can be captured using a simplistic but explicit evolutionary structure, enhancing model interpretation and reducing the need for unrecognised parameters (Zehnwirth, 1994).

There has been an increasing recognition for the need for robotic reserving in the insurance industry (McGuire, 2007; Taylor and McGuire, 2008; Sims, 2014). This need typically arises as more insurers wish to assess their outstanding claims liabilities on a more frequent basis such as quarterly or monthly. These insurers often have large portfolios of many segments which makes the task very time consuming. It is also observed that many segments do not experience dramatic changes from one period to another, making the valuation job rather repetitive (McGuire, 2007). Robotic reserving, or automated reserving, is then considered a plausible solution (Taylor and McGuire, 2008). They can perform repetitive valuation jobs for many business segments while actuaries can have more time to spend on segments that require more substantial judgements. Evolutionary models incorporate changes naturally in the model structures. Many evolutionary models are developed with filtering processes, a real-time device that updates existing estimates without the need to redo all calculations or keep track of all previous information (De Jong and Zehnwirth, 1983). Hence they can be used in the construction of reserving robots.

Evolutionary models have appeared in the loss reserving literature since as early as the 1980s. Evolutionary models are also a very common type of models used in time series analysis with a wide range of applications in engineering, physics, economics, and many more. They are often called hidden Markov models (Doucet and Johansen, 2011), or state space models (Durbin and Koopman, 2012) in these areas. These models typically assume that the

development over time of the system under study is determined by an unobserved series of dynamic factors (Durbin and Koopman, 2012, Chapter 1). The latent/unobserved factors are called states, and the relationships between system observations and these states are specified in these models.

State space models can be categorised into two groups based on the underlying distributional assumption used: Gaussian models and non-Gaussian models. In these models, the estimation of underlying factors is also important as the techniques used are not as straightforward as those used for static models with deterministic factors/parameters. We also review the estimation techniques for state space models in the subsequent sections.

2.5.2 Theory of Gaussian models and Kalman filtering

The focus of this section is on linear Gaussian state space models. The observation equation which specifies the relationship between a vector of observations $\boldsymbol{Y}_t$ and a vector of underlying states/random factors $\boldsymbol{\gamma}_t$ is

$$\boldsymbol{Y}_t = \boldsymbol{A}_t\boldsymbol{\gamma}_t + \boldsymbol{\varsigma}_t, \qquad \boldsymbol{\varsigma}_t \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{H}_t), \tag{2.113}$$

where $\boldsymbol{A}_t$ is a specified matrix and $\boldsymbol{H}_t$ is the covariance matrix of the disturbance term $\boldsymbol{\varsigma}_t$. The underlying states vector $\boldsymbol{\gamma}_t$ evolves over time according to a state equation

$$\boldsymbol{\gamma}_{t+1} = \boldsymbol{R}_t\boldsymbol{\gamma}_t + \boldsymbol{S}_t \cdot {}_{\boldsymbol{\gamma}}\boldsymbol{\epsilon}_{t+1}, \qquad {}_{\boldsymbol{\gamma}}\boldsymbol{\epsilon}_{t+1} \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{Q}_t), \tag{2.114}$$

where $\boldsymbol{R}_t$ and $\boldsymbol{S}_t$ are specified matrices, $\boldsymbol{Q}_t$ is the covariance matrix of the disturbance term ${}_{\boldsymbol{\gamma}}\boldsymbol{\epsilon}_{t+1}$. We also have the disturbances $\boldsymbol{\varsigma}_t$ and ${}_{\boldsymbol{\gamma}}\boldsymbol{\epsilon}_t$ being independent. The above structure is called a linear Gaussian state space structure due to the linearity and Gaussian assumptions specified in both equations.

In the above specification, the underlying states $\boldsymbol{\gamma}_t$ are to be estimated, as well as the covariance matrices $\boldsymbol{H}_t$ and $\boldsymbol{Q}_t$. The matrices $\boldsymbol{A}_t$, $\boldsymbol{R}_t$, $\boldsymbol{S}_t$ can either be specified, or estimated depending on the model specifications. The underlying states $\boldsymbol{\gamma}_t$ can be estimated recursively using observations obtained at each time step. This on-line estimation procedure is referred to as filtering. Underlying parameters, however, are estimated in a separate procedure. The estimation of states and parameters are reviewed in Section 2.5.2.1 and 2.5.2.2, respectively.

2.5.2.1 Kalman filtering and back smoothing for state estimation

Filtering is a real-time device that updates estimates of states recursively upon the arrival of new observations without the need to redo all calculations or keep track of all previous information. For linear Gaussian state space models, an optimal filtering algorithm can be obtained in closed-form, which is known as the Kalman filter. The Kalman filter estimates the states with minimised errors using the available information of the observation process as well as prior knowledge about the system (Durbin and Koopman, 2012). The tractability of the Kalman filter has motivated the popularity of Gaussian state space models in various fields.

The Kalman filtering procedure is summarised in Figure 2.4. In a Kalman filter, there are two sets of equations: measurement update equations and time update equations (Welch and Bishop, 1995; Durbin and Koopman, 2012). The measurement update equations are responsible for updating the current state using current observations, also known as the filtering step. The output of this step is the filtered distribution $\boldsymbol{\gamma}_t|\boldsymbol{Y}_t$. However, because of Gaussian specifications for $\boldsymbol{Y}_t$ and $\boldsymbol{\gamma}_{t+1}$ in Equation (2.113) and Equation (2.114), the distribution of $\boldsymbol{\gamma}_t|\boldsymbol{Y}_t$ is also Gaussian. It is then sufficient to estimate the mean and the covariance of this distribution

$$\widehat{\boldsymbol{\gamma}}_{t|t} = E[\boldsymbol{\gamma}_t|\boldsymbol{Y}_t], \tag{2.115}$$

$$\widehat{\boldsymbol{P}}_{t|t} = Var[\boldsymbol{\gamma}_t|\boldsymbol{Y}_t]. \tag{2.116}$$

The time update equations aim to estimate the distribution of states in the next period using current information $\boldsymbol{\gamma}_{t+1}|\boldsymbol{Y}_t$, also known as the one-step prediction step. As this distribution is also Gaussian, it is sufficient to estimate its mean and variance

$$\widehat{\boldsymbol{\gamma}}_{t+1|t} = E[\boldsymbol{\gamma}_{t+1}|\boldsymbol{Y}_t], \tag{2.117}$$

$$\widehat{\boldsymbol{P}}_{t+1|t} = Var[\boldsymbol{\gamma}_{t+1}|\boldsymbol{Y}_t]. \tag{2.118}$$

In the Kalman filter, the means and variances of states in the filtering step as well as the one-step ahead prediction step are obtained recursively. The whole history is not required to be kept and reprocessed every time new observations arrive.

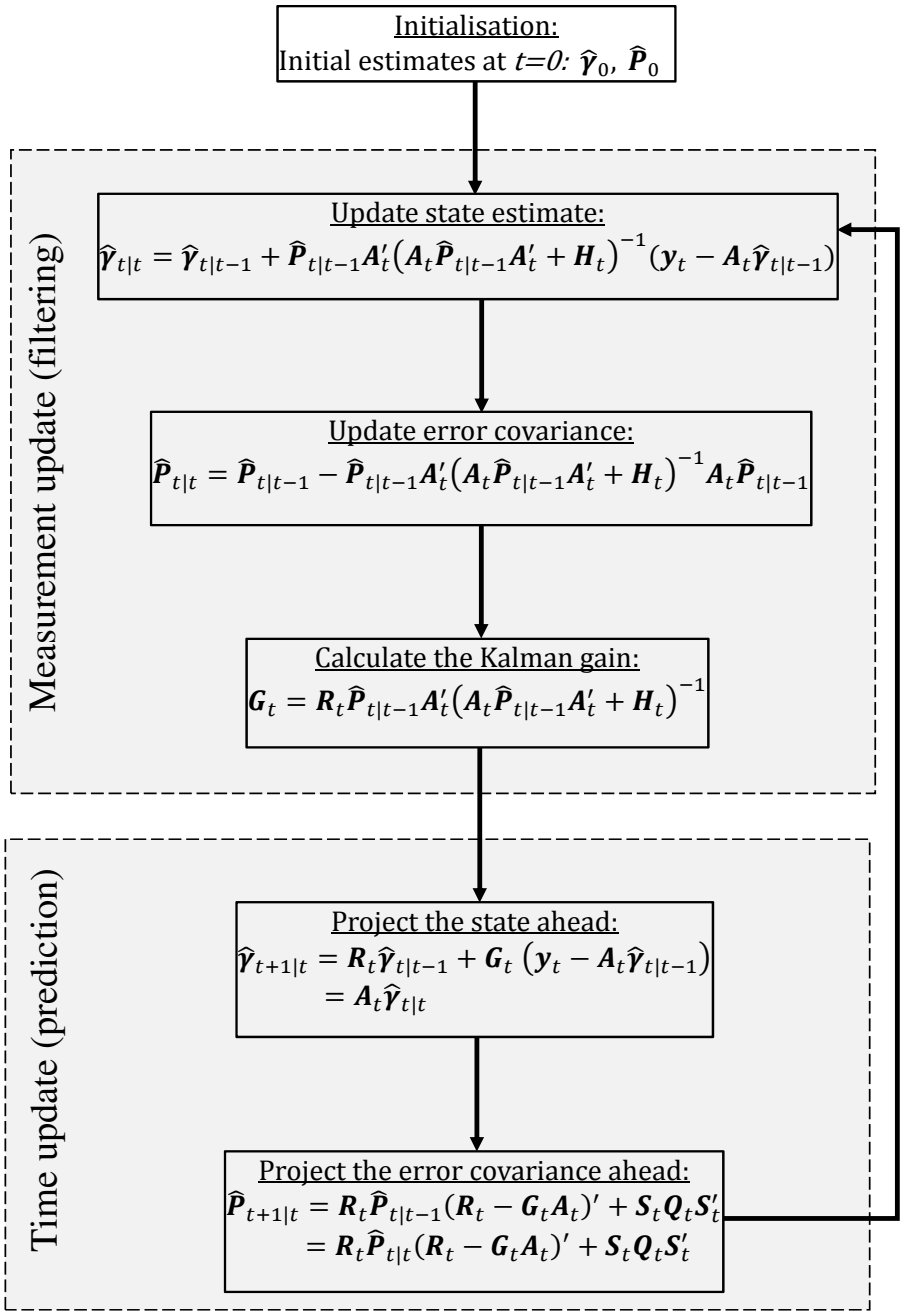

Figure 2.4: A diagram of the Kalman filter

The Kalman filter, however, is suboptimal for estimating the sequence of states. This is because it only uses new information at each time period to provide filtered estimates of states instead of the available trajectory of observations in later periods. The Kalman smoother is used to overcome this limitation. The Kalman smoother provides the backward recursive estimation of underlying states given all available observations. It is typically run after the Kalman filter is completed and all filtered estimates of states have been obtained.

Estimates of underlying states from the Kalman smoother are usually called smoothed states and are given as

$$\widehat{\boldsymbol{\gamma}}_{t|T} = E[\boldsymbol{\gamma}_t|\boldsymbol{Y}_T], \tag{2.119}$$

$$= \widehat{\boldsymbol{\gamma}}_{t|t} + \widehat{\boldsymbol{P}}_{t|t}\boldsymbol{R}'_{t+1}\widehat{\boldsymbol{P}}^{-1}_{t+1|t}\left(\widehat{\boldsymbol{\gamma}}_{t+1|T} - \widehat{\boldsymbol{\gamma}}_{t+1|t}\right), \tag{2.120}$$

where $\widehat{\boldsymbol{\gamma}}_{t|t}$, $\widehat{\boldsymbol{\gamma}}_{t+1|t}$, $\widehat{\boldsymbol{P}}_{t|t}$, $\widehat{\boldsymbol{P}}_{t+1|t}$ are estimates from the Kalman filter. The smoother initialises at $t = T$ with the estimate of $\widehat{\boldsymbol{\gamma}}_{T|T}$ from the Kalman filter.

2.5.2.2 Parameter estimation

The Kalman filter described in Section 2.5.2.1 is applied to estimate the underlying states $\boldsymbol{\gamma}_t$ conditional on known parameters $\boldsymbol{H}_t$, $\boldsymbol{Q}_t$, and possibly, $\boldsymbol{A}_t$, $\boldsymbol{R}_t$, $\boldsymbol{S}_t$. A very common method to estimate these parameters used in conjunction with the Kalman filter is maximum likelihood estimation. A comprehensive review of this estimation technique is provided in Durbin and Koopman (2012, Chapter 7).

With the use of linear Gaussian distributional assumptions for states $\boldsymbol{\gamma}_t$ and observations $\boldsymbol{Y}_t$, the likelihood function can be obtained in closed-form with the unknown parameters (denoted by $\boldsymbol{\Theta}$). In particular,

$$\log f_{\boldsymbol{Y}_{1:T}}(\boldsymbol{Y}_{1:T};\boldsymbol{\Theta}) = \sum_{t=1}^{T} \log f_{\boldsymbol{Y}_t|\boldsymbol{Y}_{t-1}}\left(\boldsymbol{y}_t|\boldsymbol{Y}_{t-1};\boldsymbol{\Theta}\right) \tag{2.121}$$

$$= -\frac{T\tilde{N}}{2}\log(2\pi)$$

$$-\frac{1}{2}\sum_{t=1}^{T}\left(\log|\boldsymbol{A}_t\widehat{\boldsymbol{P}}_{t|t-1}\boldsymbol{A}'_t + \boldsymbol{H}_t| + \left(\boldsymbol{y}_t - \boldsymbol{A}_t\widehat{\boldsymbol{\gamma}}_{t|t-1}\right)'\left(\boldsymbol{A}_t\widehat{\boldsymbol{P}}_{t|t-1}\boldsymbol{A}'_t + \boldsymbol{H}_t\right)^{-1}\left(\boldsymbol{y}_t - \boldsymbol{A}_t\widehat{\boldsymbol{\gamma}}_{t|t-1}\right)\right), \tag{2.122}$$

where $\tilde{N}$ is the number of observations in each vector $\boldsymbol{Y}_t$. The above log-likelihood function comes from

$$\boldsymbol{Y}_t|\boldsymbol{Y}_{t-1} \sim \text{Normal}\left(\boldsymbol{A}_t\widehat{\boldsymbol{\gamma}}_{t|t-1}, \boldsymbol{A}_t\widehat{\boldsymbol{P}}_{t|t-1}\boldsymbol{A}'_t + \boldsymbol{H}_t\right). \tag{2.123}$$

The log likelihood function can then be maximised to obtain estimates of unknown parameters.

2.5.3 Univariate Gaussian models

The majority of evolutionary reserving models in the literature are Gaussian models, which can be applied to data on the log-scale. This ultimately implies a log-normal assumption on claims on the original scale. A good review of Gaussian models, as well as non-Gaussian models in reserving can be found in Chukhrova and Johannssen (2017).

The very first model introduced to the literature is De Jong and Zehnwirth (1983). A typical Gaussian state space model structure is used in this model. The Hoerl curve (see also Section 2.3.3) is used for observations specification

$$\log(Y_{i,j}) = a_i + \log(j) + j + \tilde{\varsigma}_{i,j}, \qquad \tilde{\varsigma}_{i,j} \sim \text{Normal}(0, \sigma^2_{\tilde{\varsigma}}), \tag{2.124}$$

and the state equation is specified as

$$a_i = a_{i-1} + {}_a\epsilon_i, \qquad {}_a\epsilon_i \sim \text{Normal}(0, \sigma^2_{a\epsilon}). \tag{2.125}$$

The underlying state that evolves in this model is the accident period factor a_i. The above structure can be more complex with more polynomial terms added. The Kalman filter is applied for state estimation, and maximum likelihood estimation is used to estimate parameters. The filter is applied on the row-wise flow of data. This means that states are estimated recursively for each row using estimates from the previous row.

Verrall (1989) introduced a Gaussian model with the chain-ladder mean structure

$$\log(Y_{i,j}) = a_i + b_j + \tilde{\varsigma}_{i,j}, \qquad \tilde{\varsigma}_{i,j} \sim \text{Normal}(0, \sigma^2_{\tilde{\varsigma}}), \tag{2.126}$$

where the underlying states were specified such that

$$a_i = a_{i-1} + {}_a\epsilon_i, \qquad {}_a\epsilon_i \sim \text{Normal}(0, \sigma^2_{a\epsilon}), \tag{2.127}$$

$$b_j = b_{j-1} + {}_b\epsilon_j, \qquad {}_a\epsilon_j \sim \text{Normal}(0, \sigma^2_{a\epsilon}). \tag{2.128}$$

In this model the Kalman filter is also used to estimate the underlying states a_i and b_j. The filter is applied with a diagonal-wise flow of data. This means that factors in each diagonal (calendar period) are estimated recursively using estimates from the previous diagonal (calendar period). A similar model structure is used in Verrall (1994) where the

only underlying random factor is b_j. The Kalman filter is applied with the row-wise flow of data in this model.

Ntzoufras and Dellaportas (2002) also developed an evolutionary model which is similar to the model in Verrall (1989). They used Bayesian inference to estimate the underlying states a_i and b_j as well as the unknown parameters. This is a hierarchical Bayesian framework where underlying states are treated as unknown parameters, the distributions of which are driven by unknown parameters on the upper level in the hierarchy. This approach is known as the off-line estimation of states where they are not recursively updated upon the arrival of new data but are estimated all at once using all observed information.

Another Gaussian model is developed in Atherino et al. (2010). In this model, claim cells are stacked together by rows to form a univariate time series with missing values representing claims in the lower triangle. Each claim cell is assumed to have the following structure

$$\log(Y_t) = \tilde{a}_t + \tilde{b}_t + \tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{\gamma}} + \tilde{\varsigma}_t, \qquad \tilde{\varsigma}_t \sim \text{Normal}(0, \sigma^2_{\tilde{\varsigma}}), \tag{2.129}$$

where t is position of the observation in the series, $\tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{\gamma}}$ is the linear predictor with fixed effects, and $\tilde{a}_t$, $\tilde{b}_t$ are random factors specified such that

$$\tilde{a}_t = \tilde{a}_{t-1} + {}_{\tilde{a}}\epsilon_t, \qquad {}_{\tilde{a}}\epsilon_t \sim \text{Normal}(0, \sigma^2_{{}_{\tilde{a}}\epsilon}), \tag{2.130}$$

$$\tilde{b}_t = -\sum_{j=1}^{J-1} \tilde{b}_{t+1-j} + {}_{\tilde{b}}\epsilon_t, \qquad {}_{\tilde{b}}\epsilon_t \sim \text{Normal}(0, \sigma^2_{{}_{\tilde{b}}\epsilon}). \tag{2.131}$$

In this structure, the claims development pattern is captured using seasonal effects via the term $\tilde{b}_t$. Two methods are considered to calculate the mean square error of the total outstanding claims: block method and cumulating method. The block method uses the Kalman filtering algorithm to obtain the covariance matrix for all observations (including missing values), which is then used to calculate the mean square error of total outstanding claims. The cumulating method treats cumulated unobserved claims as separate states in the model, then obtains the estimates and the mean square errors of these states using the Kalman filter. However, when the cumulating method is used, the log-transformation cannot be applied hence the Gaussian assumption applies directly on claims on the original scale $Y_{i,j}$.

De Jong (2006) introduced a number of evolutionary models with parameters evolution in various dimensions, including by development period, accident period and calendar period. These models are applied on log-link ratios

$$D_{i,j} = \log\left(\frac{X_{i,j}}{X_{i,j-1}}\right). \tag{2.132}$$

A model with development period evolution is represented such that

$$D_{i,j} = \ddot{b}_j + \tilde{b}_j({}_D\epsilon_{i,j} + {}_{\tilde{b}}\tau_j \cdot {}_D\epsilon_{i,j-1}), \qquad {}_D\epsilon_{i,j} \sim \text{Normal}(0, \sigma^2_{D\epsilon}), \tag{2.133}$$

where $\ddot{b}_j$ is the mean component, and the noise component is composed of two parts: the current noise term ${}_D\epsilon_{i,j}$ and the previous noise term ${}_D\epsilon_{i,j-1}$ carried over with an adjustment coefficient ${}_{\tilde{b}}\tau_j$. The whole noise component is adjusted by a coefficient $\tilde{b}_j$.

With a similar time series approach, a model with the evolution in the accident period direction is introduced using

$$D_{i,j} = \ddot{d}_{i,j} + \tilde{b}_j \cdot {}_D\epsilon_{i,j}, \qquad {}_D\epsilon_{i,j} \sim \text{Normal}(0, \sigma^2_{D\epsilon}), \tag{2.134}$$

$$\ddot{d}_{i+1,j} = \ddot{d}_{i,j} + {}_{\ddot{d}}\tau_j \cdot {}_{\ddot{d}}\epsilon_{i+1,j}, \qquad {}_{\ddot{d}}\epsilon_{i+1,j} \sim \text{Normal}(0, \sigma^2_{D\epsilon}), \tag{2.135}$$

where the evolutionary factor is $\ddot{d}_{i+1,j}$ which evolves from one accident period to another. It is also the mean of the log-link ratio $D_{i,j}$. Two coefficient terms for the disturbances in the observation $D_{i,j}$ and the state $\ddot{d}_{i,j}$ specifications are $\tilde{b}_j$ and ${}_{\ddot{d}}\tau_j$, respectively.

The evolution in the calendar period dimension can also be captured in a similar manner with

$$D_{i,j} = \ddot{b}_j + \tilde{b}_j(\ddot{o}_{t=i+j-1} + {}_D\epsilon_{i,j}), \qquad {}_D\epsilon_{i,j} \sim \text{Normal}(0, \sigma^2_{D\epsilon}), \tag{2.136}$$

$$\ddot{o}_{t+1} = \ddot{o}_t + {}_{\ddot{o}}\tau \cdot {}_{\ddot{o}}\epsilon_{t+1}, \qquad {}_{\ddot{o}}\epsilon_{t+1} \sim \text{Normal}(0, \sigma^2_{o\epsilon}), \tag{2.137}$$

where the evolving state is the calendar period factor $\ddot{o}_t$. The column specific term in the mean structure of $D_{i,j}$ is $\ddot{b}_j$. The noise component is denoted by ${}_D\epsilon_{i,j}$ which is adjusted by a coefficient $\tilde{b}_j$. The observation equation also involves a calendar period term $\ddot{o}_t$ which evolves from one calendar period to another in a random walk. The noise in this random walk is adjusted by the coefficient ${}_{\ddot{o}}\tau$. The Kalman filter is used for state estimation in these models.

Shi et al. (2012) also considered evolutionary calendar factors in their reserving model for multiple business segments. This model is described in detail in Section 2.4.3.1. In this model, claims follow multivariate log-normal distributions with a modified chain ladder mean structure with an additional term for calendar period effects. These calendar period effects evolve over time in a random walk

$$h_t = h_{t-1} + {}_h\epsilon_t, \qquad {}_h\epsilon_t \sim \text{Normal}(0, \sigma^2_{{}_h\epsilon}), \tag{2.92}$$

or an AR(1) process

$$h_t = {}_h\tau \cdot h_{t-1} + {}_h\epsilon_t, \qquad {}_h\epsilon_t \sim \text{Normal}(0, \sigma^2_{{}_h\epsilon}). \tag{2.93}$$

Calendar period factors are common shocks and they deduce the calendar period dependence within and across segments. Hierarchical Bayesian estimation is used where these calendar period factors are treated as unknown parameters. Similar to Ntzoufras and Dellaportas (2002), this estimation is also off-line.

2.5.4 Theory of non-Gaussian models and particle filtering

The Kalman filter provides an optimal tractable recursive estimation of states in linear Gaussian models. The tractability of the Kalman filter is particularly attractive and this explains the popularity of Gaussian models not only in the reserving field but also many other fields such as engineering, physics and medicine. Linear Gaussian models typically rely on the linearity and Gaussian assumptions of states and observations, as shown in Equation (2.113) and Equation (2.114). Relaxing these assumptions result in non-linear and/or non-Gaussian state space models.

A general state space model can be introduced with

$$\boldsymbol{Y}_t \sim f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{Y}_t|\boldsymbol{\gamma}_t), \tag{2.138}$$

$$\boldsymbol{\gamma}_t \sim f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}), \tag{2.139}$$

where $f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{Y}_t|\boldsymbol{\gamma}_t)$ and $f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1})$ are specified densities. Either one or both of these distributions can be non-Gaussian.

A subclass of non-Gaussian models is EDF state space models (Durbin and Koopman, 2012). In a typical EDF state space model, observations $\boldsymbol{Y}_t$ follow a distribution from the EDF

$$f_{\boldsymbol{Y}_t}(\boldsymbol{y}_t|\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) = v\left(\boldsymbol{y}_t, \boldsymbol{\phi}_t\right) \exp\left\{\frac{\boldsymbol{y}_t\boldsymbol{\theta}_t - \kappa(\boldsymbol{\theta}_t)}{\boldsymbol{\phi}_t}\right\}, \tag{2.140}$$

where the canonical parameter $\boldsymbol{\theta}_t$ is called the signal in state space models and specified such that

$$\boldsymbol{\theta}_t = \boldsymbol{A}_t\boldsymbol{\gamma}_t, \tag{2.141}$$

with $\boldsymbol{\gamma}_t$ being the evolving states. When the state specification is linear with the Gaussian assumption, i.e.

$$\boldsymbol{\gamma}_{t+1} = \boldsymbol{R}_t\boldsymbol{\gamma}_t + \boldsymbol{S}_t \cdot {}_{\boldsymbol{\gamma}}\boldsymbol{\epsilon}_{t+1}, \qquad {}_{\boldsymbol{\gamma}}\boldsymbol{\epsilon}_{t+1} \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{Q}_t) \tag{2.142}$$

we have a model with linear Gaussian signals (Durbin and Koopman, 2012).

Non-Gaussian models provide much more flexibility in terms of model structure and distributional assumptions used. These models, however, lose their estimation tractability as the Kalman filter is no longer an optimal filtering algorithm. Section 2.5.4.1 reviews particle filtering, a very popular simulation-based filtering method for non-Gaussian models. Parameter estimation techniques are reviewed in Section 2.5.4.2.

2.5.4.1 Particle filtering

Various filtering methods have been considered for non-linear and/or non-Gaussian models since the optimal filters for these model are no longer tractable. A number of filters have been developed which approximate the famous Kalman filter for Gaussian models. The approximation can be done in various ways, including linearising the state and observation equations (also known as the extended Kalman filter), applying a transformation to observations and states to bring them closer to Gaussian systems, performing the mode approximation to obtain approximating linear Gaussian models (also known as the mode estimation method). A comprehensive review of these methods can be found in Durbin and Koopman (2012, Chapter 10).

Particle filters are procedures introduced to solve filtering problems in non-linear and/or non-Gaussian state space models using Monte Carlo algorithms, to be precise, sequential Monte Carlo algorithms. Ever since they were introduced in the 1990s, they have become very popular estimation tools for these models. Particle filtering has seen a quite extensive area of research and is still receiving attention. Excellent reviews can be found in Doucet et al. (2001), Cappé et al. (2007), Fearnhead (2008), Doucet and Johansen (2011), and Creal (2012), just to name a few. The purpose of particle filters is the same as that of the Kalman filter: to sequentially update the filtered distribution of states $\boldsymbol{\gamma}_t|\boldsymbol{Y}_t$ and the predictive distribution of states $\boldsymbol{\gamma}_{t+1}|\boldsymbol{Y}_t$ upon the arrival of new observations. Because these distributions are no longer linear Gaussian, sequential Monte Carlo algorithms are used to approximate them. The summary of particle filtering provided in this section follows Cappé et al. (2007) and Creal (2012).

Recall that we want to approximate, or estimate important statistics of the filtering distribution $f_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}$. Consider any function $\tilde{g}(.)$ of states $\boldsymbol{\gamma}_{0:t}$, which can be a mean function or a variance function. We have

$$E\left[\tilde{g}(\boldsymbol{\gamma}_{0:t})\right] = \int \tilde{g}(\boldsymbol{\gamma}_{0:t}) f_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t}) d\boldsymbol{\gamma}_{0:t}, \tag{2.143}$$

$$\approx \frac{1}{M}\sum_{m=1}^{M} \tilde{g}\left(\boldsymbol{\gamma}_{0:t}^{(m)}\right), \tag{2.144}$$

where M is the number of samples drawn from the distribution $f_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}$. In brief, we can approximate the expected value using a sample average with $\boldsymbol{\gamma}_{0:t}^{(m)}$ being drawn from its distribution. However, this distribution can be very difficult, or in many cases, impossible to draw sample from. We look for an alternative distribution $\tilde{f}_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t})$ which is close to $f_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}$ in some sense and easier to simulate from. The expected value can be rewritten as

$$E\left[\tilde{g}(\boldsymbol{\gamma}_{0:t})\right] = \int \tilde{g}(\boldsymbol{\gamma}_{0:t}) \frac{f_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t})}{\tilde{f}_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t})} \tilde{f}_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t}) d\boldsymbol{\gamma}_{0:t}, \tag{2.145}$$

$$= \int \tilde{g}(\boldsymbol{\gamma}_{0:t}) \omega_t \tilde{f}_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t}) d\boldsymbol{\gamma}_{0:t}, \tag{2.146}$$

$$\approx \sum_{m=1}^{M} \frac{\omega_t^{(m)}}{\sum_{m=1}^{N} \omega_t^{(m)}} \tilde{g}\left(\boldsymbol{\gamma}_{0:t}^{(m)}\right), \tag{2.147}$$

where ω_t is called the importance weight and is defined such that

$$\omega_t = \frac{f_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t})}{\tilde{f}_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t})}. \tag{2.148}$$

Since samples are drawn from the approximating distribution $\tilde{f}_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t})$, a correction needs to be made so that the obtained estimate is an unbiased estimator of $E\left[\tilde{g}(\boldsymbol{\gamma}_t)\right]$. This correction involves re-weighting the samples with weights ω_t, resulting in a weighted average of samples as shown in Equation (2.147).

Since new observations arrives at each time point, it can be desirable to avoid recalculating the weights for the whole state vector $\boldsymbol{\gamma}_{0:t}$ at each t. This can be done with a recursive decomposition of the filtering distribution

$$\tilde{f}_{\boldsymbol{\gamma}_{0:t}|\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_{0:t}|\boldsymbol{y}_{1:t}) = \tilde{f}_{\boldsymbol{\gamma}_{0:t-1}|\boldsymbol{Y}_{1:t-1}}(\boldsymbol{\gamma}_{0:t-1}|\boldsymbol{y}_{1:t-1})\tilde{f}_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{0:t-1},\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{0:t-1},\boldsymbol{y}_{1:t}). \tag{2.149}$$

The importance weights can then be calculated recursively

$$\omega_t = \frac{f_{\boldsymbol{\gamma}_{0:t-1}|\boldsymbol{Y}_{1:t-1}}(\boldsymbol{\gamma}_{0:t-1}|\boldsymbol{y}_{1:t-1})}{\tilde{f}_{\boldsymbol{\gamma}_{0:t-1}|\boldsymbol{Y}_{1:t-1}}(\boldsymbol{\gamma}_{0:t-1}|\boldsymbol{y}_{1:t-1})} \times \frac{f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t|\boldsymbol{\gamma}_t)f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1})}{\tilde{f}_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1},\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1},\boldsymbol{y}_{1:t})}, \tag{2.150}$$

$$= \omega_{t-1}\frac{f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t|\boldsymbol{\gamma}_t)f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1})}{\tilde{f}_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1},\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1},\boldsymbol{y}_{1:t})}, \tag{2.151}$$

where $f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t|\boldsymbol{\gamma}_t)$ and $f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1})$ are specified in Equation (2.138) and Equation (2.139) of the state space model.

Particle filtering applies the concept of the sequential Monte Carlo described above. Samples drawn from the proposal distributions are often called particles. A particle filtering algorithm can be formulated as follows based on the review in Doucet and Johansen (2011).

Sequential Monte Carlo particle filter

Step 1. Initialisation: At $t = 0$, for $m = 1, ..., M$, draw

$$\boldsymbol{\gamma}_0^{(m)} \sim \tilde{f}_{\boldsymbol{\gamma}_0}(\boldsymbol{\gamma}_0), \tag{2.152}$$

and calculate the importance weight

$$\omega_0^{(m)} = \frac{f_{\boldsymbol{\gamma}_0}(\boldsymbol{\gamma}_0^{(m)})}{\tilde{f}_{\boldsymbol{\gamma}_0}(\boldsymbol{\gamma}_0^{(m)})}. \tag{2.153}$$

Step 2. Set $t = t + 1$.

For $m = 1, ..., M$, draw

$$\boldsymbol{\gamma}_t^{(m)} \sim \tilde{f}_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1},\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}, \boldsymbol{y}_{1:t}). \tag{2.154}$$

Step 3. For $m = 1, ..., M$, compute the importance weights

$$\omega_t^{(m)} = \omega_{t-1} \frac{f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t|\boldsymbol{\gamma}_t^{(m)}) f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t^{(m)}|\gamma_{t-1}^{(m)})}{\tilde{f}_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1},\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_t^{(m)}|\boldsymbol{\gamma}_{t-1}^{(m)}, \boldsymbol{y}_{1:t})}. \tag{2.155}$$

If the proposal distribution used is

$$\tilde{f}_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1},\boldsymbol{Y}_{1:t}}(\boldsymbol{\gamma}_t^{(m)}|\boldsymbol{\gamma}_{t-1}^{(m)}, \boldsymbol{y}_{1:t}) = f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t^{(m)}|\boldsymbol{\gamma}_{t-1}^{(m)}), \tag{2.156}$$

then the importance weights are simplified to

$$\omega_t^{(m)} = \omega_{t-1} f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t|\boldsymbol{\gamma}_t^{(m)}). \tag{2.157}$$

Normalise the importance weights

$$\tilde{\omega}_t^{(m)} = \frac{\omega_t^{(m)}}{\sum_{m=1}^{M} \omega_t^{(m)}}. \tag{2.158}$$

Step 4. Re-sample M particles with probabilities $\{\tilde{\omega}_t^{(m)}\}_{m=1}^{M}$, and set $\omega_t^{(m)} = \dfrac{1}{M}$ for $m = 1, ..., M$.

Step 5. Repeat steps 2-4 until $t = T$.

The particle filtering algorithm presented above is the classical algorithm. Various modifications and improvements have been made to this algorithm to improve the efficiency and accuracy of the filtering process.

A fundamental difficulty with particle filtering is particle degeneracy, see for example, Doucet and Johansen (2011); Li et al. (2014); Arulampalam et al. (2002). This refers to a situation where all but a few particles have negligible weights after a number of iterations. Indeed, this issue has been proven in Doucet et al. (2000) to be an inherent default of the sequential Monte Carlo algorithms used in particle filtering. In particular, it has been shown

that the variance of the importance weights increases over time. Consequently, the majority of particles have normalised weights very close to 0 after a few time steps.

As the degeneracy issue of sequential Monte Carlo algorithms is unavoidable, re-sampling is incorporated into the particle filtering algorithm to mitigate it. In this step, the particles with significant weights are multiplied while those with negligible weights are abandoned. However, the diversity of the particles deteriorates significantly as a consequence of this re-sampling scheme. This is because only a small number of particles that have large weights are likely to be drawn and the resultant sample will only contain repeated copies of these particles. This problem has the same effect as the degeneracy issue without re-sampling mentioned previously. It is also referred to in the literature as the weight degeneracy, or particle impoverishment issue (Li et al., 2014; Arulampalam et al., 2002). Many advanced particle filters have been developed which aim to address this issue. A review of these techniques can be found in Li et al. (2014).

2.5.4.2 Parameter estimation

Various parameter estimation techniques have been considered for non-Gaussian state space models. They can be classified into two groups: off-line estimation methods and on-line estimate methods (Kantas et al., 2009). Off-line estimation methods refer to the estimation of parameters using all observations at once. On-line estimation methods, on the other hand, often integrate the estimation of parameters with the filtering of states. Hence parameters are updated recursively in the same manner as the filtering of states upon the arrival of new observations. The particle filtering approaches that provide on-line estimation of states and parameters are also called particle learning (Lopes and Tsay, 2011).

A simplistic off-line estimation method is maximum likelihood estimation. The likelihood function can be updated recursively as a by-product of the filtering process. As with the likelihood function of a Gaussian model, we can write the likelihood function as

$$f_{\boldsymbol{Y}_{1:T}}(\boldsymbol{Y}_{1:T};\boldsymbol{\Theta}) = f_{\boldsymbol{Y}_0}(\boldsymbol{y}_0;\boldsymbol{\Theta})\prod_{t=1}^{T} f_{\boldsymbol{Y}_t|\boldsymbol{Y}_{1:t-1}}(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1};\boldsymbol{\Theta}), \tag{2.159}$$

with

$$f_{\boldsymbol{Y}_t|\boldsymbol{Y}_{1:t-1}}(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1};\boldsymbol{\Theta}) = \int f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t|\boldsymbol{\gamma}_t;\boldsymbol{\Theta}) f_{\boldsymbol{\gamma}_t|\boldsymbol{Y}_{1:t-1}}(\boldsymbol{\gamma}_t|\boldsymbol{Y}_{1:t-1};\boldsymbol{\Theta}) d\boldsymbol{\gamma}_t \approx \frac{1}{M}\sum_{m=1}^{M} \omega_t^{(m)}, \tag{2.160}$$

where $\omega_t^{(m)}$ is the importance weight calculated in Step 3 of the particle filtering algorithm in Section 2.5.4.1. The likelihood can then be maximised using numerical methods to obtain parameter estimates (Durbin and Koopman, 2012).

On-line estimation, or particle learning methods allow parameters to be updated in the filtering process as new observations arrive. These parameters, however, are static, meaning that they are fixed and do not change over time in the same way that states do. Hence, they need to be treated differently (Carvalho et al., 2010).

A well-known particle learning technique in the literature is developed in Liu and West (2001), which is often called the Liu and West filter. In this filter, artificial dynamic noise is added to the static parameters with controlled variance inflation. This allows parameters to be treated as evolving "states" in the filter. The filter can be described as follows.

Liu and West filter (Liu and West, 2001)

Step 1. Initialisation: At $t = 0$, for $m = 1, ..., M$, draw parameters from their prior densities

$$\boldsymbol{\Theta}_0^{(m)} \sim \tilde{f}_{\boldsymbol{\Theta}_0}(\boldsymbol{\Theta}_0^{(m)}), \tag{2.161}$$

states from their initial distribution

$$\boldsymbol{\gamma}_0^{(m)} \sim \tilde{f}_{\boldsymbol{\gamma}_0}(\boldsymbol{\gamma}_0; \boldsymbol{\Theta}_t^{(m)}), \tag{2.162}$$

and calculate the importance weights

$$\omega_0^{(m)} = \frac{f_{\boldsymbol{\gamma}_0}(\boldsymbol{\gamma}_0^{(m)}; \boldsymbol{\Theta}_t^{(m)})}{\tilde{f}_{\boldsymbol{\gamma}_0}(\boldsymbol{\gamma}_0^{(m)}; \boldsymbol{\Theta}_t^{(m)})}. \tag{2.163}$$

Step 2. For $m = 1, ..., M$, compute

$$\widehat{\boldsymbol{\Theta}}_t^{(m)} = \xi \boldsymbol{\Theta}_{t-1}^{(m)} + (1-\xi)\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\Theta}_{t-1}^{(m)}, \tag{2.164}$$

where ξ is the shrinkage coefficient. Also compute

$$\widehat{\boldsymbol{\gamma}}_t^{(m)} = E[\boldsymbol{\gamma}_t | \boldsymbol{\gamma}_{t-1}^{(m)}, \boldsymbol{\Theta}_{t-1}^{(m)}]. \tag{2.165}$$

Step 3. For $m = 1, ..., M$, compute the look-ahead importance weights

$$\omega_t^{(m)} = \omega_{t-1}^{(m)} f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t | \widehat{\boldsymbol{\gamma}}_t^{(m)}; \widehat{\boldsymbol{\Theta}}_t^{(m)}). \tag{2.166}$$

Normalise the importance weights

$$\tilde{\omega}_t^{(m)} = \frac{\omega_t^{(m)}}{\sum_{m=1}^{M} \omega_t^{(m)}}. \tag{2.167}$$

Step 4. Re-sample M particles $\{\boldsymbol{\gamma}_{t-1}^{(m)}; \widehat{\boldsymbol{\Theta}}_t^{(m)}\}_{m=1}^{M}$ with probabilities $\{\widehat{\omega}_t^{(m)}\}_{m=1}^{M}$.

Step 5. For $m = 1, ..., M$, draw

$$\boldsymbol{\Theta}_t^{(m)} \sim \text{Normal}(\widehat{\boldsymbol{\Theta}}_t^{(m)}, (1-\xi^2)\Sigma_{\boldsymbol{\Theta}_{t-1}}), \tag{2.168}$$

where $\Sigma_{\boldsymbol{\Theta}_{t-1}}$ is the covariance matrix of $\{\boldsymbol{\Theta}_{t-1}^{(m)}\}_{m=1}^{M}$.

Also sample

$$\boldsymbol{\gamma}_t^{(m)} \sim f_{\boldsymbol{\gamma}_t|\boldsymbol{\gamma}_{t-1}}(\boldsymbol{\gamma}_t | \boldsymbol{\gamma}_{t-1}^{(m)}; \boldsymbol{\Theta}_t^{(m)}). \tag{2.169}$$

Step 6. For $m = 1, ..., M$, calculate the importance weights

$$\omega_t^{(m)} = \frac{f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t | \gamma_t^{(m)}; \boldsymbol{\Theta}_t^{(m)})}{f_{\boldsymbol{Y}_t|\boldsymbol{\gamma}_t}(\boldsymbol{y}_t | \widehat{\gamma}_t^{(m)}; \widehat{\boldsymbol{\Theta}}_t^{(m)})}. \tag{2.170}$$

Normalise the importance weights

$$\tilde{\omega}_t^{(m)} = \frac{\omega_t^{(m)}}{\sum_{m=1}^{M} \omega_t^{(m)}}. \tag{2.171}$$

Step 7. Repeat steps 2-6 until $t = T$.

In the Liu and West algorithm, the re-sampling step is done prior to the filtering step. This is to improve the efficiency in the selection of particles by using information in the subsequent period. This aims to reduce the particle degeneracy issue mentioned at the end of Section 2.5.4.1.

2.5.5 Univariate non-Gaussian models

There have been some non-Gaussian models introduced in the reserving literature. These models focus on single business segments.

Taylor and McGuire (2009) developed a framework with a focus on some specific members of the EDF. Specific examples are provided for models with gamma distributions and ODP distributions. A modified Hoerl curve is used for the mean structure with a log-link

$$\log(\mu_{i,j}) = a_i + r_i \log(j) + s_i j + \tilde{r}_i \min(j, 16), \tag{2.172}$$

where a different development level pass the development period 16 is allowed through the additional factors $\tilde{r} \min(j, 16)$. Note that this mean structure is tailored to the specific data set used for illustration in the paper. All coefficients of the curve, including a_i, r_i, s_i, $\tilde{r}_i$ are accident period-specific factors and they evolve as we proceed from one accident period to another. A filter called the second-order Bayesian revision is then used to give closed-form filtered estimates of these random factors.

The second-order Bayesian revision technique was developed in Taylor (2008) to provide closed-form solutions to state space models that use the EDF with conjugate prior specifications. This technique aims at a group of distributions from the EDF which can produce linear signals using their canonical link functions

$$\theta_t = a_i + r_i \log(j) + s_i j + \tilde{r}_i \min(j, 16). \tag{2.173}$$

The revision can be considered a "replica" of the Kalman filter for Gaussian models. With the use of canonical links or conjugate canonical links, a second-order Taylor series can be applied to approximate the filtered distribution of interest. For the special case of a Gaussian model, the second-order Bayesian approximation is exactly equal to the Kalman filter. It

is also noted in Taylor (2008) and Taylor and McGuire (2009) that this approach is only analytically tractable in a limited set of distributions with canonical or conjugate canonical links.

Sims (2011) developed a particle filtering algorithm for the non-Gaussian state space model in Taylor and McGuire (2009) to utilise the developments in MCMC and improve the flexibility in the choice of models (where the canonical link/conjugate canonical link requirement is relaxed). The particle filtering procedure used in Sims (2011) is the standard sequential Monte Carlo particle filtering algorithm described in Section 2.5.4.1. In this particle filter, parameters are estimated before running the filter using an initial residual analysis. Sims (2011) experienced the degeneracy issue, and also observed that the particle filter used could not always keep track of the changes in claim activity over time.

Dong and Chan (2013) developed an evolutionary framework using the generalised beta family of distributions. This is a class of distributions consisting of both light-tailed and heavy-tailed distributions such as gamma distributions, Weibull distributions, and Pareto distributions. In this framework, each individual claim cell $Y_{i,j}$ is assumed to follow a distribution from the generalised beta family. The chain ladder mean structure is then used

$$a_i + b_j. \tag{2.174}$$

These factors of the mean structure are then assumed to follow AR(1) processes. Off-line estimation is used to estimate those factors as well as other unknown parameters of the framework, i.e. all parameters are estimated at once using all observations. The estimation is performed using Bayesian inference.

2.6 Literature summary and areas for development

We provide a summary of the literature review in Section 2.6.1. Areas for development are then identified and provided in Section 2.6.2.

2.6.1 Literature summary

Loss reserving is a topic of significant importance to insurers as the valuation of outstanding claims is crucial for their financial stability and fulfilment of regulatory requirements. It is an area of research that has been receiving increasing attention, as shown (briefly) in this chapter.

Traditionally, deterministic reserving methods are used. They produce single mean estimates of the outstanding claims liabilities. Three traditional methods are reviewed in Section 2.2, including the well-known chain ladder algorithm (Section 2.2.1), the Bornhuetter-Ferguson algorithm (Section 2.2.2), and the Berquist-Sherman technique (Section 2.2.3). Estimates from the chain ladder algorithm are purely driven by the data, while the Bornhuetter-Ferguson algorithm incorporates some expert knowledge into the estimation. When there are changes in claim activity over time, the Berquist-Sherman technique can be used.

While deterministic methods are simple and convenient to apply, they do not provide estimates of uncertainty associated with the single mean estimates of outstanding claims. This can have significant consequences on the financial stability as well as solvency of an insurer, as explained in Section 2.3.1. This has motivated the development of stochastic models, which are the focus of Section 2.3. An important and popular type of stochastic models is developed using the EDF, and in most cases, its Tweedie sub-family. Many models have been introduced to the literature which use members from this family, including ODP distributions, gamma distributions, Tweedie's compound Poisson distributions, and others. Some of the theory of the EDF and the Tweedie sub-family is given in Section 2.3.2 which covers the definition, as well as many interesting properties of the family. The extensive use of this family in the reserving literature is shown in the review in Section 2.3.3 which is on the GLM framework and Section 2.3.4 which is on Tweedie models.

The focus of Section 2.4 is on multivariate reserving models. A general insurance company often operates in multiple business segments whose risks are dependent to some extent, but not monotonically. In the valuation of the outstanding claims liability on the aggregate level, it is important to consider the dependence structure amongst business segments to account for diversification benefits, and to also satisfy regulatory requirements. These are explained in detail in Section 2.4.1. These motivate the development of multivariate

models for reserving. Three types of models are reviewed: copula models, multivariate models with specific marginals, and common shock models. The copula modelling approach is a very popular multivariate reserving approach due to its flexibility in marginal modelling and dependence modelling. The extensive use of copulas in the reserving literature is summarised in Section 2.4.2. Besides copulas, multivariate models with specific marginals have also been used and are described in Section 2.4.3. Common shock approaches have also been considered for multivariate outstanding claims modelling. Common shock models have many distinctive strengths, including explicit dependence structures, ease of interpretation, and parsimonious and discipline construction of correlation matrices. These models are reviewed in Section 2.4.4.

Section 2.5 explores another segment of the reserving literature which focuses on evolutionary modelling. Insurers typically experience changes in claims activity over time. Evolutionary models with evolving factors are a parsimonious and elegant solution that allows changes to be incorporated naturally in such cases. The benefits of evolutionary models are described in detail in Section 2.5.1. Evolutionary models can be classified into two groups based on the distributional assumptions used: Gaussian models and non-Gaussian models. Some of the theory of Gaussian models is given in Section 2.5.2. This section also provides a description of the Kalman filter, a recursive algorithm that estimates evolving factors sequentially upon the arrival of new observations. Gaussian evolutionary models in the reserving field are reviewed in Section 2.5.3. Some of the theory of non-Gaussian models is provided in Section 2.5.4, together with particle filters, popular simulation based algorithms that estimate evolving factors recursively. A review of non-Gaussian reserving models is provided in Section 2.5.5.

2.6.2 Areas for development

The aim of our research is to develop models that incorporate realistic data features as well as desirable model features. This can improve the accuracy in the valuation of outstanding claims liabilities and enhance the practicality of models.

Insurers typically operate in multiple segments and it is essential to consider the dependence across segments in the valuation of outstanding claims on the portfolio level. This allows insurers to assess their diversification benefits appropriately. Insurers can

then set more accurate aggregate reserves and capital (see also Section 2.4.1). While searching for dependence modelling techniques, we draw our attention to common shock approaches. Common shock approaches are very popular dependence modelling tools with various strengths (Lindskog and McNeil, 2003). These approaches typically use common random factors to capture the drivers of dependence across related variables. This allows these drivers to be identified and monitored if needed. The explicit dependence structures from these approaches also enhance the ease of interpretation, which is one of the four desirable properties of multivariate models considered in Joe (1997, Chapter 4). These properties are also listed in Chapter 1. In addition, correlation matrices, which are tools used extensively by practitioners to specify the dependency in their portfolios, can be constructed in a parsimonious and disciplined manner (Avanzi, Taylor and Wong, 2018). Common shock approaches have made their various appearances in the literature, as also described in Section 2.4.4.

It has been noted that the EDF and more specifically, its Tweedie sub-family, have been used quite extensively in the loss reserving literature, either in univariate models (Section 2.3), multivariate models (Section 2.4), or evolutionary models (Section 2.5). The Tweedie family is a very rich family of distributions that covers many commonly known distributions including Poisson distributions, Tweedie's compound Poisson distributions, gamma distributions, and many more (Jorgensen, 1997; Alai et al., 2016). This motivates the development of a common shock Tweedie framework which can possess many strengths of common shock models, as well as the marginal flexibility of the Tweedie family of distributions. This development will be considered in Chapter 3.

The Tweedie family of distributions, however, has quite a complex density. This issue can further escalate in a multivariate framework. An appropriate estimation approach also needs to be considered if a common shock Tweedie framework is developed. We will address this problem in Chapter 4.

Many reserving data sets contain some variation in claim activity between different lengths of delay. In particular, it often reaches a peak in some early years, then dies out as the delay increases. Furthermore, claim activities across segments are not identical. Some segments such as Auto Property Damage have shorter delays in claim activity while segments such as Auto Bodily Injury covers have longer delays. This particular data feature can have impacts on common shock models (Avanzi, Taylor and Wong, 2018). We also aim to consider

the impacts of this feature in further detail and propose a solution to address them in Chapter 5.

Varying claim activity over time is another feature of reserving data typically encountered by insurers. This very common and important feature has motivated the development of evolutionary models which capture these changes naturally through evolving parameters (De Jong and Zehnwirth, 1983; Zehnwirth, 1994; Taylor et al., 2003). A number of evolutionary models have been considered for a single business segment (see also Section 2.5). This motivates us to develop a multivariate GLM evolutionary framework using a common shock approach to incorporate the dependence across segments. Inspired by the richness and popularity of GLMs and the EDF in reserving (Section 2.3.3), we can consider a natural extension of the traditional GLMs by letting their parameters evolve. We are also motivated to formulate filtering approaches that provide real-time updates of random factors for this evolutionary framework. This development will be the focus of Chapter 6.

CHAPTER 3

A multivariate Tweedie framework - Theory[1]

3.1 Introduction

In actuarial applications, common shock approaches have been very popular dependence modelling tools (Lindskog and McNeil, 2003). They are typically used to capture structural dependence that are due to known relationships which can be accounted for in a modelling framework (International Actuarial Association, 2004). A review of their applications in reserving is provided in Section 2.4.4. Common shock approaches have a number of properties that are particularly useful in modelling. With explicit dependence structures, common shock models offer ease of interpretation. The drivers of dependence can be identified and monitored if needed. Furthermore, the construction of correlation matrices can be put at ease using common shock models. Correlation matrices are tools used extensively by practitioners to specify dependence in the aggregation of outstanding claims liabilities and risk-based capital. Common shock approaches allow correlation matrices to be specified in a disciplined and parsimonious manner (Avanzi, Taylor and Wong, 2018). This is particularly beneficial for a portfolio of a large number of business lines or segments, which can be up to 100 in many cases.

It is also desirable for a framework to possess marginal modelling flexibility, a feature that has contributed to the popularity of copulas in many fields. This has attracted us to the Tweedie family of distributions. The Tweedie family is a major subclass of the

[1] An abbreviated version of results in Chapters 3 and 4 has been published in *Avanzi, B., Taylor, G., Vu, P.A., Wong, B., 2016. Stochastic loss reserving with dependence: A flexible multivariate Tweedie approach. Insurance: Mathematics and Economics 71, 63–78.*

EDF, consisting of symmetric and non-symmetric, light-tailed and heavy-tailed distributions (Alai et al., 2016; Jorgensen, 1997). This class and its members are frequently used in loss reserving including univariate models, to multivariate models and evolutionary models (see also Chapter 2). Some notable members of the Tweedie family include Poisson distributions and Tweedie's compound Poisson distributions. The former are frequently used in loss reserving and well known in stochastic models that underlie the traditional chain ladder algorithm (Section 2.3.4.1). The latter have probability mass at 0 hence are applicable in many data sets which contain 0's (Section 2.3.4.3). A recapitulation of properties of the Tweedie family of distributions is provided in Section 2.3.2.2.

In the search for approaches to model realistic data features while offering desirable modelling features, we have been inspired to develop a common shock Tweedie framework. This framework can inherit many benefits of common shock approaches and the Tweedie family of distributions mentioned above. With these many benefits, this development can accomplish the overall research aim of developing models that offer great practicality and accurate valuation of outstanding claims.

This section focuses on the theoretical development of the common shock Tweedie framework. The theoretical framework is described in Section 3.2. Section 3.3 provides a detailed analysis of moments driven from the framework. Remarks on theoretical model properties are given in Section 3.4.

3.2 Framework development

In this section we develop a common shock Tweedie framework for claims from multiple segments of business. The framework structure is provided in Section 3.2.1 with more detail on the parametrisation in Section 3.2.2.

3.2.1 Structure

Furman and Landsman (2010) developed a multivariate Tweedie distribution using a common shock approach. This is the ideal tool that we can use to develop our framework. Recall from Section 2.4 that a number of multivariate models have been developed to capture cell-wise dependence across loss cells coming from the same accident period and development

period across triangles. These models include Shi and Frees (2011); Côté et al. (2016); Zhang et al. (2012); Shi et al. (2012); Shi (2014). Our framework aims to capture this dependence structure as well. While this dependence structure cannot be interpreted using systematic factors such as calendar year dependence or accident year dependence mentioned in Section 2.4, it can be used to simply capture correlated noise as we will also demonstrate in the real data illustration in Chapter 4.

In a preliminary step, claims are standardised using a common unit of exposure, such as the number of policies, or total premium received for each accident year. This is to ensure consistency across accident periods within and across segments. Standardised claims $Y_{i,j}^{(n)}$ from the i-th accident period and j-th development period across all business segments are then collected into a vector

$$\boldsymbol{Y}_{i,j} = \begin{pmatrix} Y_{i,j}^{(1)} \\ Y_{i,j}^{(2)} \\ \vdots \\ Y_{i,j}^{(N)} \end{pmatrix}. \tag{3.1}$$

Following the definition of the multivariate Tweedie distribution in Furman and Landsman (2010), each element of the above vector is assumed to be a sum of two components

$$Y_{i,j}^{(n)} = \frac{\tilde{\theta}}{\ddot{\theta}_{i,j}^{(n)}} U_{i,j} + Z_{i,j}^{(n)}, \tag{3.2}$$

where $U_{i,j}$ is referred to as the "common shock", $Z_{i,j}^{(n)}$ is referred to as the "idiosyncratic effect", $\tilde{\theta}$ and $\ddot{\theta}$ are the canonical parameters, $\tilde{\vartheta}$ and $\ddot{\vartheta}_{i,j}^{(n)}$ are the index parameters of $U_{i,j}$ and $Z_{i,j}^{(n)}$, respectively. In this framework, all cell-wise claims that are in the same position (i, j) share a common stochastic component $U_{i,j}$. The dependence across all cell-wise claims is introduced explicitly using this common shock. It can also be observed from this construction that the effect of the common shock $U_{i,j}$ to each loss cell $Y_{i,j}^{(n)}$ is scaled by the factor $\tilde{\theta}/\ddot{\theta}_{i,j}^{(n)}$. This scaling factor aims to adjust effects of the common shock to different business segments. Unique cell effects are captured by the idiosyncratic component $Z_{i,j}^{(n)}$.

The two components are assumed to be independent and have additive Tweedie

specifications, denoted by Tweedie*,

$$U_{i,j} \sim \text{Tweedie}_p^*(\tilde{\theta}, \tilde{\vartheta}), \tag{3.3}$$

$$Z_{i,j}^{(n)} \sim \text{Tweedie}_p^*(\ddot{\theta}_{i,j}^{(n)}, \ddot{\vartheta}_{i,j}^{(n)}), \tag{3.4}$$

where p is the power parameter of these additive Tweedie distributions. The definition of additive Tweedie distributions as well as other properties of the Tweedie family of distributions are provided in Section 2.3.2.2.

With the particular construction specified as above, following Furman and Landsman (2010), the marginal distribution of $Y_{i,j}^{(n)}$ is

$$Y_{i,j}^{(n)} \sim \begin{cases} \text{Tweedie}_1^*(\tilde{\theta}, \tilde{\vartheta} + \ddot{\vartheta}_{i,j}^{(n)}), & p = 1, \\ \text{Tweedie}_p^*\left(\ddot{\theta}_{i,j}^{(n)}, \tilde{\vartheta}\left(\dfrac{\tilde{\theta}}{\ddot{\theta}_{i,j}^{(n)}}\right)^{\frac{p-2}{p-1}} + \ddot{\vartheta}_{i,j}^{(n)}\right), & p \neq 1. \end{cases} \tag{3.5}$$

The marginal densities in this multivariate model are then Tweedie distributions. Hence this construction provides distributional tractability. This closure under the taking of marginals, together with ease of interpretation of the dependence structure, are also two of the four desirable properties of a multivariate model considered in Joe (1997, Chapter 4) (also listed in Chapter 1). However, it is worth noting that a common power parameter p is required for the common shock components $U_{i,j}$, the idiosyncratic components $Z_{i,j}^{(n)}$, as well as all the claim observations $Y_{i,j}^{(n)}$. This is a necessary requirement for the multivariate Tweedie framework to have the closure property under the taking of margins. However, this can become a limitation when dealing with common shock and idiosyncratic effects with drastically different properties, or a large number of business segments with a large variation in claim activities. As mentioned earlier, while the cell-wise dependence structure cannot be interpreted using systematic factors such as calendar year dependence or accident year dependence mentioned in Section 2.4, it can be used to simply capture correlated noise as we will also demonstrate in the real data illustration in Chapter 4. In such cases, the common shocks in the proposed framework are drivers of the correlated noises observed in the data.

The multivariate density of the vector $\boldsymbol{Y}_{i,j}$ is then

$$f_{\boldsymbol{Y}_{i,j}}\left(y_{i,j}^{(1)},...,y_{i,j}^{(N)}\right)=\int_0^{\min\left(\frac{\ddot{\theta}_{i,j}^{(1)}}{\tilde{\theta}}y_{i,j}^{(1)},...,\frac{\ddot{\theta}_{i,j}^{(N)}}{\tilde{\theta}}y_{i,j}^{(N)}\right)} f_{U_{i,j}}^{*}(u_{i,j})\prod_{n=1}^{N} f_{Z_{i,j}^{(n)}}^{*}\left(y_{i,j}^{(n)}-\frac{\tilde{\theta}}{\ddot{\theta}_{i,j}^{(n)}}u_{i,j}\right)du_{i,j}, \tag{3.6}$$

where $f_{\cdot}^{*}(.)$ is the additive Tweedie density. It is straightforward that the multivariate density is the probability density of the convolutions of pairs of independent random variables $U_{i,j}$ and $Z_{i,j}^{(n)}$ for claims in position (i,j). The common shock variable $U_{i,j}$ is common across all these convolutions.

3.2.2 Parametrisation

In this section, model parametrisation is considered in more detail. One of the properties of the Tweedie family and the EDF in general, is the availability of two representations, the additive form and the reproductive form, as summarised in Section 2.3.2. The general framework introduced in the previous section utilises the additive representation in the original multivariate Tweedie distribution by Furman and Landsman (2010). However, the reproductive representation has a location parameter and a dispersion parameter that specify the mean and the dispersion of the distribution. This form is easier to interpret and more convenient to work within the loss reserving context. This has been shown in the representation of all existing EDF and Tweedie models in the literature (Section 2.3.3 and 2.3.4). Therefore, we consider the reproductive representation for further parametrisation.

In the model parametrisation, we consider two separate cases: $p \neq 1$ in Section 3.2.2.1 and $p = 1$ in Section 3.2.2.2 which is the case of common shock ODP models.

3.2.2.1 Case 1: $p \neq 1$

Model parametrisation is performed on the reproductive Tweedie representation for the case $p \neq 1$. In the marginal reproductive representation equivalent to the additive

representation in Equations 3.3 and 3.4, we have

$$U_{i,j} \sim \text{Tweedie}_p(\tilde{\alpha}, \tilde{\phi}), \tag{3.7}$$

$$Z_{i,j}^{(n)} \sim \text{Tweedie}_p(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}, \ddot{\phi}_{i,j}^{(n)}), \tag{3.8}$$

where $\tilde{\alpha}$ and $\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}$ are the location parameters and $\tilde{\phi}$ and $\ddot{\phi}_{i,j}^{(n)}$ are the dispersion parameters. Parameter $\ddot{\alpha}_i^{(n)}$ is the accident period parameter, and $\ddot{\beta}_j^{(n)}$ is the development period parameter. To ensure the estimates of $\ddot{\alpha}_i^{(n)}, \ddot{\beta}_j^{(n)}$ are unique for each n, we set $\ddot{\alpha}_1^{(n)} = 1$, $n = 1, ..., N$. After performing an appropriate claims standardisation, it can be further assumed that $\ddot{\phi}_{i,j}^{(n)} = \ddot{\phi}^{(n)}$. This simplification is commonly used in univariate Tweedie reserving models to simplify model calibration (see also Section 2.3.4). It corresponds to the simplification $\ddot{\vartheta}_{i,j}^{(n)} = \ddot{\vartheta}^{(n)}$ in the additive form. As noted in Boucher and Davidov (2011), it is justified to have column-specific dispersion parameters, i.e. $\ddot{\phi}_j^{(n)}$, and these dispersion parameters and the p parameter are dependent. However, for the sake of simplifying the model calibration, as mentioned in Alai and Wüthrich (2009) and performed in many existing models in Section 2.3.4, we have chosen the above simplification. Note also that this simplification is not needed to carry out analyses on theoretical properties of the model.

Using the duality between the additive form and the reproductive form of a Tweedie distribution (Section 2.3.2.2), the relationships between parameters of the additive and reproductive representations are given by

$$\tilde{\alpha} = \tilde{\vartheta} \left(\tilde{\theta}(1-p) \right)^{\frac{1}{1-p}}, \tag{3.9}$$

$$\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} = \ddot{\vartheta}^{(n)} \left(\ddot{\theta}_{i,j}^{(n)}(1-p) \right)^{\frac{1}{1-p}}, \tag{3.10}$$

$$\tilde{\phi} = \tilde{\vartheta}^{1-p}, \tag{3.11}$$

$$\ddot{\phi}^{(n)} = (\ddot{\vartheta}^{(n)})^{1-p}. \tag{3.12}$$

With the duality transformation, we also obtain

$$Y_{i,j}^{(n)} = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}. \tag{3.13}$$

In this expression, the effect of the common shock $U_{i,j}$ is scaled by a product of a ratio of dispersion parameters $\tilde{\phi}$ and $\ddot{\phi}^{(n)}$, and a ratio of mean parameters $\tilde{\alpha}$ and $\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}$.

Using the relationship between the additive and reproductive representations of the

Tweedie family specified in Equation (2.32) in Section 2.3.2.2, the marginal distribution of $Y_{i,j}^{(n)}$ is given by

$$Y_{i,j}^{(n)} \sim \text{Tweedie}_p\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\left[\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}+1\right], \ddot{\phi}^{(n)}\left[\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}+1\right]^{1-p}\right). \tag{3.14}$$

The multivariate density in the reproductive representation is then

$$f_{\boldsymbol{Y}_{i,j}}\left(y_{i,j}^{(1)},...,y_{i,j}^{(N)}\right) = \int_0^{B_{i,j}} f_{U_{i,j}}(u_{i,j})\prod_{n=1}^{N} f_{Z_{i,j}^{(n)}}\left(y_{i,j}^{(n)} - \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{1-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}u_{i,j}\right)du_{i,j}, \tag{3.15}$$

where

$$B_{i,j} = \min\left(\left(\frac{\ddot{\alpha}_i^{(1)}\ddot{\beta}_j^{(1)}}{\tilde{\alpha}}\right)^{1-p}\frac{\tilde{\phi}}{\ddot{\phi}^{(1)}}y_{i,j}^{(1)},...,\left(\frac{\ddot{\alpha}_i^{(N)}\ddot{\beta}_j^{(N)}}{\tilde{\alpha}}\right)^{1-p}\frac{\tilde{\phi}}{\ddot{\phi}^{(N)}}y_{i,j}^{(N)}\right), \tag{3.16}$$

and where $f_{\cdot}(.)$ is the Tweedie density in reproductive form.

A special case which is a multivariate gamma model can be obtained by letting $p = 2$. Using the equivalent gamma distribution notations we have

$$U_{i,j} \sim \text{Tweedie}_2(\tilde{\alpha},\tilde{\phi}) = \text{Gamma}\left(\frac{1}{\tilde{\phi}},\frac{1}{\tilde{\alpha}\tilde{\phi}}\right), \tag{3.17}$$

$$Z_{i,j}^{(n)} \sim \text{Tweedie}_2(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)},\ddot{\phi}^{(n)}) = \text{Gamma}\left(\frac{1}{\ddot{\phi}^{(n)}},\frac{1}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\ddot{\phi}^{(n)}}\right). \tag{3.18}$$

and

$$Y_{i,j}^{(n)} = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{-1}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}U_{i,j} + Z_{i,j}^{(n)}, \tag{3.19}$$

$$\sim \text{Tweedie}\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\left[\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}+1\right], \ddot{\phi}^{(n)}\left[\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}+1\right]^{-1}\right), \tag{3.20}$$

$$= \text{Gamma}\left(\frac{1}{\ddot{\phi}^{(n)}}+\frac{1}{\tilde{\phi}},\frac{1}{\alpha_i^{(n)}\ddot{\beta}_j^{(n)}\ddot{\phi}^{(n)}}\right). \tag{3.21}$$

This is the common shock gamma model developed in Vu (2013) (Section 2.4.3.2).

3.2.2.2 Case 2: $p = 1$

By letting $p = 1$, we arrive at common shock ODP models where

$$U_{i,j} \sim \text{ODP}\left(\tilde{\alpha}, \tilde{\phi}\right), \tag{3.22}$$

$$Z_{i,j}^{(n)} \sim \text{ODP}\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}, \ddot{\phi}_{i,j}^{(n)}\right), \tag{3.23}$$

or equivalently,

$$\frac{U_{i,j}}{\tilde{\phi}} \sim \text{Poisson}\left(\frac{\tilde{\alpha}}{\tilde{\phi}}\right), \tag{3.24}$$

$$\frac{Z_{i,j}^{(n)}}{\ddot{\phi}_{i,j}^{(n)}} \sim \text{Poisson}\left(\frac{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}{\ddot{\phi}_{i,j}^{(n)}}\right), \tag{3.25}$$

where

$$\frac{\tilde{\alpha}}{\tilde{\phi}} = \tilde{\vartheta}\exp(\tilde{\theta}), \tag{3.26}$$

$$\frac{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}{\ddot{\phi}_{i,j}^{(n)}} = \ddot{\vartheta}_{i,j}^{(n)}\exp(\tilde{\theta}). \tag{3.27}$$

Parameters $\tilde{\phi}$ and $\ddot{\phi}_{i,j}^{(n)}$ are also called over-dispersion parameters. As with the general case, it can be further assumed that $\ddot{\phi}_{i,j}^{(n)} = \ddot{\phi}^{(n)}$ to simplify the analysis.

Incremental claims are then assumed to be

$$Y_{i,j}^{(n)} = \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}U_{i,j} + Z_{i,j}^{(n)}. \tag{3.28}$$

The effect of the common shock $U_{i,j}$ to each loss cell is scaled by a factor which is a ratio of dispersion parameters $\ddot{\phi}^{(n)}/\tilde{\phi}$. Consequently the marginal distribution of each loss cell is

$$Y_{i,j}^{(n)} \sim \text{ODP}\left(\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}\tilde{\alpha} + \ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)}\right). \tag{3.29}$$

The multivariate mass function of $\boldsymbol{Y}_{i,j}$ is then given by

$$f_{\boldsymbol{Y}_{i,j}}\left(y_{i,j}^{(1)}, ..., y_{i,j}^{(N)}\right) = \sum_{u_{i,j}=0}^{B_{i,j}} p_{U_{i,j}}(u_{i,j}) \prod_{n=1}^{N} p_{Z_{i,j}^{(n)}}\left(y_{i,j}^{(n)} - \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}u_{i,j}\right), \tag{3.30}$$

where

$$B_{i,j} = \min\left(\frac{\tilde{\phi}}{\ddot{\phi}^{(1)}} y_{i,j}^{(1)}, ..., \frac{\tilde{\phi}}{\ddot{\phi}^{(N)}} y_{i,j}^{(N)}\right). \tag{3.31}$$

The multivariate Tweedie distribution used in this new framework generalises the multivariate Poisson distribution in Kocherlakota and Kocherlakota (1992). In this original distribution, unit dispersions are used, i.e. $\tilde{\phi} = \ddot{\phi}^{(n)} = 1$. In this chapter, we relax this assumption to also allow for over-dispersion.

A summary of model parametrisation is provided in Table 3.1.

p	Additive form	Reproductive form/Poisson notation
$p \neq 1$	$Y_{i,j}^{(n)} = \dfrac{\tilde{\theta}}{\ddot{\theta}_{i,j}^{(n)}} U_{i,j} + Z_{i,j}^{(n)}$	$Y_{i,j}^{(n)} = \left(\dfrac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{1-p} \dfrac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}$
	$Y_{i,j}^{(n)} \sim \text{Tweedie}_p^*\left(\ddot{\theta}_{i,j}^{(n)}, \tilde{\vartheta}\left(\dfrac{\tilde{\theta}}{\ddot{\theta}_{i,j}^{(n)}}\right)^{\frac{p-2}{p-1}} + \ddot{\vartheta}_{i,j}^{(n)}\right)$	$Y_{i,j}^{(n)} \sim \text{Tweedie}_p\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\left[\left(\dfrac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\dfrac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1\right], \ddot{\phi}^{(n)}\left[\left(\dfrac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\dfrac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1\right]^{1-p}\right)$
	$U_{i,j} \sim \text{Tweedie}_p^*\left(\tilde{\theta}, \tilde{\vartheta}\right)$	$U_{i,j} \sim \text{Tweedie}_p(\tilde{\alpha}, \tilde{\phi})$
	$Z_{i,j}^{(n)} \sim \text{Tweedie}_p^*\left(\ddot{\theta}_{i,j}^{(n)}, \ddot{\vartheta}_{i,j}^{(n)}\right)$	$Z_{i,j}^{(n)} \sim \text{Tweedie}_p(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)})$
	Relationships between parameters: $\ddot{\vartheta}_{i,j}^{(n)} = \ddot{\vartheta}^{(n)}, \quad \tilde{\alpha} = \tilde{\vartheta}\left(\tilde{\theta}(1-p)\right)^{\frac{1}{1-p}}, \quad \ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)} = \ddot{\vartheta}^{(n)}\left(\ddot{\theta}_{i,j}^{(n)}(1-p)\right)^{\frac{1}{p-1}}, \quad \tilde{\phi} = \tilde{\vartheta}^{1-p}, \quad \ddot{\phi}^{(n)} = (\ddot{\vartheta}^{(n)})^{1-p}$	
$p = 1$	$Y_{i,j}^{(n)} = U_{i,j} + Z_{i,j}^{(n)}$	$Y_{i,j}^{(n)} = U_{i,j} + Z_{i,j}^{(n)}$
	$Y_{i,j}^{(n)} \sim \text{Tweedie}_1^*\left(\tilde{\theta}, \tilde{\vartheta} + \ddot{\vartheta}_{i,j}^{(n)}\right)$	$Y_{i,j}^{(n)} \sim \text{Poisson}\left(\tilde{\alpha} + \ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)$
	$U_{i,j} \sim \text{Tweedie}_1^*\left(\tilde{\theta}, \vartheta\right)$	$U_{i,j} \sim \text{Poisson}\left(\tilde{\alpha}\right)$
	$Z_{i,j}^{(n)} \sim \text{Tweedie}_1^*\left(\tilde{\theta}, \vartheta_{i,j}^{(n)}\right)$	$Z_{i,j}^{(n)} \sim \text{Poisson}\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)$
		Extension to ODP: $Y_{i,j}^{(n)} = \dfrac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}, \quad Y_{i,j}^{(n)} \sim \text{ODP}\left(\dfrac{\ddot{\phi}^{(n)}}{\tilde{\phi}}\tilde{\alpha} + \ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)}\right)$ $U_{i,j} \sim \text{ODP}\left(\tilde{\alpha}, \tilde{\phi}\right), \quad Z_{i,j}^{(n)} \sim \text{ODP}\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)}\right)$
	Relationships between parameters: Poisson: $\tilde{\alpha} = \tilde{\vartheta}\exp(\tilde{\theta}), \quad \ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)} = \ddot{\vartheta}_{i,j}^{(n)}\exp(\tilde{\theta})$ ODP: $\dfrac{\tilde{\alpha}}{\tilde{\phi}} = \tilde{\vartheta}\exp(\tilde{\theta}), \quad \dfrac{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}{\ddot{\phi}^{(n)}} = \ddot{\vartheta}_{i,j}^{(n)}\exp(\tilde{\theta})$	

Table 3.1: Summary table of model parametrisation for the multivariate Tweedie framework

3.3 Analysis of moments

The Tweedie framework developed in the previous section has a strong advantage of allowing moments of each claim cell $Y_{i,j}^{(n)}$, and consequently, moments of the total sum of outstanding claims to be obtained in closed-form. In this section, we provide an analysis of moments obtained from the framework. Section 3.3.1 gives general expressions for moment- and cumulant-generating functions for individual claim cells $Y_{i,j}^{(n)}$. In Section 3.3.2, closed-form expressions of the mean, variance, and covariance of $Y_{i,j}^{(n)}$—the main moments and cumulants of interest in loss reserving—are provided. Closed-form expressions of the mean and variance of the total outstanding claims in the claims portfolio are given in Section 3.3.3.

3.3.1 Moment- and cumulant-generating functions

The moment generating function of ODP models with $p = 1$ can be obtained using properties of Poisson distributions. Moment-generating functions of the common shock $U_{i,j}$ and the idiosyncratic component $Z_{i,j}^{(n)}$ for the non-Poisson case with $p \neq 1$ are provided in Furman and Landsman (2010) using the additive representation. Using parameters in the reproductive representation as represented in the previous section, the moment generating function of each component is then given by

$$M_{U_{i,j}}(t) = \begin{cases} \exp\left(\tilde{\phi}^{\frac{1}{1-p}}\left[\kappa\left(\dfrac{\tilde{\alpha}^{1-p}}{\tilde{\phi}(1-p)} + t\right) - \kappa\left(\dfrac{\tilde{\alpha}^{1-p}}{\tilde{\phi}(1-p)}\right)\right]\right), & p \neq 1, \\ \exp\left(\dfrac{\tilde{\alpha}}{\tilde{\phi}}(\exp(\tilde{\phi}t) - 1)\right), & p = 1, \end{cases} \tag{3.32}$$

$$M_{Z_{i,j}^{(n)}}(t) = \begin{cases} \exp\left(\left(\ddot{\phi}^{(n)}\right)^{\frac{1}{1-p}}\left[\kappa\left(\dfrac{\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)^{1-p}}{\ddot{\phi}^{(n)}(1-p)} + t\right) - \kappa\left(\dfrac{\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)^{1-p}}{\ddot{\phi}^{(n)}(1-p)}\right)\right]\right), & p \neq 1, \\ \exp\left(\dfrac{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}{\ddot{\phi}^{(n)}}(\exp(\ddot{\phi}^{(n)}t) - 1)\right), & p = 1., \end{cases} \tag{3.33}$$

where $\kappa(.)$ is the corresponding unit cumulant function of the variables defined in Section 2.3.2.2 as

$$\kappa(\theta) = \begin{cases} \exp(\theta), & p = 1, \\ -\log(-\theta), & p = 2, \\ \dfrac{1}{2-p}\left[\theta(1-p)\right]^{\frac{p-2}{p-1}}, & p \notin (0,1] \cup [2]. \end{cases} \tag{2.31}$$

The m^{th} cumulants of $U_{i,j}$ and $Z_{i,j}^{(n)}$, denoted by $K_{U_{i,j}}^{(m)}$ and $K_{Z_{i,j}^{(n)}}^{(m)}$, can be derived from the moment generating functions

$$K_{U_{i,j}}^{(m)} = \frac{\partial^m \log M_{U_{i,j}}(t)}{\partial t^m}\bigg|_{t=0} = \begin{cases} \tilde{\phi}^{\frac{1}{1-p}} \kappa^{(m)}\left(\dfrac{\tilde{\alpha}^{1-p}}{\tilde{\phi}(1-p)}\right), & p \neq 1, \\ \tilde{\phi}^{m-1}\tilde{\alpha}, & p = 1, \end{cases} \tag{3.34}$$

$$K_{Z_{i,j}^{(n)}}^{(m)} = \frac{\partial^m \log M_{Z_{i,j}^{(n)}}(t)}{\partial t^m}\bigg|_{t=0} = \begin{cases} \left(\ddot{\phi}^{(n)}\right)^{\frac{1}{1-p}} \kappa^{(m)}\left(\dfrac{\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)^{1-p}}{\ddot{\phi}^{(n)}(1-p)}\right), & p \neq 1, \\ \ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\left(\ddot{\phi}^{(n)}\right)^{m-1}, & p = 1, \end{cases} \tag{3.35}$$

where $\kappa^{(m)}(.)$ is the m^{th} derivative of the unit cumulant function.

As a result of the independence between these two components, the moment-generating function of the incremental claims $Y_{i,j}^{(n)}$ is given by

$$M_{Y_{i,j}^{(n)}}(t) = \begin{cases} \exp\left(\tilde{\phi}^{\frac{1}{1-p}}\left[\kappa\left(\dfrac{\tilde{\alpha}^{1-p}}{\tilde{\phi}(1-p)} + \left(\dfrac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{1-p}\dfrac{\ddot{\phi}^{(n)}}{\tilde{\phi}}t\right) - \kappa\left(\dfrac{\tilde{\alpha}^{1-p}}{\tilde{\phi}(1-p)}\right)\right]\right) \\ \times \exp\left(\left(\ddot{\phi}^{(n)}\right)^{\frac{1}{1-p}}\left[\kappa\left(\dfrac{\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)^{1-p}}{\ddot{\phi}^{(n)}(1-p)} + t\right) - \kappa\left(\dfrac{\left(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)^{1-p}}{\ddot{\phi}^{(n)}(1-p)}\right)\right]\right), & p \neq 1, \\ \exp\left(\dfrac{\tilde{\alpha}}{\tilde{\phi}}(\exp(\ddot{\phi}^{(n)}t) - 1) + \dfrac{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}{\ddot{\phi}^{(n)}}(\exp(\ddot{\phi}^{(n)}t) - 1)\right), & p = 1. \end{cases} \tag{3.36}$$

Consequently, moments of $Y_{i,j}^{(n)}$ can be expressed in a general form as

$$E\left[\left(Y_{i,j}^{(n)}\right)^m\right] = \sum_{r=0}^{m}\binom{m}{r}\left(\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{1-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}\right)^r E\left[U_{i,j}^r\right] E\left[\left(Z_{i,j}^{(n)}\right)^{m-r}\right], \tag{3.37}$$

and cumulants of $Y_{i,j}^{(n)}$ are

$$K_{Y_{i,j}^{(n)}}^{(m)} = \begin{cases} \left(\left(\dfrac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \dfrac{\ddot{\phi}^{(n)}}{\tilde{\phi}} \right)^m \tilde{\phi}^{\frac{1}{1-p}} \kappa^{(m)} \left(\dfrac{\tilde{\alpha}^{1-p}}{\tilde{\phi}(1-p)} \right) \\ + \left(\ddot{\phi}^{(n)} \right)^{\frac{1}{1-p}} \kappa^{(m)} \left(\dfrac{\left(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} \right)^{1-p}}{\ddot{\phi}^{(n)}(1-p)} \right), & p \neq 1, \\ \dfrac{\tilde{\alpha}}{\tilde{\phi}} (\ddot{\phi}^{(n)})^m + \ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} \left(\ddot{\phi}^{(n)} \right)^{m-1}, & p = 1. \end{cases} \tag{3.38}$$

3.3.2 Analysis of mean, variance and covariance

Utilising the formulas of moments and cumulants given in Section 3.3, closed-form expressions of any moments and cumulants can be obtained. We provide an analysis of the mean, variance and covariance of $Y_{i,j}^{(n)}$, which are the main moments and cumulants of interest in loss reserving.

Using the moments formula in Equation (3.37), the mean of $Y_{i,j}^{(n)}$ is given by

$$E[Y_{i,j}^{(n)}] = \ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} \left[\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{2-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1 \right], \tag{3.39}$$

$$= \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} \tilde{\alpha} + \ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}. \tag{3.40}$$

The mean of $Y_{i,j}^{(n)}$ is also the location parameter of its marginal distribution.

The common shock structure provides a convenient interpretation to the effect of dependence on the mean of the marginal claim cells. As shown in Equation (3.40), the first part comes from the common shock $U_{i,j}$ and the second part comes from the idiosyncratic component $Z_{i,j}^{(n)}$. The mean $\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}$ of idiosyncratic component $Z_{i,j}^{(n)}$ incorporates the accident period effect $\ddot{\alpha}_i^{(n)}$ and the development period effect $\ddot{\beta}_j^{(n)}$. This is the total expected claim in cell (i, j) in the n^{th} loss triangle assuming business segments are independent. The common shock effect $U_{i,j}$ introduces an expected additional claims level α. This effect on each of the business segment is scaled by a scaling factor as given in framework specification in Equation (3.13).

The variance of $Y_{i,j}^{(n)}$ can also be derived using the general cumulant formula in Equation

(3.38)

$$Var[Y_{i,j}^{(n)}] = \ddot{\phi}^{(n)} \left(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}\right)^p \left[\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{2-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1 \right], \tag{3.41}$$

$$= \left(\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} \right)^2 \tilde{\phi}\tilde{\alpha}^p + \ddot{\phi}^{(n)} \left(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}\right)^p. \tag{3.42}$$

The variance can also be obtained using the mean parameter and the dispersion parameter of the marginal density of $Y_{i,j}^{(n)}$ in Equation (3.14). The over-dispersion impact of the common shocks is, again, clearly identifiable. Similar to the mean, the variance of $Y_{i,j}^{(n)}$ can also be computed as the sum of variances of two components $U_{i,j}$ and $Z_{i,j}^{(n)}$ as shown in Equation (3.42). This is due to the independence between these components in the model construction. This suggests that the dependence due to a stochastic factor $U_{i,j}$ can increase the variance of the outstanding claims $Y_{i,j}^{(n)}$. As shown in Equation (3.42), the variance of common shock $U_{i,j}$, after being scaled for each line of business, is added to the existing variance coming from the unique idiosyncratic component $Z_{i,j}^{(n)}$. We can relate this to practical situations in which different business segments have a positive dependence structure, and factors causing claims volatility to increase in one business segment can likely increase claims volatility in other segments.

The covariance between corresponding cells can be given by

$$Cov[Y_{i,j}^{(n)}, Y_{i,j}^{(m)}] = \frac{\tilde{\alpha}^{2-p}}{\left(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} \ddot{\alpha}_i^{(m)} \ddot{\beta}_j^{(m)}\right)^{1-p}} \frac{\ddot{\phi}^{(n)} \ddot{\phi}^{(m)}}{\tilde{\phi}}, \qquad m \neq n. \tag{3.43}$$

This comes directly from the construction of the model in which the common shock term is the generator of dependence across segments of business. In particular, the covariance is calculated as the product of the variance of the common shock $U_{i,j}$ and its scaling factors in each individual business segment. This transparent introduction of dependence allows an explicit expression of the covariance to be obtained. This covariance is null when $\alpha = 0$.

3.3.3 Mean and variance of the sum

The common shock structure in the common shock Tweedie framework also allows us to obtain closed-form expressions for cumulants of the sum of claims. Because all cell-wise claims $Y_{i,j}^{(n)}$ in the same position (i, j) share a common shock component $U_{i,j}$, the sum of

these claims can be given by

$$\sum_{n=1}^{N} Y_{i,j}^{(n)} = \left(\sum_{n=1}^{N} \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} \right) U_{i,j} + \sum_{n=1}^{N} Z_{i,j}^{(n)}. \tag{3.44}$$

Due to the independence between the common shock $U_{i,j}$ and the idiosyncratic components $Z_{i,j}^{(n)}$, any cumulant of $\sum_{n=1}^{N} Y_{i,j}^{(n)}$ (when it is defined), can be calculated as the sum of corresponding cumulants of $U_{i,j}$ and $Z_{i,j}^{(n)}$ with appropriate scaling factors. Two special cumulants, the mean and variance, of the sum $\sum_{n=1}^{N} Y_{i,j}^{(n)}$ (when they are defined) can be obtained in closed-form as

$$E\left[\sum_{n=1}^{N} Y_{i,j}^{(n)} \right] = \left(\sum_{n=1}^{N} \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} \right) \tilde{\alpha} + \sum_{n=1}^{N} \ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}, \tag{3.45}$$

$$Var\left[\sum_{n=1}^{N} Y_{i,j}^{(n)} \right] = \left(\sum_{n=1}^{N} \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} \right)^2 \tilde{\phi} \tilde{\alpha}^p + \sum_{n=1}^{N} \ddot{\phi}^{(n)} \left(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} \right)^p. \tag{3.46}$$

The sum of all claims in all loss triangles of a company can be calculated by aggregating all sums of cell-wise claims $\sum_{n=1}^{N} Y_{i,j}^{(n)}$ for all i and j. Using the independence between claims from different positions within a loss triangle, any cumulant of the sum of all claims in the portfolio can be obtained as the sum of corresponding cumulants of $\sum_{n=1}^{N} Y_{i,j}^{(n)}$ for all i and j.

3.4 Remarks on theoretical properties

In this chapter we have introduced a common shock Tweedie framework. Some remarks on theoretical properties of the framework are provided in this section. These include explicit dependence structure through the use of a common shock approach, marginal flexibility and closure under the taking of marginals through the use of the rich Tweedie family of distributions, ability to handle masses at 0, closed-form moments as well as notes on the performance of the framework on unbalanced data.

3.4.1 Explicit dependence structure

In the Tweedie framework developed in this chapter, dependence across segments is introduced with the help of an easily identifiable and explicit common shock structure. This structure is seamlessly applicable on more than two dimensions. It can also provide ease of interpretation for the dependence structure and a simplified tractable specification of correlation matrices. This is particularly beneficial when dealing with a portfolio of numerous segments, which can be up to 100 in some cases (Avanzi, Taylor and Wong, 2018). The dependence captured in this framework is the cell-wise dependence across segments. This type of dependency is also considered in, for example, Shi and Frees (2011); Côté et al. (2016); Zhang et al. (2012); Shi et al. (2012); Shi (2014).

3.4.2 Marginal flexibility and closure under the taking of marginals

This multivariate Tweedie framework provides flexible marginal modelling with a flexible selection of power parameter p. This is a significant improvement over existing multivariate models such as multivariate Poisson and multivariate gamma frameworks with restricted p values. This can improve the suitability of the framework for a larger variety of data sets.

It is also worth noting that a common power parameter p is required for all segments of business in implementing the framework. This allows the multivariate Tweedie framework to have the closure property under the taking of margins, one of the four desirable properties of multivariate distributions considered in Joe (1997, Chapter 4). These properties are also listed in Chapter 1. However, this can become a limitation when dealing with a large number of business segments with a large variation in claim activities.

3.4.3 Ability to handle masses at 0

The Tweedie family is a rich family of distributions which also contains the Tweedie's compound Poison-gamma distribution. This is a well-known distribution with the ability to handle masses at 0. It follows that the common shock Tweedie framework also has this ability. This can be particularly beneficial in cases which have 0's in observations.

3.4.4 Closed-form moments

One can utilise the benefit of having closed-form moments and cumulants to obtain some prediction statistics including the mean and the standard deviation of the sum of outstanding claims. This moments tractability allows the forecast of total outstanding claims to be obtained more easily and conveniently.

3.4.5 Parametrisation and unbalanced data

In the construction of the model, the most simple parametrisation is used for the common shock component $U_{i,j}$ with canonical parameter $\tilde{\theta}$ and index parameter $\tilde{\vartheta}$. This is chosen for the sake of simplicity. These parameters can be modified to vary across development years, accident years and/or calendar years.

This simplified assumption can lead to an unsatisfactory performance of the model for unbalanced data that involves different lines of business with different development durations. This limitation can be overcome by specifying column-specific or volume-related parameters for the common shock $U_{i,j}$.

CHAPTER 4

A multivariate Tweedie framework - Estimation and applications[1]

4.1 Introduction

In Chapter 3, we have introduced a common shock Tweedie framework which has the many desirable theoretical properties of common shock modelling approaches as well as the rich Tweedie family of distributions. The Tweedie family, however, has quite a complex density (Joe, 1997). This complexity can further escalate in a multivariate framework. An appropriate estimation approach also needs to be considered to improve the practicality of this framework.

In the current literature, a number of estimation approaches have been considered for some special cases of common shock Tweedie distributions. For example, Karlis (2003) developed an expectation-maximisation approach for common shock Poisson distributions (special cases of common shock Tweedie distributions with $p = 1$), Tsionas (2004) developed a Bayesian framework with a Gibbs algorithm for common shock gamma distributions (special cases of common shock Tweedie distributions with $p = 2$). The only existing estimation method that has been used for general common shock Tweedie distributions in the literature is the method of moments, see for example, Alai et al. (2016) and Furman and Landsman (2010). However, due to the small sample size often encountered in loss reserving, this method

[1] An abbreviated version of results in Chapters 3 and 4 has been published in *Avanzi, B., Taylor, G., Vu, P.A., Wong, B., 2016. Stochastic loss reserving with dependence: A flexible multivariate Tweedie approach. Insurance: Mathematics and Economics 71, 63–78.*

is not suitable for our framework.

Besides bootstrapping, Bayesian inference is a popular choice for loss reserving data due to it its ability to account for parameter uncertainty when providing a prediction of outstanding losses (Meyers and Shi, 2011). In the loss reserving literature, there has also been an increasing interest in Bayesian methods; see, for example, De Alba (2002); England and Verrall (2006); Meyers (2009); Shi et al. (2012); Zhang and Dukic (2013). This is a result of high-speed computers and advancements in Monte Carlo methods (Verrall et al., 2012). These profound developments in Monte Carlo methods, see for example, Brooks et al. (2011); Kroese et al. (2011), allow Bayesian inference to be more appropriate for complex models. In addition, a Bayesian set up allows one to incorporate expert opinions using prior distributions (Shi et al., 2012). Bayesian forecasting also incorporates all currently available information into the prediction of outstanding claims (Salzmann et al., 2012; Zhang et al., 2012). We are motivated by these benefits to provide a Bayesian set up for the parameter estimation of the multivariate Tweedie framework.

A description of the Bayesian inference used for estimation is given in Section 4.2. Illustrations using simulated data are given in Section 4.3. The development is also illustrated using a real data set from a P&C insurer in the United States in Section 4.4. Remarks on the implementation of the framework are drawn from these illustrations and provided in Section 4.5.

4.2 Bayesian inference for estimation

In this section, a Bayesian set up for the multivariate Tweedie framework is formulated. This Bayesian inference is used to estimate all unknown parameters in the common shock Tweedie framework except for the power parameter p. We propose the use of a Tweedie log-likelihood profile approach to estimate the power parameter p and to use this estimate in the Bayesian inference. This can significantly improve the stability of the algorithm, especially when the parameter vector is large. The conventional Bayesian inference with only one single step of estimation using the multivariate Tweedie density also suffers from instability due to a large number of parameters that need to be estimated at once. In addition, the multivariate Tweedie density of claims in this framework, as given in Equation (3.15), involves an integration over the common shock component that can take time to compute.

This results in a rather inefficient Bayesian inference for estimation.

To improve the efficiency of estimation, a two-stage procedure is proposed which is a non-conventional Bayesian procedure. The idea comes from the closure under the taking of marginals property of common shock Tweedie distributions, as mentioned in the remark in Section 3.4.2. Even though it is assumed that pair-wise claim cells follow a multivariate Tweedie distribution, each claim $Y_{i,j}^{(n)}$ still has its own marginal Tweedie distribution with specified location and dispersion parameters given in Equation (3.14). In the marginal estimation stage, this property is utilised and the likelihood is evaluated using marginal densities of $Y_{i,j}^{(n)}$ for all i, j and n. In the specification of the marginal distribution of $Y_{i,j}^{(n)}$, the two parameters of the common shock $\tilde{\alpha}$ and $\tilde{\phi}$ always appear in a ratio which we denote by Λ

$$\Lambda = \frac{\tilde{\alpha}^{2-p}}{\tilde{\phi}}. \tag{4.1}$$

As a result, there is parameter redundancy if these parameters are included as separate estimands in the marginal estimation stage. Therefore, instead of estimating both $\tilde{\alpha}$ and $\tilde{\phi}$, only the newly defined parameter Λ is estimated in the first stage. The second stage is the multivariate estimation stage in which $\tilde{\alpha}$ and $\tilde{\phi}$ are estimated separately conditioning on parameters estimated in the first stage.

In a Bayesian set up, one needs to specify prior distributions and the likelihood function of the framework in use. Details of these are provided in Section 4.2.1 and Section 4.2.2. An MCMC algorithm is then specified for the Bayesian inference. A Metropolis-Hastings algorithm is chosen due to its ability to work with posterior densities that are in unrecognisable forms. Details of this algorithm are provided in Section 4.2.3. Section 4.2.4 then provides a description of the procedure used to obtain the predictive distribution of outstanding claims as well as other quantities of interest.

A variety of Bayesian computational packages can be readily applied for special cases of the multivariate Tweedie framework including multivariate ODP models (i.e. $p = 1$) and multivariate gamma models (i.e. $p = 2$). These tools are developed in a number of software, for example, WinBUGS and R. However, there are no existing computational tools that are ready for use for the general multivariate Tweedie framework. Hence the focus is placed on the estimation of the general framework with $p \neq 1$. Of course, the developments for the $p \neq 1$ case can be adapted to the $p = 1$ case.

4.2.1 Selection of prior distributions

Prior distributions can be chosen informatively using prior knowledge (Koop, 2003). They can also be uninformative and assign equal possibilities to all values in the feasible set of parameter values. However, it is worth noting that more informative prior distributions can result in faster convergence (Congdon, 2010, Chapter 1). It is also mentioned in Brooks et al. (2011, Chapter 23) that in highly parametrised models, somewhat informative prior distributions are necessary.

For our framework, a preliminary analysis can be performed for each segment separately and results can be used to specify more informative prior distributions. Recall from the specification of the framework that each claim cell $Y_{i,j}^{(n)}$ is the sum of a common shock component $U_{i,j}$ and an idiosyncratic component $Z_{i,j}^{(n)}$. Each claim cell $Y_{i,j}^{(n)}$ also follows a Tweedie marginal distribution. A naive analysis can be performed, assuming that $Y_{i,j}^{(n)}$ has a marginal distribution with the chain ladder mean structure. This is indeed the univariate Tweedie model developed in Alai and Wüthrich (2009); Peters et al. (2009) and Wüthrich (2003) (see also Section 2.3.4.5). Maximum likelihood estimates of parameters in this univariate model can give guidance for the selection of informative prior distributions. We can analyse the level of dependence across segments of business to choose informative prior distributions for the location parameter $\tilde{\alpha}$ of the common shock $U_{i,j}$. Analyses are done heuristically for $\tilde{\phi}$ and $\tilde{\alpha}$, hence relatively uninformative prior distributions are recommended for these parameters.

4.2.2 Specification of likelihood functions and posterior distributions

We estimate the parameters of the multivariate Tweedie framework using a two step approach. The first step in the estimation is the marginal estimation step. In this step, all parameters can be estimated, except $\tilde{\alpha}$ and $\tilde{\phi}$. However, this estimation step allows us to estimate Λ, which is a function of $\tilde{\alpha}$ and $\tilde{\phi}$ as defined in Equation (4.1). The posterior distribution of the parameter vector in step utilises the marginal densities of $Y_{i,j}^{(n)}$ and is given by

$$f_{\boldsymbol{\Theta}|\boldsymbol{Y}^U}(\boldsymbol{\Theta}|\boldsymbol{Y}^U) \propto \left(\prod_{n=1}^{N} \prod_{i=1}^{I} \prod_{j=1}^{I-i+1} f_{Y_{i,j}^{(n)}} \left(y_{i,j}^{(n)} | \boldsymbol{\Theta} \right) \right) f_{\Lambda}(\Lambda) f_{\ddot{\boldsymbol{\alpha}}}(\ddot{\boldsymbol{\alpha}}) f_{\ddot{\boldsymbol{\beta}}}(\ddot{\boldsymbol{\beta}}) f_{\ddot{\boldsymbol{\phi}}}(\ddot{\boldsymbol{\phi}}), \tag{4.2}$$

where

$$\boldsymbol{\Theta} = \begin{pmatrix} \Lambda \\ \ddot{\boldsymbol{\alpha}} \\ \ddot{\boldsymbol{\beta}} \\ \ddot{\boldsymbol{\phi}} \end{pmatrix}, \quad \ddot{\boldsymbol{\alpha}}_{\boldsymbol{i}} = \begin{pmatrix} \ddot{\alpha}_i^{(1)} \\ \ddot{\alpha}_i^{(2)} \\ \vdots \\ \ddot{\alpha}_i^{(N)} \end{pmatrix}, \quad \ddot{\boldsymbol{\alpha}} = \begin{pmatrix} \ddot{\alpha}_1 \\ \ddot{\alpha}_2 \\ \vdots \\ \ddot{\alpha}_I \end{pmatrix}, \quad \ddot{\boldsymbol{\beta}}_{\boldsymbol{j}} = \begin{pmatrix} \ddot{\beta}_j^{(1)} \\ \ddot{\beta}_j^{(2)} \\ \vdots \\ \ddot{\beta}_j^{(N)} \end{pmatrix}, \quad \ddot{\boldsymbol{\beta}} = \begin{pmatrix} \ddot{\beta}_1 \\ \ddot{\beta}_2 \\ \vdots \\ \ddot{\beta}_J \end{pmatrix}, \quad \ddot{\boldsymbol{\phi}} = \begin{pmatrix} \ddot{\phi}^{(1)} \\ \ddot{\phi}^{(2)} \\ \vdots \\ \ddot{\phi}^{(N)} \end{pmatrix}. \tag{4.3}$$

From the model structure in Equation (3.14), we have that all claims $Y_{i,j}^{(n)}$ are independent conditional on common shock. Hence, the joint likelihood can be written as a product of two separate parts: a product of density of claims conditional on common shock, and the density of common shock. In this stage stage, the likelihood obtained is the first part of the joint likelihood. However, since the common shock is not observed, we work with the joint likelihood directly in the second stage.

In the second step of the estimation process, $\tilde{\alpha}$ and $\tilde{\phi}$ are estimated. The multivariate density needs to be used because $\tilde{\alpha}$ and $\tilde{\phi}$ have separate roles in the multivariate density. To avoid instability in the MCMC when dealing with the multivariate density, all other parameters are held fixed at their marginal estimates. A restriction is also implied on $\tilde{\alpha}$ and $\tilde{\phi}$ using the estimate of Λ. The posterior distribution of the parameter vector is given by

$$f_{\tilde{\alpha}|\boldsymbol{Y}^U,\boldsymbol{\Theta}}(\tilde{\alpha}|\boldsymbol{Y}^U,\boldsymbol{\Theta}) \propto \left(\prod_{i=1}^{I} \prod_{j=1}^{I-i+1} f_{\boldsymbol{Y}_{i,j}}\left(\boldsymbol{Y}_{i,j}|\tilde{\alpha},\boldsymbol{\Theta}\right) \right) f_{\tilde{\alpha}}(\tilde{\alpha}). \tag{4.4}$$

The posterior distributions in Equations (4.2) and (4.4) are not in recognisable forms. Hence the Metropolis algorithm is used to simulate from these distributions. The Metropolis algorithm is a special case of the Metropolis-Hastings algorithm with symmetrical proposal density. Summary statistics including the median and standard deviation are then calculated using samples drawn from these posterior distributions.

4.2.3 Metropolis algorithm

The Metropolis-Hastings algorithm is a popular MCMC method used in Bayesian inference to create posterior simulators for a wide range of models, especially when posterior distributions are not in recognisable forms (Koop, 2003, Chapter 5). There have been a

number of applications of the Metropolis-Hastings algorithm in the loss reserving literature such as Meyers (2009) and Peters et al. (2009). We consider a special case of the Metropolis-Hastings algorithm, the Metropolis algorithm, in the evaluation of posterior distributions in our Bayesian inference. This algorithm has symmetric proposal densities, hence is less computationally expensive than the traditional Metropolis-Hastings algorithm.

Recall that the goal of our marginal estimation step is to obtain $f_{\boldsymbol{\Theta}|\boldsymbol{Y}^U}(\boldsymbol{\Theta}|\boldsymbol{Y}^U)$. The Metropolis algorithm used for this valuation then proceeds as follows based on the general review in Brooks et al. (2011).

Metropolis algorithm

Step 1. Specify initial values of parameters $\boldsymbol{\Theta}^{(0)}$.

Step 2. Draw $\boldsymbol{\Theta}^* \sim \tilde{f}_{.|\boldsymbol{\Theta}}(.|\boldsymbol{\Theta}^{(t-1)})$, where $\tilde{f}_{.|\boldsymbol{\Theta}}(.|\boldsymbol{\Theta}^{(t-1)})$ is called a proposal distribution, which is symmetric. A choice for the proposal distribution $\tilde{f}_{.|\boldsymbol{\Theta}}(.|\boldsymbol{\Theta}^{(t-1)})$ can be Normal$(\boldsymbol{\Theta}^{(t-1)}, \boldsymbol{\Sigma}_{\boldsymbol{\Theta}})$ where $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}$ is the covariance matrix of the parameter vector $\boldsymbol{\Theta}$ and it needs to be adjusted such that the acceptance rate is within a certain range. This process of choosing $\boldsymbol{\Sigma}_{\boldsymbol{\Theta}}$ is also called "tuning". When a normal distribution is used for for the proposal distribution, the Metropolis algorithm is also called random walk Metropolis algorithm.

Step 3. Compute the acceptance ratio

$$\text{MR} = \frac{f_{\boldsymbol{\Theta}|\boldsymbol{Y}^U}(\boldsymbol{\Theta}^*|\boldsymbol{Y}^U)\tilde{f}_{.|\boldsymbol{\Theta}}(\boldsymbol{\Theta}^{(t-1)}|\boldsymbol{\Theta}^*)}{f_{\boldsymbol{\Theta}|\boldsymbol{Y}^U}(\boldsymbol{\Theta}^{(t-1)}|\boldsymbol{Y}^U)\tilde{f}_{.|\boldsymbol{\Theta}}(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}^{(t-1)})} = \frac{f_{\boldsymbol{\Theta}|\boldsymbol{Y}^U}(\boldsymbol{\Theta}^*|\boldsymbol{Y}^U)}{f_{\boldsymbol{\Theta}|\boldsymbol{Y}^U}(\boldsymbol{\Theta}^{(t-1)}|\boldsymbol{Y}^U)}, \tag{4.5}$$

as $\tilde{f}_{.|\boldsymbol{\Theta}}(\boldsymbol{\Theta}^{(t-1)}|\boldsymbol{\Theta}^*) = \tilde{f}_{.|\boldsymbol{\Theta}}(\boldsymbol{\Theta}^*|\boldsymbol{\Theta}^{(t-1)})$ due the symmetry of the proposal distribution.

Step 4. Draw $u \sim \text{Uniform}(0, 1)$.

Step 5. Decide on the next value of the parameter vector:

$$\boldsymbol{\Theta}^{(t)} = \begin{cases} \boldsymbol{\Theta}^*, & \text{if } u < \text{MR}, \\ \boldsymbol{\Theta}^{(t-1)}, & \text{otherwise.} \end{cases} \tag{4.6}$$

As a rule of thumb, the optimal acceptance probability is 0.234 for a vector of parameters and is 0.44 for a single parameter (Brooks et al., 2011, Chapter 4). However,

a rate that is neither too low nor too high can generally be accepted (Brooks et al., 2011). To achieve the desired acceptance rate, tuning of the variance of the proposal distribution is required. This can be performed manually using trial runs. Alternatively, this tuning process can also be done automatically using an adaptive Metropolis-Hastings algorithm with a coerced acceptance rate (see, for instance,Vihola, 2012). A similar algorithm can be written for the posterior density in the multivariate estimation step simply by using the relevant parameters and densities.

Using the Metropolis algorithm, posterior draws of parameters in the model are obtained. A burn-in period might be required to remove the initial unstable portion of the chain. We also need to thin the chain by keeping every m^{th} draw to break the serial dependence between draws (Kruschke, 2011, Chapter 23). The thinned draws are then used to compute the posterior median, standard deviation and credibility interval for parameters in the model.

4.2.4 Predictive distribution of outstanding claims

Given the sample draws from the posterior distributions, one can then proceed to make the prediction of outstanding claims in the lower loss triangles. To generate the m^{th} sample of outstanding claims, the m^{th} draw from the thinned simulation draws of parameters is used. This includes the m^{th} draw of $\boldsymbol{\Theta}$ in the marginal estimation and m^{th} draw of $\tilde{\alpha}$ and $\tilde{\phi}$ in the multivariate estimation. Outstanding claims in the lower triangles are then calculated using

$$Y_{i,j}^{(n)} = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}. \tag{4.7}$$

where

$$U_{i,j} \sim \text{Tweedie}_p(\tilde{\alpha}, \tilde{\phi}), \tag{4.8}$$

$$Z_{i,j}^{(n)} \sim \text{Tweedie}_p(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)}), \tag{4.9}$$

for $1 < i \leq I$, $I - i + 2 \leq j \leq J$, $1 \leq n \leq N$.

The predictive distribution of the total outstanding claims can then be obtained using simulated samples of outstanding claims. The distribution of total claims can be calculated for each accident period, each loss triangle, and the aggregate portfolio. Summary statistics

such as the mean, variance and quantiles can also be obtained.

4.3 Simulation illustrations

Two simulated data sets are used to assess the effectiveness of the estimation approach. The first data set is generated from a multivariate Tweedie distribution with parameters chosen to replicate an empirical data set. It contains two standardised triangles of loss ratios with ten accident periods and ten development periods. Recall that loss ratios are calculated as incremental claims standardised using total premiums earned for the corresponding accident period. This data set is presented in Table 4.12 and Table 4.13 in Appendix 4.A.1.

A subset of the Tweedie family of distributions, Tweedie's compound Poisson distributions with $1 < p < 2$, is a useful subset of distributions in loss reserving with the ability to accommodate masses at 0; see, for example, Alai and Wüthrich (2009); Boucher and Davidov (2011); Wüthrich (2003). The second simulation illustration is performed on a simulated data set with an observation of 0 to assess the performance of the estimation in presence of 0's. This data set is presented in Table 4.15 and Table 4.16 in Appendix 4.A.1.

4.3.1 Estimation of power parameter p

To find the power parameter p of the model, a univariate Tweedie GLM log-likelihood profile for the combined data set of two loss triangles is set up in which the log-likelihood function is written as a function of power parameter p. The mean structure of incremental claim $Y_{i,j}^{(n)}$ in this log-likelihood profile is given by

$$a_i^{(1)} \mathbb{1}_{\{n=1\}} + b_j^{(1)} \mathbb{1}_{\{n=1\}} + a_i^{(2)} \mathbb{1}_{\{n=2\}} + b_j^{(2)} \mathbb{1}_{\{n=2\}}, \tag{4.10}$$

where $a_i^{(n)}$ and $b_j^{(n)}$ are coefficients in the GLM regression and $\mathbb{1}_{\{.\}}$ is the indicator function. The power parameter p is found numerically by testing a range of values for p on the Tweedie GLM log-likelihood profile of the data set. When there exists at least one observation of 0 in the data set, the range of the power parameter p is restricted to $(1, 2)$. The value of p that provides the highest likelihood is selected. The 95% confidence interval (CI) of the estimate is also obtained using a χ_1^2 distribution approximation to the likelihood. The estimation results for simulated data set 1 and simulated data set 2 are provided in Table 4.1.

Simulated data set	True value	Estimate	95% CI
1	1.32	1.36	(1.21, 1.56)
2	1.32	1.24	(1.14, 1.37)

Table 4.1: Estimates of power parameter p with 95% confidence intervals for two simulated data sets

This analysis has an underlying assumption of independence between lines. However, it can provide a reasonable estimation for parameter p as shown in Table 4.1. Bayesian inference is then performed conditioning on these estimates of p. While fixing the power parameter p in the Bayesian framework is rather an adhoc procedure, it can significantly improve the stability and convergence of the MCMC.

4.3.2 Marginal estimation

The marginal estimation step allows us to estimate all parameters except $\tilde{\alpha}$ and $\tilde{\phi}$. We apply a log transformation on the parameters to avoid any issues with the positive constraint of parameters. Prior distributions are chosen using information from the marginal maximum likelihood estimation as described in Section 4.2.1. Standard deviations of the proposal distributions in the random walk Metropolis algorithm are chosen so that the acceptance rate is reasonably close to 0.234. For both simulated data sets, 150,000 simulations are run and the first 50,000 iterations are discarded as the burn-in period. After this burn-in period, the MCMC approaches the stationary state. We provide sample paths of parameters in the simulated data set 1 in Figure 4.1 for illustration. We then use every 5^{th} iteration to thin the sample.

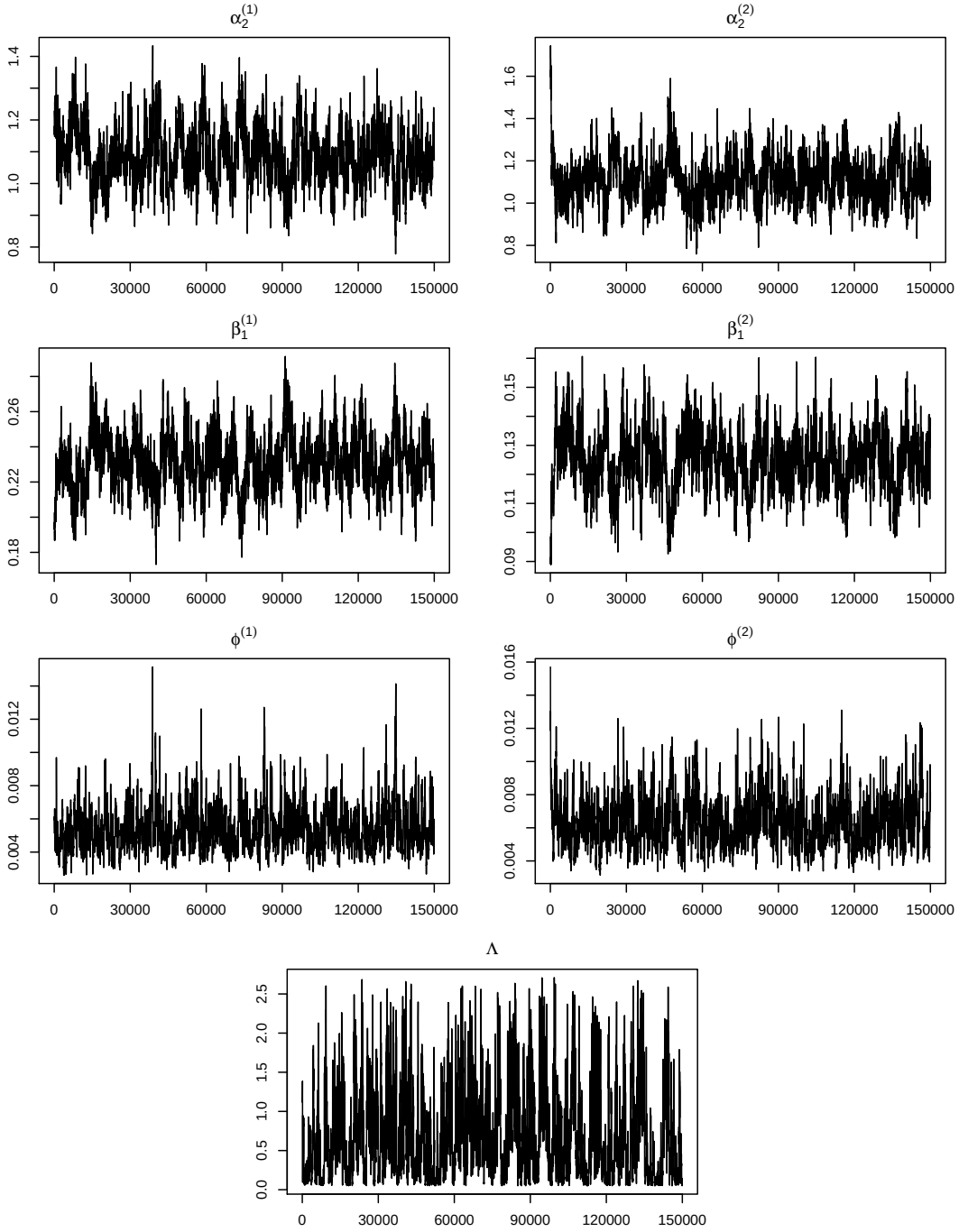

Figure 4.1: MCMC sample paths of some parameters

Estimation results for simulated data set 1 and simulated data set 2 are given in Table 4.14 and Table 4.17 in Appendix 4.A.1, respectively. The results represented include the posterior medians, standard deviations and 90% confidence intervals of the posterior distributions of parameters. Parameter estimates used are posterior medians due to the positive skewness of posterior distributions. The posterior medians are also closer to the true parameter values than the posterior means. The results show that the true values of

parameters are reasonably close to their estimates and they all lie within the corresponding confidence intervals.

4.3.3 Multivariate estimation

This step is used to estimate $\tilde{\alpha}$ and $\tilde{\phi}$ conditioning on estimates of all other parameters. An estimate of Λ is used to imply a restriction on $\tilde{\alpha}$ and $\tilde{\phi}$. In particular, only $\tilde{\alpha}$ is estimated and $\tilde{\phi}$ is computed as a function of $\tilde{\alpha}$ and Λ. For both simulated data sets, 30,000 iterations are used with the first 10,000 iterations discarded as the burn-in period. Estimates of $\tilde{\alpha}$ and $\tilde{\phi}$ are provided in Table 4.14 for simulated data set 1 and Table 4.17 for simulated data set 2 in Appendix 4.A.1. Actual values of parameters are also reasonably close to their estimates and they all fall within their corresponding 90% confidence interval.

Both simulation illustrations show that the calibration approach is reasonably accurate. It will be used to fit the model to a real data set in the subsequent section.

4.4 Illustration using real data

This section provides an illustration using real data. This data set consists of two business lines: Personal Auto line (denoted by (1)) and Commercial Auto line (denoted by (2)) and is collected for the period 1988-1997. It belongs to Pennsylvania National Insurance Group (Schedule P), and was used for an illustration in Zhang and Dukic (2013). The data set is provided in Tables 4.18 and 4.19 in Appendix 4.A.2.

4.4.1 Preliminary data analysis

A preliminary analysis is conducted on this data set. The first step of the analysis is to standardise incremental claims using the total premium earned in the corresponding accident year. This results in two triangles of loss ratios. Incremental loss ratios are converted into cumulative loss ratios whose plots are provided in Figure 4.2. It can be observed that the development patterns are quite similar for all years in the Personal Auto line. However, some volatility in claim activities are observed in the Commercial Auto line. In both lines of business, claims development tends to reach maturity at the end of the 10 year period,

suggesting that these are both short-tailed lines.

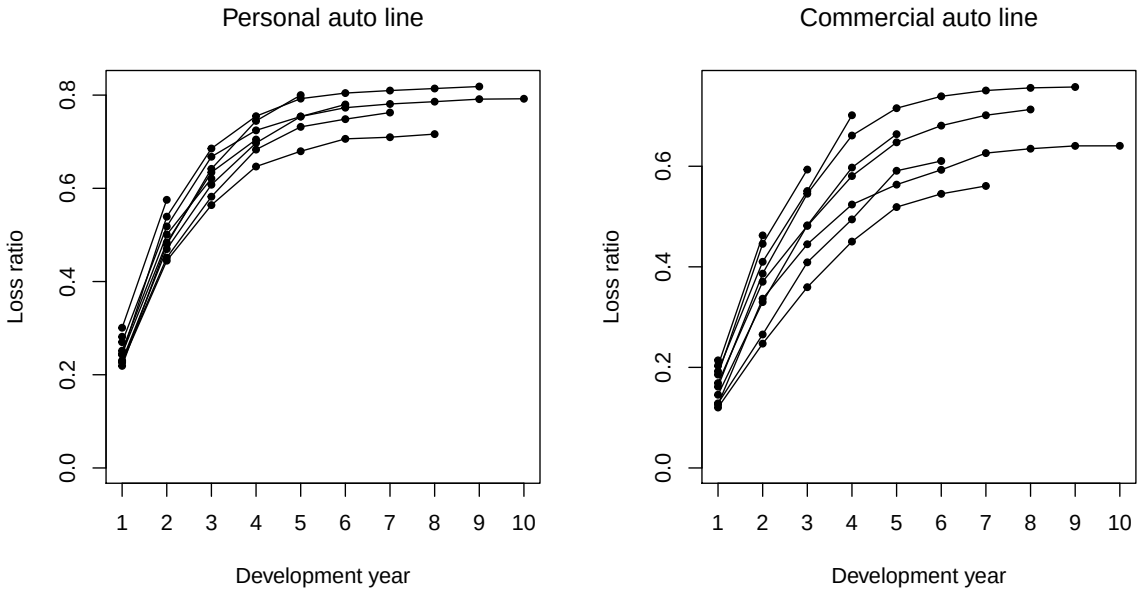

Figure 4.2: Cumulative loss ratios in real data from the Schedule P

4.4.1.1 Estimation of power parameter p

The power parameter p is estimated using a Tweedie GLM log-likelihood profile. This analysis is performed for each line of business, as well as for the combined portfolio of both lines. The mean structure of incremental claim $Y_{i,j}^{(n)}$ in this log-likelihood profile is given by

$$a_i^{(1)}\mathbb{1}_{\{n=1\}} + b_j^{(1)}\mathbb{1}_{\{n=1\}} + a_i^{(2)}\mathbb{1}_{\{n=2\}} + b_j^{(2)}\mathbb{1}_{\{n=2\}}, \tag{4.11}$$

Their estimates and confidence intervals are both provided in Table 4.2.

Line	$\widehat{p}$	95% CI
Personal Auto	1.15	(1.07, 1.40)
Commercial Auto	1.39	(1.24, 1.63)
Both lines	1.32	(1.21, 1.47)

Table 4.2: Estimates of power parameter p and their 95% confidence intervals

As required in the multivariate Tweedie framework, all lines of business need to have the same power parameter p. It can be observed from Table 4.2 that the estimate of p that provides the best fit to the combined portfolio is 1.32. This value is within the confidence

interval of the estimates from the two individual lines. Hence it will be used in the model fitting. Note that the above results are based on the assumption of segment-specific dispersion parameters $\phi^{(n)}$. As mentioned in Boucher and Davidov (2011), a different specification (such as column-specific) of dispersion parameters can result in a different outcome due to the dependence between the power parameter p and the dispersion parameter ϕ of a Tweedie distribution. The outcome might even be more favourable where the power parameters p could be more similar across the two lines. However, for the sake of simplification for the calibration, we only consider segment-specific dispersion $\phi^{(n)}$. This is an acceptable option as the individual power parameters p are all within the confidence interval of the chosen common power parameter p.

4.4.1.2 Exploratory dependence analysis

As pointed out in Avanzi, Taylor and Wong (2016), careful modelling is needed before committing to any measure of correlation. Hence a careful examination of the dependence structure is carried out.

The dependence across lines of business is first assessed by analysing the residuals after removing accident year and development year trends. This is performed by applying the GLM framework to both lines independently. To get the most accurate results, the GLM framework used for each line is a Tweedie dispersion with the power parameter p that provides the best fit to that line. In particular, a Tweedie distribution with power parameter $p = 1.15$ and a Tweedie distribution with power parameter $p = 1.39$ are used for the Personal Auto line and the Commercial Auto line respectively. GLM Pearson residuals are computed using the formula

$$\frac{Y_{i,j}^{(n)} - \mu_{i,j}^{(n)}}{\left(\mu_{i,j}^{(n)}\right)^{p}}. \tag{4.12}$$

Cell-wise correlations and their p-values are then calculated on the residuals. Hypothesis tests of the Pearson's correlation and the Spearman's correlation are performed using asymptotic t distributions. The p-value of the Kendall's correlation is calculated using a normal approximation. Results are provided in Table 4.3.

Pearson	Spearman	Kendall
0.3879 (0.0034)	0.3752 (0.0050)	0.2538 (0.0062)

Table 4.3: Correlation coefficients between cell-wise GLM residuals and their corresponding p-values

It can be observed that the correlation coefficients are quite strong and significant. This can also be compared with Kendall's correlation coefficient of 0.271 from log-normal marginal fitting in Zhang and Dukic (2013).

The strong positive correlation coefficients may come from some calendar year effects that have impact on both lines simultaneously. To further investigate this, another GLM analysis is performed with a mean structure that also includes fixed calendar year effects $h_t^{(n)}$

$$a_i^{(n)} + b_j^{(n)} + h_{t=i+j-1}^{(n)}. \tag{4.13}$$

The optimal power parameters for this particular mean structure are $p = 1.08$ for Personal Auto line and $p = 1.34$ for Commercial Auto line. Correlation coefficients between GLM Pearson residuals are given in Table 4.4. The correlation coefficients have reduced, however, they are still quite strong and significant at 5%.

Pearson	Spearman	Kendall
0.295 (0.0287)	0.3413 (0.0111)	0.2256 (0.0150)

Table 4.4: Correlation coefficients between cell-wise GLM residuals and their corresponding p-values after removing calendar year trend

Heat maps are also plotted for residuals from both lines of business. Residuals are calculated as the ratios of observed values to GLM fitted values. These are presented in Figure 4.3. An examination and comparison of these heat maps show some non-randomness in the residuals with similar patterns. The most obvious feature is that accident years 1 and 2 both exhibit behaviour different from that of other accident years. Specifically, accident year 1 is characterized by low payments in development years 4 and 5, with compensating payments acceleration in development years 6-10, and accident year 2 is characterized by low payments in development years 6-9. Consequently, a decision could be made whether these systematic deviations represent a signal that needs to be modelled or whether these are simply correlated noise. If these deviations were to be treated as a signal, an augmented model allowing these deviations as fixed effects could be considered. As a result, these additional features would

be allowed for in the forecast of outstanding claims. By allowing more fixed effects in the model, the correlation coefficients between lines of business could be significantly reduced. However, there is no obvious physical interpretation of these systematic deviations, hence it is equally possible that they are simply correlated noise. For illustration purposes, all variations are considered as correlated noise and the multivariate Tweedie framework is applied to this data set. In this application, the correlated noise is interpreted as being driven by cell-wise common shocks in the multivariate Tweedie framework. However, it is worth noting that there can be other possible modelling approaches to capture features represented in this data set.

Year	1	2	3	4	5	6	7	8	9	10
1	1.000	1.041	1.036	0.809	0.870	1.170	1.160	0.953	1.123	1.000
2	0.961	1.064	1.116	0.964	0.964	0.608	0.664	0.745	0.878	
3	0.907	1.023	1.048	1.186	0.820	1.351	0.444	1.340		
4	0.976	0.976	0.967	1.154	0.992	0.717	1.733			
5	1.037	0.996	0.915	0.938	1.106	1.205				
6	0.948	0.986	0.998	1.047	1.180					
7	1.091	0.948	1.020	0.887						
8	1.066	0.987	0.913							
9	1.021	0.979								
10	1.000									

Year	1	2	3	4	5	6	7	8	9	10
1	1.000	1.063	0.838	0.802	0.737	1.298	1.671	1.130	1.532	1.000
2	0.932	1.107	1.186	1.255	0.869	0.803	0.550	0.571	0.474	
3	1.018	1.027	0.949	0.989	0.885	1.149	0.896	1.322		
4	0.942	0.932	1.109	0.940	1.162	1.037	0.899			
5	1.067	0.854	1.083	0.835	1.373	0.743				
6	1.061	0.906	1.122	0.971	0.946					
7	1.030	0.975	0.835	1.194						
8	1.014	1.050	0.919							
9	0.953	1.043								
10	1.000									

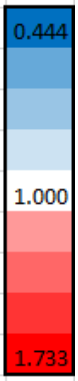

Figure 4.3: Heat maps of ratios of observed values to GLM fitted values (top: Personal Auto line, bottom: Commercial Auto line)

4.4.2 Bayesian inference and estimation results

	Median	SD	90% CI		Median	SD	90% CI
$\ddot{\alpha}_2^{(1)}$	1.0203	0.0928	(0.8727; 1.1816)	$\ddot{\alpha}_2^{(2)}$	1.1468	0.1200	(0.9694; 1.3567)
$\ddot{\alpha}_3^{(1)}$	0.9316	0.0844	(0.8023; 1.0796)	$\ddot{\alpha}_3^{(2)}$	1.1306	0.1176	(0.9581; 1.3390)
$\ddot{\alpha}_4^{(1)}$	1.0220	0.0947	(0.8685; 1.1841)	$\ddot{\alpha}_4^{(2)}$	0.8999	0.0966	(0.7598; 1.0727)
$\ddot{\alpha}_5^{(1)}$	1.0479	0.0979	(0.8966; 1.2208)	$\ddot{\alpha}_5^{(2)}$	1.0157	0.1066	(0.8564; 1.2036)
$\ddot{\alpha}_6^{(1)}$	1.1024	0.1021	(0.9458; 1.2837)	$\ddot{\alpha}_6^{(2)}$	1.1413	0.1262	(0.9546; 1.3704)
$\ddot{\alpha}_7^{(1)}$	1.0089	0.0979	(0.8578; 1.1802)	$\ddot{\alpha}_7^{(2)}$	1.3633	0.1456	(1.1518; 1.6222)
$\ddot{\alpha}_8^{(1)}$	1.0028	0.1009	(0.8515; 1.1808)	$\ddot{\alpha}_8^{(2)}$	1.4040	0.1634	(1.1638; 1.6974)
$\ddot{\alpha}_9^{(1)}$	1.2071	0.1323	(1.0050; 1.4402)	$\ddot{\alpha}_9^{(2)}$	1.5176	0.1921	(1.2371; 1.8635)
$\ddot{\alpha}_{10}^{(1)}$	1.1863	0.1679	(0.9340; 1.4898)	$\ddot{\alpha}_{10}^{(2)}$	1.5957	0.3030	(1.2073; 2.1689)
$\ddot{\beta}_1^{(1)}$	0.2334	0.0178	(0.2064; 0.2652)	$\ddot{\beta}_1^{(2)}$	0.1311	0.0116	(0.1123; 0.1503)
$\ddot{\beta}_2^{(1)}$	0.2369	0.0178	(0.2098; 0.2680)	$\ddot{\beta}_2^{(2)}$	0.1680	0.0142	(0.1448; 0.1914)
$\ddot{\beta}_3^{(1)}$	0.1343	0.0109	(0.1172; 0.1533)	$\ddot{\beta}_3^{(2)}$	0.1153	0.0109	(0.0975; 0.1335)
$\ddot{\beta}_4^{(1)}$	0.0779	0.0074	(0.0664; 0.0907)	$\ddot{\beta}_4^{(2)}$	0.0922	0.0092	(0.0776; 0.1077)
$\ddot{\beta}_5^{(1)}$	0.0405	0.0047	(0.0334; 0.0487)	$\ddot{\beta}_5^{(2)}$	0.0593	0.0071	(0.0485; 0.0720)
$\ddot{\beta}_6^{(1)}$	0.0186	0.0030	(0.0141; 0.0239)	$\ddot{\beta}_6^{(2)}$	0.0232	0.0040	(0.0171; 0.0303)
$\ddot{\beta}_7^{(1)}$	0.0066	0.0017	(0.0042; 0.0096)	$\ddot{\beta}_7^{(2)}$	0.0176	0.0034	(0.0124; 0.0236)
$\ddot{\beta}_8^{(1)}$	0.0043	0.0015	(0.0023; 0.0071)	$\ddot{\beta}_8^{(2)}$	0.0065	0.0021	(0.0036; 0.0105)
$\ddot{\beta}_9^{(1)}$	0.0039	0.0016	(0.0020; 0.0072)	$\ddot{\beta}_9^{(2)}$	0.0027	0.0013	(0.0014; 0.0054)
$\ddot{\beta}_{10}^{(1)}$	0.0005	0.0004	(0.0002; 0.0016)	$\ddot{\beta}_{10}^{(2)}$	0.0001	0.0001	(0.0001; 0.0003)
$\ddot{\phi}^{(1)}$	0.0058	0.0014	(0.0040; 0.0085)	$\ddot{\phi}^{(2)}$	0.0076	0.0019	(0.0054; 0.0114)
Λ	0.8127	0.6487	(0.0986; 2.1341)				

Table 4.5: Posterior statistics of parameters from marginal estimation

For the marginal estimation, 300,000 iterations are run and the first 150,000 iterations are discarded as a burn-in sample. We then thin the sample to reduce the sequential dependence between each iteration in use. The posterior statistics of parameters obtained from these draws are given in Table 4.5.

We then use the multivariate estimation procedure to estimate $\tilde{\alpha}$ and $\tilde{\phi}$. Estimates of $\tilde{\alpha}$ and $\tilde{\phi}$ are provided in Table 4.6.

	Median	SD	90% CI
$\tilde{\alpha}$	0.0041	0.0047	(0.0011; 0.0157)
$\tilde{\phi}$	0.0293	0.0197	(0.0117; 0.0731)

Table 4.6: Posterior statistics of parameters from multivariate estimation

4.4.3 Goodness-of-fit analysis and comparisons

We assess the marginal and multivariate goodness-of-fit. This is then compared with the goodness-of-fit of a common shock normal model ($p = 0$) and a common shock gamma model ($p = 2$) in Vu (2013).

Quantile-quantile (QQ) plots of Pearson residuals are used to assess the marginal goodness of fit. These are provided in Figure 4.4. These plots suggest that the multivariate Tweedie framework with $p = 1.32$ provides a good fit to the data.

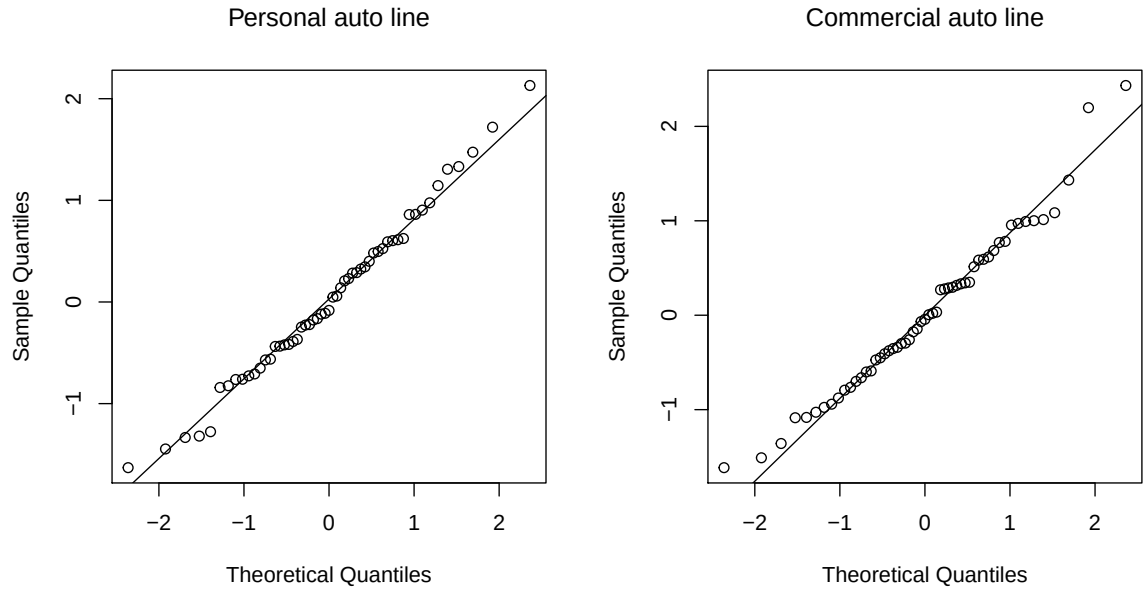

Figure 4.4: QQ plots of residuals from common shock Tweedie model ($p = 1.32$)

To assess the multivariate goodness of fit, a plot of empirical copula of observed values is compared with a plot empirical copula of one set of back-fitted values. These plots are given in Figure 4.5. This figure shows that the positive dependence structure in the data is captured by the model quite well.

A multivariate Gaussian model and a multivariate gamma are also fitted to the data set (which are special cases of the multivariate Tweedie framework with $p = 0$ and $p = 2$). The calibration of these models also follows the Bayesian procedure proposed for the general multivariate Tweedie framework. QQ plots are made for residuals from the two special cases, and provided in Figures 4.6 and 4.7. It can be observed that the marginal goodness-of-fit of the multivariate normal model and the multivariate gamma model is not as satisfactory as

that of the multivariate Tweedie framework, especially in the tails. This further indicates the advantage of having flexible power parameter p over restricting it to some special members of the Tweedie family.

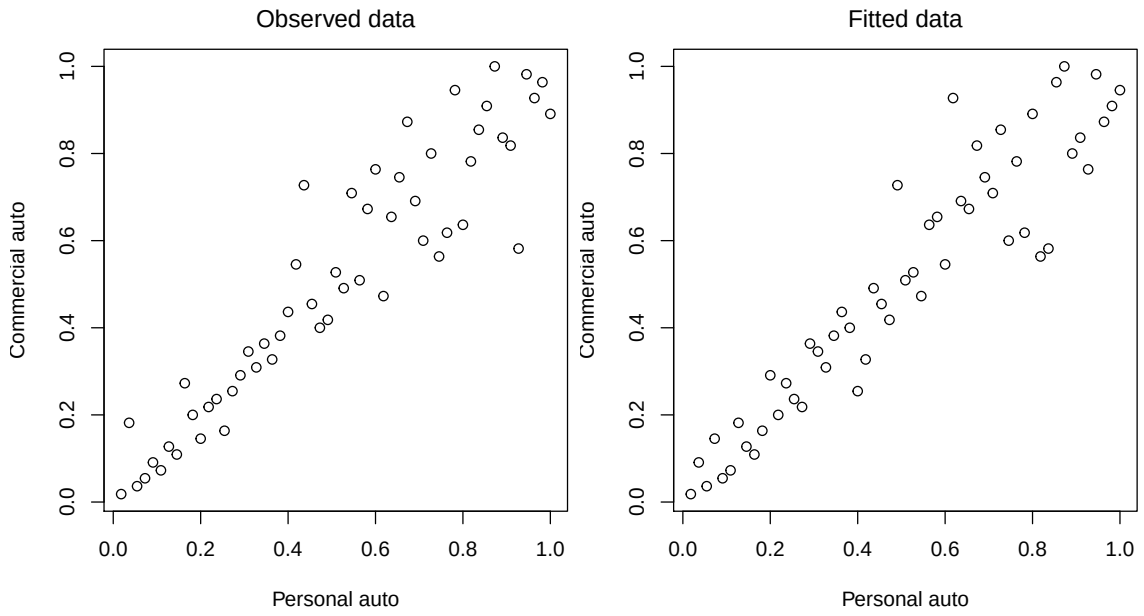

Figure 4.5: Plots of empirical copulas for observed values and back-fitted values from common shock Tweedie model ($p = 1.32$)

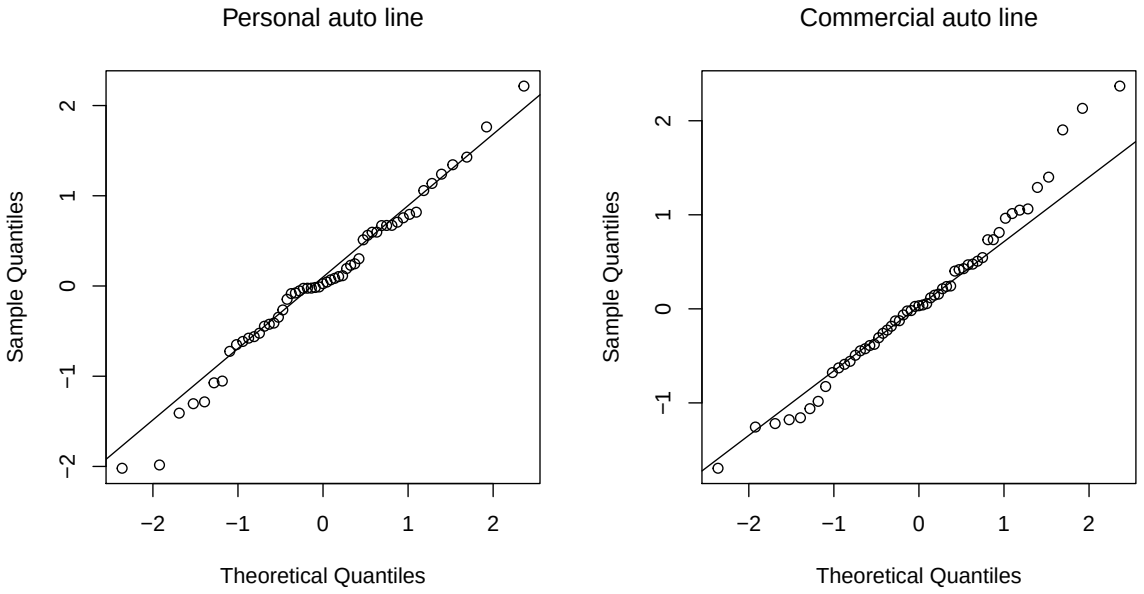

Figure 4.6: QQ plots of residuals from the common shock normal model ($p = 0$)

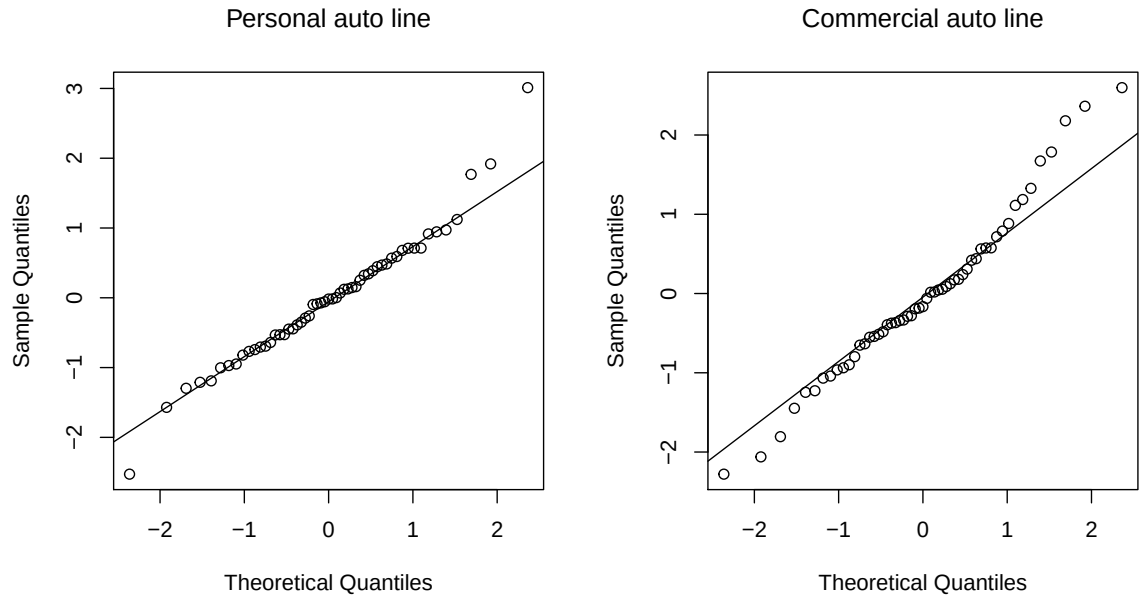

Figure 4.7: QQ plots of residuals from the common shock gamma model ($p = 2$)

Empirical copula plots are also provided for back-fitted values from the multivariate normal model and the multivariate gamma model in Figure 4.8. In comparison with empirical copula plots in Figure 4.5, it can be observed that the multivariate Tweedie framework provides a closer fit to the dependence structure present in this data set.

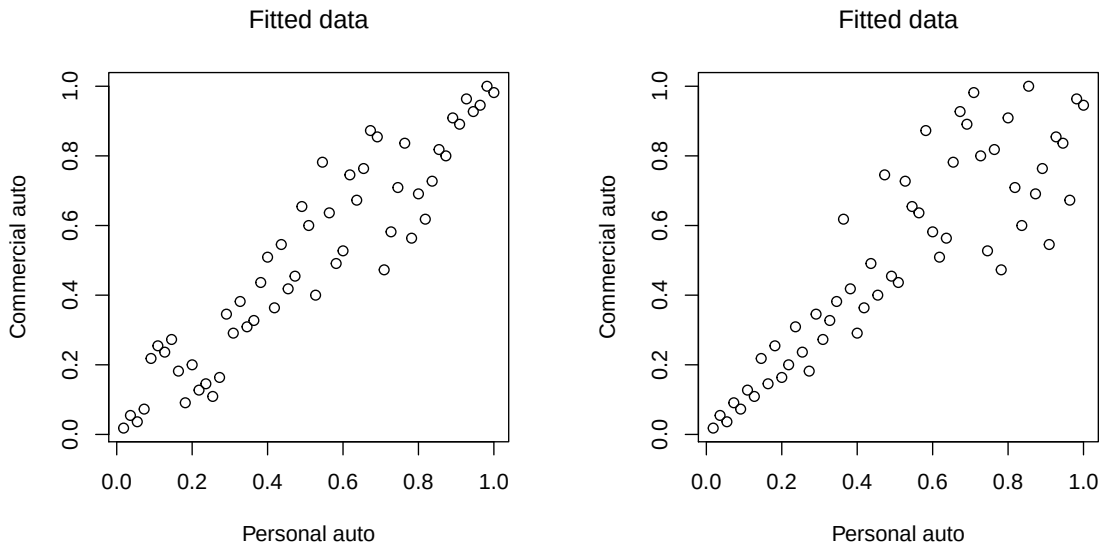

Figure 4.8: Plot of empirical copula of back-fitted values from the common shock normal model (left) and the common shock gamma model (right)

4.4.4 Outstanding claims forecast

Year	Personal Auto		Commercial Auto		Both lines	
	Mean	SD	Mean	SD	Mean	SD
2	67.44	59.64	20.21	21.30	87.65	70.72
3	361.23	186.29	209.11	127.79	570.35	244.59
4	747.07	288.92	472.89	202.02	1,219.96	386.17
5	1,417.86	430.06	1,441.71	429.71	2,859.57	662.39
6	3,555.91	821.73	3,301.36	774.04	6,857.27	1,224.53
7	8,306.58	1,536.38	9,095.14	1,644.76	17,401.72	2,375.35
8	16,000.76	2,489.37	16,304.02	2,509.34	32,304.78	3,678.74
9	27,541.14	3,714.87	23,708.28	3,463.91	51,249.42	5,190.42
10	45,677.21	6,623.84	34,340.32	6,547.81	80,017.53	9,562.36

Table 4.7: Outstanding claims statistics by accident period (numbers are in $1,000's)

Using the posterior parameter samples from the Bayesian inference, a predictive distribution of outstanding claims can be obtained. 30,000 sets of lower loss triangles are simulated using 30,000 posterior parameter samples. Table 4.7 summarises claims forecasts by accident period for each line of business and the total portfolio. Results represented include the mean and standard deviation of total claims from each accident period.

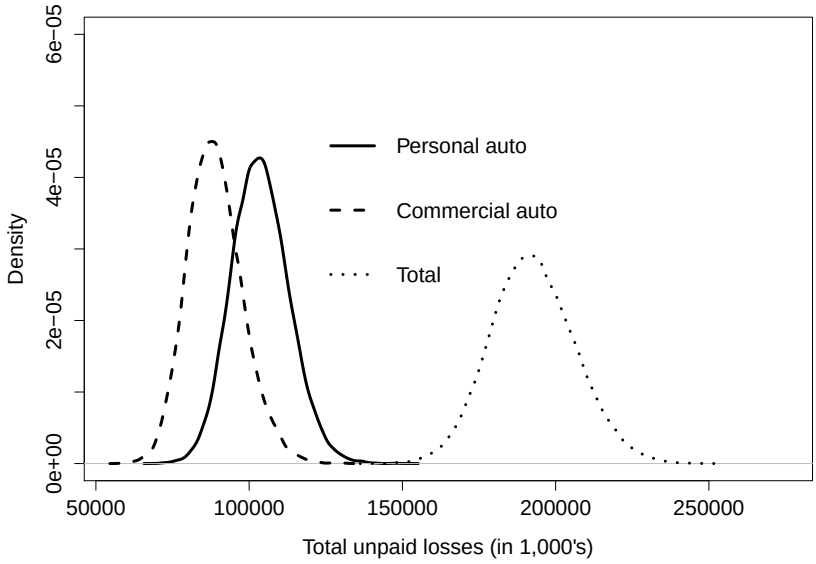

Figure 4.9: Kernel densities of predictive distributions of total outstanding claims in each line of business and in the aggregate portfolio (in $1,000's)

The predictive distributions of total outstanding claims for each line of business and the total portfolio can also be obtained. Their kernel densities are represented in Figure 4.9. Table 4.8 summarises estimation statistics of the projected total outstanding claims. Results represented include the mean, variance, $\text{VaR}_{75\%}$ and $\text{VaR}_{95\%}$ calculated directly from the MCMC samples. We can observe that the sum of the standard deviations of total claims from each line is larger than the standard deviation of the total claims portfolio. This suggests a diversification benefit due to the lack of a perfectively positive dependence structure between the two lines. This can be supported by the Pearson's correlation of 0.1212 between the total sum of claims from the two lines, obtained from pairs of total outstanding claims from each line in the set of 30,000 simulated lower loss triangles.

	Personal Auto	Commercial Auto	Both lines
Mean	103,675.21	88,893.05	192,568.26
SD	9,372.74	9,028.83	13,779.84
$\text{VaR}_{75\%}$	109,765.72	94,542.25	201,636.91
$\text{VaR}_{95\%}$	119,584.51	104,657.06	215,961.11

Table 4.8: Summary statistics of outstanding claims distributions (numbers are in \$1,000's)

In assessing diversification benefits, it is also useful to look at diversification benefits obtained for risk margins. Risk margins are often held by insurers as a regulatory requirement to provide some protection against unexpected volatility from the central estimates. We follow the definition of risk margins used in the regulatory system in Australia (also mentioned in Chapter 2) with a slight modification to also include a flexible quantile specification

$$\text{Risk margin}_{\chi\%}[Y] = \max\left\{\text{VaR}_{\chi\%}[Y] - E[Y];\ \frac{1}{2}SD[Y]\right\}. \tag{4.14}$$

Using results from Table 4.8, risk margins calculated using $\text{VaR}_{75\%}$ and $\text{VaR}_{95\%}$, as well as an indicator of diversification benefits are provided in Table 4.9. Diversification benefits are calculated as

$$\text{DB} = \frac{\big(\text{Risk margin}_{\chi\%}[Y_1] + \text{Risk margin}_{\chi\%}[Y_2]\big) - \text{Risk margin}_{\chi\%}[Y_1 + Y_2]}{\text{Risk margin}_{\chi\%}[Y_1] + \text{Risk margin}_{\chi\%}[Y_2]} \times 100\%. \tag{4.15}$$

This is to show the level of reduction in the total variation (indicated using the risk margins) that can be obtained as a result of the allowance of dependence across segments. Quite significant diversification benefits of approximately 22.8% and 26.1% can be observed for $\text{Risk margin}_{75\%}$ and $\text{Risk margin}_{95\%}$, respectively in this data set.

	Personal Auto	Commercial Auto	Both lines	DB
Risk margin$_{75\%}$	6,090.51	5,649.20	9,068.65	22.8%
Risk margin$_{95\%}$	15,909.30	15,764.01	23,392.85	26.1%

Table 4.9: Risk margin (in 1,000's) and diversification benefits statistics

The bell-shaped curves in Figure 4.9 suggest that a Gaussian distribution would fit the aggregate outstanding liabilities relatively well. However, the goodness of fit comparisons in Section 4.4.3 indicate otherwise. For comparison, loss reserves prediction using a multivariate normal distribution is still obtained. Summary statistics of outstanding liabilities from this model calibration are provided in Table 4.10. This can be compared with results from the multivariate Tweedie framework with $p = 1.32$ in Table 4.8.

The comparison indicates that even though the distributions of total outstanding claims have bell-shaped curves, an appropriate choice of marginal distribution at the cell level is still of importance, as it defines which distribution is maximised in order to get parameters. This changes the prediction results, including the means, standard deviations and quantiles of outstanding liabilities.

	Personal Auto	Commercial Auto	Both lines
Mean	101,548.13	84,966.69	186,514.82
SD	9,793.32	9,831.62	14,271.80
VaR$_{75\%}$	108,081.21	91,180.29	195,729.32
VaR$_{95\%}$	117,697.14	100,769.71	209,902.82

Table 4.10: Summary statistics of outstanding claims distributions from the multivariate normal model (numbers are in 1,000's)

Such a nicely shaped distribution as a final result is, in fact, not surprising. This is even in the case of heavily skewed individual cells. A similar example can be found in Taylor (2000, Chapter 11). One might very well be able to estimate (moderate) quantiles of the aggregate by means of a Gaussian approximation. However, this cannot be determined before all calculations are made and Figure 4.9 is produced.

4.4.5 The use of closed-form moments

To utilise the benefit of having closed-form moments and cumulants, one can also obtain prediction statistics including the mean and standard deviation using the results in Section

3.3 from Chapter 3. Recall from Section 3.3.3 that the mean and variance of the total sum of claims can be obtained in closed-form conditioning on known parameter values. That is, when there is a consideration of parameter uncertainty in a Bayesian framework, the mean and the variance of the sum of all claims given in Section 3.3.3 are random variables. In this Bayesian inference, parameter uncertainty also needs to be incorporated into the estimation. The mean and variance, allowing for parameter uncertainty, can be computed using the law of total expectation and the law of total variance

$$E\left[\sum Y_{i,j}^{(n)}\right] = E\left[E\left[\sum Y_{i,j}^{(n)}|\boldsymbol{\Theta},\tilde{\alpha}\right]\right], \tag{4.16}$$

$$Var\left[\sum Y_{i,j}^{(n)}\right] = Var\left[E\left[\sum Y_{i,j}^{(n)}|\boldsymbol{\Theta},\tilde{\alpha}\right]\right] + E\left[Var\left[\sum Y_{i,j}^{(n)}|\boldsymbol{\Theta},\tilde{\alpha}\right]\right], \tag{4.17}$$

where $E\left[\sum Y_{i,j}^{(n)}|\boldsymbol{\Theta},\tilde{\alpha}\right]$ and $Var\left[\sum Y_{i,j}^{(n)}|\boldsymbol{\Theta},\tilde{\alpha}\right]$ can be calculated using their analytical forms.

Being able to obtain the mean of outstanding claims in closed-form also allows us to compute quantiles of interest using the control variate method. This method is well-known for producing estimates with lower standard error, see for example, Glasserman (2003); Givens and Hoeting (2005); Kroese et al. (2011).

Table 4.11 summarises estimation statistics of total outstanding claims forecast. Results represented include the mean, variance, $\text{VaR}_{75\%}$ and $\text{VaR}_{95\%}$ calculated using the closed-form conditional mean and variance as mentioned above. VaR estimates computed using the control variate method are also provided. It can be observed that sample mean and variance estimates of total outstanding claims are very close to the estimates obtained using conditional mean and variance in analytical form. The sample $\text{VaR}_{75\%}$ and $\text{VaR}_{95\%}$ estimates are also very close to the corresponding control variate estimates.

	Personal Auto	Commercial Auto	Both lines
Mean	104,012.00	88,270.82	192,282.82
SD	9,845.48	8,559.48	13,971.95
$\text{VaR}_{75\%}$	110,218.22	93,815.78	201,360.63
$\text{VaR}_{95\%}$	120,540.94	102,878.18	215,699.59

Table 4.11: Summary statistics of outstanding claims distributions calculated using closed-form moments (numbers are in 1,000's)

The MCMC used produces very low standard errors and uses minimal computational time. However, in cases where MCMC cannot provide adequate accuracy or efficiency, one could further utilise the benefit of having closed-form cumulants to obtain prediction statistics

including the mean and the standard deviation of the sum of outstanding claims.

4.5 Remarks on applications of the framework

In this chapter, Bayesian inference is used to develop an estimation procedure for the multivariate Tweedie framework. The following remarks on practical model properties are drawn from the analysis and results of illustrations in the previous sections.

4.5.1 Marginal flexibility and closure under the taking of marginals

This multivariate Tweedie framework provides a framework for modelling outstanding claims with a flexible selection of power parameter p. This can improve model practicality as it can be applied to a larger variety of data sets. As seen in the illustration with real data, this framework allows the optimal estimate of p to be used, which can provide a much better marginal and multivariate goodness-of-fit compared to multivariate models with specific marginal choices such as multivariate normal models. However, it is also worth noting that a limitation of the framework comes from the requirement of using the same power parameter p as well as the same common shock for all business segments.

4.5.2 Ability to handle masses at 0

The Tweedie family is a rich family that consists of Tweedie's compound Poison-gamma distributions. These are well-known distributions with the ability to handle masses at 0. This gives the multivariate Tweedie framework a significant benefit over models with other marginal choices, such as gamma distributions. Simulation illustration 2 in Section 4.3 further confirms that the estimation procedure does not have any issue when it is applied on data sets that contain 0's.

4.5.3 Closed-form moments

The proposed framework has an advantage of having moments in closed-form. However, due to the complexity of the marginal and multivariate Tweedie densities, Bayesian inference with an MCMC procedure is used for model estimation. Parameter error is also incorporated

naturally in the results through the specification of prior distributions in the Bayesian inference. This comes in with a cost where moments (allowing for parameter error) are no longer obtained in analytical form. However, the benefit of having theoretical moments in closed form can still be exploited to an extent to obtain prediction statistics of the sum of outstanding claims. In particular, the mean and the variance of the sum of all claims given in Section 3.3.3 are random variables conditional on parameters. The unconditional mean and variance can be computed using the law of total expectation and the law of total variance. The control variate method can be used to calculate the unconditional quantiles. The use of these calculations can enhance the efficiency of MCMC, especially when it is too computationally expensive.

4.5.4 Dependence structure

A cell-wise dependence structure is proposed following a branch of literature, as also mentioned in Chapter 3. While this dependence structure cannot be interpreted using systematic factors such as calendar year dependence or accident year dependence, it can be used to capture simply correlated noise, as also demonstrated in Section 4.4.1.2. In such cases, the common shocks in the multivariate Tweedie framework are drivers of the cell-wise correlated noise across triangles. It is also worth noting that correlation coefficients, measures used to capture dependence, are dependent of the models used as raised in Avanzi, Taylor and Wong (2016). In the events where calendar year factors, or accident year factors affect claims from multiple segments simultaneously, these events can be modelled exclusively using data segmentation or deterministic effects. The dependence structure in the remainder of the data can then be modelled using the proposed cell-wise Tweedie framework if appropriate. Nevertheless, future research could explore other dependence structures such as calendar year dependence using the proposed framework to utilise its many advantages.

4.A Appendices

4.A.1 Simulated data sets and estimation results

Year	1	2	3	4	5	6	7	8	9	10
1	0.240151	0.274318	0.094119	0.085875	0.031807	0.013806	0.007190	0.009306	0.001866	0.002008
2	0.253992	0.251274	0.138049	0.068166	0.054149	0.018325	0.009745	0.006340	0.002740	
3	0.234966	0.283883	0.125763	0.069204	0.033658	0.014891	0.004963	0.004028		
4	0.225352	0.244446	0.107632	0.079135	0.045705	0.020370	0.003769			
5	0.218054	0.223552	0.154835	0.087825	0.039442	0.014322				
6	0.302805	0.275805	0.143042	0.085948	0.043431					
7	0.239969	0.296280	0.129593	0.090236						
8	0.200759	0.221279	0.157823							
9	0.291852	0.243345								
10	0.244305									

Table 4.12: Simulated loss triangle 1 (data set 1)

Year	1	2	3	4	5	6	7	8	9	10
1	0.147347	0.196444	0.103671	0.092007	0.047120	0.020597	0.016839	0.005074	0.003454	0.000225
2	0.138343	0.200295	0.112917	0.094464	0.075802	0.019854	0.022214	0.004711	0.008808	
3	0.138050	0.184805	0.145138	0.100257	0.053106	0.028627	0.043037	0.002824		
4	0.099135	0.119242	0.104156	0.108336	0.032670	0.028404	0.025760			
5	0.124336	0.217746	0.137297	0.091600	0.064107	0.028242				
6	0.148453	0.155605	0.131527	0.134106	0.060104					
7	0.187909	0.246437	0.149696	0.116582						
8	0.194526	0.233601	0.172806							
9	0.211970	0.280598								
10	0.213647									

Table 4.13: Simulated loss triangle 2 (data set 1)

	True value	Median	SD	90% CI		True value	Median	SD	90% CI
$\ddot{\alpha}_2^{(1)}$	1.0362	1.0814	0.0875	(0.9460; 1.2351)	$\ddot{\alpha}_2^{(2)}$	1.1957	1.0996	0.1004	(0.9463; 1.2765)
$\ddot{\alpha}_3^{(1)}$	0.9446	1.0024	0.0857	(0.8653; 1.1473)	$\ddot{\alpha}_3^{(2)}$	1.1738	1.1498	0.1020	(0.9955; 1.3311)
$\ddot{\alpha}_4^{(1)}$	1.0263	0.9828	0.0833	(0.8508; 1.1201)	$\ddot{\alpha}_4^{(2)}$	0.9506	0.8706	0.0833	(0.7416; 1.0146)
$\ddot{\alpha}_5^{(1)}$	1.0580	1.0111	0.0828	(0.8803; 1.1504)	$\ddot{\alpha}_5^{(2)}$	1.0549	1.1370	0.1077	(0.9764; 1.3330)
$\ddot{\alpha}_6^{(1)}$	1.1140	1.1671	0.0974	(1.0164; 1.3385)	$\ddot{\alpha}_6^{(2)}$	1.1873	1.1495	0.1099	(0.9755; 1.3402)
$\ddot{\alpha}_7^{(1)}$	1.0215	1.1116	0.0961	(0.9650; 1.2783)	$\ddot{\alpha}_7^{(2)}$	1.4094	1.3904	0.1309	(1.1910; 1.6181)
$\ddot{\alpha}_8^{(1)}$	1.0198	0.9772	0.0895	(0.8441; 1.1343)	$\ddot{\alpha}_8^{(2)}$	1.4505	1.4697	0.1492	(1.2543; 1.7436)
$\ddot{\alpha}_9^{(1)}$	1.2193	1.1174	0.1190	(0.9329; 1.3257)	$\ddot{\alpha}_9^{(2)}$	1.5616	1.6773	0.1783	(1.4039; 1.9882)
$\ddot{\alpha}_{10}^{(1)}$	1.1992	1.0530	0.1428	(0.8416; 1.3095)	$\ddot{\alpha}_{10}^{(2)}$	1.6646	1.6936	0.2721	(1.3143; 2.1981)
$\ddot{\beta}_1^{(1)}$	0.2343	0.2308	0.0158	(0.2071; 0.2584)	$\ddot{\beta}_1^{(2)}$	0.1295	0.1250	0.0095	(0.1096; 0.1407)
$\ddot{\beta}_2^{(1)}$	0.2376	0.2426	0.0165	(0.2178; 0.2706)	$\ddot{\beta}_2^{(2)}$	0.1651	0.1658	0.0122	(0.1464; 0.1871)
$\ddot{\beta}_3^{(1)}$	0.1349	0.1245	0.0094	(0.1108; 0.1412)	$\ddot{\beta}_3^{(2)}$	0.1133	0.1123	0.0090	(0.0980; 0.1276)
$\ddot{\beta}_4^{(1)}$	0.0788	0.0757	0.0063	(0.0662; 0.0871)	$\ddot{\beta}_4^{(2)}$	0.0910	0.0934	0.0079	(0.0803; 0.1064)
$\ddot{\beta}_5^{(1)}$	0.0412	0.0388	0.0040	(0.0326; 0.0457)	$\ddot{\beta}_5^{(2)}$	0.0589	0.0504	0.0053	(0.0422; 0.0594)
$\ddot{\beta}_6^{(1)}$	0.0190	0.0152	0.0022	(0.0120; 0.0192)	$\ddot{\beta}_6^{(2)}$	0.0234	0.0227	0.0032	(0.0181; 0.0287)
$\ddot{\beta}_7^{(1)}$	0.0072	0.0058	0.0013	(0.0039; 0.0081)	$\ddot{\beta}_7^{(2)}$	0.0195	0.0246	0.0030	(0.0192; 0.0289)
$\ddot{\beta}_8^{(1)}$	0.0051	0.0057	0.0014	(0.0036; 0.0082)	$\ddot{\beta}_8^{(2)}$	0.0052	0.0035	0.0009	(0.0026; 0.0055)
$\ddot{\beta}_9^{(1)}$	0.0034	0.0022	0.0007	(0.0016; 0.0037)	$\ddot{\beta}_9^{(2)}$	0.0032	0.0052	0.0016	(0.0030; 0.0084)
$\ddot{\beta}_{10}^{(1)}$	0.0009	0.0016	0.0010	(0.0007; 0.0039)	$\ddot{\beta}_{10}^{(2)}$	0.0001	0.0002	0.0001	(0.0001; 0.0004)
$\ddot{\phi}^{(1)}$	0.0049	0.0052	0.0014	(0.0037; 0.0080)	$\ddot{\phi}^{(2)}$	0.0062	0.0061	0.0015	(0.0042; 0.0092)
Λ	0.5700	0.5170	0.5891	(0.0719; 1.9067)					
$\tilde{\alpha}$	0.0060	0.0042	0.0047	(0.0011; 0.0160)	$\tilde{\phi}$	0.0664	0.0578	0.0361	(0.0242; 0.1372)

Table 4.14: Posterior statistics of parameters (data set 1)

Year	1	2	3	4	5	6	7	8	9	10
1	0.283732	0.226835	0.164292	0.089160	0.029220	0.015849	0.013507	0.011700	0.003694	0.007537
2	0.357937	0.251767	0.177964	0.078834	0.051595	0.029831	0.001342	0.009320	0.009119	
3	0.203430	0.172260	0.109408	0.084336	0.027601	0.008246	0.014475	0.007355		
4	0.223365	0.227374	0.153072	0.054501	0.049119	0.038225	0			
5	0.279627	0.201581	0.191577	0.113466	0.059034	0.023083				
6	0.196836	0.323561	0.199684	0.113159	0.065494					
7	0.214060	0.110836	0.177897	0.098204						
8	0.253372	0.184663	0.149160							
9	0.213116	0.320763								
10	0.210801									

Table 4.15: Simulated loss triangle 1 (data set 2)

Year	1	2	3	4	5	6	7	8	9	10
1	0.104007	0.227287	0.129769	0.108286	0.069063	0.039293	0.025876	0.010197	0.003272	0.001153
2	0.157652	0.220918	0.174850	0.104639	0.092709	0.029221	0.020151	0.003365	0.004267	
3	0.144728	0.199293	0.139864	0.118673	0.084007	0.033882	0.039586	0.007387		
4	0.113809	0.138278	0.080297	0.080579	0.086192	0.029228	0.023066			
5	0.110246	0.191452	0.116688	0.145530	0.064009	0.032577				
6	0.126303	0.178031	0.146857	0.145787	0.071391					
7	0.247466	0.268015	0.195683	0.204113						
8	0.208114	0.202835	0.167232							
9	0.131089	0.309236								
10	0.306811									

Table 4.16: Simulated loss triangle 2 (data set 2)

	True value	Median	SD	90% CI		True value	Median	SD	90% CI
$\ddot{\alpha}_2^{(1)}$	1.0362	1.1948	0.2014	(0.9195; 1.5760)	$\ddot{\alpha}_2^{(2)}$	1.1956	1.1364	0.1249	(0.9482; 1.3529)
$\ddot{\alpha}_3^{(1)}$	0.9445	0.7882	0.1294	(0.6380; 1.0514)	$\ddot{\alpha}_3^{(2)}$	1.1737	1.1188	0.1237	(0.9299; 1.3358)
$\ddot{\alpha}_4^{(1)}$	1.0262	0.9545	0.1602	(0.7222; 1.2449)	$\ddot{\alpha}_4^{(2)}$	0.9506	0.8099	0.0901	(0.6969; 0.9858)
$\ddot{\alpha}_5^{(1)}$	1.0579	1.1306	0.1930	(0.8649; 1.4904)	$\ddot{\alpha}_5^{(2)}$	1.0549	1.0010	0.1230	(0.8183; 1.2216)
$\ddot{\alpha}_6^{(1)}$	1.1140	1.2226	0.2040	(0.9405; 1.6071)	$\ddot{\alpha}_6^{(2)}$	1.1873	1.0788	0.1296	(0.8713; 1.3015)
$\ddot{\alpha}_7^{(1)}$	1.0214	0.8522	0.1570	(0.6605; 1.1706)	$\ddot{\alpha}_7^{(2)}$	1.4093	1.6650	0.1628	(1.3958; 1.9331)
$\ddot{\alpha}_8^{(1)}$	1.0197	0.9316	0.1761	(0.6890; 1.2700)	$\ddot{\alpha}_8^{(2)}$	1.4504	1.3178	0.1778	(1.0435; 1.6355)
$\ddot{\alpha}_9^{(1)}$	1.2193	1.1846	0.2366	(0.8652; 1.6185)	$\ddot{\alpha}_9^{(2)}$	1.5616	1.3745	0.2127	(1.0434; 1.7579)
$\ddot{\alpha}_{10}^{(1)}$	1.1991	0.9273	0.2526	(0.6534; 1.4578)	$\ddot{\alpha}_{10}^{(2)}$	1.6645	1.8487	0.1497	(1.5154; 1.9944)
$\ddot{\beta}_1^{(1)}$	0.2342	0.2238	0.0280	(0.1818; 0.2753)	$\ddot{\beta}_1^{(2)}$	0.1295	0.1282	0.0100	(0.1111; 0.1440)
$\ddot{\beta}_2^{(1)}$	0.2375	0.2051	0.0268	(0.1620; 0.2501)	$\ddot{\beta}_2^{(2)}$	0.1651	0.1782	0.0155	(0.1545; 0.2059)
$\ddot{\beta}_3^{(1)}$	0.1348	0.1520	0.0219	(0.1177; 0.1896)	$\ddot{\beta}_3^{(2)}$	0.1132	0.1204	0.0127	(0.1015; 0.1431)
$\ddot{\beta}_4^{(1)}$	0.0787	0.0788	0.0141	(0.0583; 0.1050)	$\ddot{\beta}_4^{(2)}$	0.0909	0.1035	0.0061	(0.0903; 0.1098)
$\ddot{\beta}_5^{(1)}$	0.0412	0.0370	0.0092	(0.0238; 0.0539)	$\ddot{\beta}_5^{(2)}$	0.0588	0.0711	0.0094	(0.0573; 0.0881)
$\ddot{\beta}_6^{(1)}$	0.0189	0.0165	0.0061	(0.0089; 0.0285)	$\ddot{\beta}_6^{(2)}$	0.0234	0.0279	0.0059	(0.0195; 0.0389)
$\ddot{\beta}_7^{(1)}$	0.0071	0.0045	0.0024	(0.0027; 0.0103)	$\ddot{\beta}_7^{(2)}$	0.0195	0.0223	0.0058	(0.0144; 0.0336)
$\ddot{\beta}_8^{(1)}$	0.0050	0.0049	0.0030	(0.0027; 0.0123)	$\ddot{\beta}_8^{(2)}$	0.0051	0.0044	0.0020	(0.0027; 0.0087)
$\ddot{\beta}_9^{(1)}$	0.0034	0.0041	0.0028	(0.0022; 0.0111)	$\ddot{\beta}_9^{(2)}$	0.0032	0.0026	0.0014	(0.0016; 0.0060)
$\ddot{\beta}_{10}^{(1)}$	0.0009	0.0023	0.0038	(0.0004; 0.0122)	$\ddot{\beta}_{10}^{(2)}$	0.0002	0.0004	0.0007	(0.0001; 0.0023)
$\ddot{\phi}^{(1)}$	0.0200	0.0201	0.0022	(0.0185; 0.0250)	$\ddot{\phi}^{(2)}$	0.0120	0.0110	0.0012	(0.0101; 0.0139)
Λ	0.7994	0.8264	0.4250	(0.1224; 1.5379)					
$\tilde{\alpha}$	0.0360	0.0794	0.2445	(0.0087; 0.7758)	$\tilde{\phi}$	0.1000	0.1764	0.3080	(0.0327; 0.9978)

Table 4.17: Posterior statistics of parameters (data set 2)

4.A.2 Empirical data set

This data set is drawn from Zhang and Dukic (2013).

Year	Premium	1	2	3	4	5	6	7	8	9	10
1	62,467	16,864	15,508	9,341	3,537	1,853	1,184	500	308	338	50
2	59,821	14,528	17,727	8,747	4,149	2,252	715	325	261	255	
3	62,968	14,241	13,763	7,512	5,207	2,068	1,674	219	421		
4	64,453	14,765	14,323	8,426	6,513	3,144	1,067	913			
5	71,185	16,395	17,038	9,826	6,381	4,037	1,839				
6	82,793	18,136	21,582	13,415	8,519	4,583					
7	100,826	24,727	24,037	15,181	7,105						
8	98,358	24,749	24,501	11,830							
9	76,653	23,063	21,035								
10	71,326	20,083									

Table 4.18: Personal Auto line (in 1,000s)

Year	Premium	1	2	3	4	5	6	7	8	9	10
1	42,847	5,407	9,015	4,641	3,384	1,695	1,262	1,425	373	241	6
2	38,829	6,279	8,725	6,172	4,494	2,110	919	447	202	69	
3	43,001	7,256	8,667	4,778	4,262	2,884	1,427	889	493		
4	41,840	5,028	5,317	4,697	3,795	2,871	1,100	657			
5	44,525	5,721	6,097	6,389	3,802	4,306	862				
6	50,923	7,413	9,385	7,772	5,850	3,383					
7	56,601	10,868	12,337	7,966	8,531						
8	54,609	10,143	14,193	8,070							
9	47,204	9,596	12,235								
10	42,412	9,076									

Table 4.19: Commercial Auto line (in 1,000s)

CHAPTER 5

Unbalanced data and common shock models[2]

5.1 Introduction

In loss reserving data, it is often observed that claim activity typically declines towards the right side of a triangle. In addition, different business segments can also have different claim activities, or claims development patterns where some are longer-tailed than others. In brief, a significant variation can be observed across loss cells within a single loss triangle, as well as across multiple triangles. For an illustration, we provide in Figure 5.1 the plots of incremental loss ratios for the accident year 2003 from the Bodily Injury line and the Accident Benefits line of a Canadian insurance company. This data set is available in Côté et al. (2016). Loss ratios are incremental claims standardised using the total premium earned for the corresponding accident period (see also Chapters 2 and 4). The variation across loss ratios within a single line and across lines is quite evident in this figure. Due to this feature, reserving data can be referred to as "unbalanced data".

Common shock approaches are useful dependence modelling tools with many benefits as also mentioned in Chapters 1, 2 and 3. A common shock model, however, can create problems in the absence of careful modelling. Because loss reserving data typically has an unbalanced nature, when a common shock model is applied to multiple claim cells within a triangle and/or across triangles, it may be desirable to ensure that the magnitude of the common shock does not contribute disproportionately to the total observation. In brief, one

[2] An abbreviated version of results in Chapter 5 has been submitted and is under review in *Avanzi, B., Taylor, G., Vu, P.A., Wong, B., 2018. On unbalanced data and common shock models in stochastic loss reserving.*

may wish to confine the relation of the common shock to total observations over the entire range of the triangles. A solution to such problem is to introduce parameters which specify the magnitude of the common shock proportion relative to the observation in each claim cell. This however can create over-fitting problems for a loss reserving portfolio which is often of a very small sample size. Hence there is a need to find a parsimonious approach that allows common shock models to be applied more appropriately to unbalanced data. This is the motivation for this chapter.

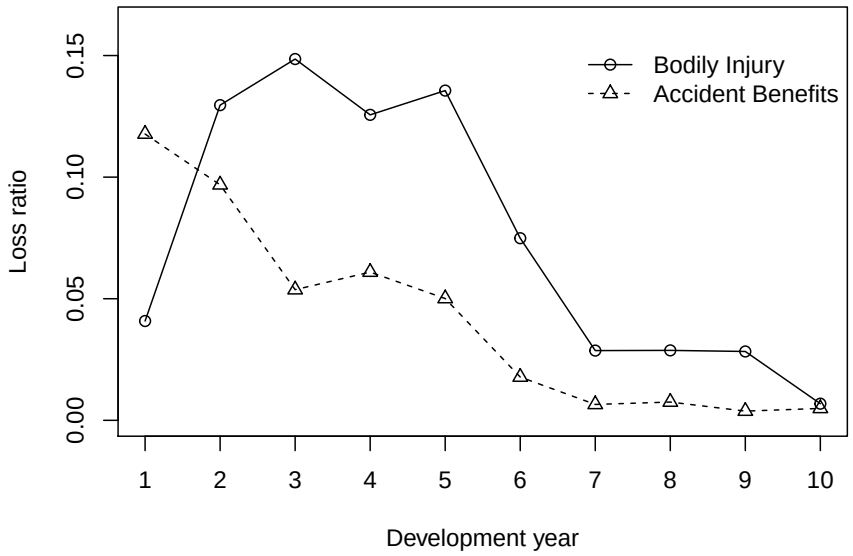

Figure 5.1: Plots of loss ratios for accident year 2003 of a Canadian insurance company

Section 5.2 investigates issues stemming from unbalanced data for common shock models. A solution to this issue is introduced in Section 5.3 which is used to modify the multivariate Tweedie framework developed in Chapters 3 and 4 for unbalanced data. Simulation illustrations are provided in Section 5.4, including an illustration with unbalanced data, and a comparison of the performances of the previously developed common shock Tweedie framework and the modified Tweedie framework on unbalanced data. An illustration using real data is provided in Section 5.5. Section 5.6 gives remarks on the performance of the proposed approach on unbalanced data.

5.2 Challenges for common shock models

In this section we examine some issues that one needs to be aware of when applying common shock models in the presence of unbalanced reserving data. These include the need

to balance common shock contributions to total observations over the entire range of the triangles, maintain model parsimony and preserve distributional tractability in some cases.

Many multivariate models in the literature with different types of dependence can be generalised using the common shock framework in Avanzi, Taylor and Wong (2018) (see also the review in Section 2.4.4). In this framework, incremental claim cells $Y_{i,j}^{(n)}$ are specified such that

$$Y_{i,j}^{(n)} = {}_{U}\lambda_{i,j}^{(n)} \cdot U_{\boldsymbol{\pi}_{(i,j)}} + {}_{\tilde{U}}\lambda_{i,j}^{(n)} \cdot \tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}} + Z_{i,j}^{(n)}, \tag{5.1}$$

where $U_{\boldsymbol{\pi}_{(i,j)}}, \tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}, Z_{i,j}^{(n)}$ are independent stochastic variates, representing the common shock for set $\boldsymbol{\pi}_{(i,j)}$ of claims from all business segments, the common shock for set $\boldsymbol{\pi}_{(i,j)}^{(n)}$ of claims from business segment (n) and the idiosyncratic component, respectively. Set $\boldsymbol{\pi}_{(i,j)}$ is the subset of claims that the common shock $U_{\boldsymbol{\pi}_{(i,j)}}$ has effects on. For example, $\boldsymbol{\pi}_{(i,j)} = \{Y_{i,1}^{(1)}, ..., Y_{i,I-i+1}^{(1)}\}$ gives a set of all claims from accident period i. In this case the common shock $U_{\boldsymbol{\pi}_{(i,j)}}$ introduces accident period dependence across business segments. Similarly, if $\boldsymbol{\pi}_{(i,j)}^{(n)} = \{Y_{1,j}^{(n)}, ..., Y_{I,j}^{(n)}\}$, the common shock term $\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}$ captures development period dependence within business segment n. Scaling factors, denoted by ${}_{U}\lambda_{i,j}^{(n)}$, ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$, scale the effects of common shocks so that they contribute proportionately to the total observation $Y_{i,j}^{(n)}$.

5.2.1 Balancing common shock proportions in loss triangles

For illustration, we use a real data set from a Canadian insurance company provided in Côté et al. (2016). This data set contains losses from 6 segments of business over the period 2003-2012. The two loss triangles chosen for illustration of the model are from Auto Insurance in Ontario. One triangle is for Bodily Injury coverage (denoted by (1)), and the other is for Accident Benefits excluding disability income (denoted by (2)). Incremental losses are given in Table 5.15 and 5.16 in Appendix 5.A.3. Claims are standardised using premium earned in the corresponding accident years.

5.2.1.1 Variation in claim activity within a single triangle

Within a single loss triangle, we can observe quite significant variation in claim observations within various dimensions. Variations can occur across accident periods (i.e.

within the same column in a loss triangle) due to changes in business volume, however this variation can often be removed by standardising claims using a common unit of exposure. We show in Figure 5.2 the heat map of incremental loss ratios from the Bodily Injury line. The loss ratios are observations standardised using the total premium earned in each accident period, hence the variation across rows is relatively small. However, significant variation is observed across development periods (i.e. within the same row in the loss triangle). As observed in Figure 5.2, more claims are paid within periods 2-5, and less in the first period as well as the last few periods.

Year	1	2	3	4	5	6	7	8	9	10
1	0.041	0.130	0.149	0.126	0.136	0.075	0.029	0.029	0.028	0.007
2	0.012	0.118	0.079	0.112	0.114	0.046	0.029	0.007	0.003	
3	0.014	0.090	0.143	0.085	0.085	0.061	0.029	0.002		
4	0.011	0.098	0.102	0.089	0.086	0.057	0.042			
5	0.016	0.109	0.105	0.113	0.051	0.037				
6	0.014	0.094	0.135	0.129	0.089					
7	0.011	0.141	0.114	0.127						
8	0.011	0.130	0.107							
9	0.008	0.071								
10	0.005									

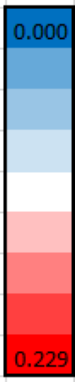

Figure 5.2: Heat map of loss ratios - Bodily Injury line

Consider a special case of the general common shock framework in Equation (5.1) with only dependence within a segment (i.e. $U_{\boldsymbol{\pi}_{(i,j)}} = 0$),

$$Y_{i,j}^{(n)} = {}_{\tilde{U}}\lambda_{i,j}^{(n)} \cdot \tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}} + Z_{i,j}^{(n)}. \tag{5.2}$$

Consequently, the contribution of the common shock to the total expected observation is

$$\frac{{}_{\tilde{U}}\lambda_{i,j}^{(n)} E\left[\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}\right]}{{}_{\tilde{U}}\lambda_{i,j}^{(n)} E\left[\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}\right] + E\left[Z_{i,j}^{(n)}\right]}. \tag{5.3}$$

If we set ${}_{\tilde{U}}\lambda_{i,j}^{(n)} = 1$, setting $E\left[\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}\right]$ such that it is comparable to a large cell, $E\left[Z_{i,j}^{(n)}\right]$ will dominate smaller cells (with smaller $E\left[Z_{i,j}^{(n)}\right]$) with common shock, whereas a comparability to a small cell will diminish the relative effects of common shock in larger cells (with large $E\left[Z_{i,j}^{(n)}\right]$).

The above simplification of the scale factor may not create a major issue for development period dependence (i.e. $\boldsymbol{\pi}^{(n)}_{(i,j)} = \{Y^{(n)}_{1,j}, ..., Y^{(n)}_{I,j}\}$) because claims within a development lag (across accident periods) often have similar sizes. However, the above issue can be quite noticeable for accident period dependence and calendar period dependence (i.e. $\boldsymbol{\pi}^{(n)}_{(i,j)} = \{Y^{(n)}_{i,1}, ..., Y^{(n)}_{i,J-i+1}\}$ and $\boldsymbol{\pi}^{(n)}_{(i,j)} = \{Y^{(n)}_{1,t}, ..., Y^{(n)}_{I,t-i+1}\}$) because claim observations within either of these dimensions can vary quite significantly, as also observed in the heat map in Figure 5.2.

Consider an illustrative example of accident period dependence where the expected value of common shock in each row is set to 5% of the total actual observation in the first development period of the corresponding row

$$E\left[\tilde{U}_{\boldsymbol{\pi}^{(n)}_{(i,j)}}\right] = 5\% \times E[Y^{(n)}_{i,1}]. \tag{5.4}$$

The contributions of common shocks in all cells within the loss triangle are shown using a heat map in Figure 5.3. The observed colour pattern is somewhat opposite to that in Figure 5.2. Without scaling factors (i.e. ${}_{\tilde{U}}\lambda^{(n)}_{i,j} = 1$), common shock proportions are significantly understated in the middle region of the triangle where claim observations are high and overstated in the tail region where observations are low.

Year	1	2	3	4	5	6	7	8	9	10
1	5.0%	1.6%	1.4%	1.6%	1.5%	2.7%	7.1%	7.1%	7.2%	29.9%
2	5.0%	0.5%	0.8%	0.5%	0.5%	1.3%	2.0%	8.0%	19.9%	
3	5.0%	0.8%	0.5%	0.8%	0.8%	1.2%	2.5%	34.4%		
4	5.0%	0.6%	0.5%	0.6%	0.6%	1.0%	1.3%			
5	5.0%	0.7%	0.7%	0.7%	1.5%	2.1%				
6	5.0%	0.7%	0.5%	0.5%	0.8%					
7	5.0%	0.4%	0.5%	0.4%						
8	5.0%	0.4%	0.5%							
9	5.0%	0.5%								
10	5.0%									

Figure 5.3: Heat map of common shock proportions - Bodily Injury line

5.2.1.2 Variation in claim activity across triangles

Variation in claim activity can also be observed in a portfolio of dependent segments which have various tail lengths. One often does not expect dependence across such lines, for

example, an Auto Property Damage line is often independent of a Workers Compensation line. However, lines with different tail lengths can still have some association. One of such examples is a portfolio of Auto Property Damage line and Auto Bodily Injury line in Australia. These two lines are usually dependent due to their overlap in insured events. One also often observes that the Auto Bodily Injury line is much longer-tailed than the Auto Property Damage line. Another example is the two business lines Bodily Injury and Accident Benefits in the data set in Côté et al. (2016) that are being considered for illustration. The heat map of loss ratios for the Accident Benefit line is provided in Figure 5.4. As observed from Figure 5.2 and Figure 5.4, the Accident Benefit line is shorter tailed than the Bodily Injury line as the majority of claims are paid within the first 3-4 years and much less are paid in the last 4-5 years. In addition to the variation across development periods within a single triangle, variation can also be observed for observations within the same development period across triangles.

Year	1	2	3	4	5	6	7	8	9	10
1	0.118	0.097	0.054	0.061	0.050	0.018	0.007	0.008	0.004	0.005
2	0.062	0.087	0.074	0.040	0.030	0.007	0.013	0.007	0.0004	
3	0.074	0.132	0.100	0.049	0.058	0.006	0.015	0.001		
4	0.067	0.105	0.070	0.060	0.039	0.025	0.011			
5	0.099	0.128	0.072	0.042	0.019	0.030				
6	0.087	0.172	0.115	0.049	0.033					
7	0.101	0.229	0.098	0.064						
8	0.100	0.165	0.069							
9	0.048	0.081								
10	0.044									

0.000

0.229

Figure 5.4: Heat map of loss ratios - Accident Benefit line

We consider a special case of the common shock model in Equation (5.1) for dependence across segments only (i.e. $\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}} = 0$),

$$Y_{i,j}^{(n)} = {}_U\lambda_{i,j}^{(n)} \cdot U_{\boldsymbol{\pi}_{(i,j)}} + Z_{i,j}^{(n)}. \tag{5.5}$$

The contribution of the common shock to the total expected observation is then given by

$$\frac{{}_U\lambda_{i,j}^{(n)} E\left[U_{\boldsymbol{\pi}_{(i,j)}}\right]}{{}_U\lambda_{i,j}^{(n)} E\left[U_{\boldsymbol{\pi}_{(i,j)}}\right] + E\left[Z_{i,j}^{(n)}\right]}. \tag{5.6}$$

By setting the scaling parameter ${}_U\lambda_{i,j}^{(n)} = 1$, a disproportion in common shock contributions can be observed within and across business lines which is similar to the observation in the case of dependence within a single segment in the previous section.

Consider an illustrative example of accident period dependence across segments (i.e. $\boldsymbol{\pi}_{(i,j)} = \{Y_{i,1}^{(1)}, ..., Y_{i,I-i+1}^{(1)}, Y_{i,1}^{(2)}, ..., Y_{i,I-i+1}^{(2)}\}$) where the expected common shock is set to 5% of total claims observed in the first accident period of the Bodily Injury line (1)

$$E\left[U_{\boldsymbol{\pi}_{(i,j)}}\right] = 5\% \times E[Y_{i,1}^{(1)}]. \tag{5.7}$$

Heat maps of common shock contributions to the total observations in the two lines are given in Figure 5.3 and 5.5. The common shock proportions are significantly low in the first 4-5 development years (including development year 1) in the Accident Benefits line.

Year	1	2	3	4	5	6	7	8	9	10
1	1.7%	2.1%	3.8%	3.4%	4.1%	11.5%	31.3%	27.2%	54.1%	41.8%
2	1.0%	0.7%	0.8%	1.5%	2.0%	8.7%	4.6%	8.8%	153.7%	
3	1.0%	0.5%	0.7%	1.5%	1.2%	11.4%	4.9%	48.7%		
4	0.8%	0.5%	0.8%	0.9%	1.4%	2.2%	4.8%			
5	0.8%	0.6%	1.1%	1.9%	4.1%	2.6%				
6	0.8%	0.4%	0.6%	1.4%	2.1%					
7	0.5%	0.2%	0.5%	0.8%						
8	0.6%	0.3%	0.8%							
9	0.8%	0.5%								
10	0.6%									

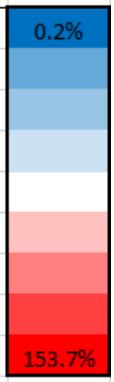

Figure 5.5: Heat map of common shock proportions - Accident Benefit line

Quite significant variations in common shock proportions can be observed within and across segments in the absence of careful modelling as a result of the unbalanced nature of reserving data. Overall, common shock contributions are understated in regions where the total observations are high, and vice versa. One may wish to confine the relation of the common shock to total observations over the entire range of the triangles. The most obvious solution to this is to have cell-specific scaling factors, as also specified in the general framework in Equation (5.1).

5.2.2 Maintaining model parsimony

Scaling factors ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$ play useful roles in scaling the within-triangle common shock effects $\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}$ such that their contributions are proportional to the total observations $Y_{i,j}^{(n)}$. However, this implies that $I \times J \times N$ new parameters are required with one parameter for each cell in the upper and lower triangles. Given that claim observations often vary significantly in size across development periods, simplifying ${}_{\tilde{U}}\lambda_{i,j}^{(n)} = {}_{\tilde{U}}\lambda_{j}^{(n)}$ can fulfil the purpose of balancing common shock proportions while reducing the required number of parameters. However, this still results in J new parameters, or $J \times N$ new parameters for N loss triangles.

Similarly, scaling factors ${}_{U}\lambda_{i,j}^{(n)}$ can be used to scale the between-triangles common shocks $U_{\boldsymbol{\pi}_{(i,j)}}$ such that their contributions are proportional to the total observations. This results in $I \times J \times N$ new parameters with one parameter for each cell in the upper and lower triangles. With the simplification ${}_{U}\lambda_{i,j}^{(n)} = {}_{U}\lambda_{j}^{(n)}$, there are still $J \times N$ new parameters to be estimated.

Loss triangles data is usually known to have a small sample size. While having scaling factors, either cell-specific factors or column-specific factors, can mitigate the impact of the unbalanced nature of reserving data, it also adds a lot more parameters to the model. If this solution is pursued, it can result in over-fitting where the number of parameters to be estimated is larger than the number of observations.

5.2.3 Obtaining distributional tractability

To further complicate the parametrisation, on some occasions, scaling factors ${}_{U}\lambda_{i,j}^{(n)}$ and ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$ also need to be specified such that the total observation $Y_{i,j}^{(n)}$ follows a specific distribution (Avanzi, Taylor and Wong, 2018), which we refers to as distributional tractability or closure under the taking of marginals.

Consider an example of the common shock Tweedie framework developed in Chapters 3 and 4. This framework is developed for cell-wise dependence across business segments (i.e. $\boldsymbol{\pi}_{(i,j)} = \{Y_{i,j}^{(1)}, ..., Y_{i,j}^{(N)}\}$). Recall that we have the specification

$$Y_{i,j}^{(n)} = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}. \tag{3.13}$$

The common shock $U_{i,j}$ and idiosyncratic component $Z_{i,j}^{(n)}$ are assumed to be independent and have Tweedie distributions

$$U_{i,j} \sim \text{Tweedie}_p(\tilde{\alpha}, \tilde{\phi}), \tag{3.7}$$

$$Z_{i,j}^{(n)} \sim \text{Tweedie}_p(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)}), \tag{3.8}$$

where $\tilde{\alpha}$ and $\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}$ are location (mean) parameters, $\tilde{\phi}$ and $\ddot{\phi}^{(n)}$ are dispersion parameters, and p is the power parameter of the Tweedie distributions. Fitting this into the general common shock framework we have the scaling factor

$${}_U\lambda_{i,j}^{(n)} = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}, \tag{5.8}$$

and the across-triangle common shock effect $U_{\boldsymbol{\pi}_{(i,j)}}$ is simplified to the notation $U_{i,j}$. The most simple parametrisation is used for the common shock component $U_{i,j}$ with parameters $\tilde{\alpha}$ and $\tilde{\phi}$, as also stated in the remark in Section 3.4.5. It follows from the closure under addition property of the Tweedie family of distributions, as proved in Jorgensen (1997, Chapter 3) and reviewed in Section 2.3.2.2, that the specification of the scaling factor in Equation (5.8) is needed to ensure $Y_{i,j}^{(n)}$ has a Tweedie distribution.

Recall from Section 3.3.2 that the mean expression can be written by

$$E\left[Y_{i,j}^{(n)}\right] = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{2-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} \ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} + \ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}, \tag{5.9}$$

where the first term in the summation is the contribution from the common shock and the second term is the contribution from the idiosyncratic component. The contribution of common shock to the expected total observation is then

$$\frac{\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{2-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}}{\left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{2-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1}. \tag{5.10}$$

The following observation can be made pending on the value of the power parameter p:

– If $p < 2$: Within a single segment of business, as the development period factor $\ddot{\beta}_j^{(n)}$ typically reaches a peak then decreases as the lag j increases, the above ratio increases.

As a result, the proportion of common shock is understated in early development lags, and overstated in late development lags (Avanzi, Taylor and Wong, 2018). For a portfolio of segments with varying tail lengths, the common shock also contributes disproportionately, as explained in Section 5.2.1.2. The larger the discrepancy between $\ddot{\beta}_j^{(n)}$ and $\ddot{\beta}_i^{(m)}$, the larger the variation in the common shock contributions.

– If $p > 2$: The opposite observation is made for this case. A larger variation in common shock proportions is still observed as the difference between development factors $\ddot{\beta}_j^{(n)}$ and $\ddot{\beta}_i^{(m)}$ become more significant.

– If $p = 2$: In this special case, the common shock contributions are simplified to

$$\frac{\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}}{\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1}, \tag{5.11}$$

which are now independent of accident and development period factors. The common shock contributes proportionately to total observations over the entire range of the triangles.

The above analyses and examples show that the choices of scaling factors ${}_U\lambda_{i,j}^{(n)}$ and ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$ are subject to many constraints. To accurately capture the dependence structure, these parameters are required to balance the common shock proportions within all cells over the entire range of the triangles. However, this can result in over-fitting, which can be a quite critical issue in loss reserving due to small sample size data. Furthermore, the specification of these parameters may be restricted in some cases for the purpose of preserving distributional tractability. It is then the aim of this chapter to find a solution that compromises between these conflicting issues.

5.3 A multivariate Tweedie approach to unbalanced data

In this section, we propose a solution that compromises between conflicting challenges encountered by common shock models when they are applied to reserving data due to the unbalanced feature of the data. Section 5.3.1 describes the general approach which involves careful parametrisation and is used to develop a modified common shock Tweedie framework for unbalanced data in Section 5.3.2. Estimation approach for this modified multivariate

Tweedie framework is then given in Section 5.3.3.

5.3.1 General approach

Recall the general common shock framework in Avanzi, Taylor and Wong (2018)

$$Y_{i,j}^{(n)} = {}_U\lambda_{i,j}^{(n)} \cdot U_{\boldsymbol{\pi}_{(i,j)}} + {}_{\tilde{U}}\lambda_{i,j}^{(n)} \cdot \tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}} + Z_{i,j}^{(n)}. \tag{5.1}$$

As illustrated in the previous section, to ensure proportionate contributions of the common shocks to the total observations within and across loss triangles, it may be desirable for scaling factors of the common shocks ${}_U\lambda_{i,j}^{(n)}$ and ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$ to be cell-specific, or column-specific at the very least. This, however, can create an over-fitting problem.

A parsimonious solution to this problem is to specify the scaling factor for the within-segment common shock $\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}$ as

$${}_{\tilde{U}}\lambda_{i,j}^{(n)} \propto \left(\prod_{i,j} E\left[Z_{i,j}^{(n)} \mathbf{1}_{\boldsymbol{\pi}_{(i,j)}^{(n)}} \right] \right)^{\frac{1}{\#\left(\boldsymbol{\pi}_{(i,j)}^{(n)}\right)}}, \tag{5.12}$$

where $\#\left(\boldsymbol{\pi}_{(i,j)}^{(n)}\right)$ is the number of observations in the set $\boldsymbol{\pi}_{(i,j)}^{(n)}$. In simple words, ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$ is proportional to the geometric average of $E[Z_{i,j}^{(n)}]$ of claims $Y_{i,j}^{(n)}$ in the set $\boldsymbol{\pi}_{(i,j)}^{(n)}$ on which the common shock $\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}$ is applied. The geometric average is used to reasonably account for the skewness in the values of $E[Z_{i,j}^{(n)}]$ within the set $\boldsymbol{\pi}_{(i,j)}^{(n)}$. This skewness is a result of the unbalanced nature of loss triangles.

Similarly, we can have the scaling factor for across-segment common shock $\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}$ specified as

$${}_U\lambda_{i,j}^{(n)} \propto \left(\prod_{i,j,n} E\left[Z_{i,j}^{(n)} \mathbf{1}_{\boldsymbol{\pi}_{(i,j)}} \right] \right)^{\frac{1}{\#\left(\boldsymbol{\pi}_{(i,j)}\right)}}, \tag{5.13}$$

where $\#\left(\boldsymbol{\pi}_{(i,j)}\right)$ is the number of observations in the set $\boldsymbol{\pi}_{(i,j)}$. That is, the scaling factor ${}_U\lambda_{i,j}^{(n)}$ is proportional to the geometric average of of $E[Z_{i,j}^{(n)}]$ of all claims $Y_{i,j}^{(n)}$ in the set $\boldsymbol{\pi}_{(i,j)}$ on which the common shock term $\tilde{U}_{\boldsymbol{\pi}_{(i,j)}^{(n)}}$ is applied.

These specifications do not provide a complete balance of common shock proportions

across cells because the effect of each individual cell is taken to the power of $\frac{1}{\#\left(\boldsymbol{\pi}_{(i,j)}^{(n)}\right)}$ or $\frac{1}{\#\left(\boldsymbol{\pi}_{(i,j)}\right)}$. However, it can reduce the imbalance in common shock proportions to a certain extent. Importantly, it reduces the number of parameters required for scaling common shock components from $I \times J \times N$ (or $2 \times N \times J$ if scaling factors are column-specific) to $N+1$ (N parameters to specify ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$ and one parameter to specify ${}_{U}\lambda_{i,j}^{(n)}$).

Further restrictions may apply in the specification of ${}_{U}\lambda_{i,j}^{(n)}$ and ${}_{\tilde{U}}\lambda_{i,j}^{(n)}$ for other purposes, such as distributional tractability (see also the previous section). The above specifications may be modified differently depending on the situation. This will be illustrated using the multivariate Tweedie framework in the section below.

5.3.2 Application to the multivariate Tweedie framework

In this application, we focus on the general case $p \neq 1$. The application to the special case $p = 1$ is similar with some straightforward modifications in the specification of distributions.

Recall the following specification of the multivariate Tweedie framework introduced in Chapters 3 and 4

$$Y_{i,j}^{(n)} = \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}. \tag{3.13}$$

The common shock $U_{i,j}$ and idiosyncratic component $Z_{i,j}^{(n)}$ are assumed to be independent and have Tweedie distributions

$$U_{i,j} \sim \text{Tweedie}_p(\tilde{\alpha}, \tilde{\phi}), \tag{3.7}$$

$$Z_{i,j}^{(n)} \sim \text{Tweedie}_p(\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)}). \tag{3.8}$$

Applying the solution proposed in the previous section, we can set the scaling factor of the common shock proportional to

$$\left(\prod_{i,j,n} E\left[Z_{i,j}^{(n)} \mathbf{1}_{\boldsymbol{\pi}_{(i,j)}}\right]\right)^{\frac{1}{\#\left(\boldsymbol{\pi}_{(i,j)}\right)}} = \left(\prod_{n} \ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}\right)^{\frac{1}{N}} \approx \left(\prod_{n} \ddot{\beta}_j^{(n)}\right)^{\frac{1}{N}}, \tag{5.14}$$

where $\boldsymbol{\pi}_{(i,j)} = \{Y_{i,j}^{(1)}, ..., Y_{i,j}^{(N)}\}$ as the framework is used to capture cell-wise dependence. The

simplification that removes accident period factors is used because we can quite reasonably expect limited variation across accident periods as a result of claims standardisation, assuming no significant changes occur across accident periods.

It can be observed that to maintain the same specification as in Equation (5.8) for distributional tractability, while satisfying Equation (5.14) to balancing common shock proportions, we can replace $\tilde{\alpha}$ with

$$\tilde{\alpha}_j = \zeta \sqrt[N]{\ddot{\beta}_j^{(1)} ... \ddot{\beta}_j^{(N)}}, \tag{5.15}$$

where ζ is a constant to be estimated. The parameter $\tilde{\alpha}_j$ is also the location parameter of the common shock $U_{i,j}$.

The multivariate Tweedie framework modified for unbalanced data can then be represented as

$$Y_{i,j}^{(n)} + \xi^{(n)} = \left(\frac{\tilde{\alpha}_j}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} U_{i,j} + Z_{i,j}^{(n)}, \tag{5.16}$$

where

$$U_{i,j} \sim \text{Tweedie}_p(\tilde{\alpha}_j, \tilde{\phi}), \qquad \tilde{\alpha}_j = \zeta \sqrt[N]{\ddot{\beta}_j^{(1)} ... \ddot{\beta}_j^{(N)}}, \tag{5.17}$$

$$Z_{i,j}^{(n)} \sim \text{Tweedie}_p(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}, \ddot{\phi}^{(n)}). \tag{5.18}$$

To ensure the estimates of $\ddot{\alpha}_i^{(n)}, \ddot{\beta}_j^{(n)}$ are unique for each n, we use a constraint $\ddot{\alpha}_1^{(n)} = 1$, $n = 1, ..., N$.

Another new feature introduced to the framework is the treatment of negative claims

$$\xi^{(n)} = \begin{cases} 0 & \text{if } \min\{Y_{i,j}^{(n)}, \forall i, j\} \geq 0, \\ \geq -\min\{Y_{i,j}^{(n)}\} & \text{if } \min\{Y_{i,j}^{(n)}, \forall i, j\} < 0, \end{cases} \tag{5.19}$$

which is a translation factor. The translation is only needed for a loss triangle if it contains at least one negative observation and it is bounded below by the smallest negative observation. Note that in this case, while its lower bound is deterministic, the actual value of $\xi^{(n)}$ still has to be estimated. As mentioned in Chapter 2, negative claims can be occasionally observed in loss triangles due to, for example, salvage recoveries, or payment from third parties. Many

models for incremental claims are unable to handle negative observations due to their lack of support for negative masses. Note that Gaussian distributions, members of the Tweedie family of distributions with $p = 0$ can handle negative claims. However restricting the marginal specification to Gaussian distributions for any data set with negative observations is not always ideal as loss distributions can be asymmetric and heavier tailed. This motivates the new development to treat negative payments in multivariate loss reserving data. The generalisation of this treatment to the overall common shock framework in Avanzi, Taylor and Wong (2018) is straightforward.

The marginal density is given by

$$\begin{aligned} &Y_{i,j}^{(n)} + \xi^{(n)} \\ &\sim \text{Tweedie}_p \left(\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)} \left[\left(\frac{\tilde{\alpha}_j}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{2-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1 \right], \ddot{\phi}^{(n)} \left[\left(\frac{\tilde{\alpha}_j}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{2-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} + 1 \right]^{1-p} \right), \end{aligned} \tag{5.20}$$

where the first parameter is the location parameter and also the mean of $Y_{i,j}^{(n)} + \xi^{(n)}$, and the second parameter is the dispersion parameter.

It also follows that the vector of translated cell-wise claims

$${}_{\xi}\boldsymbol{Y}_{i,j} = \begin{pmatrix} Y_{i,j}^{(1)} + \xi^{(1)} \\ Y_{i,j}^{(2)} + \xi^{(2)} \\ \vdots \\ Y_{i,j}^{(N)} + \xi^{(N)} \end{pmatrix}, \tag{5.21}$$

has a multivariate Tweedie distribution with the multivariate density

$$\begin{aligned} &f_{{}_{\xi}\boldsymbol{Y}_{i,j}} \left(y_{i,j}^{(1)} + \xi^{(1)}, ..., y_{i,j}^{(N)} + \xi^{(N)} \right) \\ &= \int_0^{{}_{\xi}B_{i,j}} f_{U_{i,j}}(u_{i,j}) \prod_{n=1}^{N} f_{Z_{i,j}^{(n)}} \left(y_{i,j}^{(n)} + \xi^{(n)} - \left(\frac{\tilde{\alpha}}{\ddot{\alpha}_i^{(n)} \ddot{\beta}_j^{(n)}} \right)^{1-p} \frac{\ddot{\phi}^{(n)}}{\tilde{\phi}} u_{i,j} \right) du_{i,j}, \end{aligned} \tag{5.22}$$

where

$${}_{\xi}B_{i,j} = \min \left(\left(\frac{\ddot{\alpha}_i^{(1)} \ddot{\beta}_j^{(1)}}{\tilde{\alpha}} \right)^{1-p} \frac{\tilde{\phi}}{\ddot{\phi}^{(1)}} \left(y_{i,j}^{(1)} + \xi^{(1)} \right), ..., \left(\frac{\ddot{\alpha}_i^{(N)} \ddot{\beta}_j^{(N)}}{\tilde{\alpha}} \right)^{1-p} \frac{\tilde{\phi}}{\ddot{\phi}^{(N)}} \left(y_{i,j}^{(N)} + \xi^{(N)} \right) \right). \tag{5.23}$$

The common shock proportion in this framework is given by

$$\frac{\left(\frac{\tilde{\alpha}_j}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}}{\left(\frac{\tilde{\alpha}_j}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}+1}=\frac{\left(\frac{\zeta\sqrt[N]{\ddot{\beta}_j^{(1)}...\ddot{\beta}_j^{(N)}}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}}{\left(\frac{\zeta\sqrt[N]{\ddot{\beta}_j^{(1)}...\ddot{\beta}_j^{(N)}}}{\ddot{\alpha}_i^{(n)}\ddot{\beta}_j^{(n)}}\right)^{2-p}\frac{\ddot{\phi}^{(n)}}{\tilde{\phi}}+1}. \tag{5.24}$$

This does not provide a perfect balance of common shock proportions because the balancing effect of $\ddot{\beta}_j^{(n)}$ is reduced by a factor $\sqrt[N]{\ddot{\beta}_j^{(n)}}$. However it still provides quite a significant improvement over the original framework. This will be illustrated in a simulation illustration in Section 5.4.

As mentioned in Section 5.2.3, the common shock gamma model does not suffer from issues of unbalanced data. Hence in a portfolio for which a gamma dispersion is appropriate, a multivariate gamma model can be a good candidate.

5.3.3 Estimation of the modified multivariate Tweedie framework

As with the estimation for the original common shock Tweedie framework in Chapter 4, Bayesian inference is also used to estimate parameters in the new Tweedie framework modified for unbalanced data. We also take a step further to incorporate the estimation of the power parameter p and translation parameters $\xi^{(n)}$ into the Bayesian set-up to account for their parameter uncertainty. This is to formalise the estimation of these parameters as they are often estimated heuristically in practice.

A two step estimation procedure is used for estimation, similar to that in Chapter 4. The first stage is the estimation of all parameters except parameters ζ and $\tilde{\phi}$ of the common shock. This stage, however, gives the estimate of the ratio

$$\Lambda_\zeta = \frac{\zeta^{2-p}}{\tilde{\phi}} \tag{5.25}$$

of these parameters, as can observed from Equation (5.20) with the specification $\tilde{\alpha}_j = \zeta\sqrt[N]{\ddot{\beta}_j^{(1)}...\ddot{\beta}_j^{(N)}}$. This is then followed by the multivariate stage that estimates ζ and $\tilde{\phi}$ conditional on estimates of other parameters from the first stage. The motivation for this procedure comes from properties of the multivariate Tweedie framework. Cell-

wise observations in this framework follow a multivariate Tweedie distribution, and each observation itself also has a marginal Tweedie distribution. In addition, the multivariate density has an integral calculation, as shown in Equation (5.22). This can prolong the estimation of the posterior density, making the tuning and convergence of MCMC much more difficult.

A Bayesian set-up requires the specifications of the likelihood functions, prior densities and a MCMC algorithm to approximate the posterior densities if they are not in recognisable forms. The likelihood functions follow Equation (5.20) for the first stage and Equation (5.22) for the second stage. Prior densities can be chosen to be informative or uninformative. Uninformative priors assign equal possibilities to all values in the feasible set of parameter values, whereas informative priors convey some prior preference for certain values of the parameters. Some guidance for the selection of informative prior densities can be found in Section 4.2.1 of the original framework. Regarding the prior densities for p and $\xi^{(n)}$, some constraints need to be taken into account. In particular, p is not defined in $(0, 1)$, and $\xi^{(n)}$ has a lower bound as per its specification in Equation (5.19).

Putting together the likelihood and prior specifications, the posterior density in the first stage is given by

$$f_{\boldsymbol{\Theta}|\boldsymbol{Y}^U}(\boldsymbol{\Theta}|\boldsymbol{Y}^U) \propto \left(\prod_{i,j,n} f_{Y_{i,j}^{(n)}+\xi^{(n)}} \left(y_{i,j}^{(n)} + \xi^{(n)} | \boldsymbol{\Theta} \right) \right) f_p(p) f_{\boldsymbol{\xi}}(\boldsymbol{\xi}) f_{\Lambda_\zeta}(\Lambda_\zeta) f_{\ddot{\boldsymbol{\alpha}}}(\ddot{\boldsymbol{\alpha}}) f_{\ddot{\boldsymbol{\beta}}}(\ddot{\boldsymbol{\beta}}) f_{\ddot{\boldsymbol{\phi}}}(\ddot{\boldsymbol{\phi}}), \tag{5.26}$$

where

$$\ddot{\boldsymbol{\alpha}}_{\boldsymbol{i}} = \begin{pmatrix} \ddot{\alpha}_i^{(1)} \\ \ddot{\alpha}_i^{(2)} \\ \vdots \\ \ddot{\alpha}_i^{(N)} \end{pmatrix}, \quad \ddot{\boldsymbol{\alpha}} = \begin{pmatrix} \ddot{\alpha}_1 \\ \ddot{\alpha}_2 \\ \vdots \\ \ddot{\alpha}_I \end{pmatrix}, \quad \ddot{\boldsymbol{\beta}}_{\boldsymbol{j}} = \begin{pmatrix} \ddot{\beta}_j^{(1)} \\ \ddot{\beta}_j^{(2)} \\ \vdots \\ \ddot{\beta}_j^{(N)} \end{pmatrix}, \quad \ddot{\boldsymbol{\beta}} = \begin{pmatrix} \ddot{\beta}_1 \\ \ddot{\beta}_2 \\ \vdots \\ \ddot{\beta}_J \end{pmatrix}, \tag{5.27}$$

$$\boldsymbol{\xi} = \begin{pmatrix} \xi^{(1)} \\ \xi^{(2)} \\ \vdots \\ \xi^{(N)} \end{pmatrix}, \quad \ddot{\boldsymbol{\phi}} = \begin{pmatrix} \ddot{\phi}^{(1)} \\ \ddot{\phi}^{(2)} \\ \vdots \\ \ddot{\phi}^{(N)} \end{pmatrix}, \quad \boldsymbol{\Theta} = \begin{pmatrix} p \\ \boldsymbol{\xi} \\ \Lambda_\zeta \\ \ddot{\boldsymbol{\alpha}} \\ \ddot{\boldsymbol{\beta}} \\ \ddot{\boldsymbol{\phi}} \end{pmatrix}. \tag{5.28}$$

Conditioning on parameter estimates from this stage, the posterior density for the multivariate estimation is given by

$$f_{\zeta|\boldsymbol{Y}^U,\boldsymbol{\Theta}}(\zeta|\boldsymbol{Y}^U,\boldsymbol{\Theta}) \propto \left(\prod_{i,j} f_{\boldsymbol{\xi}\boldsymbol{Y}_{\boldsymbol{i},\boldsymbol{j}}} \left({}_{\boldsymbol{\xi}}\boldsymbol{y}_{\boldsymbol{i},\boldsymbol{j}} | \zeta, \boldsymbol{\Theta} \right) \right) f_\zeta(\zeta). \tag{5.29}$$

The posterior densities in both stages are not in recognisable forms, hence MCMC is required for the evaluation. As with the original Tweedie framework in Chapter 4, random walk Metropolis algorithms are used for marginal estimation and multivariate estimation. Proposal densities are chosen (tuned) so that the acceptance probabilities are within desirable ranges. Details are given in Section 4.2.3. It is also worth noting that the use of vectorisation in model implementation can significantly improve the computational speed if the implementation is performed in R. This is one of the main strengths of R that has been noted in the literature (Lafaye de Micheaux et al., 2013, Chapter 5). Observations and the corresponding distributional specifications should be vectorised when appropriate and programming loops should be avoided as much as possible to improve the computational speed.

5.4 Simulation illustrations

Two illustrations are performed on two data sets. The first illustration, provided in Section 5.4.1, is to assess the accuracy of the estimation procedure. The second illustration, provided in Section 5.4.2, is to compare the performance of the modified multivariate Tweedie framework and the common shock Tweedie framework in Chapters 3 and 4 on a portfolio of two segments with varying tail lengths.

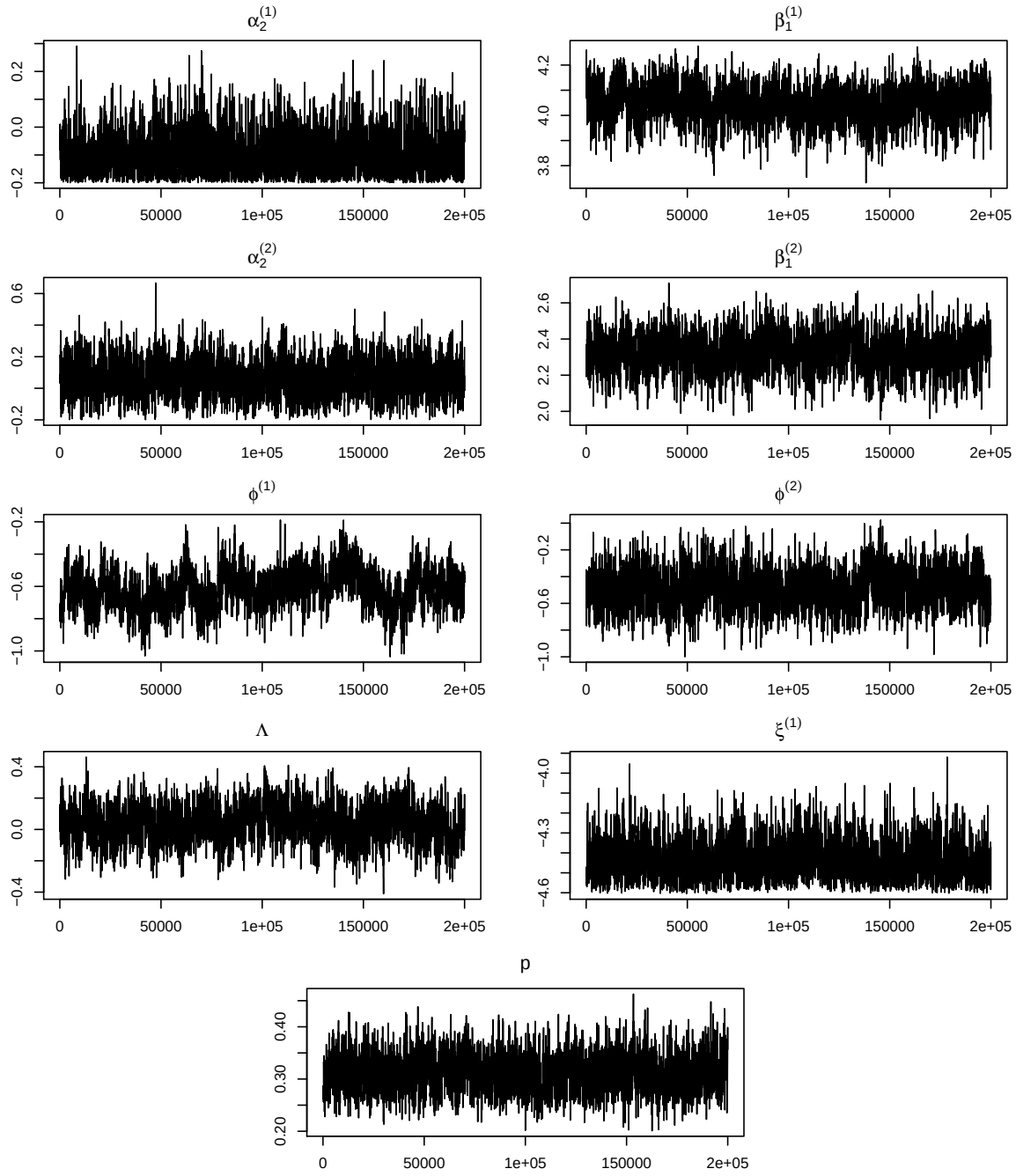

Figure 5.6: MCMC sample paths of some parameters

5.4.1 An illustration with unbalanced data and negative claims

A data set consisting of two triangles, one of which has a negative claim observation, is simulated. The two loss triangles are represented in Table 5.10 and 5.11 in Appendix 5.A.1.

The marginal fitting is first performed. Parameters are transformed using the log transformation, and uniform prior densities are used. 200,000 simulations are run and 100,000

simulations are discarded as the burn-in period. The sample chain is thinned by accepting every 5^{th} iteration to reduce the serial dependence between iterations. MCMC paths of some parameters are given in Figure 5.6. A similar procedure is performed for the multivariate estimation. The estimates of ζ and $\tilde{\phi}$ are obtained from this step. Parameter estimates are provided in Table 5.12 in Appendix 5.A.1. The results show that the estimation procedure is reasonably accurate as the true parameter values are all within the confidence intervals of their estimates.

5.4.2 A comparison of performances of the multivariate Tweedie framework with and without modification for unbalanced data

A natural question arises regarding the performance of the multivariate Tweedie approach for unbalanced data compared to the original multivariate Tweedie approach introduced in Chapters 3 and 4. To be able to assess their performances more accurately, this comparison is performed on an illustrated data set whose underlying model is known. True common shock contributions are also known and these serve as the benchmark for the comparison.

To not put any particular framework at a disadvantage, the synthetic data used for this illustration is simulated from a mixture of models. We deliberately select a (extreme) data set to which neither of the frameworks is properly adapted. In particular, two loss triangles of ten development lags and ten accident periods are generated such that the dependence is strong in the first four development lags, and a lot weaker in the last six lags. The common shock components are generated with column-specific mean parameters $\tilde{\alpha}_j = \zeta_j \sqrt{\ddot{\beta}_j^{(1)} \ddot{\beta}_j^{(2)}}$ with $\zeta_j = 0.5$ for $1 \leq j \leq 4$, and $\zeta_j = 0.02$ for $5 \leq j \leq 10$. The second segment is also simulated to be longer-tailed than the first. The two loss triangles are presented in Table 5.13 and 5.14 in Appendix 5.A.2.

Heat maps of ratios of fitted common shock proportions to true proportions are given in Figure 5.7 for triangle 1, and Figure 5.8 for triangle 2. Fitted values are calculated using parameter estimates and true values are calculated using true parameter values.

The modified Tweedie framework provides a very good fit in the first four development lags. The goodness of fit is considerably less satisfactory in the later development lags when the actual common shock proportion drops. However, it can be observed that the fitted

common shocks still contribute somewhat proportionately to the total expected observations in the later lags. The original common shock Tweedie model provides a poor goodness-of-fit overall, especially in early development lags. The proportions of common shock are underestimated in early development lags and overestimated in later lags.

Period	1	2	3	4	5	6	7	8	9	10
1	0.947	0.929	0.985	0.958	4.398	4.216	4.320	4.596	4.163	3.783
2	0.932	0.917	0.975	0.949	4.413	4.227	4.327	4.622	4.174	
3	0.995	0.965	1.018	0.987	4.802	4.598	4.710	5.028		
4	0.974	0.949	1.004	0.974	4.649	4.452	4.561			
5	0.890	0.881	0.944	0.919	4.490	4.289				
6	0.924	0.908	0.969	0.942	4.588					
7	0.895	0.886	0.948	0.923						
8	0.953	0.932	0.989							
9	0.980	0.953								
10	0.997									

0.168 — 1.000 — 11.099

Period	1	2	3	4	5	6	7	8	9	10
1	0.212	0.254	0.353	0.434	2.425	3.121	3.916	4.291	4.746	4.643
2	0.189	0.228	0.319	0.396	2.264	2.958	3.753	4.174	4.664	
3	0.211	0.250	0.346	0.426	2.539	3.300	4.177	4.617		
4	0.209	0.249	0.345	0.425	2.482	3.218	4.065			
5	0.195	0.232	0.324	0.401	2.379	3.116				
6	0.196	0.233	0.325	0.402	2.391					
7	0.211	0.251	0.348	0.429						
8	0.204	0.242	0.337							
9	0.226	0.266								
10	0.190									

Figure 5.7: Heat maps of ratios of fitted common shock proportions to true proportions for triangle 1 (top: Tweedie framework modified for unbalanced data, bottom: original common shock Tweedie framework)

Overall the modified Tweedie framework does not eliminate the issues of unbalanced data across development periods, however there is a reduction. The fitting is quite good in early development lags, but is unsatisfactory in later lags. The use of the geometric average of column factors across multiple triangles may contribute to this performance as the geometric average may not be close to some individual column factors if the development patterns are too different. However, it is worth emphasising that the example used has quite an extreme variation in common shock proportions across development lags, and one should not expect such radical variation in practice. In addition, the poor performance also arises from the

discrepancy between the modified model with a constant scaling term ζ and the true model generating the data (with column specific scaling term ζ_j). With this specification, it is not surprising that the earlier (large) development periods dominate the estimation of ζ. We do not expect good results because of model misspecification, but we can arrive at two main conclusions: the modified framework out-performs the original framework; and the common shock proportions are mis-estimated in the higher development periods, where amounts are small and do not contribute significantly to total liability.

Period	1	2	3	4	5	6	7	8	9	10
1	1.091	1.162	1.116	1.174	6.895	7.415	7.521	5.901	7.527	9.004
2	1.065	1.125	1.069	1.124	6.697	7.222	7.361	5.683	7.332	
3	1.034	1.081	1.020	1.072	6.336	6.837	6.979	5.364		
4	1.093	1.165	1.114	1.174	7.004	7.544	7.671			
5	1.130	1.218	1.175	1.240	7.458	8.023				
6	1.059	1.117	1.059	1.114	6.635					
7	1.077	1.141	1.083	1.141						
8	1.110	1.187	1.132							
9	1.047	1.101								
10	1.039									

Period	1	2	3	4	5	6	7	8	9	10
1	0.191	0.188	0.204	0.257	2.635	4.371	6.432	7.624	10.677	10.275
2	0.180	0.180	0.197	0.248	2.640	4.414	6.572	7.813	11.099	
3	0.172	0.171	0.187	0.236	2.505	4.199	6.277	7.469		
4	0.169	0.168	0.183	0.232	2.448	4.106	6.146			
5	0.184	0.183	0.200	0.253	2.665	4.446				
6	0.169	0.168	0.184	0.233	2.478					
7	0.172	0.175	0.192	0.244						
8	0.183	0.186	0.205							
9	0.188	0.193								
10	0.184									

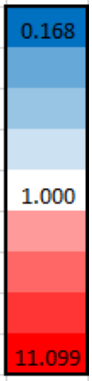

Figure 5.8: Heat maps of ratios of fitted common shock proportions to true proportions for triangle 2 (top: Tweedie framework modified for unbalanced data, bottom: original common shock Tweedie framework)

5.5 Illustration with real data

The data used for illustration is a set of two triangles from the Bodily Injury line (1) and the Accident Benefit line (2) from a Canadian insurance company provided in Côté et al.

(2016). These two triangles have also been used for illustrations in Section 5.2 and their details can be found therein.

5.5.1 Preliminary analysis

A preliminary analysis is performed to assess the suitability of this data set. This includes the assessment of the tails, as well as the dependence structure.

5.5.1.1 Analysis of the tails

Plots of loss ratios are provided in Figure 5.9. It can be observed from these plots the Bodily Injury line has longer claims development than the Accident Benefits line. This is also observed earlier in the heat maps of loss ratios in Section 5.2.

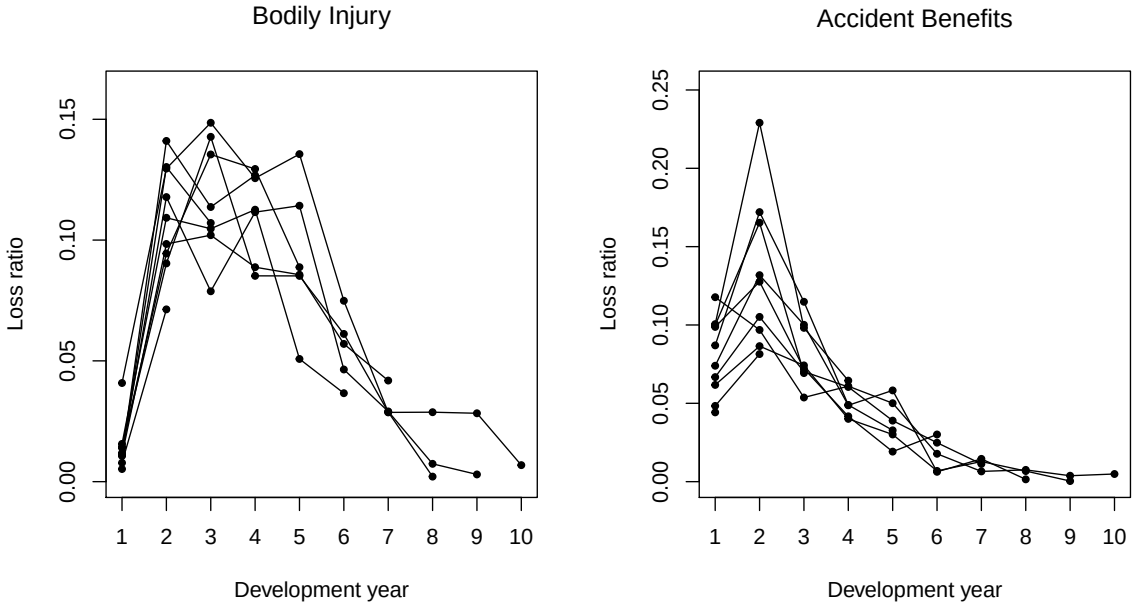

Figure 5.9: Incremental loss ratios in real data from a Canadian insurer

Tail lengths of the two business lines are assessed using chain ladder development factors. Recall the definition of chain ladder development factors from Section 2.2.1

$$d_j^{(n)} = \frac{\sum_{i=1}^{I-j} X_{i,j+1}^{(n)}}{\sum_{i=1}^{I-j} X_{i,j}^{(n)}}, \tag{2.4}$$

where $X_{i,j}^{(n)}$ are cumulative claims. Results are given in Table 5.1. It can be observed that the development factors of the Bodily Injury line dominate those of the Accident Benefits line for all development lags, except in the final year. This blip may be a false signal due to the truncation of data at the last development period and only one single observation is made in this final year. Hence the Bodily Injury line is convincingly longer-tailed than the Accident Benefits line.

Year (j)	1	2	3	4	5	6	7	8	9
$d_j^{(1)}$	8.1617	1.8968	1.4521	1.2652	1.1249	1.0624	1.0225	1.0254	1.0092
$d_j^{(2)}$	2.5844	1.3584	1.1708	1.1140	1.0481	1.0305	1.0137	1.0057	1.0118

Table 5.1: Claims development factors for each development period

5.5.1.2 Exploratory dependence analysis

A heuristic dependence analysis is performed by fitting to each line a Tweedie GLM with a log-link and the chain ladder mean structure

$$a_i^{(n)} + b_j^{(n)}. \tag{5.30}$$

This is to remove fixed accident period and development period effects. The best power parameters p chosen for the two lines with the above mean structure by assessing the respective likelihood profiles are 1.07 and 1.34. Correlations between GLM Pearson residuals of the two lines are given in Table 5.2, where the residuals are calculated using

$$\frac{Y_{i,j}^{(n)} - \mu_{i,j}^{(n)}}{\left(\mu_{i,j}^{(n)}\right)^p}. \tag{5.31}$$

The dependence between residuals is strong and significant after allowing for fixed accident period and development period effects.

Pearson	Spearman	Kendall
0.3659 (0.0060)	0.3480 (0.0096)	0.2525 (0.0065)

Table 5.2: Correlation coefficients between cell-wise GLM residuals and their corresponding p-values

To examine whether this strong correlation comes from calendar year effects that can

impact both lines simultaneously, we also perform another GLM analysis with an additional fixed calendar year effect in the mean structure

$$a_i^{(n)} + b_j^{(n)} + h_t^{(n)}. \tag{5.32}$$

With this particular mean structure, the best power parameters p chosen for the two lines are 1.05 and 1.25, respectively. It is worth noting that the power parameter estimates p change as the model structure changes. Correlations between GLM Pearson residuals of the two lines are then given in Table 5.3. The correlation coefficients have been reduced, however, not very significantly.

Pearson	Spearman	Kendall
0.3416 (0.0107)	0.3250 (0.0159)	0.2202 (0.0176)

Table 5.3: Correlation coefficients between cell-wise GLM residuals and their corresponding p-values after removing fixed calendar year effects

Year	1	2	3	4	5	6	7	8	9	10
1	1.000	0.988	0.938	1.028	0.987	0.963	0.746	1.654	1.504	1.000
2	1.132	1.070	0.906	0.941	1.281	0.997	0.854	0.697	0.243	
3	1.131	1.035	1.172	0.767	1.106	1.027	0.963	0.210		
4	1.165	0.863	0.959	0.992	0.932	1.169	1.594			
5	1.254	1.079	1.201	1.037	0.664	0.842				
6	0.983	0.883	0.994	1.111	1.009					
7	0.834	0.996	0.917	1.117						
8	0.706	1.087	0.953							
9	0.904	1.013								
10	1.000									

Year	1	2	3	4	5	6	7	8	9	10
1	1.000	0.906	0.879	1.193	1.187	1.006	0.526	1.224	1.639	1.000
2	1.157	0.994	1.112	0.876	0.984	0.431	1.162	1.515	0.219	
3	1.122	0.914	1.105	0.968	1.398	0.293	1.191	0.285		
4	0.892	0.784	1.029	1.287	1.011	1.534	1.166			
5	1.161	1.032	0.925	0.785	0.538	1.912				
6	0.962	1.058	1.130	0.864	0.821					
7	0.868	1.109	0.931	1.045						
8	0.974	1.106	0.864							
9	0.945	1.040								
10	1.000									

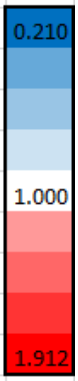

Figure 5.10: Heat maps of ratios of observed values to GLM fitted values (top: Bodily Injury line, bottom: Accident Benefits)

Heat maps of residuals from the GLM analysis with the mean structure specified in Equation (5.32) are given in Figure 5.10. Residuals are calculated as ratios of observed values to GLM fitted values. There are some common cell-wise patterns that are quite obvious from the heat maps, for example, low payments in development year 7 compensated by accelerated payments in periods 8-9 in the first accident year, payment dips in accident year 4 and development lag 2, similar development patterns in accident periods 7 and 8. This suggests some cell-wise dependence between the two business lines. Results from the preliminary analysis shows that this data set is suitable for illustration of the model. In this illustration, the common shocks in the multivariate Tweedie approach are used as drivers of the correlated noise observed in the data.

5.5.2 Estimation results

Bayesian inference is used for estimation. The marginal fitting is first performed. 400,000 simulations are run and 300,000 simulations are discarded as the burn-in period. The sample chain is thinned by accepting every 5^{th} iteration to reduce the serial dependence between iterations. The multivariate fitting is then performed with 90,000 simulations and the first 30,000 are discarded as the burn-in period. The chain is then thinned by selecting every 3^{th} iteration. Summary statistics are then computed on these posterior samples. The results are given in Table 5.4 and 5.5. Note that the value of p used in our framework is estimated to be 1.8290. This is different from the estimates obtained in the preliminary analysis as the multivariate Tweedie framework used is different from the GLM structure used earlier.

	Median	SD	90% CI		Median	SD	90% CI
$\ddot{\alpha}_2^{(1)}$	0.6390	0.0875	(0.5120; 0.7970)	$\ddot{\alpha}_2^{(2)}$	0.7770	0.1119	(0.6230; 0.9830)
$\ddot{\alpha}_3^{(1)}$	0.9010	0.1026	(0.7510; 1.0860)	$\ddot{\alpha}_3^{(2)}$	1.0050	0.1508	(0.7860; 1.2770)
$\ddot{\alpha}_4^{(1)}$	0.7340	0.1207	(0.5710; 0.9610)	$\ddot{\alpha}_4^{(2)}$	1.3690	0.1773	(1.1220; 1.6980)
$\ddot{\alpha}_5^{(1)}$	0.9680	0.1461	(0.7650; 1.2390)	$\ddot{\alpha}_5^{(2)}$	0.9980	0.1390	(0.8010;1.2550)
$\ddot{\alpha}_6^{(1)}$	0.8260	0.1149	(0.6580; 1.0310)	$\ddot{\alpha}_6^{(2)}$	1.4150	0.2656	(1.0520; 1.9120)
$\ddot{\alpha}_7^{(1)}$	1.2260	0.1464	(1.0140; 1.4880)	$\ddot{\alpha}_7^{(2)}$	1.5060	0.1727	(1.2560; 1.8240)
$\ddot{\alpha}_8^{(1)}$	0.8510	0.1276	(0.6760; 1.0940)	$\ddot{\alpha}_8^{(2)}$	1.2390	0.2478	(0.9200; 1.7170)
$\ddot{\alpha}_9^{(1)}$	0.7230	0.0772	(0.6060; 0.8600)	$\ddot{\alpha}_9^{(2)}$	1.2450	0.2061	(0.9440; 1.6230)
$\ddot{\alpha}_{10}^{(1)}$	0.2200	0.0592	(0.1480; 0.3360)	$\ddot{\alpha}_{10}^{(2)}$	1.7260	0.3244	(1.2560; 2.3350)
$\ddot{\beta}_1^{(1)}$	0.0160	0.0022	(0.0130; 0.0200)	$\ddot{\beta}_1^{(2)}$	0.0590	0.0080	(0.0470; 0.0740)
$\ddot{\beta}_2^{(1)}$	0.1430	0.0192	(0.1140; 0.1770)	$\ddot{\beta}_2^{(2)}$	0.1050	0.0139	(0.0840; 0.1300)
$\ddot{\beta}_3^{(1)}$	0.1270	0.0136	(0.1060; 0.1510)	$\ddot{\beta}_3^{(2)}$	0.0670	0.0095	(0.0530; 0.0840)
$\ddot{\beta}_4^{(1)}$	0.0930	0.0111	(0.0760; 0.1120)	$\ddot{\beta}_4^{(2)}$	0.0310	0.0040	(0.0250; 0.0390)
$\ddot{\beta}_5^{(1)}$	0.1190	0.0153	(0.0970; 0.1470)	$\ddot{\beta}_5^{(2)}$	0.0300	0.0032	(0.0250; 0.0350)
$\ddot{\beta}_6^{(1)}$	0.0510	0.0088	(0.0380; 0.0670)	$\ddot{\beta}_6^{(2)}$	0.0160	0.0019	(0.0130; 0.0190)
$\ddot{\beta}_7^{(1)}$	0.0400	0.0065	(0.0310; 0.0520)	$\ddot{\beta}_7^{(2)}$	0.0150	0.0018	(0.0120; 0.0180)
$\ddot{\beta}_8^{(1)}$	0.0100	0.0010	(0.0090; 0.0120)	$\ddot{\beta}_8^{(2)}$	0.0050	0.0008	(0.0040; 0.0070)
$\ddot{\beta}_9^{(1)}$	0.0200	0.0040	(0.0140; 0.0270)	$\ddot{\beta}_9^{(2)}$	0.0020	0.0003	(0.0020; 0.0030)
$\ddot{\beta}_{10}^{(1)}$	0.0050	0.0008	(0.0040; 0.0060)	$\ddot{\beta}_{10}^{(2)}$	0.0030	0.0004	(0.0020; 0.0030)
$\ddot{\phi}^{(1)}$	0.1400	0.0372	(0.0900; 0.2120)	$\ddot{\phi}^{(2)}$	0.1580	0.0431	(0.1030; 0.2430)
Λ	0.3240	0.0732	(0.2220; 0.4610)	p	1.8290	0.0660	(1.7120; 1.9260)

Table 5.4: Posterior statistics of parameters from marginal estimation

	Median	SD	90% CI
ζ	1.0080	4.6868	(0.0570; 17.2280)
$\tilde{\phi}$	3.0910	0.9413	(1.8920; 5.0220)

Table 5.5: Posterior statistics of parameters from multivariate estimation

5.5.3 Goodness-of-fit analysis

Marginal and multivariate goodness-of-fits are assessed. Marginal goodness-of-fit is assessed using QQ plots of residuals in Figure 5.11. The plot shows that the fit is quite off in the right tail of the Bodily Injury line, and slightly off in both tails of the Accident Benefit line. The goodness of fit in other regions, however, is quite good. This may be a result of the restriction of using the same power parameter p for both lines. However, the multivariate Tweedie framework still provides marginal flexibility with flexible choices of p. For comparison, similar QQ plots are performed for a common shock normal model in Figure

5.12. It can be observed that the Tweedie marginals provide a much better fit compared to the normal marginals (with power parameter $p = 0$).

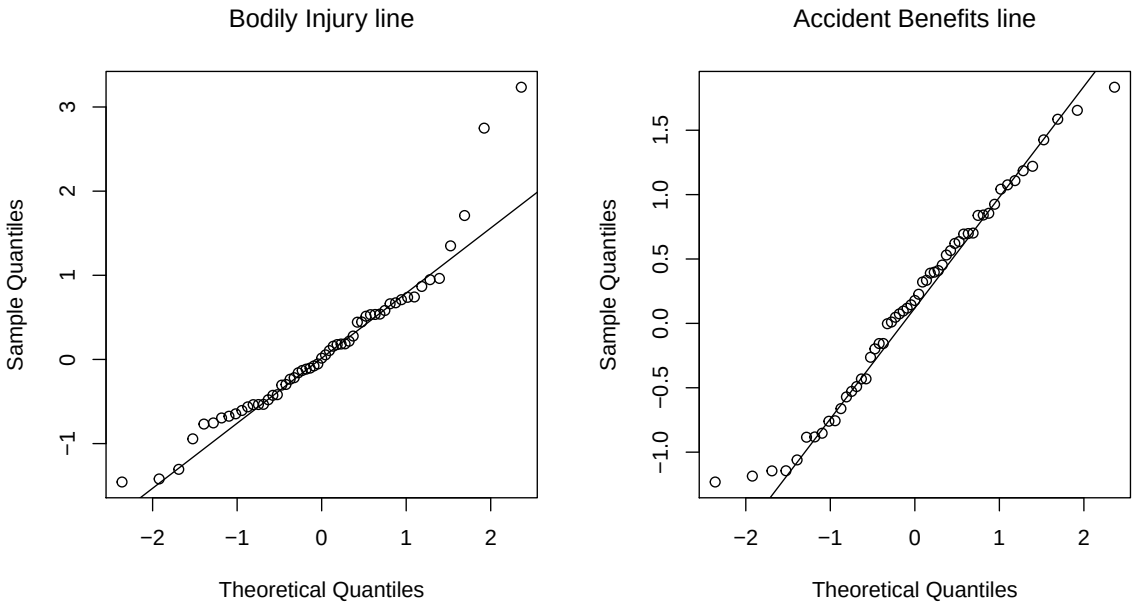

Figure 5.11: QQ plots of residuals from common shock Tweedie model ($p = 1.829$)

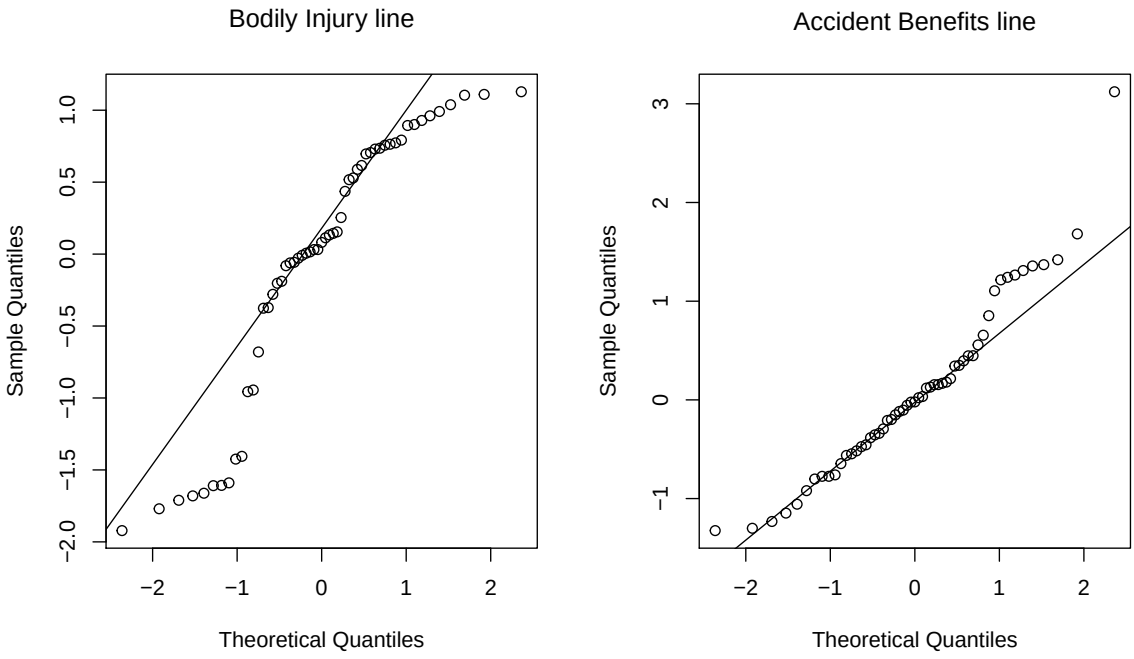

Figure 5.12: QQ plots of residuals from common shock normal model

Multivariate goodness-of-fit is assessed by comparing the empirical copula of real data

with an empirical copula of back fitted data. Because of the use of a Bayesian inference, various sets of back fitted data can be generated. A path is randomly chosen for illustration. Plots of empirical copulas are presented in Figure 5.13. It can be observed that the model can capture the general positive dependence structure in the data.

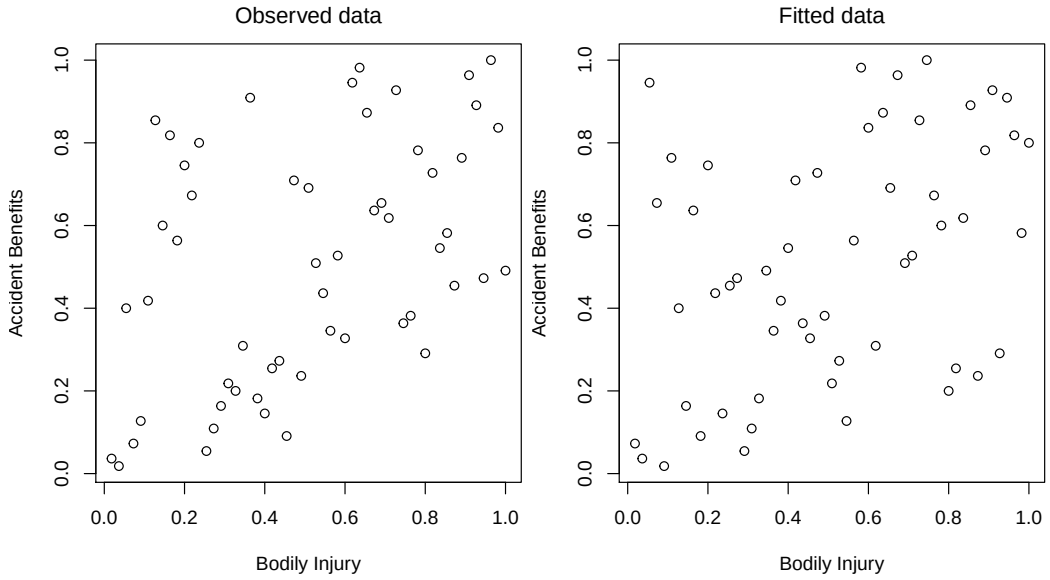

Figure 5.13: Plots of empirical copulas for observed values and back-fitted values

5.5.4 Common shock proportions

Predictive distributions of outstanding claim observations in the lower triangles can be calculated using the predictive Bayesian inference. Using parameter estimates, the contributions of common shock within each cell in the two triangles are calculated and given in Table 5.6 and 5.7. It can be observed that there is only a very mild variation in the common shock proportions within and across triangles.

Year	1	2	3	4	5	6	7	8	9	10
1	4.8%	4.2%	4.1%	4.0%	3.9%	3.9%	4.0%	4.1%	3.6%	4.2%
2	5.2%	4.6%	4.4%	4.3%	4.2%	4.2%	4.3%	4.4%	3.9%	4.5%
3	4.9%	4.3%	4.2%	4.0%	3.9%	4.0%	4.1%	4.2%	3.7%	4.2%
4	5.1%	4.5%	4.3%	4.2%	4.1%	4.2%	4.2%	4.3%	3.8%	4.4%
5	4.9%	4.3%	4.1%	4.0%	3.9%	4.0%	4.0%	4.1%	3.6%	4.2%
6	5.0%	4.4%	4.2%	4.1%	4.0%	4.1%	4.1%	4.2%	3.7%	4.3%
7	4.7%	4.1%	4.0%	3.8%	3.7%	3.8%	3.9%	4.0%	3.5%	4.0%
8	5.0%	4.3%	4.2%	4.1%	4.0%	4.1%	4.1%	4.2%	3.7%	4.3%
9	5.1%	4.5%	4.3%	4.2%	4.1%	4.2%	4.2%	4.3%	3.8%	4.4%
10	6.2%	5.4%	5.3%	5.1%	5.0%	5.1%	5.1%	5.2%	4.6%	5.3%

Table 5.6: Proportions of common shock to the expected total observations calculated using parameter estimates - Bodily Injury

Year	1	2	3	4	5	6	7	8	9	10
1	4.4%	5.0%	5.1%	5.3%	5.4%	5.4%	5.3%	5.2%	5.9%	5.1%
2	4.6%	5.2%	5.3%	5.5%	5.7%	5.6%	5.5%	5.4%	6.1%	5.3%
3	4.4%	5.0%	5.1%	5.3%	5.4%	5.3%	5.3%	5.1%	5.9%	5.1%
4	4.2%	4.7%	4.9%	5.1%	5.2%	5.1%	5.0%	4.9%	5.6%	4.8%
5	4.4%	5.0%	5.1%	5.3%	5.4%	5.4%	5.3%	5.2%	5.9%	5.1%
6	4.1%	4.7%	4.8%	5.0%	5.1%	5.1%	5.0%	4.9%	5.5%	4.8%
7	4.1%	4.7%	4.8%	5.0%	5.1%	5.0%	4.9%	4.8%	5.5%	4.7%
8	4.2%	4.8%	5.0%	5.1%	5.3%	5.2%	5.1%	5.0%	5.7%	4.9%
9	4.2%	4.8%	5.0%	5.1%	5.3%	5.2%	5.1%	5.0%	5.7%	4.9%
10	4.0%	4.6%	4.7%	4.9%	5.0%	4.9%	4.8%	4.7%	5.4%	4.6%

Table 5.7: Proportions of common shock to the expected total observations calculated using parameter estimates - Accident Benefits

5.5.5 Outstanding claims forecast

To obtain the distributions of the outstanding claims, posterior samples of parameters from the Bayesian inference are used to project claims in lower triangles. This projection utilises the specification in Equations (5.16), (5.17) and (5.18) in Section 5.3. This gives a set of samples of future claims in the lower triangles. Using this set, summary statistics of the total outstanding claims distributions are given in Table 5.8 and kernel densities of outstanding claims are given in Figure 5.14. Summary statistics provided include the posterior mean, standard deviation, $\text{VaR}_{75\%}$ and $\text{VaR}_{95\%}$ of the distribution of total outstanding claims for each line, as well as for both lines.

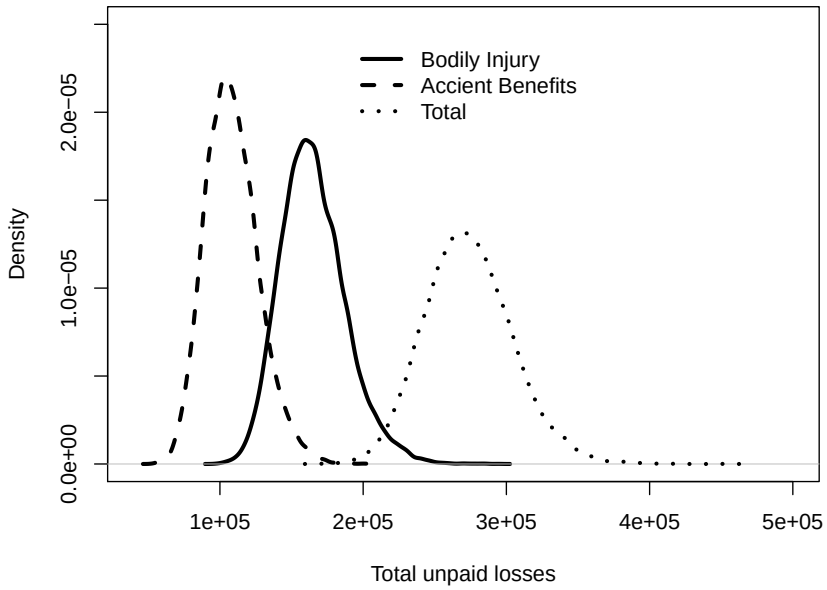

Figure 5.14: Kernel densities of predictive distributions of total outstanding claims in each line of business and in the aggregate portfolio

	Bodily Injury	Accident Benefits	Both lines
Mean	165,185.92	108,465.81	273,651.73
SD	22,720.88	18,554.65	30,538.83
$\text{VaR}_{75\%}$	179,057.18	120,100.43	293,061.56
$\text{VaR}_{95\%}$	205,752.20	141,426.24	326,177.22

Table 5.8: Summary statistics of outstanding claims distributions

The two business lines do not have a comonotonic dependence structure, and this allows the insurer to gain some diversification benefits when they set their risk margins. Recall the definitions of risk margins and diversification benefits from Section 4.4.4

$$\text{Risk margin}_{\chi\%}[Y] = \max\left\{\text{VaR}_{\chi\%}[Y] - E[Y];\ \frac{1}{2}SD[Y]\right\}, \tag{4.14}$$

$$\text{DB} = \frac{\big(\text{Risk margin}_{\chi\%}[Y_1] + \text{Risk margin}_{\chi\%}[Y_2]\big) - \text{Risk margin}_{\chi\%}[Y_1 + Y_2]}{\text{Risk margin}_{\chi\%}[Y_1] + \text{Risk margin}_{\chi\%}[Y_2]} \times 100\%. \tag{4.15}$$

$\text{Risk Margin}_{75\%}$ and $\text{Risk Margin}_{95\%}$, as well as associated diversification benefits are provided in Table 5.9. It can then be observed that quite significant diversification benefits can be gained as a result of allowing for (non-comonotonic) dependence across business lines.

	Bodily Injury	Accident Benefits	Both lines	DB
Risk Margin$_{75\%}$	13,871.26	11,634.61	19,409.83	23.9%
Risk Margin$_{95\%}$	40,566.28	32,960.43	52,525.49	28.6%

Table 5.9: Risk margin and diversification benefits statistics

5.6 Remarks on unbalanced data and the proposed treatment

The application of common shock approaches to reserving data requires careful modelling. This is due to a number of challenges that arise from the unbalanced nature of reserving data. In particular, it is desirable to use scaling factors to adjust the contributions of the common shock proportionately to the total observations over the entire range of the triangles. However, it is also important to maintain model parsimony and distributional tractability in some cases.

In this chapter we propose a solution that compromises between the conflicting problems mentioned above. This solution involves using careful parametrisation to develop a common shock Tweedie framework. This framework is the modification of the common shock Tweedie framework developed in Chapters 3 and 4 for unbalanced data. The illustrations show a significant improvement in the performance of the modified framework. While the proposed solution does not provide a complete balance of common shock contributions over the entire range of the triangles, it reduces the disproportion in these contributions quite significantly. Model parsimony and distributional tractability are still maintained with this solution.

5.A Appendices

5.A.1 Simulated data set 1 and estimation results

Year	1	2	3	4	5	6	7	8	9	10
1	85.57	43.18	20.58	13.40	4.40	2.34	1.86	0.55	0.28	0.15
2	78.22	28.65	12.74	5.08	6.97	2.82	1.50	0.07	-0.01	
3	85.90	36.58	22.21	14.29	2.23	3.31	0.82	1.86		
4	67.86	36.94	16.01	11.23	5.54	4.68	1.40			
5	83.45	33.30	21.24	10.80	4.32	3.04				
6	63.85	39.38	24.71	2.84	7.77					
7	78.80	31.17	16.96	8.27						
8	90.32	36.19	13.56							
9	97.94	35.43								
10	58.14									

Table 5.10: Simulated triangle 1 (data set 1)

Year	1	2	3	4	5	6	7	8	9	10
1	24.12	38.93	45.70	43.19	16.04	8.70	4.78	1.83	1.45	1.66
2	21.04	40.05	35.83	19.93	15.27	11.21	6.84	2.81	1.12	
3	23.98	38.59	40.73	47.22	22.01	10.36	3.49	3.53		
4	26.34	42.48	57.27	29.72	24.03	12.11	1.86			
5	29.46	33.18	44.63	39.51	25.97	11.60				
6	23.67	48.70	49.66	20.12	21.34					
7	29.10	36.51	50.52	43.98						
8	30.58	53.40	49.21							
9	31.16	50.48								
10	31.04									

Table 5.11: Simulated triangle 2 (data set 1)

	True value	Median	SD	90% CI		True value	Median	SD	90% CI
$\ddot{\alpha}_2^{(1)}$	1.0300	0.8950	0.0756	(0.8250; 1.0610)	$\ddot{\alpha}_2^{(2)}$	1.1900	1.0620	0.1285	(0.8850; 1.3050)
$\ddot{\alpha}_3^{(1)}$	1.1900	1.2820	0.1523	(1.0530; 1.5560)	$\ddot{\alpha}_3^{(2)}$	1.1700	1.2300	0.1693	(0.9950; 1.5440)
$\ddot{\alpha}_4^{(1)}$	1.1200	1.0250	0.1190	(0.8590; 1.2460)	$\ddot{\alpha}_4^{(2)}$	1.1500	1.1020	0.1507	(0.8860; 1.3810)
$\ddot{\alpha}_5^{(1)}$	1.1500	1.0600	0.1200	(0.8790; 1.2700)	$\ddot{\alpha}_5^{(2)}$	1.1500	1.3190	0.1635	(1.0710; 1.6040)
$\ddot{\alpha}_6^{(1)}$	1.1600	1.1740	0.1468	(0.9650; 1.4460)	$\ddot{\alpha}_6^{(2)}$	1.2000	1.1390	0.1421	(0.9340; 1.3970)
$\ddot{\alpha}_7^{(1)}$	1.1200	1.0010	0.1159	(0.8530; 1.2270)	$\ddot{\alpha}_7^{(2)}$	1.4000	1.4480	0.1912	(1.1690; 1.7950)
$\ddot{\alpha}_8^{(1)}$	1.1400	1.0380	0.1219	(0.8690; 1.2700)	$\ddot{\alpha}_8^{(2)}$	1.4500	1.5000	0.2095	(1.1890; 1.8750)
$\ddot{\alpha}_9^{(1)}$	1.2100	1.1310	0.1343	(0.9350; 1.3760)	$\ddot{\alpha}_9^{(2)}$	1.5600	1.4820	0.1908	(1.2110; 1.8320)
$\ddot{\alpha}_{10}^{(1)}$	1.1900	1.0680	0.1879	(0.8530; 1.4470)	$\ddot{\alpha}_{10}^{(2)}$	1.6600	1.9840	0.5113	(1.3110; 2.9670)
$\ddot{\beta}_1^{(1)}$	60.0000	56.8610	4.2527	(49.8400; 63.7100)	$\ddot{\beta}_1^{(2)}$	10.0000	10.2800	1.1138	(8.5280; 12.1490)
$\ddot{\beta}_2^{(1)}$	20.0000	21.7210	1.9812	(18.6780; 25.1880)	$\ddot{\beta}_2^{(2)}$	20.0000	22.1790	2.0222	(18.9970; 25.6420)
$\ddot{\beta}_3^{(1)}$	10.0000	10.0160	1.3134	(8.1100; 12.4490)	$\ddot{\beta}_3^{(2)}$	25.0000	27.2130	2.5612	(23.1110; 31.5170)
$\ddot{\beta}_4^{(1)}$	5.0000	4.6610	0.7082	(3.5870; 5.8940)	$\ddot{\beta}_4^{(2)}$	20.0000	21.2330	2.1120	(17.9160; 24.8910)
$\ddot{\beta}_5^{(1)}$	2.5000	2.2920	0.2783	(1.8700; 2.7830)	$\ddot{\beta}_5^{(2)}$	15.0000	13.9600	1.6821	(11.4140; 16.8640)
$\ddot{\beta}_6^{(1)}$	1.2500	1.5100	0.2095	(1.2120; 1.892)	$\ddot{\beta}_6^{(2)}$	8.0000	6.5560	0.9920	(5.0850; 8.33707)
$\ddot{\beta}_7^{(1)}$	0.6000	0.7290	0.1453	(0.5320; 1.0040)	$\ddot{\beta}_7^{(2)}$	3.0000	2.7070	0.4708	(2.0370; 3.5830)
$\ddot{\beta}_8^{(1)}$	0.3000	0.2620	0.0449	(0.2010; 0.3460)	$\ddot{\beta}_8^{(2)}$	2.0000	1.9430	0.3335	(1.4350; 2.5340)
$\ddot{\beta}_9^{(1)}$	0.1500	0.1430	0.0272	(0.1040; 0.1930)	$\ddot{\beta}_9^{(2)}$	1.0000	0.7700	0.1122	(0.6180; 1.0100)
$\ddot{\beta}_{10}^{(1)}$	0.1500	0.2040	0.0445	(0.1440; 0.2860)	$\ddot{\beta}_{10}^{(2)}$	1.0000	0.9480	0.1997	(0.6580; 1.3140)
$\ddot{\phi}^{(1)}$	0.5000	0.5530	0.0649	(0.4560; 0.6680)	$\ddot{\phi}^{(2)}$	0.7000	0.6220	0.0936	(0.4880; 0.7930)
p	1.3000	1.3640	0.0507	(1.2850; 1.4510)	$\xi^{(1)}$	0.0100	0.0120	0.0013	(0.0100; 0.0140)
Λ	1.0118	1.0420	0.1293	(0.8470; 1.2710)					
ζ	0.500	0.3620	0.6942	(0.0600; 2.2370)	$\tilde{\phi}$	0.600	0.5030	0.4603	(0.1610; 1.6020)

Table 5.12: Posterior statistics of parameters (data set 1)

5.A.2 Simulated data set 2

Year	1	2	3	4	5	6	7	8	9	10
1	47.16	36.33	18.58	10.63	3.93	0.38	0.45	0.00	0.00	0.13
2	102.30	49.12	18.47	17.05	3.30	1.70	1.77	1.04	0.00	
3	101.87	56.91	14.75	24.29	1.46	1.15	0.83	0.17		
4	97.09	35.96	27.80	10.86	3.93	3.71	0.43			
5	107.07	34.34	20.49	19.73	6.65	1.70				
6	107.10	66.55	27.03	17.09	2.38					
7	123.60	37.41	32.77	15.53						
8	107.03	50.50	18.30							
9	105.93	42.02								
10	109.09									

Table 5.13: Simulated triangle 1 (data set 2)

Year	1	2	3	4	5	6	7	8	9	10
1	19.61	45.29	44.23	28.82	24.48	3.15	3.23	3.30	1.94	0.73
2	33.52	41.13	39.33	41.78	22.46	8.69	2.09	4.85	1.88	
3	24.39	43.40	34.06	59.94	22.00	13.90	5.54	1.62		
4	27.78	37.03	41.41	31.12	31.73	5.92	7.69			
5	24.46	41.96	36.55	23.42	20.88	9.61				
6	26.36	38.68	58.52	36.25	27.15					
7	30.05	36.18	52.14	41.98						
8	30.32	53.54	52.87							
9	42.37	42.25								
10	46.49									

Table 5.14: Simulated triangle 2 (data set 2)

5.A.3 Empirical data set

This data set is drawn from Côté et al. (2016).

Year	Premium	1	2	3	4	5	6	7	8	9	10
1	85,421	3,488	14,559	27,249	37,979	49,561	55,957	58,406	60,862	63,280	63,864
2	98,579	1,169	12,781	20,550	31,547	42,808	47,385	50,251	50,978	51,272	
3	103,062	1,478	10,788	25,499	34,279	43,057	49,360	52,329	52,544		
4	108,412	1,186	11,852	22,913	32,537	41,824	48,005	52,542			
5	111,176	1,737	13,881	25,521	38,037	43,684	47,755				
6	112,050	1,571	12,153	27,329	41,832	51,779					
7	112,577	1,199	17,077	29,876	44,149						
8	113,707	1,263	16,073	28,249							
9	126,442	986	10,003								
10	130,484	683									

Table 5.15: Bodily Injury line (cumulative claims)

Year	Premium	1	2	3	4	5	6	7	8	9	10
1	116,491	13,714	24,996	31,253	38,352	44,185	46,258	47,019	47,894	48,334	48,902
2	111,467	6883	16,525	24,796	29,263	32,619	33,383	34,815	35,569	35,612	
3	107,241	7933	22,067	32,801	38,028	44,274	44,948	46,507	46,665		
4	105,687	7052	18,166	25,589	31,976	36,092	38,720	39,914			
5	105,923	10,463	23,982	31,621	36,039	38,070	41,260				
6	111,487	9697	28,878	41,678	47,135	50,788					
7	113,268	11,387	37,333	48,452	55,757						
8	121,606	12,150	32,250	40,677							
9	110,610	5348	14,357								
10	104,304	4,612									

Table 5.16: Accident Benefits (cumulative claims)

CHAPTER 6

A multivariate evolutionary GLM framework

6.1 Introduction

The overall aim of our research is to develop models that incorporate realistic and desirable model features. This is to improve the valuation of outstanding claims as well as the practicality of models. Many features are incorporated and considered in the developments in the previous chapters, including the dependency across segments, marginal flexibility, explicit dependence structures, tractable moments, and the unbalanced nature of reserving data. Additionally, insurers also typically experience changes in claims activity over time and this complicates the prediction of future outstanding claims (see also Sections 1.1.3 and 2.5.1). In such cases, the assumption of similar claims development patterns across accident periods in static models (i.e. models with deterministic parameters such as those in the previous chapters) is often invalidated. Actuarial judgements or changes in the algebraic model structures are often required (De Jong and Zehnwirth, 1983; Taylor et al., 2003; Gluck and Venter, 2009; Sims, 2012).

An elegant and plausible solution for modelling portfolios with changing claim activities are evolutionary models (De Jong and Zehnwirth, 1983; Zehnwirth, 1994; Gluck and Venter, 2009; Taylor and McGuire, 2009). These models incorporate the changes naturally to produce smooth estimates of outstanding claims liabilities over time without many subjective judgements (Sims, 2012). Model factors are not simply randomised, but change over time in a recursive manner. In addition, evolutionary models are usually accompanied by filtering processes, real-time devices that enable the adaptation of changes in the estimation of model

factors. More weight is given to more recent information in these processes, hence the estimation and prediction for immature accident periods can be performed more accurately with less reliance on judgements (Taylor, 2000; Alpuim and Ribeiro, 2003). Evolutionary models can also be used to construct reserving robots which automate repetitive valuation jobs. They are particularly useful, or even essential, when the valuation for portfolios of a large number of segments are required on a frequent basis such as quarterly or even monthly. These many benefits of evolutionary models are covered in detail in Sections 1.1.3 and 2.5.1. A number of evolutionary models have been introduced in the reserving literature, which mainly focus on a single business segment. A review of these models is provided in Section 2.5.

In the loss reserving literature, the EDF and its sub-class, the Tweedie family of distributions, have been used frequently in univariate as well as multivariate models (see also the literature review in Chapter 2). Models using the EDF are usually specified using the GLM framework which allows a flexible incorporation of covariates. Hence a natural step toward evolutionary modelling is to allow parameters/covariates in the GLM framework to evolve. As also mentioned in the previous chapters, the dependency amongst business segments can exist due to various reasons such as legislative changes, or common calendar period factors. Common shock approaches are a candidate for dependence modelling in reserving with many benefits (see also Section 2.4.4). These inspire the development of a multivariate evolutionary GLM framework in this chapter with the use of a common shock approach for dependence modelling. We are also motivated to formulate filtering approaches that provide recursive real-time updates of random factors in this framework without using the whole history of information.

The structure and specifications of the multivariate evolutionary GLM framework are described in Section 6.2. Filtering processes and parameter estimation that take into consideration features of reserving data are provided in Section 6.3. Two filters are introduced in this section: a particle filter for the evolutionary GLM framework, and a dual Kalman filter for Gaussian cases. Simulation illustrations are given in Section 6.4. An illustration using real data from a Canadian insurer is provided in Section 6.5. Remarks on properties and the estimation of the framework are then given in Section 6.6.

6.2 Framework development

In this section we introduce a multivariate evolutionary GLM framework in which parameters evolve over time. Section 6.2.1 provides the structure and specifications of the framework. The state space matrix representation of the framework is provided in Section 6.2.2. A special case of the framework, Gaussian models, are described in Section 6.2.3.

6.2.1 Structure and specifications

As with a typical state space model (see Section 2.5), the multivariate evolutionary GLM framework also has two components: the observation component and the state component. The observation component specifies the relation between observations and latent random factors. The state component specifies the evolution/dynamics of random factors.

6.2.1.1 Observation component

We assume that the incremental claim $Y_{i,j}^{(n)}$ follows a distribution from the EDF

$$Y_{i,j}^{(n)} \sim \text{EDF}\left(\theta_{i,j}^{(n)}, \phi^{(n)}\right), \tag{6.1}$$

with the density

$$f_{Y_{i,j}^{(n)}}\left(y_{i,j}^{(n)}; \theta_{i,j}^{(n)}, \phi^{(n)}\right) = v\left(y_{i,j}^{(n)}, \phi^{(n)}\right) \exp\left\{\frac{y_{i,j}^{(n)}\theta_{i,j}^{(n)} - \kappa\left(\theta_{i,j}^{(n)}\right)}{\phi^{(n)}}\right\}, \tag{6.2}$$

where $\theta_{i,j}^{(n)}$ is the canonical parameter, $\phi^{(n)}$ is the dispersion parameter, $\kappa(.)$ is the unit cumulant function, and $v(.)$ is a specified function which corresponds to the distribution used.

It then follows that

$$E[Y_{i,j}^{(n)}] = \mu_{i,j}^{(n)} = \kappa'\left(\theta_{i,j}^{(n)}\right), \tag{6.3}$$

$$Var[Y_{i,j}^{(n)}] = \phi^{(n)}\kappa''\left(\theta_{i,j}^{(n)}\right). \tag{6.4}$$

When $\kappa''\left(\theta_{i,j}^{(n)}\right) = \left(\mu_{i,j}^{(n)}\right)^p$ we have a distribution from the Tweedie sub-family with the power parameter p.

We use a modified Hoerl curve with a calendar period effect to specify the mean structure. There are various benefits of using the Hoerl curve. These include parsimonious modelling, robustness against fluctuations in observations, and extrapolation beyond the range of the observed development period. A review of its many benefits as well as its applications in reserving can be found in Section 2.3.3. The Hoerl curve has been used to specify the mean structures in many evolutionary reserving models (see Section 2.5). It is a smoothing curve used to approximate the claims development pattern, hence it allows a systematic change in the development pattern over time as its parameters change.

We have the following Hoerl curve mean structure with a log-link

$$\log(\mu_{i,j}^{(n)}) = a_i^{(n)} + r_i^{(n)} \log(j) + s_i^{(n)} j + h_{t=i+j-1}^{(n),} \tag{6.5}$$

where $a_i^{(n)}$ is the accident period effect, $r_i^{(n)}$ and $s_i^{(n)}$ are parameters of the Hoerl curve that specifies the development pattern, and $h_t^{(n)}$ is the calendar period effect. Factors $a_i^{(n)}$, $r_i^{(n)}$, $s_i^{(n)}$ are accident period-dependent, and factor $h_t^{(n)}$ is calendar period-dependent. They are all evolutionary and they evolve within their respective dimension. It is worth noting that the above mean structure as well as the link function can be modified to suit the data set under study.

6.2.1.2 State component

In the multivariate evolutionary GLM framework, factors $a_i^{(n)}$, $r_i^{(n)}$, $s_i^{(n)}$ and $h_t^{(n)}$ are random and they evolve over time. Their evolution can be specified using time series processes such as Autoregressive-Moving-Average (ARMA) or AR processes. For simplicity, we use random walk processes for states evolution

$$a_i^{(n)} = a_{i-1}^{(n)} + {}_a\epsilon_i^{(n)}, \qquad {}_a\epsilon_i^{(n)} \sim \text{Normal}\left(0, \sigma^2_{a\epsilon^{(n)}}\right), \tag{6.6}$$

$$r_i^{(n)} = r_{i-1}^{(n)} + {}_r\epsilon_i^{(n)}, \qquad {}_r\epsilon_i^{(n)} \sim \text{Normal}\left(0, \sigma^2_{r\epsilon^{(n)}}\right), \tag{6.7}$$

$$s_i^{(n)} = s_{i-1}^{(n)} + {}_s\epsilon_i^{(n)}, \qquad {}_s\epsilon_i^{(n)} \sim \text{Normal}\left(0, \sigma^2_{s\epsilon^{(n)}}\right), \tag{6.8}$$

where $\sigma^2_{a\epsilon^{(n)}}$, $\sigma^2_{r\epsilon^{(n)}}$, $\sigma^2_{s\epsilon^{(n)}}$ are variances of the disturbance terms in the evolution that often

need to be estimated.

In many cases, the dependence across business segments arises from some calendar period factors that affect claims in the same calendar period within and across segments simultaneously (Shi et al., 2012; De Jong, 2012; Wüthrich, 2010). For example, a legislative change in a particular calendar period can speed up the claims settlement processes in all business segments. A subset of the existing literature on multivariate reserving focuses on modelling calendar period dependence (see also the review in Section 2.4). We specify the evolution of the calendar period factor to also allow for this source of dependence

$$h_t^{(n)} = h_{t-1}^{(n)} + {}_h\epsilon_t^{(n)} + {}_\epsilon\lambda^{(n)} \cdot {}_h\tilde{\epsilon}_t, \tag{6.9}$$

$${}_h\epsilon_t^{(n)} \sim \text{Normal}\left(0, \sigma^2_{{}_h\epsilon^{(n)}}\right), \tag{6.10}$$

$${}_h\tilde{\epsilon}_t \sim \text{Normal}\left(0, \sigma^2_{{}_h\tilde{\epsilon}}\right). \tag{6.11}$$

There are two sources of disturbance in this evolution: the segment-specific disturbance ${}_h\epsilon_t^{(n)}$ and the common shock disturbance ${}_h\tilde{\epsilon}_t$. The variances of these terms, $\sigma^2_{{}_h\epsilon^{(n)}}$ and $\sigma^2_{{}_h\tilde{\epsilon}}$, often need to be estimated.

The calendar period dependence is deduced by the common shock term ${}_h\tilde{\epsilon}_t$. This term can represent any changes in the calendar period t that affect all segments simultaneously. The effects of this common shock on each segment, however, are usually not uniform as some segments may be more heavily affected than others. In addition, the calendar factors in different segments may vary in size, or in other words, have an unbalanced nature. It is then desirable to use the scaling factors ${}_\epsilon\lambda^{(n)}$. These factors aim to adjust the effects of the common shock on individual segments so that they are consistent with practitioners' experience. They are also used to mitigate issues of unbalanced data. This means ensuring that the contributions of the common shock to the calendar factors from individual segments are proportionate to the size of these factors themselves, as also discussed in Chapter 5.

The evolution of model factors in a single segment can be summarised in Figure 6.1. This figure shows two groups of factors: $a_i^{(n)}$, $r_i^{(n)}$, $s_i^{(n)}$ (in black) which evolve with accident period i (black arrows), and $h_t^{(n)}$ (in red) which evolves with calendar period t (red arrows). In the first accident period, all calendar factors $h_1^{(n)}, ..., h_I^{(n)}$ are present and their evolution within the first row follows Equation (6.9). These factors are mapped one to one with calendar factors in the second accident period $h_2^{(n)}, .., h_I^{(n)}$. This is because all claims within the same

diagonal are affected by the same calendar factor.

The model structure can be modified to capture other types of dependence such as accident period dependence, development period dependence, or a combination of these types. Different time series processes can also be used to specify the evolution of model factors.

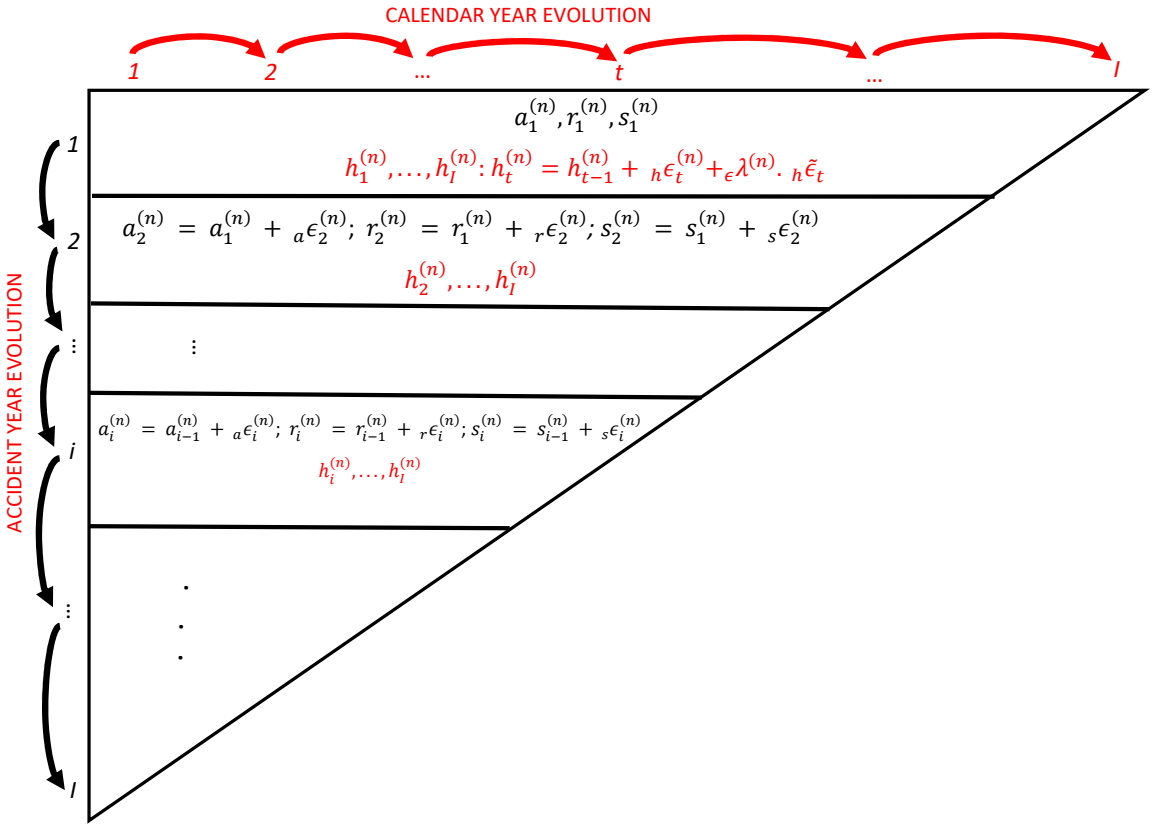

Figure 6.1: Evolution of factors in the multivariate evolutionary GLM framework

6.2.2 State space matrix representation

Evolutionary models are often presented using matrices for convenience in the model set-up and estimation (Section 2.5.2 and 2.5.4). We present a matrix representation of the above framework in this section.

6.2.2.1 Observation component

We consider each accident period as a time period when new observations arrive. The vector of observations at each time period is then a vector of all claims in the same accident

period within and across triangles

$$\boldsymbol{Y}_i = \begin{pmatrix} Y_{i,1}^{(1)} \\ \vdots \\ Y_{i,I-i+1}^{(1)} \\ Y_{i,1}^{(2)} \\ \vdots \\ Y_{i,I-i+1}^{(N)} \end{pmatrix}. \tag{6.12}$$

This vector has the distribution

$$\boldsymbol{Y}_i \sim \text{EDF}\left(\boldsymbol{\theta}_i, \boldsymbol{\phi}\right), \tag{6.13}$$

where

$$\boldsymbol{\theta}_i = \begin{pmatrix} \theta_{i,1}^{(1)} \\ \vdots \\ \theta_{i,I-i+1}^{(1)} \\ \theta_{i,1}^{(2)} \\ \vdots \\ \theta_{i,I-i+1}^{(N)} \end{pmatrix}, \qquad \boldsymbol{\phi} = \begin{pmatrix} \phi^{(1)} \\ \vdots \\ \phi^{(1)} \\ \phi^{(2)} \\ \vdots \\ \phi^{(N)} \end{pmatrix} \begin{matrix} \\ \\ \Big\} (I-i+1) \text{ rows} \\ \\ \\ \\ \end{matrix}. \tag{6.14}$$

Using properties of the EDF, the mean structure is specified such that

$$E[\boldsymbol{Y}_i] = \boldsymbol{\mu}_i = \kappa'\left(\boldsymbol{\theta}_i\right), \tag{6.15}$$

with a log-link that relates it to a linear predictor

$$\log\left(\boldsymbol{\mu}_i\right) = \boldsymbol{A}_i\boldsymbol{\gamma}_i + \boldsymbol{E}_i\boldsymbol{\psi}_I. \tag{6.16}$$

In this structure we specify

$$\boldsymbol{\gamma}_i^{(n)} = \begin{pmatrix} a_i^{(n)} \\ r_i^{(n)} \\ s_i^{(n)} \end{pmatrix}, \qquad \boldsymbol{\gamma}_i = \begin{pmatrix} \boldsymbol{\gamma}_i^{(1)} \\ \vdots \\ \boldsymbol{\gamma}_i^{(N)} \end{pmatrix}, \tag{6.17}$$

$$\boldsymbol{\psi}_I^{(n)} = \begin{pmatrix} h_1^{(n)} \\ \vdots \\ h_I^{(n)} \end{pmatrix}, \qquad \boldsymbol{\psi}_I = \begin{pmatrix} \boldsymbol{\psi}_I^{(1)} \\ \vdots \\ \boldsymbol{\psi}_I^{(N)} \end{pmatrix}, \tag{6.18}$$

$$\boldsymbol{A}_i^{(n)} = \begin{pmatrix} 1 & \log(1) & 1 \\ 1 & \log(2) & 2 \\ \vdots & \vdots & \vdots \\ 1 & \log(I-i+1) & I-i+1 \end{pmatrix}, \qquad \boldsymbol{A_i} = \begin{pmatrix} \boldsymbol{A}_i^{(1)} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_i^{(2)} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{A}_i^{(N)} \end{pmatrix}, \tag{6.19}$$

$$\boldsymbol{E}_i^{(n)} = \left.\begin{pmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & 1 \end{pmatrix}\right\}(I-i+1) \text{ rows} \quad , \boldsymbol{E}_i = \begin{pmatrix} \boldsymbol{E}_i^{(1)} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{E}_i^{(2)} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{E}_i^{(N)} \end{pmatrix}. \tag{6.20}$$

$$\underbrace{}_{(i-1) \text{ cols}} \quad \underbrace{}_{(I-i+1) \text{ cols}}$$

6.2.2.2 State component

We now present the evolution of random factors using matrices. From the model structure, the evolution of $\boldsymbol{\gamma}_i$ can be represented as

$$\boldsymbol{\gamma}_i = \boldsymbol{\gamma}_{i-1} + {}_{\gamma}\boldsymbol{\epsilon}_i, \qquad {}_{\gamma}\boldsymbol{\epsilon}_i \sim \text{Normal}(\mathbf{0}, \boldsymbol{Q}_{\gamma\epsilon}), \tag{6.21}$$

where

$$\boldsymbol{{}_{\gamma}\epsilon_i^{(n)}} = \begin{pmatrix} {}_{a}\epsilon_i^{(n)} \\ {}_{r}\epsilon_i^{(n)} \\ {}_{s}\epsilon_i^{(n)} \end{pmatrix}, \qquad {}_{\gamma}\boldsymbol{\epsilon}_i = \begin{pmatrix} {}_{\gamma}\boldsymbol{\epsilon}_i^{(1)} \\ \vdots \\ {}_{\gamma}\boldsymbol{\epsilon}_i^{(N)} \end{pmatrix}, \tag{6.22}$$

$$\boldsymbol{Q}_{\gamma\epsilon}^{(n)} = \begin{pmatrix} \sigma^2_{{}_{a}\epsilon^{(n)}} & 0 & 0 \\ 0 & \sigma^2_{{}_{r}\epsilon^{(n)}} & 0 \\ 0 & 0 & \sigma^2_{{}_{s}\epsilon^{(n)}} \end{pmatrix}, \qquad \boldsymbol{Q}_{\gamma\epsilon} = \begin{pmatrix} \boldsymbol{Q}_{\gamma\epsilon}^{(1)} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}_{\gamma\epsilon}^{(2)} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{Q}_{\gamma\epsilon}^{(N)} \end{pmatrix}. \tag{6.23}$$

For calendar factors, we have

$$\boldsymbol{\psi}_t = \boldsymbol{R}_{t-1}\boldsymbol{\psi}_{t-1} + \boldsymbol{S}_{t-1} \cdot {}_h\boldsymbol{\epsilon}_t, \qquad {}_h\boldsymbol{\epsilon}_t \sim \text{Normal}(\mathbf{0}, \boldsymbol{Q}_{{}_h\epsilon}) \tag{6.24}$$

where

$$\boldsymbol{\psi}_t^{(n)} = \begin{pmatrix} h_1^{(n)} \\ \vdots \\ h_t^{(n)} \end{pmatrix}, \qquad \boldsymbol{\psi}_t = \begin{pmatrix} \boldsymbol{\psi}_t^{(1)} \\ \vdots \\ \boldsymbol{\psi}_t^{(N)} \end{pmatrix}, \tag{6.25}$$

$${}_h\boldsymbol{\epsilon}_t = \begin{pmatrix} {}_h\epsilon_t^{(1)} \\ \vdots \\ {}_h\epsilon_t^{(N)} \\ {}_h\tilde{\epsilon}_t \end{pmatrix}, \qquad \boldsymbol{Q}_{{}_h\epsilon} = \begin{pmatrix} \sigma^2_{{}_h\epsilon^{(1)}} & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \sigma^2_{{}_h\epsilon^{(N)}} & 0 \\ 0 & \dots & 0 & \sigma^2_{{}_h\tilde{\epsilon}} \end{pmatrix}, \tag{6.26}$$

$$\boldsymbol{R}_{t-1}^{(n)} = \underbrace{\left.\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ 0 & 0 & \dots & 1 \end{pmatrix}\right\} t \text{ rows}}_{(t-1) \text{ cols}}, \; \boldsymbol{R}_{t-1} = \begin{pmatrix} \boldsymbol{R}_{t-1}^{(1)} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_{t-1}^{(2)} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{R}_{t-1}^{(N)} \end{pmatrix}, \tag{6.27}$$

$$\boldsymbol{S}_{t-1}^{(n)} = \left.\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}\right\} t \text{ rows}, \qquad {}_\epsilon\tilde{\boldsymbol{\lambda}}^{(n)} = \left.\begin{pmatrix} 0 \\ \vdots \\ 0 \\ {}_\epsilon\lambda^{(n)} \end{pmatrix}\right\} t \text{ rows} \tag{6.28}$$

$$\boldsymbol{S}_{t-1} = \begin{pmatrix} \boldsymbol{S}_{t-1}^{(1)} & \mathbf{0} & \dots & \mathbf{0} & {}_\epsilon\tilde{\boldsymbol{\lambda}}^{(1)} \\ \mathbf{0} & \boldsymbol{S}_{t-1}^{(2)} & \dots & \mathbf{0} & {}_\epsilon\tilde{\boldsymbol{\lambda}}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{S}_{t-1}^{(N)} & {}_\epsilon\tilde{\boldsymbol{\lambda}}^{(N)} \end{pmatrix}. \tag{6.29}$$

6.2.3 Special cases: Gaussian models

In Gaussian models which are special cases of the multivariate evolutionary GLM framework, Gaussian assumptions are applied on random factors and observations (where the observations can be either on the original scale or the log scale). The observations equation can be written as

$$\boldsymbol{Y}_i = \boldsymbol{A}_i\boldsymbol{\gamma}_i + \boldsymbol{E}_i\boldsymbol{\psi}_I + \boldsymbol{\varsigma}_i, \qquad \boldsymbol{\varsigma}_i \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{H}_i), \tag{6.30}$$

where

$$\boldsymbol{\varsigma}_i = \begin{pmatrix} \varsigma_{i,1}^{(1)} \\ \vdots \\ \varsigma_{i,I-i+1}^{(1)} \\ \varsigma_{i,1}^{(2)} \\ \vdots \\ \varsigma_{i,I-i+1}^{(N)} \end{pmatrix}, \tag{6.31}$$

$$\boldsymbol{H}_i^{(n)} = \begin{pmatrix} \sigma^2_{\varsigma^{(n)}} & 0 & \dots & 0 \\ 0 & \sigma^2_{\varsigma^{(n)}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \underbrace{0 \quad 0 \quad \dots}_{(I-i+1)\text{ cols}} & & & \sigma^2_{\varsigma^{(n)}} \end{pmatrix}, \qquad \boldsymbol{H}_i = \begin{pmatrix} \boldsymbol{H}_i^{(1)} & \boldsymbol{0} & \dots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}_i^{(2)} & \dots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{H}_i^{(N)} \end{pmatrix}. \tag{6.32}$$

The evolution of random factors is specified in the same way as in the general framework,

$$\boldsymbol{\gamma}_i = \boldsymbol{\gamma}_{i-1} + {}_{\gamma}\boldsymbol{\epsilon}_i, \qquad {}_{\gamma}\boldsymbol{\epsilon}_i \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{Q}_{\gamma\epsilon}), \tag{6.33}$$

$$\boldsymbol{\psi}_t = \boldsymbol{R}_{t-1}\boldsymbol{\psi}_{t-1} + \boldsymbol{S}_{t-1} \cdot {}_{h}\boldsymbol{\epsilon}_t, \qquad {}_{h}\boldsymbol{\epsilon}_t \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{Q}_{h\epsilon}). \tag{6.34}$$

6.3 Estimation

In this section we discuss the estimation of random factors as well as unknown parameters in the framework. The random factors include $\boldsymbol{\gamma}_i$ (for $\forall i$) and $\boldsymbol{\psi}_I$, and the

unknown parameters are specified using a vector

$$\boldsymbol{\Theta} = \{\boldsymbol{\phi}\left(\boldsymbol{H}_i\right), \boldsymbol{Q}_{\gamma\epsilon}, \boldsymbol{Q}_{h\epsilon}, {}_{\epsilon}\lambda^{(1)}, ..., {}_{\epsilon}\lambda^{(N)}\}. \tag{6.35}$$

We consider the on-line estimation of random factors which recursively updates the factors estimates upon the arrival of new observations (for a description of on-line estimation in state space models, see also Section 2.5.4.2). The recursive estimation of random factors and parameters is also called the filtering process, and the estimates of factors and parameters from this process are often called filtered estimates.

Section 6.3.1 describes a particle learning approach for the multivariate evolutionary GLM framework. This approach is simulation-based and incorporates the on-line estimation of parameters into the filtering of random factors. When Gaussian models are used, a closed-form filter can be used for the estimation of random factors which is called a dual Kalman filter. This filter is described in Section 6.3.2.

6.3.1 Particle learning approach

Particle learning approaches are extensions of the traditional particle filtering to also incorporate the estimation of parameters. They are also called on-line estimation approaches as they update parameters in a sequential manner upon the arrival of new observations (Lopes and Tsay, 2011). This is in contrast to off-line estimation approaches where the estimation of parameters is performed after all observations have been received (Kantas et al., 2009). By providing continuing updates of parameters, particle learning approaches allow them to be traced and used in a timely manner. There have been a number of particle learning approaches developed in the literature. For our framework, we use the Liu and West filter by Liu and West (2001). This is a very popular filter that has been used in various fields including physics, engineering, and more. A review of particle filtering, parameter estimation techniques and the Liu and West filter can be found in Sections 2.5.4.1 and 2.5.4.2.

In the Liu and West filter, the estimates of random factors as well as parameters are updated at each time step using new observations. While random factors evolve over time, parameters are indeed static. To allow them to be updated in the same manner as random factors, artificial dynamics are added to their specification in a specific way. If this step was not included, parameter estimates obtained in the previous time period would be outdated

as new observations arrive. As a result, the number of significant samples would deteriorate, which is also called the degeneracy problem (Andrieu et al., 2012; Kantas et al., 2015).

In the multivariate evolutionary GLM framework that we introduced in section 6.2 the random factors are $\boldsymbol{\gamma}_i$ (which contains accident and development effects), and $\boldsymbol{\psi}_I$ which is a vector of random calendar effects. These factors, however, evolve in different dimensions as shown in Figure 6.1. Factors $\boldsymbol{\gamma}_i$ evolve by accident periods, while $h_t^{(n)}$ evolves from one calendar period to another. This two dimensional evolution of factors is unconventional as a traditional state space model typically considers the evolution in a single time dimension. This makes the filtering of random factors as well as unknown parameters not as straight forward as it in a standard model.

To address this problem, we treat calendar factors in the same way as parameters in the filtering process. This is because all calendar factors are already present when we initiate the estimation/filter from the first accident period. These factors do not evolve as we proceed to subsequent accident periods. Hence, their nature is static as the filter runs within the dimension of accident periods. This is the same as the nature of parameters in the framework.

By having a Bayesian inference for parameter estimation, parameter errors can also be assessed. The prior distribution of parameters at the initialisation $f_{\boldsymbol{\Theta}_1}(\boldsymbol{\Theta}_1^{(m)})$ can be chosen to reflect the level of knowledge of these parameters. Modifying the Liu and West filter for our framework to also incorporate the estimation of calendar factors, we can proceed as follows.

Particle learning algorithm

Step 1. Initialisation: At $i = 1$, for $m = 1, ..., M$, draw parameters from their prior densities

$$\boldsymbol{\Theta}_1^{(m)} \sim f_{\boldsymbol{\Theta}_1}(\boldsymbol{\Theta}_1^{(m)}). \tag{6.36}$$

Draw calendar factors $\boldsymbol{\psi}_{I,1}^{(m)}$ using their specification

$$\boldsymbol{\psi}_{t,1}^{(m)} = \boldsymbol{R}_{t-1}\boldsymbol{\psi}_{t-1,1}^{(m)} + \boldsymbol{S}_{t-1}^{(m)} \cdot {}_h\boldsymbol{\epsilon}_t, \qquad {}_h\boldsymbol{\epsilon}_t \sim \text{Normal}(\mathbf{0}, \boldsymbol{Q}_{h\epsilon}^{(m)}). \tag{6.37}$$

where $\boldsymbol{S}_{t-1}^{(m)}$ and $\boldsymbol{Q}_{h\epsilon}^{(m)}$ are specified using $\boldsymbol{\Theta}_1^{(m)}$. The vector $\boldsymbol{\psi}_{I,1}^{(m)}$ represents the m^{th} sample of all calendar factors at time 1.

Draw initial samples of other factors

$$\boldsymbol{\gamma}_1^{(m)} \sim f_{\boldsymbol{\gamma}_1}(\boldsymbol{\gamma}_1; \boldsymbol{\Theta}_1^{(m)}). \tag{6.38}$$

Calculate the importance weights

$$\omega_1^{(m)} = f_{\boldsymbol{Y}_1|\boldsymbol{\gamma}_1,\boldsymbol{\psi}_I,\boldsymbol{\Theta}_1}(\boldsymbol{y}_1|\boldsymbol{\gamma}_1^{(m)}; \boldsymbol{\psi}_{I,1}^{(m)}, \boldsymbol{\Theta}_1^{(m)}). \tag{6.39}$$

For $i = 2, ..., I$ and $m = 1, ..., M$:

Step 2. Compute

$$\widehat{\boldsymbol{\Theta}}_i^{(m)} = \xi \boldsymbol{\Theta}_{i-1}^{(m)} + (1-\xi)\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\Theta}_{i-1}^{(m)}, \tag{6.40}$$

$$\widehat{\boldsymbol{\psi}}_{I,i}^{(m)} = \xi \boldsymbol{\psi}_{I,i-1}^{(m)} + (1-\xi)\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\psi}_{I,i-1}^{(m)}, \tag{6.41}$$

where ξ is a shrinkage coefficient. The vector $\widehat{\boldsymbol{\psi}}_{I,i}^{(m)}$ represents the m^{th} look-ahead sample of all calendar factors at time i.

Also compute

$$\widehat{\boldsymbol{\gamma}}_i^{(m)} = E[\boldsymbol{\gamma}_i|\boldsymbol{\gamma}_{i-1}^{(m)}, \boldsymbol{\Theta}_{i-1}^{(m)}]. \tag{6.42}$$

Step 3. Compute the look-ahead importance weights

$$\omega_i^{(m)} = \omega_{i-1}^{(m)} f_{\boldsymbol{Y}_i|\boldsymbol{\gamma}_i,\boldsymbol{\psi}_I,\boldsymbol{\Theta}_i}(\boldsymbol{y}_i|\widehat{\boldsymbol{\gamma}}_i^{(m)}, \widehat{\boldsymbol{\psi}}_{I,i}^{(m)}; \widehat{\boldsymbol{\Theta}}_i^{(m)}). \tag{6.43}$$

Normalise the importance weights

$$\tilde{\omega}_i^{(m)} = \frac{\omega_i^{(m)}}{\sum_{m=1}^{M} \omega_i^{(m)}}. \tag{6.44}$$

Step 4. Re-sample M particles $\{\boldsymbol{\gamma}_{i-1}^{(m)}, \widehat{\boldsymbol{\psi}}_{I,i}^{(m)}; \widehat{\boldsymbol{\Theta}}_i^{(m)}\}_{m=1}^{M}$ with probabilities $\{\widehat{\omega}_i^{(m)}\}_{m=1}^{M}$.

Step 5. Draw

$$\boldsymbol{\Theta}_i^{(m)} \sim \text{Normal}(\widehat{\boldsymbol{\Theta}}_i^{(m)}, (1-\xi^2)\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\Theta}_{i-1}}), \tag{6.45}$$

$$\boldsymbol{\psi}_{I,i}^{(m)} \sim \text{Normal}(\widehat{\boldsymbol{\psi}}_{I,i}^{(m)}, (1-\xi^2)\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\psi}_{I,i-1}}), \tag{6.46}$$

where $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\Theta}_{i-1}}$ is the sample covariance matrix of $\{\boldsymbol{\Theta}_{i-1}^{(m)}\}_{m=1}^{M}$, and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\psi}_{I,i-1}}$ is the sample covariance matrix of $\{\boldsymbol{\psi}_{I,i-1}^{(m)}\}_{m=1}^{M}$.

Also sample

$$\boldsymbol{\gamma}_i^{(m)} \sim f_{\boldsymbol{\gamma}_i|\boldsymbol{\gamma}_{i-1}}(\boldsymbol{\gamma}_i|\boldsymbol{\gamma}_{i-1}^{(m)}; \boldsymbol{\Theta}_i^{(m)}). \tag{6.47}$$

Step 6. Calculate the importance weights

$$\omega_i^{(m)} = \frac{f_{\boldsymbol{Y}_i|\boldsymbol{\gamma}_i,\boldsymbol{\psi}_I,\boldsymbol{\Theta}_i}(\boldsymbol{y}_i|\boldsymbol{\gamma}_i^{(m)}, \boldsymbol{\psi}_{I,i}^{(m)}; \boldsymbol{\Theta}_i^{(m)})}{f_{\boldsymbol{Y}_i|\boldsymbol{\gamma}_i,\boldsymbol{\psi}_I,\boldsymbol{\Theta}_i}(\boldsymbol{y}_i|\widehat{\boldsymbol{\gamma}}_i^{(m)}, \widehat{\boldsymbol{\psi}}_{I,i}^{(m)}; \widehat{\boldsymbol{\Theta}}_i^{(m)})}. \tag{6.48}$$

Normalise the importance weights

$$\tilde{\omega}_i^{(m)} = \frac{\omega_i^{(m)}}{\sum_{m=1}^{M} \omega_i^{(m)}}. \tag{6.49}$$

Step 7. Repeat steps 2-6 until $i = I$.

6.3.2 Dual Kalman filtering approach for Gaussian models

For Gaussian models, a (modified) Kalman filter can be used to recursively estimate random factors. As mentioned in the previous section, due to the calendar period factors h_t which behave differently to other factors, the traditional Kalman filter cannot be applied without adjustments. Using a similar treatment as with the one in the particle learning approach, we also consider calendar factors as static parameters to be updated beyond the first accident period. A modified version of the Kalman filter in the literature that fits well to this purpose is called the dual Kalman filter. It was developed by Nelson and Stear (1976) and has been used to provide sequential estimates of dynamic factors as well as static factors or parameters of Gaussian models in various fields, including civil engineering (Azam et al., 2015), vehicle systems (Wenzel et al., 2006), science (Gove and Hollinger, 2006), and others.

The dual Kalman filter involves two filters that run in parallel, one for static factors, and one for random factors. The information from one filter flows into the other for continuing updates. Applying a dual Kalman filter to Gaussian cases of our framework, we can proceed as follows.

Dual Kalman filter algorithm

Step 1. Initialisation: At $i = 1$, obtain initial estimates of calendar factors

$$\widehat{\boldsymbol{\psi}}_{I,1|0} = E[\boldsymbol{\psi}_{I,1}], \tag{6.50}$$

$$_h\widehat{\boldsymbol{P}}_{1|0} = Cov[\boldsymbol{\psi}_{I,1}]. \tag{6.51}$$

These can be obtained by simulating N samples of $\boldsymbol{\psi}_{I,1}$ using their specification

$$\boldsymbol{\psi}_{t,1} = \boldsymbol{R}_{t-1}\boldsymbol{\psi}_{t-1,1} + \boldsymbol{S}_{t-1} \cdot {}_h\boldsymbol{\epsilon}_t, \qquad {}_h\boldsymbol{\epsilon}_t \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{Q}_{h\epsilon}), \tag{6.52}$$

and calculating the sample mean and covariance matrix using these samples.
Also obtain initial estimates of other factors

$$\widehat{\boldsymbol{\gamma}}_{1|0} = \widehat{\boldsymbol{\gamma}}_{0|0} = E[\boldsymbol{\gamma}_1], \tag{6.53}$$

$$_\gamma\widehat{\boldsymbol{P}}_{1|0} = Cov[\boldsymbol{\gamma}_1], \tag{6.54}$$

which can be chosen using static GLM analyses and preliminary analyses of data.
For $i = 1, ..., I$:

Step 2. (Measurement update/filtering for calendar factors)
Calculate the Kalman gain for calendar factors

$$_h\boldsymbol{G}_i = {}_h\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{E}_i' \left(\boldsymbol{E}_i \cdot {}_h\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{E}_i' + \boldsymbol{H}_i\right)^{-1}. \tag{6.55}$$

Update estimates of calendar factors, including the mean and the error covariance matrix

$$\widehat{\boldsymbol{\psi}}_{I,i|i} = E[\boldsymbol{\psi}_{I,i}|\boldsymbol{Y}_i] = \widehat{\boldsymbol{\psi}}_{I,i|i-1} + {}_h\boldsymbol{G}_i \left(\boldsymbol{Y}_i - \boldsymbol{A}_i \cdot \widehat{\boldsymbol{\gamma}}_{i-1|i-1} - \boldsymbol{E}_i \cdot \widehat{\boldsymbol{\psi}}_{I,i|i-1}\right), \tag{6.56}$$

$$_h\widehat{\boldsymbol{P}}_{i|i} = Cov[\boldsymbol{\psi}_{I,i}|\boldsymbol{Y}_i] = {}_h\widehat{\boldsymbol{P}}_{i|i-1} - {}_h\boldsymbol{G}_i \cdot \boldsymbol{E}_i \cdot {}_h\widehat{\boldsymbol{P}}_{i|i-1}. \tag{6.57}$$

Step 3. (Measurement update/filtering for other factors)

Calculate the Kalman gain for other factors

$$ {}_{\gamma}\boldsymbol{G}_i = {}_{\gamma}\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{A}_i' \left(\boldsymbol{A}_i \cdot {}_{\gamma}\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{A}_i' + \boldsymbol{H}_i \right)^{-1}. \tag{6.58} $$

Update estimates of other factors, including the mean and the error covariance matrix

$$ \widehat{\boldsymbol{\gamma}}_{i|i} = E[\boldsymbol{\gamma}_i|\boldsymbol{Y}_i] = \widehat{\boldsymbol{\gamma}}_{i|i-1} + {}_{\gamma}\boldsymbol{G}_i \left(\boldsymbol{Y}_i - \boldsymbol{A}_i \cdot \widehat{\boldsymbol{\gamma}}_{i|i-1} - \boldsymbol{E}_i \cdot \widehat{\boldsymbol{\psi}}_{I,i|i} \right), \tag{6.59} $$

$$ {}_{\gamma}\widehat{\boldsymbol{P}}_{i|i} = Cov[\boldsymbol{\gamma}_i|\boldsymbol{Y}_i] = {}_{\gamma}\widehat{\boldsymbol{P}}_{i|i-1} - {}_{\gamma}\boldsymbol{G}_i \cdot \boldsymbol{A}_i \cdot {}_{\gamma}\widehat{\boldsymbol{P}}_{i|i-1}. \tag{6.60} $$

Step 4. (Time update/prediction of calendar factors)

Project the calendar factors ahead

$$ \widehat{\boldsymbol{\psi}}_{I,i+1|i} = E[\boldsymbol{\psi}_{I,i+1}|\boldsymbol{Y}_i] = \widehat{\boldsymbol{\psi}}_{I,i|i}, \tag{6.61} $$

and project the error covariance of these factors

$$ {}_{h}\widehat{\boldsymbol{P}}_{i+1|i} = Cov[\boldsymbol{\psi}_{I,i+1}|\boldsymbol{Y}_i] = {}_{h}\widehat{\boldsymbol{P}}_{i|i} + \boldsymbol{Q}_{{}_{h_I}\epsilon}, \tag{6.62} $$

where $\boldsymbol{Q}_{{}_{h_I}\epsilon}$ is the artificial dynamics added to the covariance specification. It can be chosen to reflect the level of uncertainty regarding the estimates of calendar factors. Greater uncertainty can be accompanied by a larger artificial noise.

Step 5. (Time update/prediction of other factors) Project other factors ahead

$$ \widehat{\boldsymbol{\gamma}}_{i+1|i} = E[\boldsymbol{\gamma}_{i+1}|\boldsymbol{Y}_i] = \widehat{\boldsymbol{\gamma}}_{i|i}, \tag{6.63} $$

and project their error covariance ahead

$$ {}_{\gamma}\widehat{\boldsymbol{P}}_{i+1|i} = Cov[\boldsymbol{\gamma}_{i+1}|\boldsymbol{Y}_i] = {}_{\gamma}\widehat{\boldsymbol{P}}_{i|i} + \boldsymbol{Q}_{\boldsymbol{\gamma\epsilon}}. \tag{6.64} $$

Step 6. Repeat step 2-5 until $i = I$.

The above algorithm is conditional on known values of parameters $\boldsymbol{\Theta}$. Maximum likelihood estimation can be used to estimate these parameters. The log likelihood function

can be written as

$$\log f_{\boldsymbol{Y}_{1:I}}(\boldsymbol{Y}_{1:I};\boldsymbol{\Theta}) = \sum_{i=1}^{I} \log f_{\boldsymbol{Y}_i|\boldsymbol{Y}_{i-1}}\left(\boldsymbol{y}_i|\boldsymbol{y}_{i-1};\boldsymbol{\Theta}\right) \tag{6.65}$$

$$\begin{aligned}
&= -\frac{I^2(I+1)}{4}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{I}\Big(\log|\boldsymbol{A}_i \cdot {}_\gamma\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{A}_i' + \boldsymbol{E}_i \cdot {}_h\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{E}_i' + \boldsymbol{H}_i| + \\
&\left(\boldsymbol{y}_i - \boldsymbol{A}_i \cdot \widehat{\boldsymbol{\gamma}}_{i|i-1} - \boldsymbol{E}_i \cdot \widehat{\boldsymbol{\psi}}_{I,i|i-1}\right)' \left(\boldsymbol{A}_i \cdot {}_\gamma\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{A}_i' + \boldsymbol{E}_i \cdot {}_h\widehat{\boldsymbol{P}}_{i|i-1} \cdot \boldsymbol{E}_i' + \boldsymbol{H}_i\right)^{-1} \\
&\times \left(\boldsymbol{y}_i - \boldsymbol{A}_i \cdot \widehat{\boldsymbol{\gamma}}_{i|i-1} - \boldsymbol{E}_i \cdot \widehat{\boldsymbol{\psi}}_{I,i|i-1}\right)\Big),
\end{aligned} \tag{6.66}$$

which can be maximised numerically to provide the maximum likelihood estimate of $\boldsymbol{\Theta}$. When maximum likelihood estimation is used for parameter estimation, bootstrapping is needed to assess the parameter uncertainty in the projection of future claims.

It is also desirable to use all available information to estimate random factors. This is accomplished using the Kalman back smoother after the dual Kalman filter is complete,

$$\widehat{\boldsymbol{\psi}}_{I,i|I} = E[\boldsymbol{\psi}_{I,i|I}|\boldsymbol{Y}_I] = \widehat{\boldsymbol{\psi}}_{I,i|i} + {}_h\widehat{\boldsymbol{P}}_{i|i} \cdot {}_h\widehat{\boldsymbol{P}}_{i+1|i}^{-1}\left(\widehat{\boldsymbol{\psi}}_{I,i+1|I} - \widehat{\boldsymbol{\psi}}_{I,i+1|i}\right), \tag{6.67}$$

$$\widehat{\boldsymbol{\gamma}}_{i|I} = E[\boldsymbol{\gamma}_i|\boldsymbol{Y}_I] = \widehat{\boldsymbol{\gamma}}_{i|i} + {}_\gamma\widehat{\boldsymbol{P}}_{i|i} \cdot {}_\gamma\widehat{\boldsymbol{P}}_{i+1|i}^{-1}\left(\widehat{\boldsymbol{\gamma}}_{i+1|I} - \widehat{\boldsymbol{\gamma}}_{i+1|i}\right). \tag{6.68}$$

The estimates obtained from the back smoother are also called smoothed estimates.

6.4 Simulation illustrations

In this section we provide two illustrations using simulated data, one for a Gaussian model with the dual Kalman filter (Section 6.4.1), and one for the multivariate evolutionary GLM framework with the particle learning approach (Section 6.4.2). The aims of these illustration are to assess performance of the estimation and to draw any remarks on practical applications of the framework.

6.4.1 Gaussian model illustration

We first perform a simulation illustration for the case of a Gaussian model. The simulated data set used for illustration consists of two 15×15 triangles that are given in Tables 6.9 and 6.10 in Appendix 6.A.1. The data is simulated from log-normal distributions,

hence the log transformation is applied before a Gaussian evolutionary model is fitted. The dual Kalman filter and smoother with maximum likelihood estimation described in Section 6.3.2 is used to provided smoothed estimates of random factors and unknown parameters. The dual Kalman filter is initialised using estimates from static GLM analysis.

6.4.1.1 Random factors estimation

Estimates of random factors are provided in Table 6.11 in Appendix 6.A.1. We provide plots of fitting ratios in Figure 6.2. The fitting ratios are calculated as ratios of true values to smoothed values. It is noted that each development pattern is fitted with a Hoerl curve (also known as the gamma curve) orchestrated by two parameters $r_i^{(n)}$ and $s_i^{(n)}$. Therefore, the most direct way to assess the goodness-of-fit for the development pattern is to consider the fitting ratios of the mean and variance of the Hoerl curve calculated using these two parameters. In particular, the mean of the Hoerl curve for accident period i and segment n is calculated by

$$\frac{r_i^{(n)} - 1}{-s_i^{(n)}}, \tag{6.69}$$

and the variance by

$$\frac{r_i^{(n)} - 1}{\left(s_i^{(n)}\right)^2}. \tag{6.70}$$

It can be observed from Figure 6.2 that the fitting is quite satisfactory for accident period factors $a_i^{(n)}$. The fitting of calendar factors $h_t^{(n)}$ also seems reasonable. The estimates of the means and the variances of the Hoerl curves are also quite close to their true values. However, the estimates of the means and the variances of the Hoerl curves are less accurate in immature accident periods. There is evidence of compensation between estimates of random factors, particularly between accident factors and calendar factors. In particular, accident period factors $a_i^{(n)}$ are consistently underestimated, and this mis-estimation is absorbed into the estimates of calendar factors $h_t^{(n)}$, resulting in an overestimation. However, it is worth noting that the filter/smoother can still capture the relative over time movements in these factors reasonably well, indicated by relatively straight lines in the fitting plots in Figure 6.2.

To assess the calendar period dependence driven by common shocks, we calculate the Pearson correlation coefficient between actual calendar factors $h_t^{(1)}$ and $h_t^{(2)}$, and the

correlation coefficient between their smoothed estimates from the dual Kalman smoother. The Pearson correlation coefficient between the smoothed estimates is 0.8992 with the 95% confidence interval (0.7174; 0.9663). The true Pearson correlation coefficient is 0.7579 which lies well within this interval. Hence the filter and smoother can capture the dependence quite well.

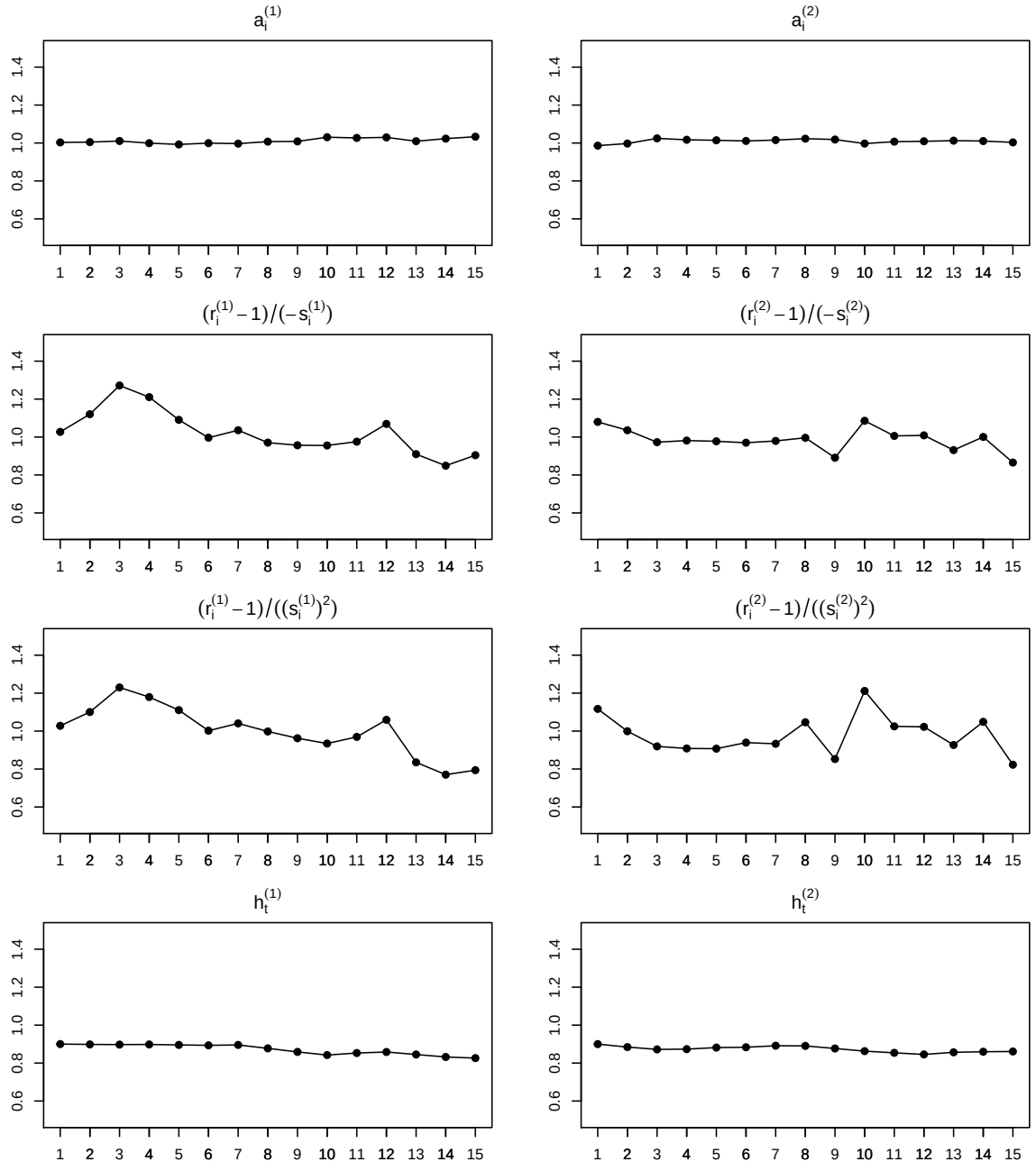

Figure 6.2: Fitted ratios of random factors in the Gaussian model illustration (true values to smoothed estimates)

6.4.1.2 Parameter estimation

	True value	Estimate	90% CI
$\sigma^2_{\varsigma^{(1)}}$	0.0200	0.0155	(0.0133; 0.0430)
$\sigma^2_{a\epsilon^{(1)}}$	0.0100	0.0043	(0.0041; 0.0121)
$\sigma^2_{r\epsilon^{(1)}}$	0.0050	0.0018	(0.0017; 0.0026)
$\sigma^2_{s\epsilon^{(1)}}$	0.0010	0.0003	(0.0003; 0.0004)
$\sigma^2_{h\epsilon^{(1)}}$	0.0050	0.0017	(0.0017; 0.0018)
${}_{\epsilon}\lambda^{(1)}$	0.6000	0.5185	(0.5106; 0.5453)
$\sigma^2_{\varsigma^{(2)}}$	0.0200	0.0126	(0.0119; 0.0428)
$\sigma^2_{a\epsilon^{(2)}}$	0.0050	0.0020	(0.0018; 0.0088)
$\sigma^2_{r\epsilon^{(2)}}$	0.0020	0.0006	(0.0006; 0.0008)
$\sigma^2_{s\epsilon^{(2)}}$	0.0005	0.0001	(0.0001; 0.0002)
$\sigma^2_{h\epsilon^{(2)}}$	0.0050	0.0017	(0.0017; 0.0018)
${}_{\epsilon}\lambda^{(2)}$	0.8000	0.7657	(0.7640; 0.7811)
$\sigma^2_{h\tilde{\epsilon}}$	0.0050	0.0017	(0.0017; 0.0018)

Table 6.1: Maximum likelihood estimates of parameters in the Gaussian model illustration

Parameter estimates are provided in Table 6.1. The results show that many parameters are not within their estimated CIs. While the variance terms $\sigma^2_{\varsigma^{(n)}}$ all lie within their respective CIs, they tend to fall on the lower side of their intervals. Other variance terms, however, fall either on the upper side within their respective CIs, or outside their CIs. This suggests a compensation across these terms. The model contains latent random factors which are not observed. The noises in the observations and random factors altogether contribute to the overall volatility that we observe in the data. The estimation may not be able to clearly distinguish between the two latent sources of disturbance and this explains the compensation in the estimation results.

6.4.1.3 Goodness-of-fit analysis

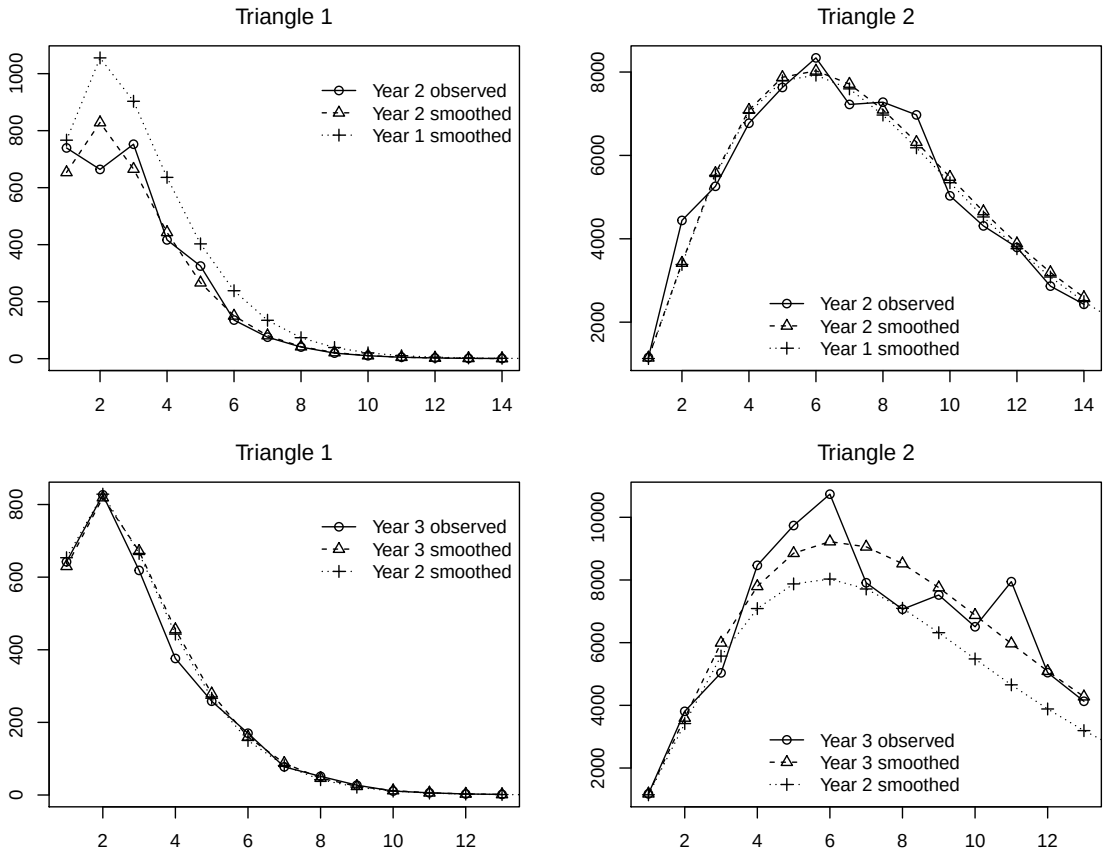

Figure 6.3: Tracking of claims development patterns for some accident years in the Gaussian model illustration

We examine the performance of the dual Kalman filter/smoother by looking at how closely the smoothed claims patterns track the actual patterns. Examples of this tracking are given for accident years 2 and 3 in Figure 6.3. There are significant changes in the claims development patterns from year 1 to year 2, and from year 2 to year 3, as shown in this figure. However, these changes are tracked quite closely by the Kalman smoother.

Heat maps of residuals are given in Figure 6.4. The residuals are calculated as ratios of observed values (on the log-scale) to smoothed values. The heat maps show that the overall fit is very good. The fit is slightly off in the tail of triangle 1, which may be a result of the initialisation where estimates chosen to initialise the filter are static GLM estimates which are not true values. However, the goodness-of-fit improves as the filter proceeds to later accident years. In addition, as triangle 1 is simulated to be shorter tailed than triangle 2, the values observed in the last few development periods are very small which can also magnify the error

ratios.

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.964	1.009	1.000	0.971	1.009	1.018	0.988	0.968	1.014	1.050	1.002	1.092	1.050	1.468	1.102
2	1.019	0.967	1.019	0.990	1.036	0.979	0.984	0.990	0.968	0.983	0.964	0.953	0.296	0.800	
3	1.003	1.001	0.987	0.969	0.986	1.012	0.972	1.026	1.038	0.954	0.999	0.958	1.351		
4	0.990	0.994	1.016	0.961	1.016	1.040	1.064	1.066	0.951	0.892	0.978	1.148			
5	1.017	1.018	0.992	0.961	1.053	1.040	0.920	0.983	0.976	0.989	0.879				
6	0.994	0.991	0.994	0.997	1.000	1.033	1.014	1.019	1.078	1.001					
7	0.986	0.980	0.961	1.023	1.055	0.994	1.002	0.972	1.007						
8	0.999	0.989	1.021	0.976	0.993	1.004	0.985	0.987							
9	1.023	1.029	1.002	0.993	0.982	1.062	1.006								
10	1.009	0.993	1.020	1.016	0.974	1.071									
11	0.995	0.991	1.003	1.006	0.948										
12	0.983	1.000	0.964	0.984											
13	1.012	0.994	0.994												
14	1.000	1.040													
15	0.988														

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.998	1.024	1.032	0.995	1.015	1.008	1.011	1.012	1.004	0.972	1.002	0.999	1.007	0.974	0.984
2	1.001	1.032	0.993	0.995	0.996	1.004	0.993	1.003	1.011	0.990	0.991	0.997	0.987	0.992	
3	1.000	1.007	0.980	1.009	1.010	1.017	0.985	0.979	0.997	0.994	1.033	0.999	0.996		
4	0.996	0.990	0.975	1.000	1.006	0.993	0.971	1.004	1.029	0.985	1.004	1.025			
5	1.006	1.016	1.002	0.989	0.980	0.991	1.002	1.014	0.984	0.989	1.009				
6	1.022	1.000	0.990	0.996	0.983	0.997	1.017	0.991	0.990	1.026					
7	1.036	1.003	0.985	0.997	1.021	0.995	0.997	0.998	0.993						
8	0.994	1.011	0.994	0.995	0.998	1.025	0.988	1.005							
9	0.963	0.999	0.979	0.990	0.993	1.015	1.016								
10	1.009	1.004	0.992	0.989	1.000	0.963									
11	1.020	1.008	0.998	1.005	0.998										
12	0.988	0.994	1.006	0.987											
13	0.993	1.007	1.029												
14	1.016	1.004													
15	0.985														

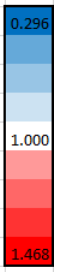

Figure 6.4: Heat maps ratios of observed values (on the log-scale) to smoothed values in the Gaussian model illustration (top: triangle 1, bottom: triangle 2)

We perform 100 simulations from the same set of parameters, and the results obtained are quite similar across all these simulations. The overall goodness-of-fit is very good, and estimates of random factors are also reasonably accurate. However, parameter estimates are not often accurate.

6.4.2 Evolutionary GLM approach illustration

A simulation illustration is performed for the evolutionary GLM framework to assess the performance of the particle learning estimation approach. The simulated data used for this illustration consists of two 15×15 triangles given in Table 6.12 and Table 6.13 in Appendix 6.A.2. The data is simulated from the Tweedie sub-family of the EDF. The particle

learning approach developed in Section 6.3.1 is used to estimate random factors and unknown parameters. We use 50,000 particles for each time period and initialise the filter using static GLM estimates.

6.4.2.1 Random factors estimation

Estimates of random factors are provided in Table 6.14 in Appendix 6.A.2. We provide plots of fitting ratios in Figure 6.5, which are calculated as ratios of true values to filtered values. The fitting ratios of the means and variances of the Hoerl curves are also given in this figure. It can be observed from Figure 6.5 that the fitting is good for accident period factors $a_i^{(n)}$. The estimates of the means and variances of the Hoerl curves are also quite close to their actual values. Estimates of calendar factor $h_t^{(n)}$ are consistently lower than their true values, however the deviations remain within 20% of actual values. The compensation between random factors is again evident in this illustration. Accident period factors $a_i^{(n)}$ are consistently underestimated, and this is compensated by an overestimation of calendar period factors.

As with the illustration for a Gaussian model, estimates in this illustration are also sensitive to values used to initialise the particle filter. Estimates from the corresponding static GLM fitting also show to be a reasonable choice for the initialisation.

To assess the calendar period dependence driven by common shocks, we calculate the Pearson correlation coefficient between actual calendar factors $h_t^{(1)}$ and $h_t^{(2)}$, and the correlation coefficient between the estimates of these factors obtained from the particle filter. The Pearson correlation coefficient between the filtered estimates is 0.7520 with a 95% confidence interval (0.3900; 0.9127). The actual Pearson correlation coefficient is 0.4634 which lies well within this interval.

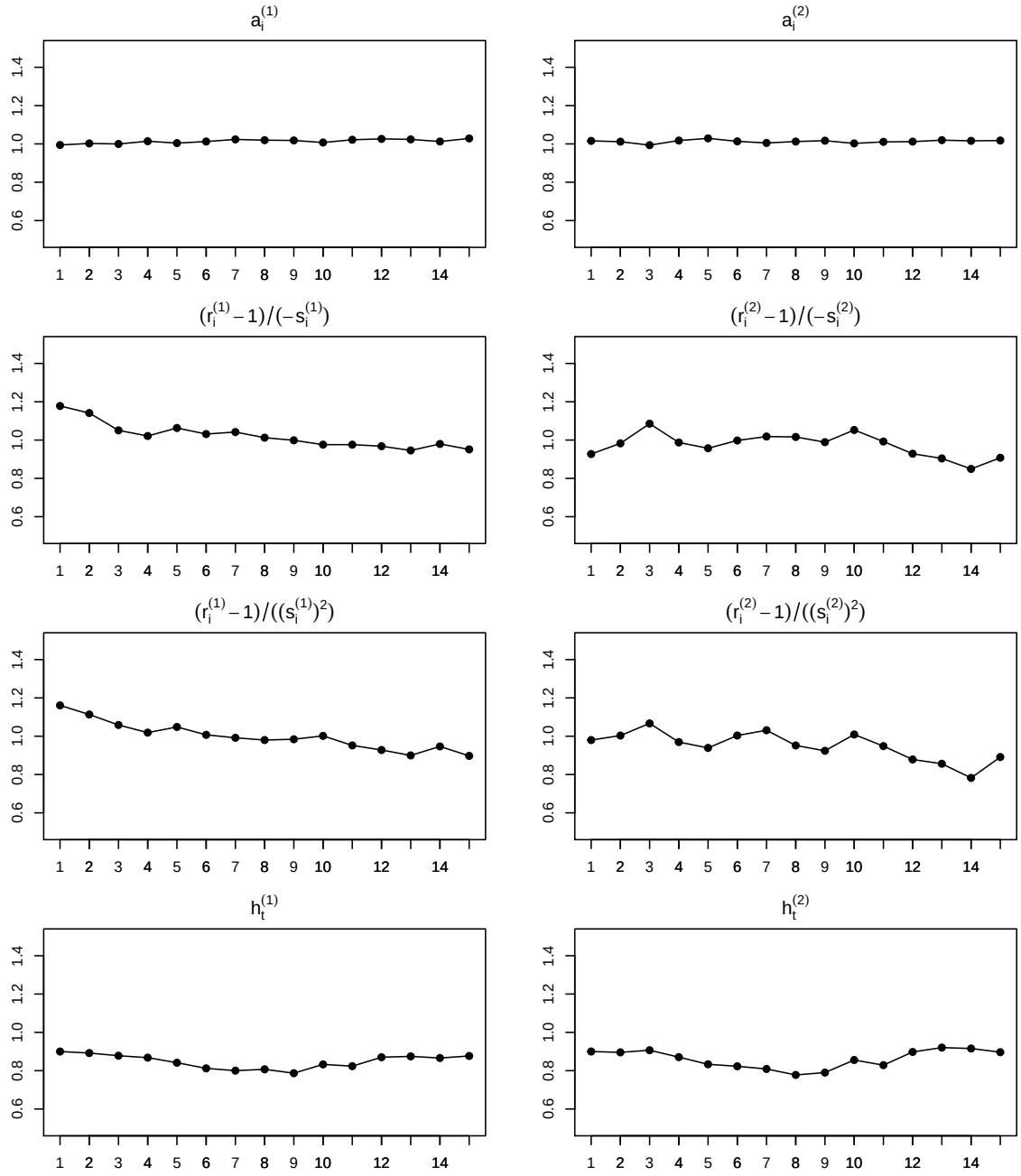

Figure 6.5: Fitted ratios of random factors in the GLM framework illustration (true values to filtered estimates)

6.4.2.2 Parameter estimation

Parameter estimates are provided in Table 6.2. The results show that the true values of many parameters fall out of their respective confidence intervals, even though they appear to be closer to their estimates compared to the results in the Gaussian illustration.

The narrow CIs that we observe in the results are mainly due to the degeneracy issue of the particle filter used (see also a discussion on this issue in Section 2.5.4.1). Particle degeneracy refers to the situation where all but a few particles have negligible weights. As a result, the re-sampling step in the particle learning algorithm focuses on multiplying the few particles with significant weights and abandons the majority of particles with negligible weights. The resultant sample then has a very low diversity of particles. This consequently results in smaller confidence intervals of parameter estimates in the particle learning (Rios and Lopes, 2013). This problem is often encountered not only in particle learning but also in particle filtering in general, see for example, Doucet et al. (2000); Andrieu et al. (2005); Carvalho et al. (2010); Creal (2012). Using informative prior distributions for parameters and initial values for random factors can help reduce this problem. This illustration also has a large vector of observations at each time period (up to 30 in the first period). This high dimension of observations makes the likelihood function very steep and it further contributes to the degeneracy problem (see, for example, Wang et al., 2017; Li et al., 2014).

	True value	Estimate	90% CI
$\phi^{(1)}$	0.4000	0.4308	(0.4308; 0.4309)
$\sigma^2_{a^{\epsilon^{(1)}}}$	0.0100	0.0075	(0.0075; 0.0075)
$\sigma^2_{r^{\epsilon^{(1)}}}$	0.0050	0.0051	(0.0051; 0.0051)
$\sigma^2_{s^{\epsilon^{(1)}}}$	0.0010	0.0021	(0.0021; 0.0021)
$\sigma^2_{h^{\epsilon^{(1)}}}$	0.0050	0.0037	(0.0036; 0.0037)
${}_{\epsilon}\lambda^{(1)}$	0.6000	0.5676	(0.5674; 0.5677)
$p^{(1)}$	1.2700	1.3915	(1.3911; 1.3923)
$\phi^{(2)}$	0.5000	0.5777	(0.5771; 0.5788)
$\sigma^2_{a^{\epsilon^{(2)}}}$	0.0050	0.0050	(0.0050; 0.0050)
$\sigma^2_{r^{\epsilon^{(2)}}}$	0.0020	0.0020	(0.0020; 0.0020)
$\sigma^2_{s^{\epsilon^{(2)}}}$	0.0005	0.0002	(0.0002; 0.0002)
$\sigma^2_{h^{\epsilon^{(2)}}}$	0.0050	0.0042	(0.0042; 0.0042)
${}_{\epsilon}\lambda^{(2)}$	0.8000	0.6327	(0.6324; 0.6329)
$p^{(2)}$	1.3500	1.4264	(1.4263; 1.4265)
$\sigma^2_{h^{\tilde{\epsilon}}}$	0.0050	0.0092	(0.0092; 0.0093)

Table 6.2: Particle learning estimates of parameters in the GLM approach illustration

As shown in Table 6.2, there appears to be some compensation between variance terms in this illustration as well. The dispersion parameters of the observations, including $\phi^{(1)}$, $\phi^{(2)}$, $p^{(1)}$, $p^{(2)}$ are overestimated, while variance terms of random factors are

underestimated. This is similar to the results that we observed in the previous illustration of a Gaussian model.

6.4.2.3 Goodness-of-fit analysis

We examine the performance of the particle filter by assessing how the filtered claim patterns closely track the observed patterns. Examples of this tracking for accident years 7 and 12 are given in Figure 6.6. There are significant changes in the claims development patterns from year 6 to year 7, and from year 11 to year 12, as shown in this figure. However, the particle filter is able to track these changes quite closely.

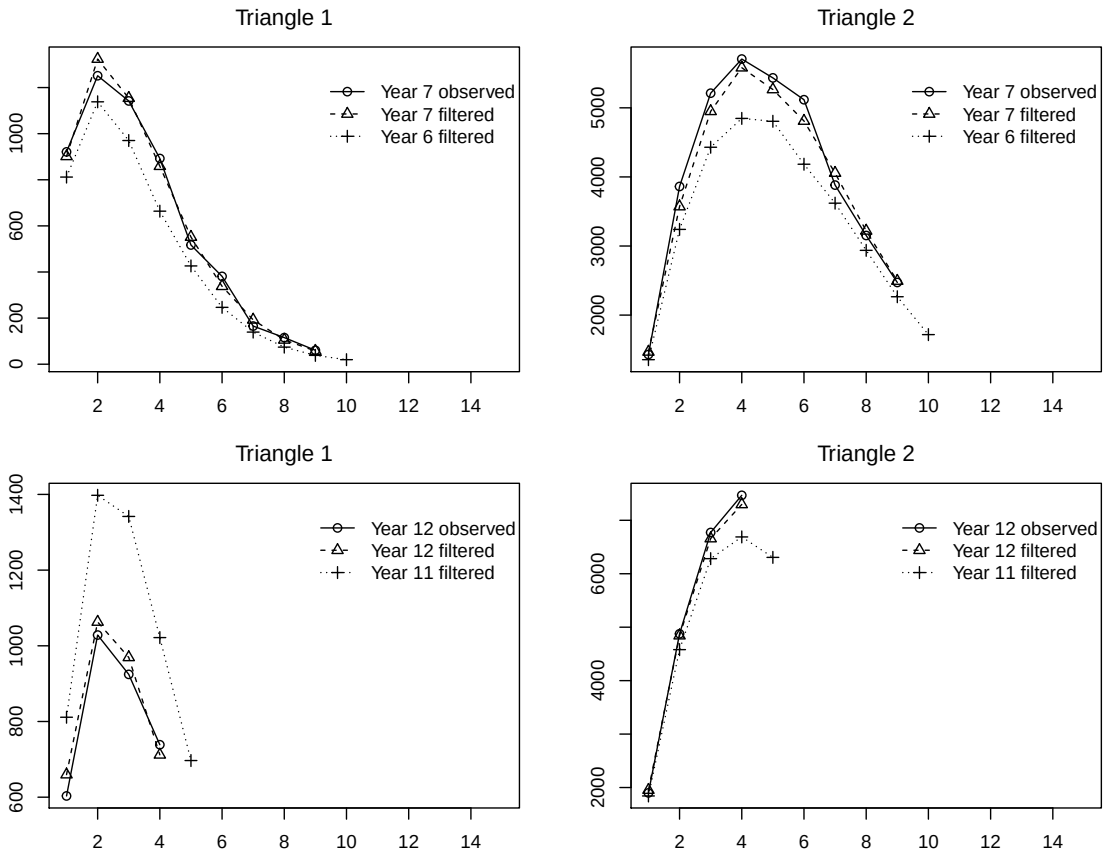

Figure 6.6: Tracking of claims development patterns for some accident years in the GLM framework illustration

Heat maps of residuals are given in Figure 6.7. Residuals are calculated as the ratios of observed values to filtered values. The heat maps show that the overall fit is good. The goodness-of-fit is slightly off in the tail of triangle 1, as a result of the initialisation where the initial values chosen to start the filter are static GLM estimates. However, the goodness-

of-fit improves as the filter proceeds to later accident years. In addition, as triangle 1 is simulated to be shorter tailed than triangle 2, values observed in the last few development periods are very small which may also magnify the error ratios. The overall goodness-of-fit in this illustration is slightly worse than the goodness-of-fit in the Gaussian illustration in the previous section. This is likely due to the fact that the particle filter is simulation-based while the dual Kalman filter is the optimal filter obtained in closed-form.

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.065	0.925	1.000	0.973	1.005	1.132	0.960	1.077	0.998	0.908	0.853	0.835	0.705	1.222	0.906
2	1.039	1.074	1.003	0.961	0.935	0.936	0.935	0.878	0.967	1.340	1.164	1.421	0.812	0.326	
3	1.155	0.954	0.966	0.955	1.000	0.984	1.203	1.015	1.057	0.874	0.699	0.806	1.656		
4	0.973	1.067	1.028	1.090	0.890	1.089	0.874	1.173	0.657	1.245	0.881	0.811			
5	1.053	1.017	1.027	0.990	1.100	1.034	1.071	1.051	1.120	0.921	1.087				
6	1.081	0.999	0.967	1.032	0.933	1.149	1.020	0.916	1.167	0.579					
7	1.022	0.945	0.989	1.039	0.937	1.127	0.860	1.094	1.055						
8	0.989	1.033	0.948	1.075	0.874	1.062	1.044	0.981							
9	0.978	1.004	0.969	1.042	1.005	1.061	0.915								
10	0.957	0.925	1.065	1.005	0.990	1.020									
11	0.946	0.967	1.058	1.051	0.927										
12	0.915	0.968	0.954	1.037											
13	0.799	1.013	0.985												
14	0.919	1.019													
15	0.997														

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.932	1.096	0.963	1.015	0.935	1.001	0.974	1.003	1.098	0.999	1.095	0.977	0.918	1.036	0.882
2	0.928	0.897	1.061	1.023	1.047	1.078	0.998	1.005	0.994	0.924	0.932	1.075	0.982	0.944	
3	0.939	0.914	1.025	1.009	1.056	0.979	1.018	1.013	1.005	0.979	0.993	0.956	1.012		
4	1.042	1.046	1.038	1.015	1.042	1.010	0.981	1.036	0.924	0.984	1.011	0.980			
5	0.974	0.875	1.012	1.007	0.926	1.080	0.982	0.966	1.005	0.947	1.002				
6	1.001	0.966	0.950	1.046	0.989	1.056	0.942	1.039	0.979	0.982					
7	0.974	1.081	1.053	1.022	1.032	1.064	0.957	0.979	0.992						
8	0.963	1.053	0.997	1.059	1.045	0.944	0.957	1.021							
9	1.015	0.981	1.019	0.973	0.963	0.986	1.050								
10	0.904	1.097	1.030	0.957	1.033	0.937									
11	1.147	0.970	0.975	1.020	0.994										
12	0.973	1.007	1.017	1.023											
13	1.009	1.038	1.013												
14	0.759	1.038													
15	1.002														

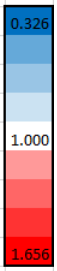

Figure 6.7: Heat maps of ratios of observed values to filtered values in the GLM framework illustration (top: triangle 1, bottom: triangle 2)

We perform 100 simulations from the same set of parameters, and the results obtained are quite similar across all these simulations. The overall goodness-of-fit is very good, and estimates of random factors are also reasonably accurate. However, parameter estimates are not accurate. Some compensations across random factors, and across variances of the observations and random factors are also observed.

6.5 Real data illustration

The data set used for this illustration is from Côté et al. (2016), a description of which is also provided in Section 5.5. The two triangles chosen for illustration in this chapter are from the Accident Benefits covers of the Auto Insurance line in Ontario. One triangle is for Accident Benefits excluding Disability Income (denoted by (1)), and the other is for Accident Benefits with Disability Income only (denoted by (2)). Incremental losses are given in Table 6.15 and 6.16 in Appendix 6.A.3. Claims are standardised using the total premium earned in the corresponding accident years.

6.5.1 Preliminary analysis

A preliminary analysis is performed to assess the suitability of the data set. This includes an assessment of any changes in the development patterns as well as the dependence across the two lines.

6.5.1.1 Analysis of development patterns

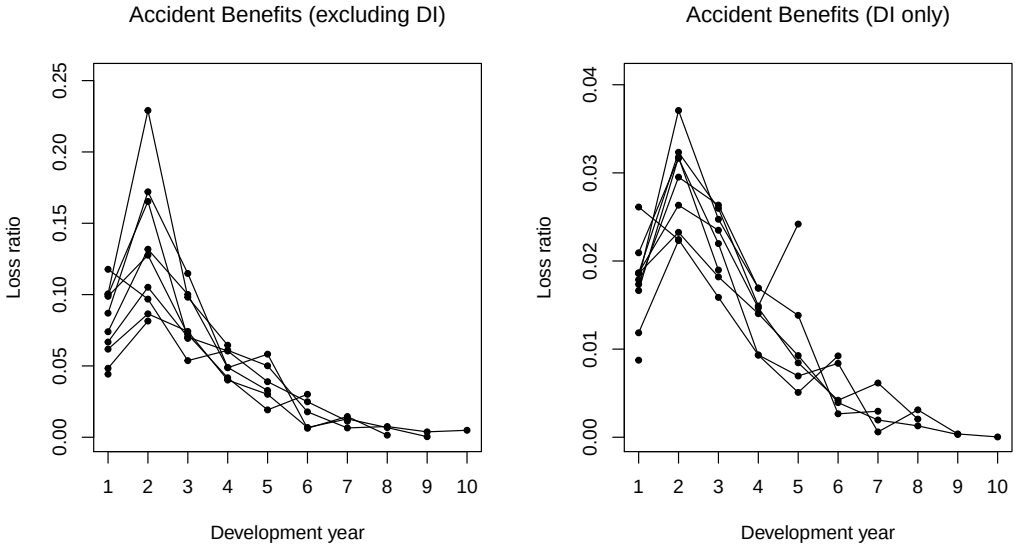

Figure 6.8: Cumulative loss ratios in real data from a Canadian insurer

Plots of loss ratios are provided in Figure 6.8. For each loss triangle, the top two values in each accident period are also highlighted to identify the peak in the development pattern. These are provided in Figure 6.9 for accident periods 1-8.

Plots of loss ratios show variations in claims development patterns over time. In the Accident Benefits excluding Disability Income line, the peak in the claims development pattern shifts across development periods 1-2 and 2-3. Hence it is desirable for the model to be able to capture this feature. In the Accident Benefits with Disability Income only line, the peak in the development shifts from development periods 1-2 in the first two accident periods to periods 2-3 onwards. It can be tempting to model the first two accident periods separately using a static modelling approach. However, using an evolutionary model can allow the changes in the prediction of claims to be smoothed out over time. In addition, it can be observed that the level of variation between claims in development periods 1 and 3 within the same accident period has also varied over time. This indicates some variation in the development patterns besides the peaks.

Year	1	2	3	4	5	6	7	8	9	10
1	0.118	0.097	0.054	0.061	0.050	0.018	0.007	0.008	0.004	0.005
2	0.062	0.087	0.074	0.040	0.030	0.007	0.013	0.007	0.000	
3	0.074	0.132	0.100	0.049	0.058	0.006	0.015	0.001		
4	0.067	0.105	0.070	0.060	0.039	0.025	0.011			
5	0.099	0.128	0.072	0.042	0.019	0.030				
6	0.087	0.172	0.115	0.049	0.033					
7	0.101	0.229	0.098	0.064						
8	0.100	0.165	0.069							

Year	1	2	3	4	5	6	7	8	9	10
1	0.026	0.022	0.016	0.009	0.007	0.008	0.001	0.003	0.000	0.000
2	0.019	0.023	0.018	0.014	0.009	0.004	0.002	0.001	0.000	
3	0.019	0.026	0.023	0.015	0.008	0.004	0.006	0.002		
4	0.017	0.030	0.026	0.017	0.014	0.003	0.003			
5	0.021	0.032	0.022	0.009	0.005	0.009				
6	0.019	0.032	0.026	0.015	0.024					
7	0.018	0.037	0.025	0.017						
8	0.017	0.032	0.019							

Figure 6.9: Loss triangles with top two values highlighted for each accident period (top: Accident Benefits excluding Disability Income, bottom: Accident Benefits - Disability Income only)

6.5.1.2 Exploratory dependence analysis

A heuristic dependence analysis is performed by fitting to each line a Tweedie GLM with the log-link and a modified Hoerl curve with additional covariates for the first two development periods

$$a_i^{(n)} + r^{(n)} \log(j) + s^{(n)} j + b_1^{(n)} \mathbb{1}_{\{j=1\}} + b_2^{(n)} \mathbb{1}_{\{j=2\}}. \tag{6.71}$$

As the peak in the development pattern shifts across development periods 1, 2 and 3, adding the two covariates for the first two development periods can improve the goodness-of-fit of the Hoerl curve. Such modification is quite common in the applications of the Hoerl curve (England and Verrall, 2001). This static GLM fitting aims to remove fixed accident period and development period effects.

Pearson	Spearman	Kendall
0.2599 (0.0554)	0.3087 (0.0222)	0.2256 (0.0150)

Table 6.3: Correlation coefficients between cell-wise GLM residuals and their corresponding p-values

Correlation coefficients between pair-wise GLM Pearson residuals of the two lines are provided in Table 6.3, where the residuals are calculated using

$$\frac{Y_{i,j}^{(n)} - \mu_{i,j}^{(n)}}{\left(\mu_{i,j}^{(n)}\right)^p}. \tag{6.72}$$

All coefficients are quite strong, however the Pearson coefficient is not significant at 5%. Another GLM analysis is also performed with the chain ladder mean structure. The Pearson correlation coefficient of residuals from this GLM analysis, however, is significant. This further illustrates the conclusion in Avanzi, Taylor and Wong (2016) that correlation coefficients are dependent on the methodology used. From these results, it is then reasonable to conclude that there may be dependence retained in the data set after removing fixed accident year and development year effects.

Year	1	2	3	4	5	6	7	8	9	10
1	1.388	0.726	0.687	1.130	1.460	0.859	0.541	1.097	0.991	2.336
2	0.975	0.869	1.271	0.995	1.176	0.443	1.427	1.323	0.136	
3	0.860	0.974	1.262	0.891	1.674	0.299	1.188	0.212		
4	0.849	0.851	0.969	1.209	1.226	1.296	1.011			
5	1.213	0.997	0.961	0.805	0.582	1.515				
6	0.877	1.103	1.256	0.776	0.817					
7	0.856	1.241	0.907	0.864						
8	1.052	1.106	0.791							
9	0.960	1.029								
10	1.000									

0.092 1.000 2.336

Year	1	2	3	4	5	6	7	8	9	10
1	1.567	0.878	0.872	0.684	0.791	1.636	0.215	2.140	0.520	0.097
2	1.164	0.951	1.044	1.072	1.102	0.801	0.731	0.929	0.452	
3	0.971	0.894	1.120	0.932	0.834	0.710	1.909	1.231		
4	0.873	0.970	1.215	1.039	1.321	0.435	0.886			
5	1.128	1.114	1.085	0.613	0.520	1.618				
6	0.775	0.878	0.990	0.756	1.909					
7	0.828	1.121	1.050	0.957						
8	0.901	1.123	0.942							
9	0.885	1.088								
10	1.000									

Figure 6.10: Heat maps of ratios of observed values to GLM fitted values (top: Accident Benefits (excluding DI), bottom: Accident Benefits (DI only)

To further explore the dependence across the two lines, we analyse heat maps of GLM residuals in Figure 6.10. The residuals in the heat maps are calculated as the ratios of observed values to GLM fitted values. We can observe some common patterns in the calendar year dimension, suggesting that the correlation coefficients results in Table 6.3 may be attributed to calendar year dependence. For clearer illustrations, plots of residuals for the last three calendar years (the last three diagonals in the heat maps) are provided in Figure 6.11. This figure shows clear evidence of calendar year dependence.

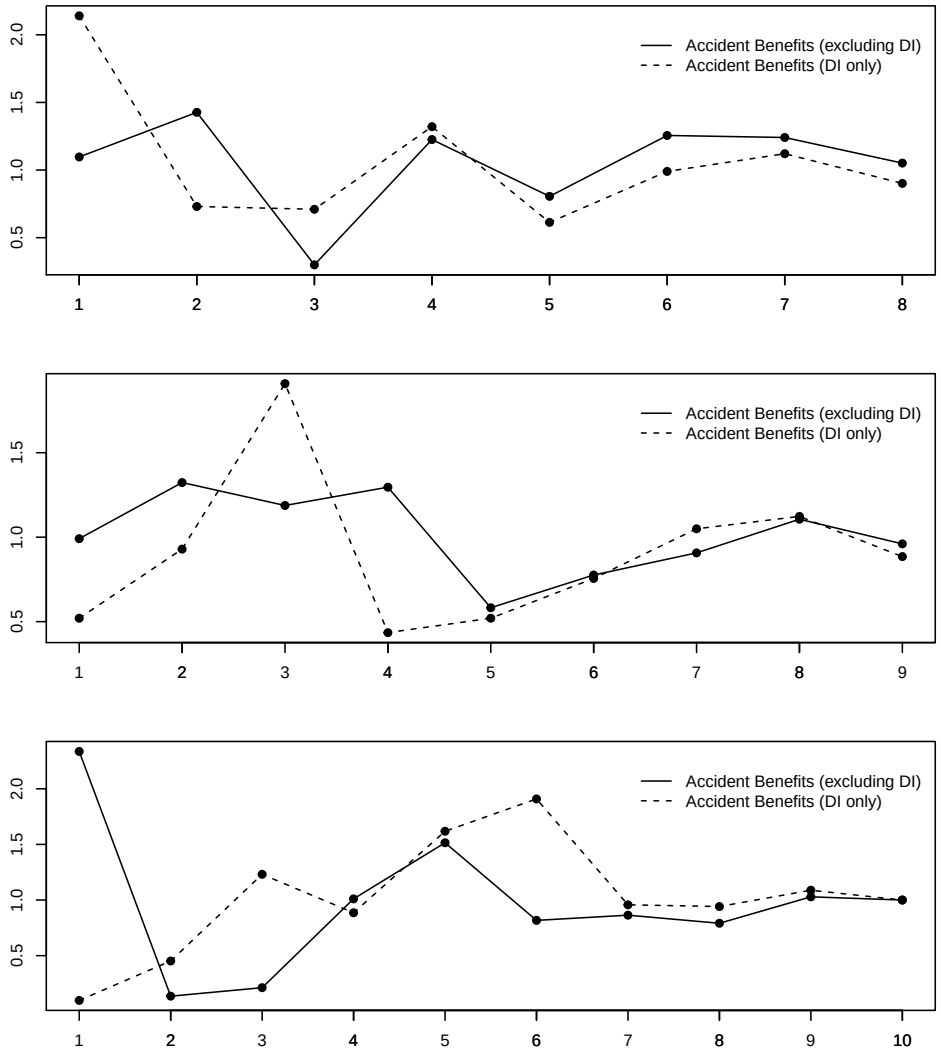

Figure 6.11: Plots of ratios of observed values to GLM fitted values for the last 3 calendar years

We look at residuals by calendar years to find another trace of calendar year trend and dependence. The residual for each calendar year is calculated as the difference between the sum of observed values and the sum of fitted values for all cells in that year, standardised using the sum of fitted values. Plots of calendar year residuals for the two lines are given in Figure 6.12. Clear evidence of calendar year dependence is observed from this figure. In particular, there are some sympathetic movements in the calendar year trends from both lines such as in calendar periods 1-2, 4-5, 7-9. This suggests some common effects that impact both lines, as well as idiosyncratic effects within individual lines. The Pearson correlation coefficient between the calendar year residuals from the two lines is 0.6976 (p-value 0.0249), which is strong and significant.

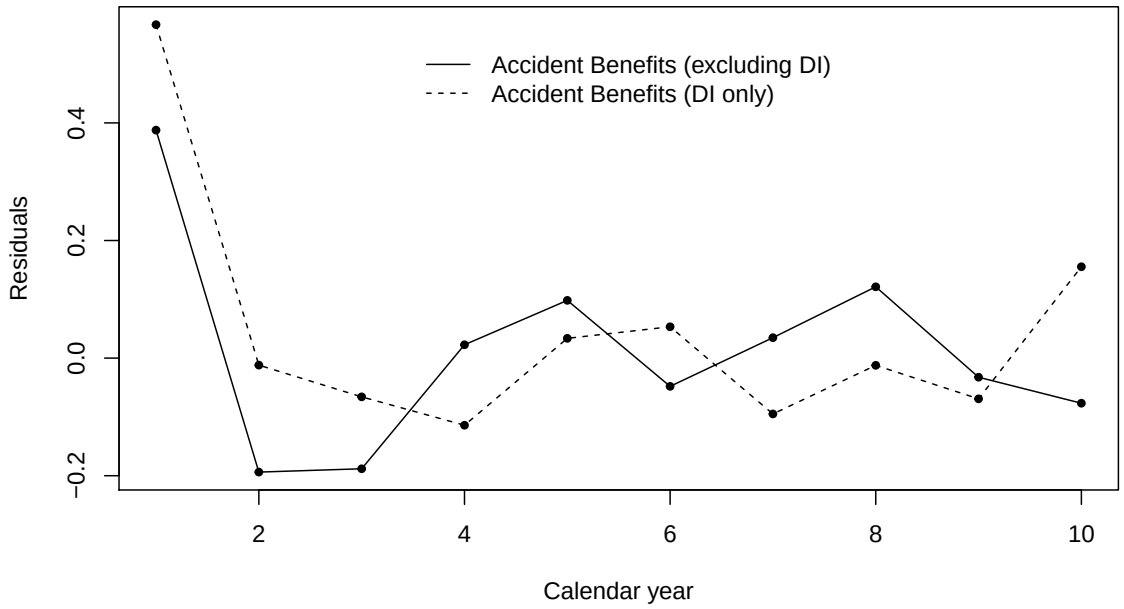

Figure 6.12: Plots of GLM residuals by calendar years

The exploratory dependence analysis shows some evidence of calendar year dependence across the two lines. Hence this data set is suitable for illustrating the multivariate evolutionary GLM framework.

6.5.2 Model used and estimation results

A multivariate evolutionary GLM is fitted to the data set. Instead of using a generic distribution from the EDF, we focus on distributions from the Tweedie sub-family of the EDF. This is because the Tweedie family is a major subclass of the EDF which covers the majority of commonly used distributions (Jorgensen, 1997; Alai et al., 2013;Avanzi, Taylor, Vu and Wong, 2016). In addition, the selection of a specific distribution from the Tweedie family simplifies conveniently to the estimation of the power parameter p. For this data set, different Tweedie distributions with different power parameters p are used, providing flexible dispersion modelling for the two lines.

The mean structure used is

$$a_i^{(n)} + r_i^{(n)} \log(j) + s_i^{(n)} j + b_{i,1}^{(n)} \mathbb{1}_{\{j=1\}} + b_{i,2}^{(n)} \mathbb{1}_{\{j=2\}} + h_t^{(n)}, \tag{6.73}$$

where $a_i^{(n)}$, $r_i^{(n)}$, $s_i^{(n)}$, $b_{i,1}^{(n)}$, $b_{i,2}^{(n)}$, $h_t^{(n)}$ are random factors that have random walk evolution

within their respective time dimension.

n	i	$a_i^{(n)}$	$r_i^{(n)}$	$s_i^{(n)}$	$b_{i,1}^{(n)}$	$b_{i,2}^{(n)}$	$h_i^{(n)}$
	1	-2.0093	1.3115	-0.6984	0.5553	0.2005	0.0000
	2	-2.0142	1.2026	-0.7005	0.0362	0.1557	-0.0028
	3	-2.0479	1.9093	-0.8420	0.2509	0.2237	0.0068
	4	-2.0474	1.9407	-0.8349	0.0359	0.1674	0.1814
	5	-2.0153	1.9509	-0.8733	0.5484	0.4951	-0.0153
	6	-1.8899	2.1597	-0.9237	0.5494	0.6518	-0.1817
	7	-1.9436	2.1534	-0.8723	0.6715	0.8886	-0.1791
(1)	8	-1.9963	2.0434	-0.8846	0.7631	0.7925	-0.2023
	9	-2.0942	1.8848	-0.8674	0.2985	0.3903	-0.2185
	10	-2.0660	1.9685	-0.9101	0.1290	0.3581	-0.2548
	1	-3.6999	1.9821	-0.8779	0.9202	0.2526	0.0000
	2	-3.6618	2.1940	-0.9283	0.6658	0.1919	0.0322
	3	-3.6864	2.0891	-0.8281	0.6168	0.2896	-0.0561
	4	-3.6632	2.2922	-0.8690	0.4334	0.1267	0.0855
	5	-3.6485	2.1814	-0.8517	0.3686	0.4682	0.2618
	6	-3.6084	2.1829	-0.7584	0.4518	0.4729	-0.2053
	7	-3.6036	2.0957	-0.7528	0.5783	0.5821	-0.3378
(2)	8	-3.6202	1.9516	-0.7664	0.5515	0.5776	-0.2683
	9	-3.5940	1.9978	-0.8600	0.3277	0.3639	-0.2342
	10	-3.6191	2.0065	-0.8329	-0.0011	0.2890	-0.1454

Table 6.4: Random factor estimates in real data illustration

We use 50,000 particles for each time step and initialise the filter using static GLM estimates with the above mean structure. We also examine the change in claims development pattern over time and perform a number of trial runs to select somewhat informative prior distributions for parameters which can reduce the degeneracy issue. The filtered estimates of random factors are given in Table 6.4. Parameter estimates are given in Table 6.5.

	$n=1$		$n=2$	
	Estimate	90% CI	Estimate	90% CI
$\phi^{(n)}$	0.0069	(0.0063; 0.0073)	0.0071	(0.0062; 0.0083)
$p^{(n)}$	1.2509	(1.2425; 1.2685)	1.3592	(1.3490; 1.3692)
$\sigma^2_{a^{\epsilon^{(n)}}}$	0.0048	(0.0043; 0.0053)	0.0035	(0.0032; 0.0040)
$\sigma^2_{r^{\epsilon^{(n)}}}$	0.0497	(0.0476; 0.0520)	0.0778	(0.0753; 0.0818)
$\sigma^2_{s^{\epsilon^{(n)}}}$	0.0078	(0.0074; 0.0083)	0.0151	(0.0138; 0.0172)
$\sigma^2_{b_{,1}{}^{\epsilon^{(n)}}}$	0.2057	(0.2002; 0.2131)	0.2058	(0.1942; 0.2131)
$\sigma^2_{b_{,2}{}^{\epsilon^{(n)}}}$	0.1313	(0.1257; 0.1353)	0.1180	(0.1021; 0.1301)
$\sigma^2_{h^{\epsilon^{(n)}}}$	0.0755	(0.0723; 0.0797)	0.1079	(0.0981; 0.1376)
${}_{\epsilon}\lambda^{(n)}$	0.7135	(0.6698; 0.7743)	0.6614	(0.6192; 0.7408)
$\sigma^2_{h^{\tilde{\epsilon}}}$	0.1114	(0.1060; 0.1209)		

Table 6.5: Parameter estimates in real data illustration

6.5.3 Goodness-of-fit analysis

The tracking of claims development patterns is provided in Figures 6.13 and 6.14. These plots show that the particle filter can track changes in claim activity quite reasonably well overall, especially in the last few years. There are also quite dramatic changes in the claims development pattern in some years, for example, from year 1 to 2, year 2 to 3, which are captured well by the model. Changes within the period from year 3 to 6 are quite rapid, which are captured by the model to some extent but not fully.

Heat maps of residuals are provided in Figure 6.15. The residuals in these heat maps are calculated as the ratios of observed values to fitted values. It can be observed that the goodness-of-fit is better than that of the traditional static GLM in Figure 6.12. The goodness-of-fit is noticeably better in early development years, especially the first two years. This is particularly useful for claims projection as early development years contribute significantly more to the total claims.

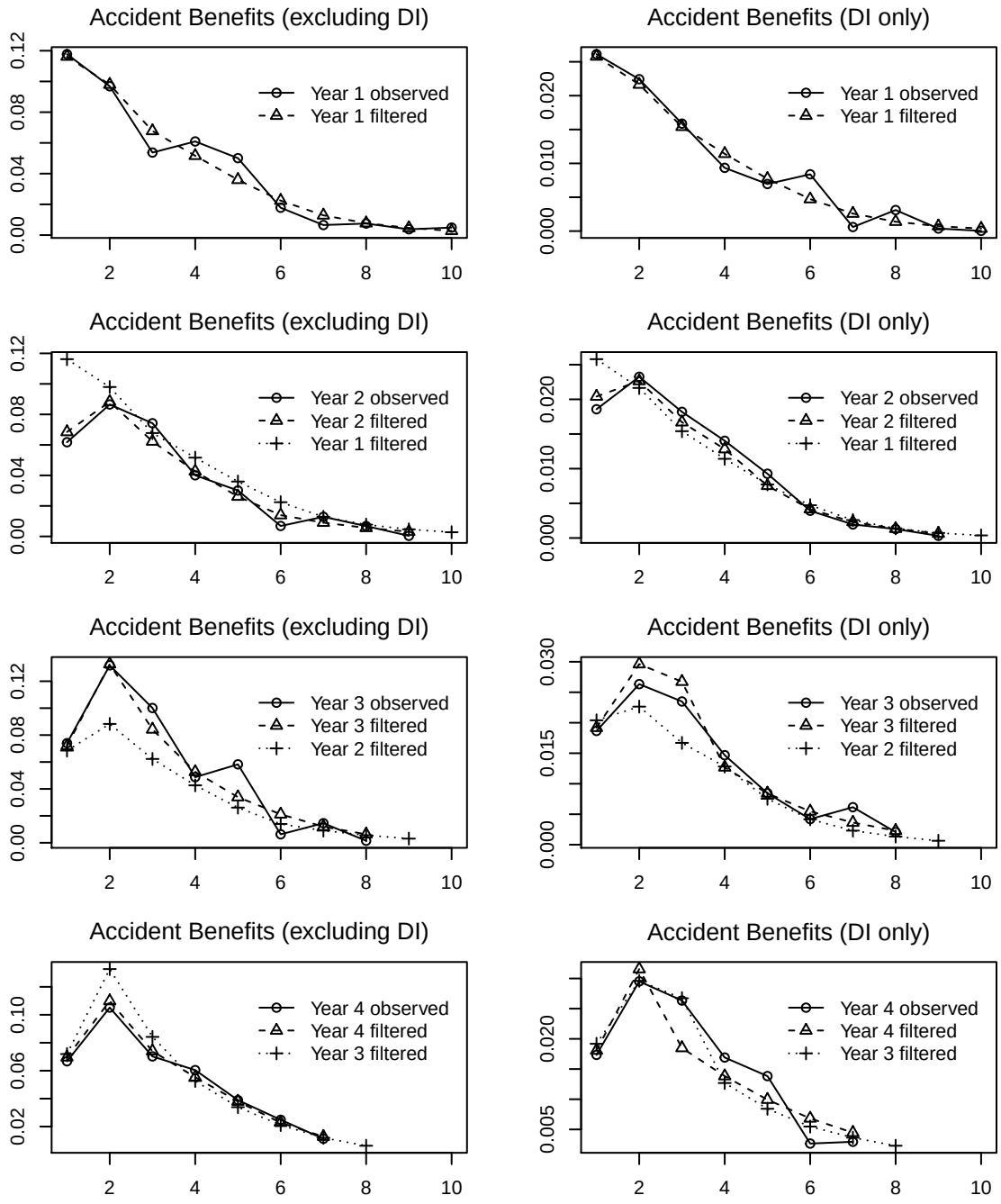

Figure 6.13: Tracking of claims development patterns in real data illustration (accident years 1-4)

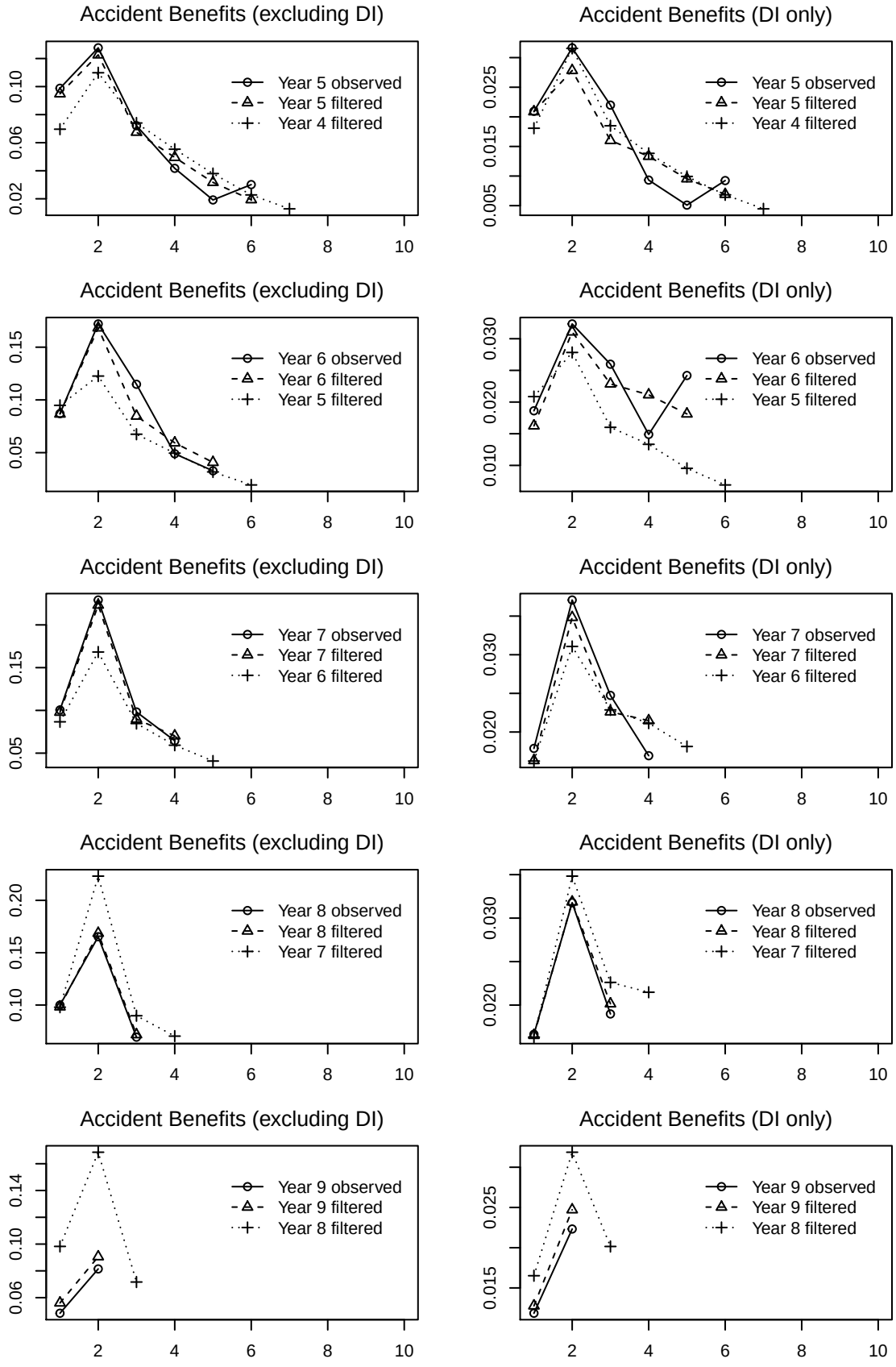

Figure 6.14: Tracking of claims development patterns in real data illustration (accident years 5-9)

Year	1	2	3	4	5	6	7	8	9	10
1	1.013	0.989	0.793	1.180	1.392	0.794	0.508	0.971	0.813	1.777
2	0.902	0.980	1.192	0.940	1.153	0.491	1.420	1.208	0.125	
3	1.029	0.993	1.189	0.926	1.717	0.297	1.213	0.233		
4	0.959	0.957	0.949	1.092	1.026	1.092	0.881			
5	1.041	1.041	1.069	0.842	0.605	1.555				
6	1.004	1.022	1.356	0.826	0.804					
7	1.027	1.026	1.094	0.918						
8	1.017	0.980	0.967							
9	0.863	0.899								
10	0.983									

Year	1	2	3	4	5	6	7	8	9	10
1	1.013	1.036	1.030	0.818	0.900	1.774	0.232	2.237	0.520	0.092
2	0.910	1.027	1.089	1.094	1.231	0.943	0.847	0.963	0.486	
3	0.973	0.889	0.879	1.157	1.004	0.767	1.693	0.896		
4	0.960	0.935	1.424	1.222	1.393	0.389	0.662			
5	1.003	1.137	1.373	0.699	0.534	1.336				
6	1.147	1.041	1.136	0.704	1.335					
7	1.096	1.065	1.094	0.789						
8	1.008	0.997	0.942							
9	0.928	0.903								
10	0.869									

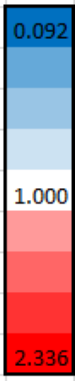

Figure 6.15: Heat maps of ratios of observed values to fitted values (top: Accident Benefits (excluding DI), bottom: Accident Benefits (DI only)

Plots of residuals in three dimensions: accident years, development years and calendar years are provided in Figure 6.16. Residuals in these plots are calculated as the average of fitted ratios of observed values to fitted values within each year in their respective dimension. The goodness-of-fit seems to deteriorate in later development years. This is due to the lack of available information for these late development lags. In addition, the use of the Hoerl curve to smooth out the whole development pattern may also attribute to this poor fit. Furthermore, observed values in these years are low, which can can also magnify the error ratios. The goodness-of-fit seems reasonable for accident year and calendar year residuals. The goodness-of-fit for calendar years 6 and 8 of Accident Benefits (DI only) is slightly worse. From examining Figures 6.13 and 6.14, the rapid change in the development pattern from year 3 to 5 seems to have contributed to this deviation.

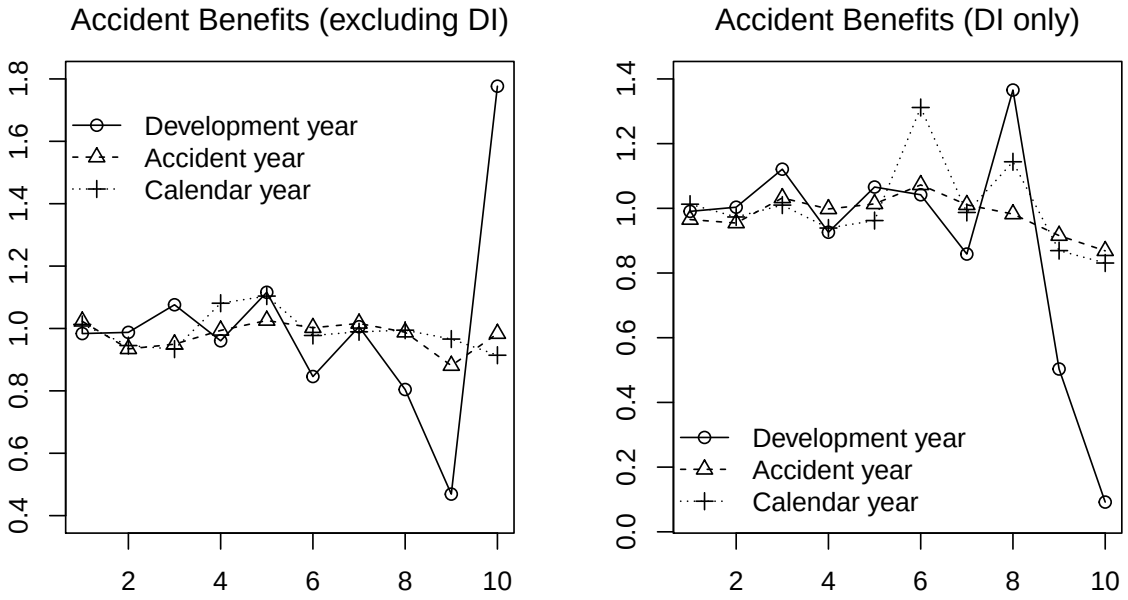

Figure 6.16: Plots of residuals by accident year, development year and calendar year

Using parameter estimates in Table 6.5, the Pearson correlation coefficient between calendar period factors $h_t^{(1)}$ and $h_t^{(2)}$ is 0.3133. The correlation coefficient between the filtered estimates of these factors is 0.7537 with 95% CI $(0.2362; 0.9381)$. This estimate is quite close to the coefficient between calendar year residuals from the two lines in the preliminary analysis.

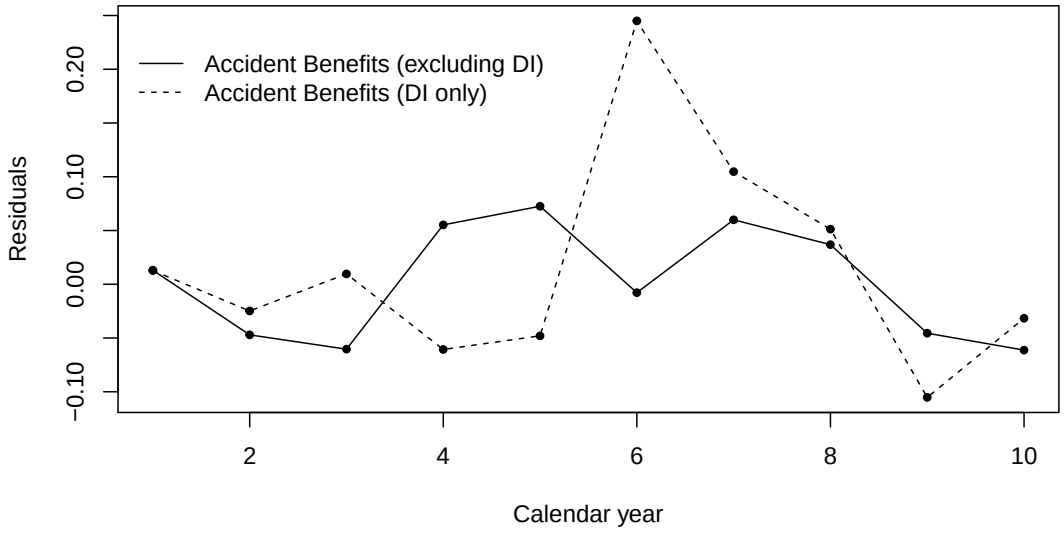

Figure 6.17: Residuals by calendar year

There is no clear evidence of calendar year dependence in the heat maps in Figure 6.15.

To further look for any trace of calendar year dependence, we plot calendar year residuals for the two lines in Figure 6.17. The residuals in this figure are calculated as the difference between the sum of observed values and the sum of fitted values for all claims in the same calendar year, standardised using the sum of fitted values. By comparing this figure with Figure 6.12, it can be observed that the clear positive dependence is no longer apparent. The Pearson correlation coefficient reduces to 0.1400 (p-value 0.6996), which is much weaker than 0.6976 and is also insignificant. Hence we can conclude that the model has captured the calendar year dependence quite well.

The goodness-of-fit in this illustration is not as good as the goodness-of-fit in the simulation illustration in Section 6.4.2. This is expected as the synthetic data set is simulated from a theoretical model, whereas the underlying model that generates the real data set is unknown. This is to say that there may be other factors in the data that are yet to be considered and captured in the model.

Before we close, we would also like to compare the results of our approach with the particle filtering results in the univariate evolutionary model by Sims (2012). It is observed that our results are more satisfactory than the results for univariate evolutionary models in Sims (2011), see also Section 2.5.5. Sims (2011) used the standard sequential Monte Carlo particle filter with the variances of the disturbance terms chosen by an initial residual analysis. Sims (2011) noted that the particle filter could not always keep track of the changes, and it also suffered from the degeneracy issue. The approach that we proposed, however, is an advancement of the traditional particle filter to also incorporate parameter estimation. This allows the variances of the disturbance terms to be updated upon the arrival of new observations. This could explain the better performance of our particle filter in tracking changes. In addition, the particle filter used in this chapter is a modification of the auxiliary particle filter. This filter typically places the re-sampling ahead of the evaluation step whereas the reverse order is typically performed in the traditional particle filter (see also the review in Doucet and Johansen, 2011). The re-sampling step utilises the importance weights calculated using the look-ahead-likelihood. This aims to reduce the degeneracy issue (Doucet and Johansen, 2011; Creal, 2012; Cappé et al., 2007). However, it is also noted that the degeneracy issue still exists to some extent in our filter, as shown in the parameter estimation of the simulation illustration in Section 6.4.2. The particle degeneracy problem is not as severe in this real data illustration perhaps because the dimension of observations is smaller (only up to 20 in the first time period whereas it is 30 in the simulation illustration).

6.5.4 Outstanding claims forecast

To forecast outstanding claims in the lower triangles, we utilise particles from the filtering for the upper triangles. These particles are samples of random factors and parameters. They are used to project future claims using the framework specification in Section 6.2. A set of samples of future claims can then be obtained, which is used to calculate summary statistics for the total outstanding claims liability.

The means and standard deviations of the total outstanding claims by accident years for each line of business and the total portfolio are summarised in Table 6.6. Summary statistics of the total outstanding claims distributions are given in Table 6.7 and their kernel densities are given in Figure 6.18. The summary statistics provided include the posterior means, standard deviations, $\text{VaR}_{75\%}$ and $\text{VaR}_{95\%}$ of the distributions of total outstanding claims for each line, as well as the total portfolio.

Year	AB (excluding DI)		AB (DI only)		Both lines	
	Mean	SD	Mean	SD	Mean	SD
1	187.49	193.56	38.30	44.31	225.79	197.85
2	579.92	361.71	216.96	142.92	796.89	388.72
3	1,448.14	698.41	526.54	299.49	1,974.68	764.69
4	2,299.10	1,117.06	940.59	446.71	3,239.70	1,188.88
5	5,504.30	2,482.17	3,695.25	1,644.89	9,199.55	2,806.42
6	13,128.14	5,506.59	5,456.64	2,618.54	18,584.78	6,155.75
7	17,550.89	8,319.55	6,530.34	3,742.70	24,081.23	9,194.94
8	19,898.82	11,057.94	7,026.08	6,163.77	26,924.90	12,297.21
9	31,035.09	20,539.99	14,891.23	24,332.31	45,926.32	32,660.82

Table 6.6: Outstanding claims statistics by accident year

	AB (excluding DI)	AB (DI only)	Both lines
Mean	91,631.90	39,321.94	130,953.84
SD	28,960.41	26,251.88	40,495.76
$\text{VaR}_{75\%}$	106,307.33	44,375.00	147,631.89
$\text{VaR}_{95\%}$	143,964.73	75,211.60	198,509.08

Table 6.7: Summary statistics of outstanding claims distributions

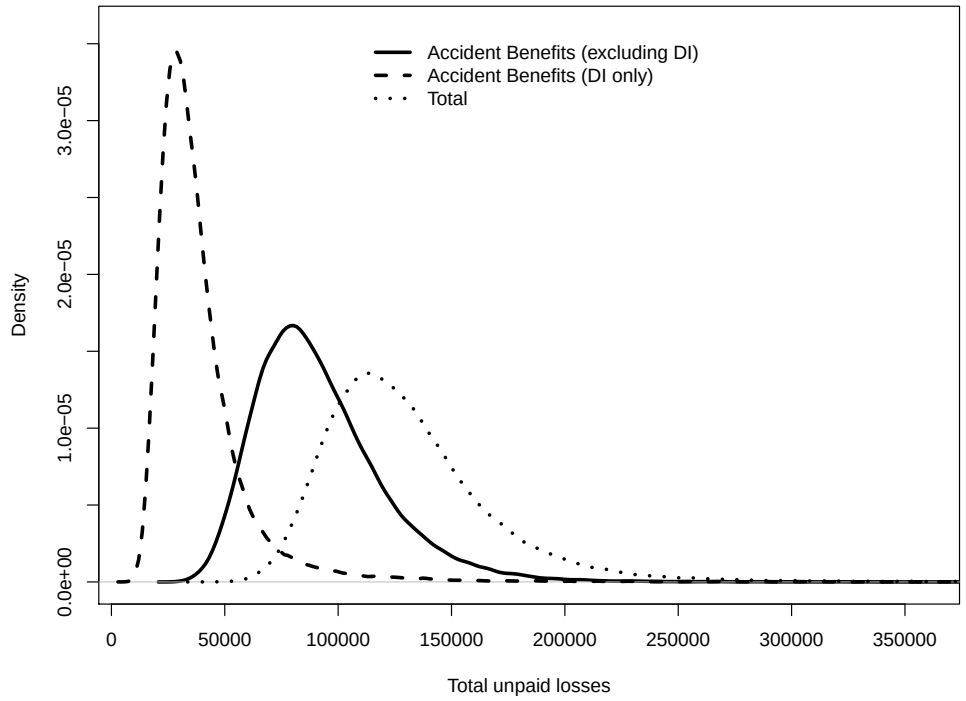

Figure 6.18: Kernel densities of predictive distributions of total outstanding claims in each line of business and in the aggregate portfolio

The two lines, however, do not have a comonotonic dependence structure, and this allows the insurer to gain a diversification benefit when they set their risk margins. Recall the definitions of risk margins and diversification benefits from Section 4.4.4

$$\text{Risk margin}_{\chi\%}[Y] = \max\left\{\text{VaR}_{\chi\%}[Y] - E[Y];\, \frac{1}{2}SD[Y]\right\}, \tag{4.14}$$

$$\text{DB} = \frac{\big(\text{Risk margin}_{\chi\%}[Y_1] + \text{Risk margin}_{\chi\%}[Y_2]\big) - \text{Risk margin}_{\chi\%}[Y_1+Y_2]}{\text{Risk margin}_{\chi\%}[Y_1] + \text{Risk margin}_{\chi\%}[Y_2]} \times 100\%. \tag{4.15}$$

The Risk $\text{Margin}_{75\%}$ and Risk $\text{Margin}_{95\%}$, as well as their corresponding diversification benefits are provided in Table 6.8. It can then be observed that quite significant diversification benefits can be gained as a result of allowing for (non-comonotonic) dependence across lines in the valuation of the total outstanding claims liability.

	AB (excluding DI)	AB (DI only)	Both lines	DB
Risk $\text{Margin}_{75\%}$	14,675.43	13,125.94	20,247.88	27.2%
Risk $\text{Margin}_{95\%}$	52,332.83	35,889.66	67,555.24	23.4%

Table 6.8: Risk margin and diversification benefits statistics

6.6 Remarks on framework properties and applications

We have introduced a multivariate evolutionary GLM framework with great flexibility and formulated filters that recursively update factors upon the arrival of new observations. In this section we provide some remarks on the properties of the framework and its applications. These include modelling flexibility, the dimension within which the filters are run, potential compensation effects in the estimation results, some caution in initialising the filters, and the use of smoothing.

6.6.1 Modelling flexibility

The framework introduced in this chapter provides great modelling flexibility. Various distributions from the EDF can be used with very flexible mean structures. The numerical illustrations in this chapter use the Tweedie subclass of the EDF. This allows model uncertainty to be considered conveniently through the estimation of the power parameter p.

We specify special random walk evolution for random factors. This can be modified easily by using more complex time series processes. However, it is worth noting that a more complex structure will increase the number of parameters in the model. This may not be desirable given the typically small sample size of reserving data. A preliminary analysis of the data and expert opinions can help select appropriate specifications for the evolution.

The dependence across segments is captured using a common shock approach in this framework. Common shock approaches provide many benefits, such as an explicit dependence structure and ease of interpretation (see also Chapters 1 and 2). We specifically target the calendar period dependence in the specification of our framework. This, however, can be modified easily to incorporate other sources of dependence such as common accident period effects, and common development period effects.

6.6.2 Dimension of filters

The particle filter and the dual Kalman filter that we introduce in this chapter are accident-period-based. It means that they proceed from one accident period to another.

This is to utilise the greater availability of data in the first accident period, which helps initialise the filters more accurately. However, given that new information typically arrives by calendar periods, it can be for better risk management purposes to have the filters run by calendar periods. This, however, creates some additional complications due to the lack of data for initialisation. In addition, this diagonal flow of the filters also misaligns with the dimension within which accident period factors and development factors typically evolve. The technique that we use to filter calendar factors in accident-period-based filters can be modified to address the latter.

6.6.3 Compensations across random factors and variances of disturbance terms

The simulation illustrations in Section 6.4 show a decent overall goodness-of-fit. This is evident in the heat maps of actual to fitted values in this section. However, there is evidence of some compensation in the estimation: a compensation between variance terms, and a compensation between random factors.

Variance terms are unknown parameters that need to be estimated in the framework. These include dispersion parameters of the observations, and variances of disturbance terms in the evolution of random factors. While the framework has two components: observations and random factors, random factors are latent and only the total volatility is observed. This may cause a mis-allocation across variance terms. This issue is unlikely to distort the projection of claims if it is handled with caution. For example, observed rapid changes in claims development patterns should be appropriately recognised with large variance estimates of development factors. As long as the overall goodness-of-fit is reasonable, the compensation across component volatilities should not have material impacts on the projection of future claims because the total volatility of future claims is the subject of interest. A careful examination of data features as well as any expert opinions can help select more informative starting values and prior distributions of variance parameters. It is also worth noting that while it is desirable for prior distributions of parameters to have large variances, these prior distributions should not be too vague to avoid particle degeneracy issues. Trial runs may be required to select the optimal variances for these prior distributions. This task is quite similar in nature to the "tuning" process of the traditional Metropolis-Hastings algorithm of Bayesian inference.

Compensation can also occur across random factors: accident period factors, development period factors and calendar period factors. As shown in Figure 6.2 and 6.5 of the simulation illustrations, compensation across accident period factors and calendar period factors is the most prominent. The relatively stable fitting ratios in these figures, however, demonstrate a consistent mis-allocation between these factors. This issue is not only specific to evolutionary models but is also very common in models that consider all three factors (Zehnwirth, 1994; Taylor, 2000; Barnett and Zehnwirth, 2000; Brydon and Verrall, 2009; Gluck and Venter, 2009; Venter et al., 2017). Due to the collinearity between these factors, they can largely offset to give an overall reasonable fit. Experienced practitioners also tend to be more interested in the combination of these factors rather than their individual trends (Venter et al., 2017). As the ultimate goal of any valuation task is to forecast outstanding claims, a question is then raised regarding the impacts of this mis-allocation on the accurateness of claims projection. McGuire et al. (2018) showed that extrapolating future trends of these factors using their corresponding estimates from the past would produce reasonably accurate future claims experience. This works well for cases with constant calendar year trends. When the trends are not constant, one should proceed with caution and some reasonableness checks of the trends can be useful.

6.6.4 Particle degeneracy issue and initialisation of the particle filter

Particle degeneracy can be a potential issue in the application of the particle learning approach. This is the situation when the set of samples contains repeated copies of a few particles with significant weights, hence the diversity of particles is significantly low (see also the discussion on this in Section 2.5.4.1). While this problem is unavoidable in particle learning and particle filtering in general (see for example, see for example, Doucet et al., 2000; Andrieu et al., 2005; Carvalho et al., 2010; Creal, 2012), it can be more severe for data of a high dimension, or when uninformative priors for parameters and initial values of random factors are used (Wang et al., 2017; Li et al., 2014). A consequence of this problem is the underestimation of parameter errors and observation errors. One should be mindful of this issue when applying the particle learning approach. A careful selection of priors and initial values may be required to reduce this issue.

In the selection of initial values for the filters, it may be useful to use GLM estimates for random factors. A number of trial runs may be required to determine the appropriate

starting values of parameters as well as their prior distributions.

6.6.5 Smoothing

It is often desirable to perform back smoothing following filtering to obtain estimates using all available information. Back smoothing is a lot easier to accomplish for Gaussian models because of the availability of the estimates in closed-form. For non-Gaussian models, particle smoothing can be used, however, it is not an easy task due to particle degeneracy (Doucet and Johansen, 2011). This problem is further escalated in the particle learning algorithm in Section 6.3.1 as parameters are also incorporated in the on-line estimation. We do not perform particle smoothing for the evolutionary GLM framework, except for Gaussian cases where the Kalman smoother can be readily applied. Further research could further investigate this aspect, especially with regard to addressing the degeneracy problem.

6.A Appendices

6.A.1 Simulated data set 1 and estimation results

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	602.30	1,122.25	901.25	525.92	425.89	262.43	127.27	64.13	41.27	23.73	10.53	6.18	2.78	1.49	0.62
2	739.34	664.01	752.44	416.75	325.01	135.44	75.48	40.77	19.49	10.31	4.96	2.45	1.06	0.65	
3	642.28	826.71	618.59	376.01	258.32	170.15	77.54	51.40	27.13	10.89	6.08	2.86	1.67		
4	597.75	824.04	802.20	392.36	342.95	224.24	136.74	71.39	24.44	11.09	7.19	4.57			
5	653.93	854.13	585.56	327.69	336.58	174.33	54.79	38.31	19.46	10.25	4.26				
6	589.12	816.28	728.59	535.12	351.20	252.35	130.37	73.02	48.38	19.39					
7	515.76	677.00	514.67	537.08	403.71	169.33	99.82	48.23	29.44						
8	549.98	692.63	718.62	377.89	263.34	163.87	84.29	46.39							
9	700.79	1034.79	747.80	506.48	307.90	285.76	121.78								
10	607.15	757.41	781.25	537.82	268.81	270.59									
11	462.82	573.68	498.72	341.35	151.33										
12	403.08	553.90	350.35	262.98											
13	511.76	573.32	460.24												
14	501.46	833.57													
15	455.05														

Table 6.9: Simulated triangle 1 (data set 1)

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1,110.95	4,125.89	7,255.83	6,720.31	8,879.99	8,492.29	8,410.17	7,719.22	6,409.51	4,194.32	4,613.07	3,735.88	3,265.72	2,033.61	1,773.12
2	1,153.94	4,440.45	5,258.84	6,772.19	7,630.38	8,339.50	7,222.77	7,276.79	6,971.93	5,030.54	4,308.98	3,792.31	2,864.62	2,428.61	
3	1,175.68	3,808.38	5,035.29	8,467.15	9,737.88	10,737.90	7,911.10	7,065.64	7,520.91	6,506.90	7,946.16	5,042.42	4,129.61		
4	1,152.67	3,329.42	4,897.89	7,969.27	9,604.94	8,988.17	7,278.42	9,352.41	10,721.72	6,477.61	6,724.26	6,923.10			
5	1,255.58	4,131.81	6,093.36	7,047.69	7,421.58	8,589.54	9,307.11	9,856.34	6,881.90	6,423.12	6,670.25				
6	1,450.77	3,743.02	5,707.90	7,841.52	7,957.82	9,560.97	11,412.13	8,484.77	7,766.90	9,538.77					
7	1,633.88	3,902.83	5,523.56	7,995.11	1,1360.42	9,428.78	9,488.62	9,056.63	7,984.47						
8	1,197.88	4,152.19	5,991.96	7,882.29	9,299.56	12,460.42	8,803.62	9,782.86							
9	940.55	3,621.44	5,016.44	7,132.98	8,305.76	10,619.71	10,580.20								
10	1,308.16	3,712.33	5,478.54	6,777.43	8,380.06	6,191.61									
11	1,526.47	4,223.27	6,408.18	8,847.79	9,453.75										
12	1,255.17	3,926.82	7,308.47	8,060.79											
13	1,374.72	4,738.92	9,770.35												
14	1,631.05	4,631.16													
15	1,284.50														

Table 6.10: Simulated triangle 2 (data set 1)

n	i	$a_i^{(n)}$		$r_i^{(n)}$		$s_i^{(n)}$		$h_i^{(n)}$	
		True	Estimate	True	Estimate	True	Estimate	True	Estimate
	1	6.9947	7.0167	1.6359	1.6529	-0.8251	-0.8246	0.5000	0.4500
	2	6.8641	6.8959	1.5157	1.5883	-0.8461	-0.8614	0.4994	0.4486
	3	6.7749	6.8469	1.4596	1.6042	-0.8198	-0.8476	0.4983	0.4472
	4	6.8542	6.8484	1.5136	1.6380	-0.8145	-0.8359	0.4966	0.4460
	5	6.8415	6.7906	1.5659	1.6063	-0.8725	-0.8570	0.4965	0.4447
	6	6.7952	6.7901	1.6813	1.6750	-0.8167	-0.8120	0.4966	0.4437
	7	6.7427	6.7203	1.6303	1.6500	-0.8279	-0.8242	0.4943	0.4427
(1)	8	6.6634	6.7123	1.6671	1.6296	-0.8539	-0.8305	0.5037	0.4420
	9	6.7226	6.7797	1.6943	1.6602	-0.8145	-0.8097	0.5140	0.4414
	10	6.5345	6.7336	1.6630	1.6479	-0.7961	-0.8142	0.5236	0.4410
	11	6.4229	6.5946	1.5842	1.5735	-0.8452	-0.8506	0.5164	0.4404
	12	6.3549	6.5436	1.5084	1.5487	-0.8547	-0.8626	0.5125	0.4399
	13	6.5337	6.5954	1.5677	1.5628	-0.7860	-0.8564	0.5201	0.4395
	14	6.4938	6.6432	1.6236	1.5835	-0.7700	-0.8487	0.5280	0.4394
	15	6.4094	6.6221	1.5665	1.5835	-0.7463	-0.8500	0.5318	0.4394
	1	7.0454	6.9475	2.0693	2.1165	-0.3817	-0.3690	0.5000	0.4500
	2	6.9772	6.9574	2.0261	2.1017	-0.3516	-0.3645	0.5097	0.4507
	3	6.7969	6.9628	2.0704	2.1020	-0.3242	-0.3431	0.5175	0.4513
	4	6.8370	6.9530	2.0286	2.0908	-0.3076	-0.3323	0.5171	0.4517
	5	6.8698	6.9679	2.0191	2.0734	-0.3108	-0.3349	0.5127	0.4520
	6	6.9182	6.9927	2.0681	2.0702	-0.3193	-0.3297	0.5122	0.4524
	7	6.9022	7.0095	2.0371	2.0665	-0.3152	-0.3309	0.5080	0.4528
(2)	8	6.8453	7.0013	2.1274	2.0691	-0.3460	-0.3294	0.5088	0.4530
	9	6.8523	6.9778	2.1398	2.0612	-0.3203	-0.3347	0.5168	0.4532
	10	7.0142	6.9933	2.0757	2.0465	-0.3846	-0.3446	0.5257	0.4537
	11	7.0037	7.0529	2.0768	2.0630	-0.3419	-0.3356	0.5319	0.4542
	12	7.0128	7.0758	2.0799	2.0749	-0.3349	-0.3305	0.5378	0.4547
	13	7.0317	7.1187	2.1660	2.0908	-0.3208	-0.3224	0.5312	0.4551
	14	7.0509	7.1231	2.1437	2.0916	-0.3374	-0.3218	0.5297	0.4554
	15	7.0849	7.1081	2.1988	2.0916	-0.3068	-0.3228	0.5293	0.4557

Table 6.11: Random factor estimates (data set 1)

6.A.2 Simulated data set 2 and estimation results

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	780.22	1,001.31	976.39	701.15	479.12	332.35	165.10	105.16	53.81	26.41	13.14	6.73	2.94	2.61	0.98
2	712.18	1,044.98	846.14	576.32	356.55	212.97	121.20	62.19	37.04	26.98	12.22	7.55	2.14	0.43	
3	870.46	1,005.42	878.00	611.21	411.37	243.74	167.85	80.50	45.12	20.05	8.31	4.85	5.10		
4	796.02	1,166.45	945.30	705.18	364.32	260.10	121.88	89.35	27.32	26.98	9.74	4.62			
5	814.11	1,079.17	932.74	629.16	431.17	244.96	141.16	76.29	42.55	17.88	10.80				
6	877.20	1,137.54	938.62	685.14	397.95	283.48	142.15	67.91	44.72	11.41					
7	920.94	1,251.27	1,141.94	892.30	517.03	381.02	165.43	115.42	60.30						
8	869.26	1,342.65	1,147.71	950.62	516.04	379.84	214.98	113.79							
9	819.89	1,324.12	1,151.73	923.22	576.67	365.41	182.59								
10	797.00	1,227.72	1,356.88	965.16	634.22	411.00									
11	767.37	1,351.61	1,419.23	1,074.30	645.94										
12	603.26	1,028.40	924.51	738.57											
13	425.34	865.54	765.00												
14	457.73	828.66													
15	497.53														

Table 6.12: Simulated triangle 1 (data set 2)

Year	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1,051.49	2,908.76	3,548.39	4,180.12	3,829.17	3,787.87	3,237.76	2,821.79	2,541.13	1,856.58	1,600.69	1,107.31	795.35	679.36	433.67
2	1,022.46	2,414.12	4,009.28	4,351.30	4,447.23	4,234.74	3,417.64	2,904.03	2,378.16	1,751.58	1,390.12	1,242.83	853.65	612.16	
3	997.67	2,478.47	3,965.30	4,434.88	4,643.37	3,949.76	3,572.67	3,046.95	2,406.56	1,883.33	1,480.23	1,063.52	834.77		
4	1,484.59	3,761.86	5,391.49	6,020.91	6,196.67	5,587.35	4,898.76	4,270.64	3,200.27	2,721.99	2,145.32	1,582.65			
5	1,324.08	2,889.24	4,610.08	5,023.86	4,489.16	4,821.97	3,633.48	2,989.67	2,458.95	1,752.05	1,385.68				
6	1,356.38	3,131.53	4,208.76	5,070.06	4,751.72	4,416.58	3,408.66	3,051.95	2,217.67	1,686.76					
7	1,429.51	3,861.67	5,211.83	5,706.28	5,433.40	5,117.32	3,881.77	3,152.25	2,472.31						
8	1,704.91	4,737.91	6,515.72	7,522.14	7,422.98	6,023.96	5,078.15	4,351.21							
9	1,727.04	4,301.44	6,132.19	6,642.10	6,395.65	5,747.83	5,108.37								
10	1,547.62	4,520.63	6,008.15	6,082.58	6,159.39	4,856.79									
11	2,112.87	4,442.72	6,124.43	6,819.85	6,265.20										
12	1,899.73	4,876.19	6,771.83	7,467.21											
13	2,103.70	5,379.00	7,336.02												
14	1,320.09	4,684.38													
15	1,726.91														

Table 6.13: Simulated triangle 2 (data set 2)

n	i	$a_i^{(n)}$		$r_i^{(n)}$		$s_i^{(n)}$		$h_i^{(n)}$	
		True	Estimate	True	Estimate	True	Estimate	True	Estimate
	1	6.9830	6.9434	1.5955	1.7118	-0.7850	-0.7965	0.5000	0.4500
	2	6.8905	6.9085	1.5986	1.7002	-0.8064	-0.8266	0.5023	0.4481
	3	7.0009	6.9989	1.6360	1.6632	-0.8222	-0.8160	0.5030	0.4417
	4	6.9880	7.0861	1.5933	1.6074	-0.8155	-0.8175	0.5044	0.4379
	5	7.0329	7.0630	1.6335	1.6829	-0.8312	-0.8429	0.5111	0.4302
	6	7.0457	7.1332	1.6793	1.7175	-0.8375	-0.8576	0.5217	0.4236
	7	7.0402	7.2059	1.6851	1.7497	-0.7905	-0.8304	0.5356	0.4286
(1)	8	7.0322	7.1696	1.7397	1.7738	-0.7953	-0.8217	0.5334	0.4304
	9	7.0264	7.1542	1.8041	1.8150	-0.8248	-0.8370	0.5267	0.4144
	10	7.0400	7.0915	1.8931	1.8493	-0.8340	-0.8126	0.5353	0.4458
	11	6.9343	7.0847	1.9495	1.9501	-0.8087	-0.8291	0.5380	0.4430
	12	6.7371	6.9144	1.9621	1.9714	-0.8504	-0.8871	0.5337	0.4644
	13	6.5899	6.7457	2.0376	2.0315	-0.8852	-0.9303	0.5283	0.4621
	14	6.6038	6.6860	2.0236	2.0380	-0.9016	-0.9332	0.5284	0.4576
	15	6.4945	6.6808	2.0333	2.0421	-0.8839	-0.9370	0.5349	0.4689
	1	6.8839	6.9921	1.9463	1.8302	-0.4366	-0.4132	0.5000	0.4500
	2	6.9029	6.9835	1.9466	1.9113	-0.4371	-0.4282	0.5013	0.4490
	3	7.0123	6.9676	1.9224	2.0187	-0.4437	-0.4514	0.4982	0.4517
	4	7.1320	7.2572	1.9728	1.9782	-0.4274	-0.4353	0.5052	0.4398
	5	7.0479	7.2527	1.9766	1.9529	-0.4586	-0.4675	0.5157	0.4297
	6	7.1557	7.2514	1.9492	1.9416	-0.4740	-0.4712	0.5242	0.4314
	7	7.3043	7.3410	1.9718	1.9777	-0.4841	-0.4782	0.5297	0.4285
(2)	8	7.4428	7.5354	1.9541	2.0351	-0.4506	-0.4811	0.5466	0.4251
	9	7.3675	7.4923	1.9626	2.0190	-0.4497	-0.4814	0.5418	0.4281
	10	7.4645	7.4848	1.9253	2.0159	-0.4764	-0.4970	0.5345	0.4574
	11	7.4939	7.5738	1.9464	1.9828	-0.4691	-0.4909	0.5261	0.4360
	12	7.4921	7.5815	1.9997	1.9813	-0.4427	-0.4679	0.5159	0.4631
	13	7.5051	7.6527	2.0748	2.0262	-0.4515	-0.4767	0.5063	0.4661
	14	7.3779	7.4958	2.1793	2.0874	-0.4452	-0.4833	0.4905	0.4492
	15	7.3656	7.4958	2.1708	2.0822	-0.4737	-0.4825	0.4894	0.4387

Table 6.14: Random factor estimates (data set 2)

6.A.3 Empirical data set

This data set is drawn from Côté et al. (2016).

Year	Premium	1	2	3	4	5	6	7	8	9	10
1	116,491	13,714	24,996	31,253	38,352	44,185	46,258	47,019	47,894	48,334	48,902
2	111,467	6883	16,525	24,796	29,263	32,619	33,383	34,815	35,569	35,612	
3	107,241	7933	22,067	32,801	38,028	44,274	44,948	46,507	46,665		
4	105,687	7052	18,166	25,589	31,976	36,092	38,720	39,914			
5	105,923	10,463	23,982	31,621	36,039	38,070	41,260				
6	111,487	9697	28,878	41,678	47,135	50,788					
7	113,268	11,387	37,333	48,452	55,757						
8	121,606	12,150	32,250	40,677							
9	110,610	5348	14,357								
10	104,304	4,612									

Table 6.15: Accident Benefits excluding Disability Income (cumulative claims)

Year	Premium	1	2	3	4	5	6	7	8	9	10
1	116,491	3,043	5,656	7,505	8,593	9,403	10,380	10,450	10,812	10,856	10,860
2	111,467	2,070	4662	6,690	8,253	9,286	9,724	9,942	10,086	10,121	
3	107,241	2,001	4,825	7,344	8,918	9,824	10,274	10,934	11,155		
4	105,687	1,833	4,953	7,737	9,524	10,986	11,267	11,579			
5	105,923	2,217	5,570	7,898	8,885	9,424	10,402				
6	111,487	2,076	5,681	8,577	10,237	12,934					
7	113,268	2,025	6,225	9,027	10,945						
8	121,606	2,024	5,888	8,196							
9	110,610	1,311	3,780								
10	104,304	912									

Table 6.16: Accident Benefits - Disability Income only (cumulative claims)

CHAPTER 7

Conclusions

By considering and incorporating realistic and desirable model features, insurers can accurately assess their outstanding claims liabilities in a convenient manner. Many of these features include dependency across business segments, unbalanced nature of reserving data, changes in claim activity over time, non-positive claim observations, marginal flexibly and more.

An important aspect of reserving data is potential dependence across business segments within a portfolio. By considering this feature, insurers can more accurately allow for diversification benefits when they set risk margins for their outstanding claims liabilities as well as risk-based capital. A number of multivariate reserving models have been developed using copulas, multivariate distributions with specific marginals, and common shocks. Common shock techniques are traditional dependence modelling tools with various applications not only in insurance but also in many other areas. They are well known for their ability to capture dependence drivers transparently with explicit dependence structures. As a result, these drivers can be estimated, communicated, as well as monitored if needed. The construction of correlation matrices can also be more parsimonious and disciplined. This is particularly beneficial in practice as correlation matrices are a major tool used by practitioners to present dependence across segments in their portfolios.

In this thesis we propose a number of developments in reserving that all draw from common shock modelling techniques to utilise their many benefits. These developments consider many realistic and desirable modelling features. A summary of contributions is provided in Section 7.1, followed by limitations and areas for future research in Section 7.2.

7.1 Summary of contributions

The overall aim of this thesis is to develop approaches that have realistic and desirable features to improve the valuation of outstanding claims liability while still offering great practicality.

In Chapter 3 (some of the results of which were published in Avanzi, Taylor, Vu and Wong, 2016), we contribute to the existing literature with the development of a common shock Tweedie framework which has many desirable properties. We consider the Tweedie family of distributions, a very rich major sub-class of the EDF which covers various symmetric and non-symmetric, light-tailed and heavy-tailed distributions (Alai et al., 2016; Alai and Wüthrich, 2009; Furman and Landsman, 2010; Jorgensen, 1997). This family is characterised by the specification of a variance function μ^p, where values of the power parameter p identify the corresponding member of the family and they can be anywhere in $(-\infty, 0] \cap [1, \infty)$. We utilise the richness of the Tweedie family and develop a multivariate Tweedie framework using a common shock approach. While offering the various benefits of common shock approaches as mentioned earlier, this framework also provides great marginal flexibility and as a result, enhanced practicality. It is also a multivariate generalisation of ODP models which utilise over-dispersed Poisson distributions of the Tweedie family with $p = 1$. The univariate ODP models are well known stochastic models that underlie the famous chain-ladder algorithm.

The common shock Tweedie framework developed in Chapter 3 also provides other desirable features besides marginal flexibility and explicit dependence structure with ease of interpretation. In particular, the multivariate Tweedie's compound Poisson cases of this framework where $p \in (1, 2)$ can handle claim observations of 0's. These observations can be encountered quite often in reserving data. We also provide closed-form moments, including the mean and variance, of the sum of total outstanding claims liability in this chapter. These results are particularly beneficial when the calculation of these quantities are computationally expensive. For the sake of simplicity, the most simple parametrisation for the common shock variable is assumed, which leads to some issues with the unbalanced nature of reserving data. This issue is analysed and addressed in Chapter 5.

Chapter 4 focuses on the estimation and applications of the multivariate Tweedie framework. The Tweedie family of distributions has a complex density. The complexity is elevated in a multivariate framework. We overcome this issue with the formulation of

an efficient Bayesian approach for model estimation. The performance of model estimation was assessed using simulated data sets, and illustrated using a real data set from Schedule P (available in Zhang and Dukic, 2013). The benefit of marginal flexibility offered by the framework is emphasised in the illustration with real data. This property allows the multivariate Tweedie framework to provide a good fit to the data set compared to some special members of the family with specific power parameters p. The Bayesian estimation with efficient MCMC algorithms reduces the need to use closed-form cumulants of the sum of outstanding claims. However, we can still take the advantage of having tractable cumulants of outstanding claims to efficiently compute the mean and the variance of total outstanding losses.

We propose the multivariate Tweedie approach for outstanding claims liabilities valuation with dependence. However, the development of this approach, as well as many remarks and considerations on the theoretical and practical properties, could be applied in other contexts where additive background systematic risks exist such as mortality modelling (Alai et al., 2016), capital modelling (Furman and Landsman, 2010), and so forth.

While common shock approaches have many benefits, they may create problems when applied to reserving data without careful modelling. This is particularly the case when claim activity varies quite significantly within a single triangle, and also across triangles. It typically reaches a peak in some early periods, then dies out as the delay increases. Furthermore, different segments can have different claims experience as some can be longer-tailed than others. We refer to this characteristic as the unbalanced nature of reserving data. It is then desirable to use scaling factors to adjust the common shock contributions proportionately to the total observations over the entire range of the triangles. However, an excessive use of scaling factors can result in over-parametrisation. In some cases such as the common shock Tweedie framework developed in earlier chapters, there may also be a desire to maintain distributional tractability. In Chapter 5, we propose an approach which compromises the various conflicting problems arising from the unbalanced nature of reserving data. This approach involves using careful and parsimonious parametrisation to develop a common shock Tweedie framework modified for unbalanced data. Numerical illustrations show a substantial improvement in the performance of the framework modified for unbalanced data, compared to the original framework in Chapters 3 and 4.

It can also be quite common to observe negative entries in loss triangles due to for

example, salvage recoveries and payment from third parties. This data feature, however, can create difficulties for many models which do not offer support for negative claims. In Chapter 5, we also incorporate a translation factor to account for this feature in the modified common shock Tweedie framework. This translation factor is estimated via Bayesian inference that so that their uncertainty can be formally assessed in the valuation of outstanding claims liabilities.

Insurers typically experience changes in their claim experience over time, making the application of static models with deterministic parameters no longer straightforward. We capture this common data feature in a multivariate evolutionary GLM framework in Chapter 6. This framework utilises the very popular and rich GLM class, hence provides great flexibility in marginal modelling and mean structure. We extend the traditional GLM framework in loss reserving on two fronts. Firstly, we allow parameters of the traditional GLM framework to evolve, hence enable changes in claim experience to be captured naturally in an elegant manner. This helps provide a clear picture of the historical experience. Secondly, we introduce dependence across segments using a common shock approach with an explicit and easy-to-interpret dependence structure.

Together with the development of the multivariate evolutionary GLM framework, we also contribute to the literature with the formulation of two filters in Chapter 6: a particle filter with parameters learning for the general framework, and a dual Kalman filter for the special case of Gaussian models. These filters are real-time devices that recursively update random factors and parameters upon the arrival of new information. It gives more weight to more recent data, hence provides a more accurate projection of future claims. In the special structure of reserving data with three different time dimensions, the application of a standard filter is not straightforward. We take into account this difficulty in the development of our filters. However, there can be compensation between random factors, and across variance parameters. A careful selection of initial values is required to reduce this as well as the numerical instability of the filters.

7.2 Limitations and areas for future research

In this thesis, common shock techniques are used to develop approaches that provide many realistic and desirable properties such as marginal flexibility, unbalanced data

treatment, and evolutionary structure. However, these approaches are still bound to some limitations which could give some directions for future research. Frameworks developed in this thesis are applied to loss reserving, however, general approaches and considerations could still be considered in other fields. These limitations and areas for future research are provided in detail below.

The common shock Tweedie framework in Chapters 3 and 4 offers an advantage of marginal flexibility through flexible choices of power parameter p. This is demonstrated in the illustration using real data in Chapter 4 where this flexibility provides a much better goodness-of-fit compared to cases where the parameter p is fixed. However, all marginal distributions are required to belong to the Tweedie family with the same p. While this is to maintain closure under the taking of marginals, one of the four desirable properties of multivariate distributions in Joe (1997, Chapter 4), it restricts the marginal flexibility of the framework. Claims from all business segments are required to have the same dispersion which can be violated in cases where some segments have diverse properties. Furthermore, the multivariate Tweedie framework captures cell-wise dependence across triangles. Future research could consider modifying this dependence structure to capture other sources of dependence such as development period dependence, calendar period dependence, or accident period dependence.

Our research raises some potential issues of common shock models when they are applied to reserving data that has an unbalanced nature. These issues, however, might appear whenever common shock models are applied to heterogeneous data. These can include mortality data for different group ages, or capital modelling for different types of risks. A solution to these issues is proposed in Chapter 5, which could be extended to solve similar problems in other contexts. While this solution can reduce the problems of unbalanced data quite substantially, a complete balance in common shock proportions cannot be achieved. Future research could consider a better solution to this problem. Other multivariate models with explicit dependence structures such as mixture models could also be considered as they might be more applicable to unbalanced data.

To capture changes in claim activity over time, we propose a multivariate evolutionary GLM framework in Chapter 6. This framework specifies calendar period dependence and random walk evolution for random factors, which could be modified to provide more complex structures. The filters formulated are accident period-based as they proceed from one accident

period to another. This is to utilise the benefit of having more available data in the first accident period for initialisation. However, as data arrives by calendar years, one could consider the formulation of calendar period-based filters. The development of a particle back smoothing algorithm to obtain optimal estimates of random factors using all available information could also be an area of development for future research.

The particle learning approach used for estimation of random factors can experience the degeneracy issue when it is applied on high-dimensional data. While informative initial values and priors can be used to reduce this problem to some extent, it can still be quite severe for high-dimensional data sets of many large triangles. This problem is quite well known for particle learning and particle filtering in general. Some advanced particle filters have been formulated to overcome the problem of particle degeneracy which could be used in further research. These filters, however, are often known to have significant additional computational requirements hence they need to be evaluated carefully before they are used for future developments.

The evolutionary framework and filters that we develop can be used to build reserving robots which can automate repetitive reserving jobs. This is particularly useful, and even essential, as the demand for more frequent outstanding claims valuation has been increasing. Future research could further into the complete construction of reserving robots, which might involve more detailed selection of initial values, disturbance variances, model selection and blending of results from different models. However, as shown in the numerical illustrations of the evolutionary framework, compensation across factors can occur in automated models. Future research could further look into this issue and should be mindful of it in developing and applying these models.

The developments in this thesis provide great marginal flexibility. Chapters 3 and 4 focus on the Tweedie family of distribution, while Chapter 6 considers the EDF with the flexible GLM structure. However, the EDF and its Tweedie subclass, are relatively light-tailed. Other families of heavier tailed distributions could be considered in the future, such as the generalised beta distribution family of second kind.

Despite many distinctive strengths, common shock approaches proposed in this thesis can only capture positive dependence structures. Negative dependences can be observed in practice which may invalidate the applications of proposed approaches. In such cases, other dependence modelling approaches such as copulas (for example, Shi and Frees, 2011; De Jong,

2012) and mixture approaches (for example Lee and Lin, 2012; Willmot and Woo, 2015) could be considered.

Bibliography

Abdallah, A., Boucher, J.P., Cossette, H., 2015. Modeling dependence between loss triangles with hierarchical Archimedean copulas. ASTIN Bulletin 45, 577–599.

Abdallah, A., Boucher, J.P., Cossette, H., Trufin, J., 2016. Sarmanov family of bivariate distributions for multivariate loss reserving analysis. North American Actuarial Journal 20, 184–200.

Ajne, B., 1994. Additivity of chain-ladder projections. ASTIN Bulletin 24, 311–318.

Alai, D.H., Landsman, Z., Sherris, M., 2013. Lifetime dependence modelling using a truncated multivariate gamma distribution. Insurance: Mathematics and Economics 52, 542–549.

Alai, D.H., Landsman, Z., Sherris, M., 2016. Multivariate Tweedie lifetimes: The impact of dependence. Scandinavian Actuarial Journal 8, 692–712.

Alai, D.H., Merz, M., Wüthrich, M.V., 2009. Mean square error of prediction in the Bornhuetter-Ferguson claims reserving method. Annals of Actuarial Science 4, 7–31.

Alai, D.H., Wüthrich, M.V., 2009. Taylor approximations for model uncertainty within the Tweedie exponential dispersion family. ASTIN Bulletin 39, 453–477.

Alpuim, T., Ribeiro, I., 2003. A state space model for run-off-off triangles. Applied Stochastic Models in Business and Industry 19, 105–120.

Andrieu, C., Doucet, A., Tadić, V.B., 2005. On-line parameter estimation in general state-space models, in: Proceedings of the 44th IEEE Conference on Decision and Control, pp. 332–337.

Andrieu, C., Doucet, A., Tadić, V.B., 2012. On-line parameter estimation in general state-space models using a pseudo-likelihood approach, in: IFAC Proceedings, pp. 500–505.

Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T., 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing 50, 174–188.

Atherino, R., Pizzinga, A., Fernandes, C., 2010. A row-wise stacking of the runoff triangle: State space alternatives for IBNR reserve prediction. ASTIN Bulletin 40, 917–946.

Avanzi, B., Taylor, G., Vu, P.A., Wong, B., 2016. Stochastic loss reserving with dependence: A flexible multivariate Tweedie approach. Insurance: Mathematics and Economics 71, 63–78.

Avanzi, B., Taylor, G., Wong, B., 2018. Common shock models for claim arrays. ASTIN Bulletin 48, 1–28.

Avanzi, B., Taylor, G.C., Wong, B., 2016. Correlations between insurance lines of business: An illusion or a real phenomenon? Some methodological considerations. ASTIN Bulletin 46, 225–263.

Azam, S.E., Chatzi, E., Papadimitriou, C., 2015. A dual Kalman filter approach for state estimation via output-only acceleration measurements. Mechanical Systems and Signal Processing 60, 866–886.

Barnett, G., Zehnwirth, B., 2000. Best estimates for reserves, in: Proceedings of the Casualty Actuarial Society, pp. 245–321.

Berquist, J.R., Sherman, R.E., 1977. Loss reserve adequacy testing: A comprehensive, systematic approach, in: Proceedings of the Casualty Actuarial Society, pp. 123–184.

Bornhuetter, R.L., Ferguson, R.E., 1972. The actuary and IBNR, in: Proceedings of the Casualty Actuarial Society, pp. 181–195.

Boucher, J.P., Davidov, D., 2011. On the importance of dispersion modeling for claims reserving: An application with the Tweedie distribution. Variance 5, 158–172.

Braun, C., 2004. The prediction error of the chain ladder method applied to correlated run-off triangles. ASTIN Bulletin 34, 399–424.

Brooks, S., Gelman, A., Jones, G., Meng, X., 2011. Handbook of Markov chain Monte Carlo. Chapman & Hall.

Brydon, D., Verrall, R., 2009. Calendar year effects, claims inflation and the chain-ladder technique. Annals of Actuarial Science 4, 287–301.

Cappé, O., Godsill, S.J., Moulines, E., 2007. An overview of existing methods and recent advances in sequential Monte Carlo, in: Proceedings of the IEEE, pp. 899–924.

Carvalho, C.M., Johannes, M.S., Lopes, H.F., Polson, N.G., et al., 2010. Particle learning and smoothing. Statistical Science 25, 88–106.

Chukhrova, N., Johannssen, A., 2017. State space models and the Kalman-filter in stochastic claims reserving: Forecasting, filtering and smoothing. Risks 5, 30.

Congdon, P.D., 2010. Applied Bayesian hierarchical methods. Chapman & Hall.

Côté, M.P., Genest, C., Abdallah, A., 2016. Rank-based methods for modeling dependence between loss triangles. European Actuarial Journal 6, 377–408.

Creal, D., 2012. A survey of sequential Monte Carlo methods for economics and finance. Econometric Reviews 31, 245–296.

De Alba, E., 2002. Bayesian estimation of outstanding claim reserves. North American Actuarial Journal 6, 1–20.

De Alba, E., 2006. Claims reserving when there are negative values in the runoff triangle: Bayesian analysis using the three-parameter log-normal distribution. North American Actuarial Journal 10, 45–59.

De Alba, E., Nieto-Barajas, L.E., 2008. Claims reserving: A correlated Bayesian model. Insurance: Mathematics and Economics 43, 368–376.

De Jong, P., 2006. Forecasting runoff triangles. North American Actuarial Journal 10, 28–38.

De Jong, P., 2012. Modeling dependence between loss triangles. North American Actuarial Journal 16, 74–86.

De Jong, P., Heller, G.Z., et al., 2008. Generalized linear models for insurance data. Cambridge University Press.

De Jong, P., Zehnwirth, B., 1983. Claims reserving, state-space models and the Kalman filter. Journal of the Institute of Actuaries 110, 157–181.

Dong, A.X.D., Chan, J.S.K., 2013. Bayesian analysis of loss reserving using dynamic models with generalized beta distribution. Insurance: Mathematics and Economics 53, 355–365.

Doucet, A., De Freitas, N., Gordon, N., 2001. Sequential Monte Carlo methods in practice. Springer.

Doucet, A., Godsill, S., Andrieu, C., 2000. On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing 10, 197–208.

Doucet, A.D., Johansen, A.M., 2011. A tutorial on particle filtering and smoothing: 15 years later, in: Handbook of Nonlinear Filtering. Oxford University Press. volume 12, pp. 656–704.

Durante, F., Sempi, C., 2010. Copula theory: An introduction, in: Jaworski, P., Durante, F., Härdle, W.K., Rychlik, T. (Eds.), Copula Theory and Its Applications. Springer, pp. 3–31.

Durbin, J., Koopman, S.J., 2012. Time series analysis by state space methods. Oxford University Press.

Embrechts, P., Frey, R., McNeil, A., 2005. Quantitative risk management. Princeton University Press.

Embrechts, P., Lindskog, F., McNeil, A., 2003. Modelling dependence with copulas and applications to risk management. Handbook of Heavy Tailed Distributions in Finance 8, 329–384.

England, P.D., Verrall, R.J., 1999. Analytic and bootstrap estimates of prediction errors in claims reserving. Insurance: Mathematics and Economics 25, 281–293.

England, P.D., Verrall, R.J., 2001. A flexible framework for stochastic claims reserving, in: Proceedings of the Casualty Actuarial Society, pp. 1–38.

England, P.D., Verrall, R.J., 2002. Stochastic claims reserving in general insurance. British Actuarial Journal 8, 443–518.

England, P.D., Verrall, R.J., 2006. Predictive distributions of outstanding liabilities in general insurance. Annals of Actuarial Science 1, 221–270.

England, P.D., Verrall, R.J., Wüthrich, M.V., 2012. Bayesian over-dispersed Poisson model and the Bornhuetter & Ferguson claims reserving method. Annals of Actuarial Science 6, 258–283.

Fearnhead, P., 2008. Computational methods for complex stochastic systems: A review of some alternatives to MCMC. Statistics and Computing 18, 151–171.

Fleming, K.G., Mayer, J.H., 1988. Adjusting incurred losses for simultaneous shifts in payment patterns and case reserve adequacy levels. Technical Report. Casualty Actuarial Society.

Frees, E.W., Derrig, R.A., Meyers, G. (Eds.), 2014. Predictive modeling applications in actuarial science. volume 1 of *International Series on Actuarial Science*. Cambridge University Press.

Frees, E.W., Meyers, G., Derrig, R.A. (Eds.), 2016. Predictive modeling applications in actuarial science. volume 2 of *International Series on Actuarial Science*. Cambridge University Press.

Friedland, J., 2010. Estimating unpaid claims using basic techniques. Technical Report. Casualty Actuarial Society.

Furman, E., 2008. On a multivariate gamma distribution. Statistics and Probability Letters 78, 2353–2360.

Furman, E., Landsman, Z., 2010. Multivariate Tweedie distributions and some related capital-at-risk analyses. Insurance: Mathematics and Economics 46, 351–361.

Ghezzi, T.L., 2001. Loss reserving without loss development patterns–Beyond Berquist-Sherman, in: Casualty Actuarial Society E-forum, pp. 43–104.

Gismondi, F., Janssen, J., Manca, R., 2012. The construction of the claims reserve distribution by means of a semi-Markov backward simulation model. Annals of Actuarial Science 6, 23–64.

Givens, G.H., Hoeting, J.A., 2005. Computational statistics. John Wiley & Sons.

Glasserman, P., 2003. Monte Carlo methods in financial engineering. Springer.

Gluck, S.M., Venter, G.G., 2009. Stochastic trend models in casualty and life insurance, in: Enterprise Risk Management Symposium.

Gove, J.H., Hollinger, D.Y., 2006. Application of a dual unscented Kalman filter for simultaneous state and parameter estimation in problems of surface-atmosphere exchange. Journal of Geophysical Research: Atmospheres 111.

Hachemeister, C.A., Stanard, J.N., 1975. IBNR claims count estimation with static lag functions, in: Proceedings of the Casualty Actuarial Society.

Heberle, J., Thomas, A., 2016. The fuzzy Bornhuetter–Ferguson method: An approach with fuzzy numbers. Annals of Actuarial Science 10, 303–321.

Hess, K.T., Schmidt, K.D., 2002. A comparison of models for the chain–ladder method. Insurance: Mathematics and Economics 31, 351–364.

Hess, K.T., Schmidt, K.D., Zocher, M., 2006. Multivariate loss prediction in the multivariate additive model. Insurance: Mathematics and Economics 39, 185–191.

International Actuarial Association, 2004. A global framework for insurer solvency assessment. Website. URL: `https://www.actuaries.org/LIBRARY/Papers/Global_Framework_Insurer_Solvency_Assessment-public.pdf`. last accessed: 28/11/2017.

Joe, H., 1997. Multivariate models and dependence concepts. Chapman & Hall.

Johnson, N.L., Kotz, S., Balakrishnan, N., 1997. Discrete multivariate distributions. volume 165. John Wiley & Sons.

Johnson, N.L., Kotz, S., Balakrishnan, N., 2002. Continuous multivariate distributions, models and applications. volume 1. John Wiley & Sons.

Jorgensen, B., 1997. The theory of dispersion models. volume 76. Chapman & Hall.

Joshi, M.S., Paterson, J.M., 2013. Introduction to mathematical portfolio theory. Cambridge University Press.

Kaas, R., Goovaerts, M., Dhaene, J., Denuit, M., 2008. Modern actuarial risk theory: Using R. volume 128. Springer.

Kantas, N., Doucet, A., Singh, S.S., Maciejowski, J., Chopin, N., et al., 2015. On particle methods for parameter estimation in state-space models. Statistical Science 30, 328–351.

Kantas, N., Doucet, A., Singh, S.S., Maciejowski, J.M., 2009. An overview of sequential Monte Carlo methods for parameter estimation in general state-space models, in: IFAC Proceedings, pp. 774–785.

Karlis, D., 2003. An EM algorithm for multivariate Poisson distribution and related models. Journal of Applied Statistics 30, 63–77.

Kirschner, G.S., Kerley, C., Isaacs, B., 2002. Two approaches to calculating correlated reserve indications across multiple lines of business, in: Proceedings of the Casualty Actuarial Society, pp. 211–246.

Kocherlakota, S., Kocherlakota, K., 1992. Bivariate discrete distributions. Marcel Dekker.

Koop, G., 2003. Bayesian econometrics. John Wiley & Sons.

Kroese, D.P., Taimre, T., Botev, Z.I., 2011. Handbook of Monte Carlo Methods. John Wiley & Sons.

Kruschke, J.K., 2011. Doing Bayesian data analysis: A tutorial introduction with R. Academic Press.

Kunkler, M., 2006. Modelling negatives in stochastic reserving models. Insurance: Mathematics and Economics 38, 540–555.

Landsman, Z.M., Valdez, E.A., 2003. Tail conditional expectations for elliptical distributions. North American Actuarial Journal 7, 55–71.

Lee, S.C.K., Lin, X.S., 2012. Modeling dependent risks with multivariate Erlang mixtures. ASTIN Bulletin 42, 153–180.

Li, T., Sun, S., Sattar, T.P., Corchado, J.M., 2014. Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches. Expert Systems with Applications 41, 3944–3954.

Lindskog, F., McNeil, A.J., 2003. Common Poisson shock models: Applications to insurance and credit risk modelling. ASTIN Bulletin 33, 209–238.

Liu, J., West, M., 2001. Combined parameter and state estimation in simulation-based filtering, in: Doucet, A., de Freitas, N., Gordon, N. (Eds.), Sequential Monte Carlo methods in practice. Springer, pp. 197–223.

Lopes, H.F., Tsay, R.S., 2011. Particle filters and bayesian inference in financial econometrics. Journal of Forecasting 30, 168–209.

Mack, T., 1991. A simple parametric model for rating automobile insurance or estimating IBNR claims reserves. ASTIN Bulletin 21, 93–109.

Mack, T., 1993. Distribution-free calculation of the standard error of chain ladder reserve estimates. ASTIN Bulletin 23, 213–225.

Mack, T., Venter, G., 2000. A comparison of stochastic models that reproduce chain ladder reserve estimates. Insurance: Mathematics and Economics 26, 101–107.

Mathai, A.M., Moschopoulos, P.G., 1991. On a multivariate gamma. Journal of Multivariate Analysis 39, 135–153.

McCullagh, P., Nelder, J.A., 1989. Generalized linear models (2ed). Chapman & Hall.

McGuire, G., 2007. Building a reserving robot, in: Biennial Convention - Actuaries Institute.

McGuire, G., Taylor, G., Miller, H., 2018. Self-assembling insurance claim models using regularized regression and machine learning. SSRN .

Merz, M., Wüthrich, M.V., 2008. Prediction error of the multivariate chain ladder reserving method. North American Actuarial Journal 12, 175–197.

Merz, M., Wüthrich, M.V., 2009a. Combining chain-ladder and additive loss reserving method for dependent lines of business. Variance 3, 270–291.

Merz, M., Wüthrich, M.V., 2009b. Prediction error of the multivariate additive loss reserving method for dependent lines of business. Variance 3, 131–151.

Merz, M., Wüthrich, M.V., Hashorva, E., 2013. Dependence modelling in multivariate claims run-off triangles. Annals of Actuarial Science 7, 3–25.

Meyers, G., 2009. Bayesian analysis with the Metropolis-Hastings algorithm. The Actuarial Review 36, 32–33.

Meyers, G.G., 2007. The common shock model for correlated insurance losses. Variance 1, 40–52.

Meyers, G.G., Shi, P., 2011. The retrospective testing of stochastic loss reserve models. Casualty Actuarial Society E-forum Summer, 1–37.

Lafaye de Micheaux, P., Drouilhet, R., Liquet, B. (Eds.), 2013. The R software: Fundamentals of programming and statistical analysis. Springer.

Miller, M.B., 2013. Mathematics and statistics for financial risk management. John Wiley & Sons.

Miranda, M.D.M., Nielsen, J.P., Verrall, R., 2012. Double chain ladder. ASTIN Bulletin 42, 59–76.

Nelsen, R.B., 1999. An introduction to copulas. Springer.

Nelson, L., Stear, E., 1976. The simultaneous on-line estimation of parameters and states in linear systems. IEEE Transactions on Automatic Control 21, 94–98.

Ntzoufras, I., Dellaportas, P., 2002. Bayesian modelling of outstanding liabilities incorporating claim count uncertainty. North American Actuarial Journal 6, 113–125.

Peters, G.W., Shevchenko, P., Wüthrich, M., 2009. Model uncertainty in claims reserving within Tweedie's compound Poisson models. ASTIN Bulletin 39, 1–33.

Renshaw, A.E., 1989. Chain ladder and interactive modelling (Claims reserving and GLIM). Journal of the Institute of Actuaries 116, 559–587.

Renshaw, A.E., 1994. Modelling the claims process in the presence of covariates. ASTIN Bulletin 24, 265–285.

Renshaw, A.E., 1996. Claims reserving by joint modelling. Insurance Mathematics and Economics 3, 239–240.

Renshaw, A.E., Verrall, R.J., 1998. A stochastic model underlying the chain-ladder technique. British Actuarial Journal 4, 903–923.

Rios, M.P., Lopes, H.F., 2013. The extended Liu and West filter: Parameter learning in Markov switching stochastic volatility models, in: Zeng, Y., Wu, S. (Eds.), State-space models: Applications in economics and finance. Springer, pp. 23–61.

Saluz, A., Gisler, A., 2014. Best estimate reserves and the claims development results in consecutive calendar years. Annals of Actuarial Science 8, 351–373.

Salzmann, R., Wüthrich, M.V., 2012. Modeling accounting year dependence in runoff triangles. European Actuarial Journal 2, 227–242.

Salzmann, R., Wütrich, M.V., Merz, M., 2012. Higher moments of the claims development result in general insurance. ASTIN Bulletin 42, 355–384.

Schmidt, K.D., 2002. A note on the over-dispersed Poisson family. Insurance: Mathematics and Economics 30, 21–25.

Schmidt, K.D., 2006. Optimal and additive loss reserving for dependent lines of business. Casualty Actuarial Society E-forum Fall, 319–351.

Schmidt, K.D., Zocher, M., 2008. The Bornhuetter-Ferguson principle. Variance 2, 85–110.

Shi, P., 2014. A copula regression for modeling multivariate loss triangles and quantifying reserving variability. ASTIN Bulletin 44, 85–102.

Shi, P., 2017. A multivariate analysis of intercompany loss triangles. Journal of Risk and Insurance 84, 717–737.

Shi, P., Basu, S., Meyers, G.G., 2012. A Bayesian log-normal model for multivariate loss reserving. North American Actuarial Journal 16, 29–51.

Shi, P., Frees, E.W., 2011. Dependent loss reserving using copulas. ASTIN Bulletin 41, 449–486.

Shi, P., Valdez, E.A., 2014. Multivariate negative binomial models for insurance claim counts. Insurance: Mathematics and Economics 55, 18–29.

Sims, J., 2011. Evolutionary reserving models - Are particle filters the way to go?, in: GIRO conference.

Sims, J., 2012. Seeing the bigger picture in claims reserving, in: General Insurance Seminar, Actuaries Institute.

Sims, J., 2014. Robotic reserving - Are we there yet?, in: General Insurance Seminar.

State Insurance Regulatory Authority, 2018. CTP green slip reforms. Website. URL: `https://www.sira.nsw.gov.au/fraud-and-regulation/reforms/ctp-green-slip-reforms`. last accessed: 26/08/2018.

Taylor, G., 2000. Loss reserving: An actuarial perspective. Kluwer Academic Publishers.

Taylor, G., 2008. Second-order Bayesian revision of a generalised linear model. Scandinavian Actuarial Journal 2008, 202–242.

Taylor, G., 2009. The chain ladder and Tweedie distributed claims data. Variance 3, 96–104.

Taylor, G., 2011. Maximum likelihood and estimation efficiency of the chain ladder. ASTIN Bulletin 41, 131–155.

Taylor, G., McGuire, G., 2008. Robotic reserving, in: GIRO Convention.

Taylor, G., McGuire, G., 2009. Adaptive reserving using Bayesian revision for the exponential dispersion family. Variance 3, 105–130.

Taylor, G., McGuire, G., 2016. Stochastic loss reserving using generalized linear models. Casualty Actuarial Society Monograph.

Taylor, G., McGuire, G., Greenfield, A., 2003. Loss reserving: Past, present and future. University of Melbourne Research Paper.

Taylor, G., Sullivan, J., 2016. Generalized linear models as predictive claim models, in: Frees, E.W., Meyers, G., Derrig, R.A. (Eds.), Predictive Modeling Applications in Actuarial Science. Cambridge University Press. volume 2, pp. 60–99.

Taylor, G.C., 1977. Separation of inflation and other effects from the distribution of non-life insurance claim delays. ASTIN Bulletin 9, 219–230.

Taylor, G.C., 2018. Observations on industry practice in the construction of large correlation structures for risk and capital margins. European Actuarial Journal 8, 517–543.

Taylor, G.C., Ashe, F., 1983. Second moments of estimates of outstanding claims. Journal of Econometrics 23, 37–61.

Ting Lee, M.L., 1996. Properties and applications of the Sarmanov family of bivariate distributions. Communications in Statistics-Theory and Methods 25, 1207–1222.

Tsionas, E.G., 2004. Bayesian inference for multivariate gamma distributions. Statistics and Computing 14, 223–233.

Venter, G., Gutkovich, R., Gao, Q., 2017. Parameter reduction in actuarial triangle models. Variance In press.

Verrall, R., Hössjer, O., Björkwall, S., 2012. Modelling claims run-off with reversible jump Markov chain Monte Carlo methods. ASTIN Bulletin 42, 35–58.

Verrall, R.J., 1989. A state space representation of the chain ladder linear model. Journal of the Institute of Actuaries 116, 589–609.

Verrall, R.J., 1994. A method for modelling varying run-off evolutions in claims reserving. ASTIN Bulletin 24, 325–332.

Verrall, R.J., 2000. An investigation into stochastic claims reserving models and the chain-ladder technique. Insurance: Mathematics and Economics 26, 91–99.

Vihola, M., 2012. Robust adaptive Metropolis algorithm with coerced acceptance rate. Statistics and Computing 22, 997–1008.

Vu, P.A., 2013. Stochastic loss reserving with dependence: A multivariate gamma approach. Honours thesis. University of New South Wales.

Wang, X., Li, T., Sun, S., Corchado, J.M., 2017. A survey of recent advances in particle filters and remaining challenges for multitarget tracking. Sensors 17, 2707.

Welch, G., Bishop, G., 1995. An introduction to the Kalman filter.

Wenzel, T.A., Burnham, K., Blundell, M., Williams, R., 2006. Dual extended Kalman filter for vehicle state and parameter estimation. Vehicle System Dynamics 44, 153–171.

Willmot, G.E., Woo, J.K., 2015. On some properties of a class of multivariate Erlang mixtures with insurance applications. ASTIN Bulletin 45, 151–173.

Wright, T.S., 1990. A stochastic method for claims reserving in general insurance. Journal of the Institute of Actuaries 117, 677–731.

Wüthrich, M.V., 2003. Claim reserving using Tweedie's compound Poisson model. ASTIN Bulletin 33, 331–346.

Wüthrich, M.V., 2010. Accounting year effects modeling in the stochastic chain ladder reserving method. North American Actuarial Journal 14, 235–255.

Wüthrich, M.V., Merz, M., 2008. Stochastic claims reserving methods in insurance. John Wiley & Sons.

Zehnwirth, B., 1989. Regression methods – Applications, in: Casualty Loss Reserve Seminar.

Zehnwirth, B., 1994. Probabilistic development factor models with applications to loss reserve variability, prediction intervals and risk based capital. Casualty Actuarial Society E-forum 2, 447–606.

Zhang, Y., 2010. A general multivariate chain ladder model. Insurance: Mathematics and Economics 46, 588–599.

Zhang, Y., Dukic, V., 2013. Predicting multivariate insurance loss payments under the Bayesian copula framework. Journal of Risk and Insurance 80, 891–919.

Zhang, Y., Dukic, V., Guszcza, J., 2012. A Bayesian non-linear model for forecasting insurance loss payments. Journal of the Royal Statistical Society 175, 637–656.

APPENDIX A

R codes

A.1 R codes for Chapter 4

A.1.1 Simulation illustration 1

```
library(tweedie)
library(MASS)
library(statmod)
library(truncnorm)
library(rootSolve)
library(LaplacesDemon)

##########Simulation##########
ralpha1 = c(1, 1.03625, 0.94456, 1.02627, 1.05796, 1.11404, 1.02149, 1.01978,
     1.21930, 1.199152)
ralpha2 = c(1, 1.19566, 1.17375, 0.95060, 1.05491, 1.18730, 1.40939, 1.45048,
     1.56165, 1.66456)

rbeta1 = c(0.23426, 0.23756, 0.13488, 0.07877, 0.04121,0.01899, 0.00719,
    0.00506, 0.00344, 0.00089)
rbeta2 = c(0.12952, 0.16514, 0.11326, 0.09096, 0.05887, 0.02342, 0.01951,
    0.00517, 0.00323, 0.00015)

rphi1 = 0.00486
rphi2 = 0.00616

ralphatil = 0.006
rphitil = 0.06640

sp=1.32

stau=(sp-2)/(sp-1)

#Generate loss triangles
loss1 = matrix(NA, ncol=10, nrow=10)
loss2 = matrix(NA, ncol=10, nrow=10)

r1 = rtweedie(150,mu=ralphatil,phi=rphitil,xi=sp)
r2 = r1[!r1==0]

random_w = matrix(NA, nrow=10,ncol=10)
loss1 = matrix(NA, nrow=10,ncol=10)
loss2 = matrix(NA, nrow=10, ncol=10)
```

```
for(i in 1:1){
for (j in 1:(10-i+1)){
random_w[i,j] = r2[j]
loss1[i,j] = (ralphatil/((ralpha1[i]*rbeta1[j])))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j], phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil/((ralpha2[i]*rbeta2[j])))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j], phi = rphi2,xi=sp)
}}


for (i in 2:10){
for (j in 1:(10-i+1)){
random_w[i,j] = r2[10*(i-1)-sum(0:(i-2))+j]
loss1[i,j] = (ralphatil/((ralpha1[i]*rbeta1[j])))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil/((ralpha2[i]*rbeta2[j])))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}}

##########Premliminary analysis###########
plot(loss1[1,],type = "l",lwd=1,ylim=c(0,0.3))
for(i in 2:10){
lines(loss1[i,],lwd=1)
}

plot(loss2[1,],type = "l",lwd=1,ylim=c(0,0.3))
for(i in 2:10){
lines(loss2[i,],lwd=1)
}

###########Find p###########
vloss1 = as.vector(t(loss1))
vloss2 = as.vector(t(loss2))

#Set up llh profile for each line
i = rep(1:10,each=10)
j = rep(1:10, 10)

ci.vec = seq(1,4,by=0.01)

llh1 = rep(0,length(ci.vec))
llh2 = llh1

for (t in 1:length(ci.vec)){
out1 <- glm(vloss1~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp1 <- summary(out1)$dispersion
mu1 <- fitted(out1)
den1 <- dtweedie(out1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2 <- glm(vloss2~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp2 <- summary(out2)$dispersion
mu2 <- fitted(out2)
den2 <- dtweedie(out2$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2[t] <- sum(log(den2))
}

#Set up llh profile for both lines combined
i1 = c(rep(1:10,each=10),rep(11:20, each=10))
j1 = c(rep(1:10,10),rep(11:20,10))
cbine=c(vloss1,vloss2)
allh = rep(0, length(ci.vec))

for (t in 1:length(ci.vec)){
outa <- glm(cbine~as.factor(i1) + as.factor(j1), fam=tweedie(var.power=ci.vec
    [t]))
disp <- summary(outa)$dispersion
```

```
mu <- fitted(outa)
den <- dtweedie(outa$y, mu = mu, phi = disp, power = ci.vec[t])
allh[t] <- sum(log(den))
}

#Find p that maximises llh
ci1 = ci.vec[which.max(llh1)]

ci2 = ci.vec[which.max(llh2)]

aci = ci.vec[which.max(allh)]

#Find CI for p
outci1 = rep(0,length(llh1))
for (k in 1:length(llh1)){
if (isTRUE(2*abs(llh1[k]-max(llh1)) <= 3.84))  {
outci1[k] = ci.vec[k]
}
else{
outci1[k] = 0
}
}

ci1low = outci1[11]
ci1hi = outci1[73]

outci2 = rep(0,length(llh2))
for (k in 1:length(llh2)){
if (isTRUE(2*abs(llh2[k]-max(llh2)) <= 3.84))  {
outci2[k] = ci.vec[k]
}
else{
outci2[k] = 0
}
}

ci2low = outci2[5]
ci2hi = outci2[58]

outaci = rep(0,length(allh))
for (k in 1:length(allh)){
if (isTRUE(2*abs(allh[k]-max(allh)) <= 3.84))  {
outaci[k] = ci.vec[k]
}
else{
outaci[k] = 0
}
}

acilow = outaci[11]
acihi = outaci[56]

##Ln L profile plot
plot(ci.vec,llh1, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(ci1, max(llh1), pch=15, cex=1)
points(aci, llh1[34], pch=18, cex=1)
abline(v=ci1low)
abline(v=ci1hi)

plot(ci.vec,llh2, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(ci2, max(llh2), pch=15, cex=1)
points(aci, llh2[34], pch=18, cex=1)
abline(v=ci2low)
abline(v=ci2hi)

plot(ci.vec,allh, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(aci, max(allh), pch=15, cex=1)
abline(v=acilow)
abline(v=acihi)

#Results
```

```
pest = rbind(c(ci1,ci1low,ci1hi), c(ci2,ci2low,ci2hi), c(aci,acilow,acihi))

##########Choose initial values and get information for prior distributions
    selection##########
#Estimate alpha and beta
solvefn <- function(x,data){
F2 <- x[1]*sum(x[10:18]^(2-aci)) - as.numeric(crossprod(data[2,1:9],x
    [10:18]^(1-aci)))
F3 <- x[2]*sum(x[10:17]^(2-aci)) - as.numeric(crossprod(data[3,1:8],x
    [10:17]^(1-aci)))
F4 <- x[3]*sum(x[10:16]^(2-aci)) - as.numeric(crossprod(data[4,1:7],x
    [10:16]^(1-aci)))
F5 <- x[4]*sum(x[10:15]^(2-aci)) - as.numeric(crossprod(data[5,1:6],x
    [10:15]^(1-aci)))
F6 <- x[5]*sum(x[10:14]^(2-aci)) - as.numeric(crossprod(data[6,1:5],x
    [10:14]^(1-aci)))
F7 <- x[6]*sum(x[10:13]^(2-aci)) - as.numeric(crossprod(data[7,1:4],x
    [10:13]^(1-aci)))
F8 <- x[7]*sum(x[10:12]^(2-aci)) - as.numeric(crossprod(data[8,1:3],x
    [10:12]^(1-aci)))
F9 <- x[8]*sum(x[10:11]^(2-aci)) - as.numeric(crossprod(data[9,1:2],x
    [10:11]^(1-aci)))
F10 <- x[9]*sum(x[10:10]^(2-aci)) - as.numeric(crossprod(data[10,1],x
    [10:10]^(1-aci)))

F11 <- x[10]*sum(c(1,x[1:9])^(2-aci)) - as.numeric(crossprod(data[,1],c(1,x
    [1:9])^(1-aci)))
F12 <- x[11]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:9,2],c(1,
    x[1:8])^(1-aci)))
F13 <- x[12]*sum(c(1,x[1:7])^(2-aci)) - as.numeric(crossprod(data[1:8,3],c(1,
    x[1:7])^(1-aci)))
F14 <- x[13]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:7,4],c(1,
    x[1:6])^(1-aci)))
F15 <- x[14]*sum(c(1,x[1:5])^(2-aci)) - as.numeric(crossprod(data[1:6,5],c(1,
    x[1:5])^(1-aci)))
F16 <- x[15]*sum(c(1,x[1:4])^(2-aci)) - as.numeric(crossprod(data[1:5,6],c(1,
    x[1:4])^(1-aci)))
F17 <- x[16]*sum(c(1,x[1:3])^(2-aci)) - as.numeric(crossprod(data[1:4,7],c(1,
    x[1:3])^(1-aci)))
F18 <- x[17]*sum(c(1,x[1:2])^(2-aci)) - as.numeric(crossprod(data[1:3,8],c(1,
    x[1:2])^(1-aci)))
F19 <- x[18]*sum(c(1,x[1:1])^(2-aci)) - as.numeric(crossprod(data[1:2,9],c(1,
    x[1:1])^(1-aci)))
F20 <-x[19]*sum(1^(2-aci)) - as.numeric(crossprod(data[1,10],1^(1-aci)))

c(F2=F2,F3=F3,F4=F4,F5=F5,F6=F6,F7=F7,F8=F8,F9=F9,F10=F10,
F11=F11,F12=F12,F13=F13,F14=F14,F15=F15,F16=F16,F17=F17,F18=F18,F19=F19,F20=
    F20)
}

ss1 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(loss1,na.rm=T)),
    data=loss1)
ss2 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(loss2,na.rm=T)),
    data=loss2)

#Estimate phi1 and phi2
alphaest1 = c(1,ss1$root[1:9])
betaest1 = ss1$root[10:19]
resid1= matrix(NA,nrow=10,ncol=10)

for (i in 1:10){
for(j in 1:(10-i+1)){
resid1[i,j] = (loss1[i,j]-alphaest1[i]*betaest1[j])^2/((alphaest1[i]*betaest1
    [j])^aci)
}
}
phi1est = sum(resid1,na.rm=T)/45

alphaest2 = c(1,ss2$root[1:9])
betaest2 = ss2$root[10:19]
```

```
resid2= matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for(j in 1:(10-i+1)){
resid2[i,j] = (loss2[i,j]-alphaest2[i]*betaest2[j])^2/((alphaest2[i]*betaest2
    [j])^aci)
}
}
phi2est = sum(resid2,na.rm=T)/45

#Estimate alphatil and betatil - by trials
betatilest = 0.009
alphatilest=0.003
minyest=matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for (j in 1:(10-i+1)){
minyest[i,j] <- min((((alphatilest/(alphaest1[i]*betaest1[j]))^(1-aci)*(
    phi1est/betatilest)))*alphatilest,(((alphatilest/(alphaest2[i]*betaest2[j
    ]))^(1-aci)*(phi2est/betatilest)))*alphatilest)
}}

lambdaest = alphatilest^(2-aci)/betatilest

##########Marginal estimatio###########

#Posterior function
model1 <- function(parm,data){
alpha1 <- exp(parm[1:9])
beta1 <-exp(parm[10:19])
alpha2<-exp(parm[20:28])
beta2<-exp(parm[29:38])
phi1 <- exp(parm[39])
phi2<-exp(parm[40])
lambda <- exp(parm[41])

alpha1.prior <- sum(dunif(log(alpha1),min=-0.5,max=1.5,log=T))
beta1.prior <- sum(dunif(log(beta1[1:2]),min=-2.5,max=-0.5,log=T)) +sum(dunif
    (log(beta1[3]),min=-3,max=-1,log=T)) +sum(dunif(log(beta1[4:5]),min=-4,max
    =-2,log=T)) +sum(dunif(log(beta1[6]),min=-5,max=-3,log=T))+sum(dunif(log(
    beta1[7:8]),min=-6,max=-4,log=T))+sum(dunif(log(beta1[9]),min=-6.5,max
    =-4.5,log=T)) +sum(dunif(log(beta1[10]),min=-7.5,max=-5,log=T))

alpha2.prior <- sum(dunif(log(alpha2),min=-0.5,max=1.5,log=T))
beta2.prior <- sum(dunif(log(beta2[1:3]),min=-3,max=-1,log=T)) +sum(dunif(log
    (beta2[4:5]),min=-4,max=-2,log=T)) +sum(dunif(log(beta2[6]),min=-5,max=-3,
    log=T))+sum(dunif(log(beta2[7]),min=-5.5,max=-3.5,log=T)) +sum(dunif(log(
    beta2[8]),min=-6,max=-4,log=T))+sum(dunif(log(beta2[9]),min=-6.5,max=-4.5,
    log=T))+sum(dunif(log(beta2[10]),min=-9.5,max=-7.5,log=T))

phi1.prior <- dunif(log(phi1),min=-7,max=-3,log=T)
phi2.prior <- dunif(log(phi2),min=-7,max=-3,log=T)

lambda.prior <- dunif(log(lambda),min=-3,max=1,log=T)

alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

l <- matrix(NA, nrow=10, ncol=10)
loss1<- data$loss1
loss2 <-data$loss2

for (i in 1:10){
for (j in 1:(10-i+1)){
A1=alphaf1[i]*beta1[j]*((phi1/((alphaf1[i]*beta1[j])^((2-aci))))*lambda+1)
B1 = phi1*((phi1/((alphaf1[i]*beta1[j])^((2-aci))))*lambda+1)^{(1-aci)}

A2=alphaf2[i]*beta2[j]*((phi2/((alphaf2[i]*beta2[j])^((2-aci))))*lambda+1)
B2 = phi2*((phi2/((alphaf2[i]*beta2[j])^((2-aci))))*lambda+1)^{(1-aci)}

l[i,j] = log(dtweedie(loss1[i,j],xi=aci,mu=A1,phi=B1))+log(dtweedie(loss2[i,j
    ],xi=aci,mu=A2,phi=B2))
}}
```

```
LL <- sum(l, na.rm=T)
LP <- LL + alpha1.prior+beta1.prior+alpha2.prior+beta2.prior+phi1.prior+phi2.
    prior+lambda.prior

list(LP = LP, Monitor = c(LP, alpha1, beta1,alpha2,beta2,phi1,phi2,lambda),
    parm=parm)
}

#Prepare for MCMC
data=list(loss1=loss1,loss2=loss2)
init <-  log(c(alphaest1[2:10],betaest1,alphaest2[2:10],betaest2,phi1est,
    phi2est,lambdaest))
lower = c(rep(-0.5,9),rep(-2.5,2),-3,rep(-4,2),-5,rep(-6,2),-6.5,-7.5,rep
    (-0.5,9),rep(-3,3),rep(-4,2),rep(-5,1),-5.5,-6,-6.5,-9.5,-7,-7,-3)
upper = c(rep(1.5,9),rep(-0.5,2),-1,rep(-2,2),-3, rep(-4,2),-4.5,-5, rep
    (1.5,9),rep(-1,3),rep(-2,2),rep(-3,1),-3.5,-4,-4.5,-7.5,-3,-3,1)
std = c(0.0187, 0.0210, 0.0234, 0.0210, 0.0217, 0.0218, 0.0241, 0.0261,
    0.0308, 0.0239, 0.0218, 0.0243, 0.0217, 0.0285, 0.0330, 0.0458, 0.0477,
    0.0643, 0.0873, 0.0250, 0.0277, 0.0276, 0.0280, 0.0281, 0.0281, 0.0336,
    0.0350, 0.0315, 0.0230, 0.0238, 0.0236, 0.0254, 0.0352, 0.0352, 0.0389,
    0.0655, 0.0624, 0.0919, 0.0533, 0.0535, 0.2165)

Iterations=150000
Status=1000
Thinning=5
LogFile = ""
cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

Acceptance <- 0
Mo0 <- model1(init, data)
Mon <- matrix(NA, Iterations, length(Mo0[["Monitor"]]), byrow=TRUE)
dimension <- length(init)
thinned <- matrix(NA, floor(Iterations/Thinning)+1,length(init))
thinned[1,] <- init

#Run MCMC
set.seed(11)
for (iter in 1:Iterations) {

if(iter %% Status == 0) cat("Iteration: ", iter, sep="")

#Metropolis algorithm
prop <- rtruncnorm(1,mean = Mo0[["parm"]],sd = std, a = lower, b=upper)
Mo1 <- try(model1(prop, data), silent=TRUE)

log.u <- log(runif(1))
log.alpha <- Mo1[["LP"]] - Mo0[["LP"]] + log(dtruncnorm(Mo0[["parm"]], mean =
     prop, sd = std, a = lower, b =  upper)) - log(dtruncnorm(prop, mean=Mo0[[
    "parm"]], sd = std, a = lower, b =  upper))

if((is.finite(log.alpha)) && (!inherits(Mo1, "try-error")) &&  ((is.finite(
    Mo1[["Monitor"]]))) && (log.u < log.alpha)) {
Mo0 <- Mo1
Acceptance <- Acceptance + 1}

Mon[iter,] <- Mo0[["Monitor"]]

#Thin samples
if(iter %% Thinning == 0) {
t.iter <- floor(iter / Thinning) + 1
thinned[t.iter,] <- Mo0[["parm"]]}

#Show tracking
if(iter %% Status == 0){
cat(", LP:", round(Mo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(Acceptance/iter, 2), "\n", sep = "", file =
    LogFile, append = TRUE)}
}
```

```
cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#Plot sample paths
thinnedplot = matrix(NA,nrow=15000,ncol=41)
freq = 30

for(n in 1:150000){
if(n%%freq == 0){
tp.iter <- floor(n / freq)
thinnedplot[tp.iter,] <- Mon[n,2:42]}
}

xticks <- seq(0, 150000, 30000)
xuse <-seq(0,5000, 1000)
layout(matrix(c(1,1,2,2,3,3,4,4,5,5,6,6,0,7,7,0), 4, 4, byrow=TRUE))
par(mar=c(2.5,2.5,2.5,2.5))

plot((thinnedplot[1:5000,1]),type="l",main=expression(paste(alpha[2]^(1))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,20]),type="l",main=expression(paste(alpha[2]^(2))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,10]),type="l",main=expression(paste(beta[1]^(1))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,29]),type="l",main=expression(paste(beta[1]^(2))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,39]),type="l",main=expression(paste(phi^(1))),ylab="
    ",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,40]),type="l",main=expression(paste(phi^(2))),ylab="
    ",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,41]),type="l",main=expression(paste(Lambda)),ylab=""
    ,xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)

#Summary statistics for marginal estimation
true = c(ralpha1[2:10],rbeta1,ralpha2[2:10],rbeta2,rphi1,rphi2,ralphatil^(2-
    aci)/rphitil)
used = exp(thinned[10000:30000,])
mean = apply(used,2,median,na.rm=T)
st.dev = apply(used,2,sd,na.rm=T)
lCI = rep(0,41)
uCI = rep(0,41)
for (i in 1:41){
lCI[i] = quantile(used[,i],p=0.05)
uCI[i] = quantile(used[,i],p=0.95)
}
bothline = cbind(true,mean,st.dev,lCI,uCI)

##########Multivariate estimation (adaptive Metropolis)###########

#Use parameter estimates from marginal estimation
alpha1 <- mean[1:9]
beta1 <- mean[10:19]
alpha2<- mean[20:28]
beta2<- mean[29:38]
phi1 <- mean[39]
phi2<- mean[40]
lambda <- mean[41]
alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

#Posterior function
model2 <- function(parm,data){
alpha <- exp(parm)
```

```
beta <- alpha^(2-aci)/lambda

alpha.prior <- dunif(log(alpha),min=-7,max=-4,log=T)

l <- matrix(NA, nrow=10, ncol=10)
loss1<- data$loss1
loss2 <- data$loss2
miny <- matrix(NA,nrow=10,ncol=10)
prior = c(alpha.prior)

if(all(is.finite(prior))){
for (i in 1:10){
for (j in 1:(10-i+1)){
miny[i,j] <- min((loss1[i,j]/((alpha/(alphaf1[i]*beta1[j]))^(1-aci)*(phi1/
    beta))), (loss2[i,j]/((alpha/(alphaf2[i]*beta2[j]))^(1-aci)*(phi2/beta))))

if (miny[i,j]<=1e-200000000){
l[i,j]=log(dtweedie(loss1[i,j],xi=aci, mu=alphaf1[i]*beta1[j], phi = phi1))+
    log(dtweedie(loss2[i,j],xi=aci, mu=alphaf2[i]*beta2[j], phi = phi2))
}
else {
f<- function(z){
dtweedie(loss1[i,j]-(alpha/(alphaf1[i]*beta1[j]))^(1-aci)*(phi1/beta)*z,xi=
    aci, mu=alphaf1[i]*beta1[j], phi = phi1)*dtweedie(loss2[i,j]-(alpha/(
    alphaf2[i]*beta2[j]))^(1-aci)*(phi2/beta)*z,xi=aci, mu=alphaf2[i]*beta2[j
    ], phi = phi2)*dtweedie(z,xi=aci, mu=alpha, phi = beta)
}
llh<- try(integrate(f,lower=1e-2000000000, upper=miny[i,j]),silent=T)
if(inherits(llh, 'try-error') ){
l[i,j] <- log(0)
}
else{
l[i,j] <- log(llh$value)}}
}}}

LL <- sum(l, na.rm=T)
LP <- LL + alpha.prior

list(LP = LP, Monitor = c(LP, alpha,beta), parm=parm)
}

#Prepare for MCMC
ainit <-  -5.116

Iterations=30000
Status=100
Thinning=5
alpha.star=0.2
Periodicity=1
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

aAcceptance <- 0
aMo0 <- model2(ainit, data)
aMon <- matrix(NA,nrow=Iterations,ncol=length(aMo0[["Monitor"]]))
adimension <- length(ainit)
athinned <- matrix(NA, floor(Iterations/Thinning)+1,length(ainit)+1)
athinned[1,] <- ainit
aScaleF <- 0.0001/sqrt(adimension)
aVarCov <- matrix(0, adimension, adimension)
diag(aVarCov) <- rep(aScaleF, adimension)
aS <- t(chol(aVarCov))

#Run MCMC
set.seed(11)

for (aiter in 1:Iterations) {
if(aiter %% Status == 0) cat("Iteration: ", aiter, sep="")
```

```
#Adaptive Metropolis
aU <- rnorm(adimension)
aprop <- as.vector(aMo0[["parm"]] + aS %*% aU)
aMo1 <- try(model2(aprop, data), silent=TRUE)

alog.u <- log(runif(1))
alog.alpha <- aMo1[["LP"]] - aMo0[["LP"]]
if((is.finite(alog.alpha)) && (!inherits(aMo1, "try-error")) &&  ((is.finite(
    aMo1[["Monitor"]]))) && (alog.u < alog.alpha)) {
aMo0 <- aMo1
aAcceptance <- aAcceptance + 1}

aMon[aiter,] <- aMo0[["Monitor"]]

if({aiter >= 2} & {aiter %% Periodicity == 0}) {
aeta <- min(1, adimension*aiter^(-2/3))
aVarCov.test <- aS %*% (diag(adimension) + aeta*(min(1, exp(alog.alpha)) -
    alpha.star) * aU %*% t(aU) / sqrt(sum(aU^2))) %*% t(aS)
if(!all(is.finite(aVarCov.test))) {aVarCov.test <- aVarCov}
if(!is.symmetric.matrix(aVarCov.test)){aVarCov.test <- as.symmetric.matrix(
    aVarCov.test)}
if(is.positive.definite(aVarCov.test)){ aS.z <- try(t(chol(aVarCov)), silent=
    TRUE)
if(!inherits(aS.z, "try-error")) {
aVarCov <- aVarCov.test
aS <- aS.z}}}

#Thin samples
if(aiter %% Thinning == 0) {
at.iter <- floor(aiter / Thinning) + 1
athinned[at.iter,] <- aMo0[["Monitor"]][2:3]}

if(aiter %% Status == 0){
cat(", LP:", round(aMo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(aAcceptance/aiter, 2), "\n", sep = "", file =
     LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#MCMC trace plots
par(mfrow=c(2,1))
axticks <- seq(0, 30000, 10000)
axuse <-seq(0,30000,10000)

par(mar=c(2.5,2.5,2.5,2.5))
plot((aMon[1:30000,2]),type="l",main=expression(paste("Sample path for ",
    alpha)),ylab="",xlab="", xaxt = "n",ylim=c(0,0.02))
axis(side=1,at=axuse, labels=axticks)
plot((aMon[1:30000,3]),type="l",main=expression(paste("Sample path for ",
    beta)),ylab="",xlab="", xaxt = "n",ylim=c(0.015,0.15))
axis(side=1,at=axuse, labels=axticks)

#Summary statistics for multivariate estimation
aused = cbind(athinned[2000:6000,1:2])
atrue = c(ralphatil,rphitil)
amean = apply(aused,2,median,na.rm=T)
astd = apply(aused,2,sd,na.rm=T)
alCI = rep(0,2)
auCI = rep(0,2)
for (i in 1:2){
alCI[i] = quantile(aused[,i],p=0.05)
auCI[i] = quantile(aused[,i],p=0.95)
}
abothline = cbind(atrue,amean,astd,alCI,auCI)
```

A.1.2 Simulation illustration 2

```
library(tweedie)
library(MASS)
library(statmod)
library(truncnorm)
library(rootSolve)
library(LaplacesDemon)

##########Simulation##########
ralpha1 = c(1, 1.0362, 0.9445, 1.0262, 1.0579, 1.1140, 1.0214, 1.0197,
    1.2193, 1.1991)
ralpha2 = c(1, 1.1956, 1.1737, 0.9506, 1.0549, 1.1873, 1.4093, 1.4504,
    1.5616, 1.6645)

rbeta1 = c(0.2342, 0.2375, 0.1348, 0.0787, 0.0412,0.0189, 0.0071, 0.0050,
    0.0034, 0.0009)
rbeta2 = c(0.1295, 0.1651, 0.1132, 0.0909, 0.0588,0.0234, 0.0195, 0.0051,
    0.0032, 0.0002)

rphi1 = 0.020
rphi2 = 0.012

ralphatil = 0.0360
rphitil = 0.1

sp=1.32

stau=(sp-2)/(sp-1)

#Generate loss triangles
loss1 = matrix(NA, ncol=10, nrow=10)
loss2 = matrix(NA, ncol=10, nrow=10)

set.seed(8)
r1 = rtweedie(55,mu=ralphatil,phi=rphitil,xi=sp)

random_w = matrix(NA, nrow=10,ncol=10)
loss1 = matrix(NA, nrow=10,ncol=10)
loss2 = matrix(NA, nrow=10, ncol=10)

for(i in 1:1){
for (j in 1:(10-i+1)){
random_w[i,j] = r1[j]
loss1[i,j] = (ralphatil/((ralpha1[i]*rbeta1[j])))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j], phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil/((ralpha2[i]*rbeta2[j])))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j], phi = rphi2,xi=sp)
}}


for (i in 2:10){
for (j in 1:(10-i+1)){
random_w[i,j] = r1[10*(i-1)-sum(0:(i-2))+j]
loss1[i,j] = (ralphatil/((ralpha1[i]*rbeta1[j])))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil/((ralpha2[i]*rbeta2[j])))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}}

##########Premliminary analysis###########
plot(loss1[1,],type = "l",lwd=1,ylim=c(0,0.3))
for(i in 2:10){
lines(loss1[i,],lwd=1)
}

plot(loss2[1,],type = "l",lwd=1,ylim=c(0,0.3))
for(i in 2:10){
lines(loss2[i,],lwd=1)
}

###########Find p###########
vloss1 = as.vector(t(loss1))
vloss2 = as.vector(t(loss2))
```

```
#Set up llh profile for each line
i = rep(1:10,each=10)
j = rep(1:10, 10)

ci.vec = seq(1,2,by=0.01)

llh1 = rep(0,length(ci.vec))
llh2 = llh1

for (t in 1:length(ci.vec)){
out1 <- glm(vloss1~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp1 <- summary(out1)$dispersion
mu1 <- fitted(out1)
den1 <- dtweedie(out1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2 <- glm(vloss2~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp2 <- summary(out2)$dispersion
mu2 <- fitted(out2)
den2 <- dtweedie(out2$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2[t] <- sum(log(den2))
}

#Set up llh profile for both lines combined
i1 = c(rep(1:10,each=10),rep(11:20, each=10))
j1 = c(rep(1:10,10),rep(11:20,10))
cbine=c(vloss1,vloss2)
allh = rep(0, length(ci.vec))

for (t in 1:length(ci.vec)){
outa <- glm(cbine~as.factor(i1) + as.factor(j1), fam=tweedie(var.power=ci.vec
    [t]))
disp <- summary(outa)$dispersion
mu <- fitted(outa)
den <- dtweedie(outa$y, mu = mu, phi = disp, power = ci.vec[t])
allh[t] <- sum(log(den))
}

#Find p that maximises llh
ci1 = ci.vec[which.max(llh1)]

ci2 = ci.vec[which.max(llh2)]

aci = ci.vec[which.max(allh)]

#Find CI for p
outci1 = rep(0,length(llh1))
for (k in 1:length(llh1)){
if (isTRUE(2*abs(llh1[k]-max(llh1)) <= 3.84))  {
outci1[k] = ci.vec[k]
}
else{
outci1[k] = 0
}
}

ci1low = outci1[14]
ci1hi = outci1[39]

outci2 = rep(0,length(llh2))
for (k in 1:length(llh2)){
if (isTRUE(2*abs(llh2[k]-max(llh2)) <= 3.84))  {
outci2[k] = ci.vec[k]
}
else{
outci2[k] = 0
```

```
}
}

ci2low = outci2[24]
ci2hi = outci2[87]

outaci = rep(0,length(allh))
for (k in 1:length(allh)){
if (isTRUE(2*abs(allh[k]-max(allh)) <= 3.84))  {
outaci[k] = ci.vec[k]
}
else{
outaci[k] = 0
}
}

acilow = outaci[15]
acihi = outaci[38]

##Ln L profile plot
plot(ci.vec,llh1, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(ci1, max(llh1), pch=15, cex=1)
points(aci, llh1[34], pch=18, cex=1)
abline(v=ci1low)
abline(v=ci1hi)

plot(ci.vec,llh2, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(ci2, max(llh2), pch=15, cex=1)
points(aci, llh2[34], pch=18, cex=1)
abline(v=ci2low)
abline(v=ci2hi)

plot(ci.vec,allh, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(aci, max(allh), pch=15, cex=1)
abline(v=acilow)
abline(v=acihi)

#Results
pest = rbind(c(ci1,ci1low,ci1hi), c(ci2,ci2low,ci2hi), c(aci,acilow,acihi))

##########Choose initial values and get information for prior distributions
    selection##########
#Estimate alpha and beta
solvefn <- function(x,data){
F2 <- x[1]*sum(x[10:18]^(2-aci)) - as.numeric(crossprod(data[2,1:9],x
    [10:18]^(1-aci)))
F3 <- x[2]*sum(x[10:17]^(2-aci)) - as.numeric(crossprod(data[3,1:8],x
    [10:17]^(1-aci)))
F4 <- x[3]*sum(x[10:16]^(2-aci)) - as.numeric(crossprod(data[4,1:7],x
    [10:16]^(1-aci)))
F5 <- x[4]*sum(x[10:15]^(2-aci)) - as.numeric(crossprod(data[5,1:6],x
    [10:15]^(1-aci)))
F6 <- x[5]*sum(x[10:14]^(2-aci)) - as.numeric(crossprod(data[6,1:5],x
    [10:14]^(1-aci)))
F7 <- x[6]*sum(x[10:13]^(2-aci)) - as.numeric(crossprod(data[7,1:4],x
    [10:13]^(1-aci)))
F8 <- x[7]*sum(x[10:12]^(2-aci)) - as.numeric(crossprod(data[8,1:3],x
    [10:12]^(1-aci)))
F9 <- x[8]*sum(x[10:11]^(2-aci)) - as.numeric(crossprod(data[9,1:2],x
    [10:11]^(1-aci)))
F10 <- x[9]*sum(x[10:10]^(2-aci)) - as.numeric(crossprod(data[10,1],x
    [10:10]^(1-aci)))

F11 <- x[10]*sum(c(1,x[1:9])^(2-aci)) - as.numeric(crossprod(data[,1],c(1,x
    [1:9])^(1-aci)))
F12 <- x[11]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:9,2],c(1,
    x[1:8])^(1-aci)))
F13 <- x[12]*sum(c(1,x[1:7])^(2-aci)) - as.numeric(crossprod(data[1:8,3],c(1,
    x[1:7])^(1-aci)))
F14 <- x[13]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:7,4],c(1,
    x[1:6])^(1-aci)))
```

```
F15 <- x[14]*sum(c(1,x[1:5])^(2-aci)) - as.numeric(crossprod(data[1:6,5],c(1,
    x[1:5])^(1-aci)))
F16 <- x[15]*sum(c(1,x[1:4])^(2-aci)) - as.numeric(crossprod(data[1:5,6],c(1,
    x[1:4])^(1-aci)))
F17 <- x[16]*sum(c(1,x[1:3])^(2-aci)) - as.numeric(crossprod(data[1:4,7],c(1,
    x[1:3])^(1-aci)))
F18 <- x[17]*sum(c(1,x[1:2])^(2-aci)) - as.numeric(crossprod(data[1:3,8],c(1,
    x[1:2])^(1-aci)))
F19 <- x[18]*sum(c(1,x[1:1])^(2-aci)) - as.numeric(crossprod(data[1:2,9],c(1,
    x[1:1])^(1-aci)))
F20 <-x[19]*sum(1^(2-aci)) - as.numeric(crossprod(data[1,10],1^(1-aci)))

c(F2=F2,F3=F3,F4=F4,F5=F5,F6=F6,F7=F7,F8=F8,F9=F9,F10=F10,
F11=F11,F12=F12,F13=F13,F14=F14,F15=F15,F16=F16,F17=F17,F18=F18,F19=F19,F20=
    F20)
}

ss1 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(loss1,na.rm=T)),
    data=loss1)
ss2 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(loss2,na.rm=T)),
    data=loss2)

#Estimate phi1 and phi2
alphaest1 = c(1,ss1$root[1:9])
betaest1 = ss1$root[10:19]
resid1= matrix(NA,nrow=10,ncol=10)

for (i in 1:10){
for(j in 1:(10-i+1)){
resid1[i,j] = (loss1[i,j]-alphaest1[i]*betaest1[j])^2/((alphaest1[i]*betaest1
    [j])^aci)
}
}
phi1est = sum(resid1,na.rm=T)/45

alphaest2 = c(1,ss2$root[1:9])
betaest2 = ss2$root[10:19]
resid2= matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for(j in 1:(10-i+1)){
resid2[i,j] = (loss2[i,j]-alphaest2[i]*betaest2[j])^2/((alphaest2[i]*betaest2
    [j])^aci)
}
}
phi2est = sum(resid2,na.rm=T)/45

#Estimate alpha and beta
betatilest = 0.009
alphatilest=0.003
minyest=matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for (j in 1:(10-i+1)){
minyest[i,j] <- min((((alphatilest/(alphaest1[i]*betaest1[j]))^(1-aci)*(
    phi1est/betatilest)))*alphatilest,(((alphatilest/(alphaest2[i]*betaest2[j
    ]))^(1-aci)*(phi2est/betatilest)))*alphatilest)
}}

lambdaest = alphatilest^(2-aci)/betatilest

############Marginal estimatio############

#Posterior function
model1 <- function(parm,data){
alpha1 <- exp(parm[1:9])
beta1 <-exp(parm[10:19])
alpha2<-exp(parm[20:28])
beta2<-exp(parm[29:38])
phi1 <- exp(parm[39])
phi2<-exp(parm[40])
lambda <- exp(parm[41])
```

```
alpha1.prior <- sum(dunif(log(alpha1),min=-0.5,max=1.5,log=T))
beta1.prior <- sum(dunif(log(beta1[1:2]),min=-2.5,max=-0.5,log=T)) +sum(dunif
    (log(beta1[3]),min=-3,max=-1,log=T)) +sum(dunif(log(beta1[4:5]),min=-4,max
    =-2,log=T)) +sum(dunif(log(beta1[6]),min=-5,max=-3,log=T))+sum(dunif(log(
    beta1[7:8]),min=-6,max=-4,log=T))+sum(dunif(log(beta1[9]),min=-6.2,max
    =-4.2,log=T)) +sum(dunif(log(beta1[10]),min=-8,max=-4,log=T))

alpha2.prior <- sum(dunif(log(alpha2),min=-0.4,max=0.7,log=T))
beta2.prior <- sum(dunif(log(beta2[1]),min=-3.5,max=-1.9,log=T)) + sum(dunif(
    log(beta2[2:3]), min=-2.7, max=-0.8))+sum(dunif(log(beta2[4:5]),min=-3.2,
    max=-2.2,log=T)) +sum(dunif(log(beta2[6]),min=-4.2,max=-2.5,log=T))+sum(
    dunif(log(beta2[7]),min=-4.5,max=-2.5,log=T)) +sum(dunif(log(beta2[8]),min
    =-6,max=-4,log=T))+sum(dunif(log(beta2[9]),min=-6.5,max=-4.5,log=T))+sum(
    dunif(log(beta2[10]),min=-9.5,max=-5.5,log=T))

phi1.prior <- dunif(log(phi1),min=-4,max=0,log=T)
phi2.prior <- dunif(log(phi2),min=-4.6,max=-0.5,log=T)

lambda.prior <- dunif(log(lambda),min=-5,max=3,log=T)

alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

l <- matrix(NA, nrow=10, ncol=10)
loss1<- data$loss1
loss2 <-data$loss2

for (i in 1:10){
for (j in 1:(10-i+1)){
A1=alphaf1[i]*beta1[j]*((phi1/((alphaf1[i]*beta1[j])^((2-aci))))*lambda+1)
B1 = phi1*((phi1/((alphaf1[i]*beta1[j])^((2-aci))))*lambda+1)^{(1-aci)}

A2=alphaf2[i]*beta2[j]*((phi2/((alphaf2[i]*beta2[j])^((2-aci))))*lambda+1)
B2 = phi2*((phi2/((alphaf2[i]*beta2[j])^((2-aci))))*lambda+1)^{(1-aci)}

l[i,j] = log(dtweedie(loss1[i,j],xi=aci,mu=A1,phi=B1))+log(dtweedie(loss2[i,j
    ],xi=aci,mu=A2,phi=B2))
}}

LL <- sum(l, na.rm=T)
LP <- LL + alpha1.prior+beta1.prior+alpha2.prior+beta2.prior+phi1.prior+phi2.
    prior+lambda.prior

list(LP = LP, Monitor = c(LP, alpha1, beta1,alpha2,beta2,phi1,phi2,lambda),
    parm=parm)
}

#Preparing for MCMC
data=list(loss1=loss1,loss2=loss2)
init <-  c(0.036, -0.057,  0.026,  0.056,  0.108,  0.021,  0.020,  0.198,
    0.182, -1.452, -1.438, -2.004, -2.542, -3.189, -3.969, -4.948, -5.298,
    -5.684, -7.013,  0.179,  0.160, -0.051,  0.053,  0.172,  0.343,  0.372,
    0.446,  0.510, -2.044, -1.801, -2.179, -2.398, -2.834, -3.755, -3.937,
    -5.279, -5.745, -8.517, -3.912, -4.423, -0.357)
lower = c(rep(-0.5,9),rep(-2.5,2),-3,rep(-4,2),-5,rep(-6,2),-6.2,-8,rep
    (-0.4,9),-3.5,rep(-2.7,2),rep(-3.2,2),-4.2,-4.5,-6,-6.5,-9.5,-4,-4.6,-5)
upper = c(rep(1.5,9),rep(-0.5,2),-1,rep(-2,2),-3, rep(-4,2),-4.2,-4, rep
    (0.7,9),-1.9,rep(-0.8,2),rep(-2.2,2),-2.5,-2.5,-4,-4.5,-5.5,0,-0.5,3)
std = c( 0.04057, 0.0400, 0.0434, 0.045, 0.0457, 0.043, 0.0431, 0.0481,
    0.0508, 0.0359, 0.0358,0.0353, 0.0447, 0.0485, 0.065, 0.075, 0.0857,
    0.0903, 0.15, 0.0340, 0.0367, 0.0366,0.037, 0.0371, 0.0341, 0.0416,
    0.0420, 0.0355, 0.0300, 0.0318, 0.0316, 0.0314, 0.0442,0.0442, 0.0409,
    0.0655, 0.070, 0.75, 0.0553, 0.0555, 0.3)-0.007

Iterations=150000
Status=1000
Thinning=5
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()
```

```
Acceptance <- 0
Mo0 <- model1(init, data)
Mon <- matrix(NA, Iterations, length(Mo0[["Monitor"]]), byrow=TRUE)
dimension <- length(init)
thinned <- matrix(NA, floor(Iterations/Thinning)+1,length(init))
thinned[1,] <- init

#Run MCMC
set.seed(11)
for (iter in 1:Iterations) {

if(iter %% Status == 0) cat("Iteration: ", iter, sep="")

#Metropolis algorithm
prop <- rtruncnorm(1,mean = Mo0[["parm"]],sd = std, a = lower, b=upper)
Mo1 <- try(model1(prop, data), silent=TRUE)

log.u <- log(runif(1))
log.alpha <- Mo1[["LP"]] - Mo0[["LP"]] + log(dtruncnorm(Mo0[["parm"]], mean =
     prop, sd = std, a = lower, b =  upper)) - log(dtruncnorm(prop, mean=Mo0[[
    "parm"]], sd = std, a = lower, b =  upper))

if((is.finite(log.alpha)) && (!inherits(Mo1, "try-error")) &&  ((is.finite(
    Mo1[["Monitor"]]))) && (log.u < log.alpha)) {
Mo0 <- Mo1
Acceptance <- Acceptance + 1}

Mon[iter,] <- Mo0[["Monitor"]]

#Thin samples
if(iter %% Thinning == 0) {
t.iter <- floor(iter / Thinning) + 1
thinned[t.iter,] <- Mo0[["parm"]]}

#Show tracking
if(iter %% Status == 0){
cat(", LP:", round(Mo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(Acceptance/iter, 2), "\n", sep = "", file =
    LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#Plot sample paths
thinnedplot = matrix(NA,nrow=15000,ncol=41)
freq = 30

for(n in 1:150000){
if(n%%freq == 0){
tp.iter <- floor(n / freq)
thinnedplot[tp.iter,] <- Mon[n,2:42]}
}

xticks <- seq(0, 150000, 30000)
xuse <-seq(0,5000, 1000)
layout(matrix(c(1,1,2,2,3,3,4,4,5,5,6,6,0,7,7,0), 4, 4, byrow=TRUE))
par(mar=c(2.5,2.5,2.5,2.5))

plot((thinnedplot[1:5000,1]),type="l",main=expression(paste(alpha[2]^(1))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,20]),type="l",main=expression(paste(alpha[2]^(2))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,10]),type="l",main=expression(paste(beta[1]^(1))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,29]),type="l",main=expression(paste(beta[1]^(2))),
    ylab="",xlab="Iteration", xaxt = "n")
```

```
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,39]),type="l",main=expression(paste(phi^(1))),ylab="
    ",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,40]),type="l",main=expression(paste(phi^(2))),ylab="
    ",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,41]),type="l",main=expression(paste(Lambda)),ylab=""
    ,xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)

#Summary statistics for marginal estimation
true = c(ralpha1[2:10],rbeta1,ralpha2[2:10],rbeta2,rphi1,rphi2,ralphatil^(2-
    aci)/rphitil)
used = exp(thinned[10000:30000,])
mean = apply(used,2,median,na.rm=T)
st.dev = apply(used,2,sd,na.rm=T)
lCI = rep(0,41)
uCI = rep(0,41)
for (i in 1:41){
lCI[i] = quantile(used[,i],p=0.05)
uCI[i] = quantile(used[,i],p=0.95)
}
bothline = cbind(true,mean,st.dev,lCI,uCI)

##########Multivariate estimation (adaptive Metropolis)###########

#Use parameter estimates from marginal estimation
alpha1 <- mean[1:9]
beta1 <- mean[10:19]
alpha2<- mean[20:28]
beta2<- mean[29:38]
phi1 <- mean[39]
phi2<- mean[40]
lambda <- mean[41]
alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

#Posterior function
model2 <- function(parm,data){
alpha <- exp(parm)
beta <- alpha^(2-aci)/lambda

alpha.prior <- dunif(log(alpha),min=-7,max=-4,log=T)

l <- matrix(NA, nrow=10, ncol=10)
loss1<- data$loss1
loss2 <- data$loss2
miny <- matrix(NA,nrow=10,ncol=10)
prior = c(alpha.prior)

if(all(is.finite(prior))){
for (i in 1:10){
for (j in 1:(10-i+1)){
miny[i,j] <- min((loss1[i,j]/((alpha/(alphaf1[i]*beta1[j]))^(1-aci)*(phi1/
    beta))), (loss2[i,j]/((alpha/(alphaf2[i]*beta2[j]))^(1-aci)*(phi2/beta))))

if (miny[i,j]<=1e-200000000){
l[i,j]=log(dtweedie(loss1[i,j],xi=aci, mu=alphaf1[i]*beta1[j], phi = phi1))+
    log(dtweedie(loss2[i,j],xi=aci, mu=alphaf2[i]*beta2[j], phi = phi2))
}
else {
f<- function(z){
dtweedie(loss1[i,j]-(alpha/(alphaf1[i]*beta1[j]))^(1-aci)*(phi1/beta)*z,xi=
    aci, mu=alphaf1[i]*beta1[j], phi = phi1)*dtweedie(loss2[i,j]-(alpha/(
    alphaf2[i]*beta2[j]))^(1-aci)*(phi2/beta)*z,xi=aci, mu=alphaf2[i]*beta2[j
    ], phi = phi2)*dtweedie(z,xi=aci, mu=alpha, phi = beta)
}
llh<- try(integrate(f,lower=1e-2000000000, upper=miny[i,j]),silent=T)
if(inherits(llh, 'try-error') ){
```

```
l[i,j] <- log(0)
}
else{
l[i,j] <- log(llh$value)}}
}}}

LL <- sum(l, na.rm=T)
LP <- LL + alpha.prior

list(LP = LP, Monitor = c(LP, alpha,beta), parm=parm)
}

#Prepare for MCMC
ainit <-  -3.324

Iterations=30000
Status=100
Thinning=5
alpha.star=0.2
Periodicity=1
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

aAcceptance <- 0
aMo0 <- model2(ainit, data)
aMon <- matrix(NA,nrow=Iterations,ncol=length(aMo0[["Monitor"]]))
adimension <- length(ainit)
athinned <- matrix(NA, floor(Iterations/Thinning)+1,length(ainit)+1)
athinned[1,] <- ainit
aScaleF <- 0.0001/sqrt(adimension)
aVarCov <- matrix(0, adimension, adimension)
diag(aVarCov) <- rep(aScaleF, adimension)
aS <- t(chol(aVarCov))

#Run MCMC
set.seed(11)

for (aiter in 1:Iterations) {
if(aiter %% Status == 0) cat("Iteration: ", aiter, sep="")

#Adaptive Metropolis
aU <- rnorm(adimension)
aprop <- as.vector(aMo0[["parm"]] + aS %*% aU)
aMo1 <- try(model2(aprop, data), silent=TRUE)

alog.u <- log(runif(1))
alog.alpha <- aMo1[["LP"]] - aMo0[["LP"]]
if((is.finite(alog.alpha)) && (!inherits(aMo1, "try-error")) &&  ((is.finite(
    aMo1[["Monitor"]]))) && (alog.u < alog.alpha)) {
aMo0 <- aMo1
aAcceptance <- aAcceptance + 1}

aMon[aiter,] <- aMo0[["Monitor"]]

if({aiter >= 2} & {aiter %% Periodicity == 0}) {
aeta <- min(1, adimension*aiter^(-2/3))
aVarCov.test <- aS %*% (diag(adimension) + aeta*(min(1, exp(alog.alpha)) -
    alpha.star) * aU %*% t(aU) / sqrt(sum(aU^2))) %*% t(aS)
if(!all(is.finite(aVarCov.test))) {aVarCov.test <- aVarCov}
if(!is.symmetric.matrix(aVarCov.test)){aVarCov.test <- as.symmetric.matrix(
    aVarCov.test)}
if(is.positive.definite(aVarCov.test)){ aS.z <- try(t(chol(aVarCov)), silent=
    TRUE)
if(!inherits(aS.z, "try-error")) {
aVarCov <- aVarCov.test
aS <- aS.z}}}

#Thin samples
if(aiter %% Thinning == 0) {
```

```
at.iter <- floor(aiter / Thinning) + 1
athinned[at.iter,] <- aMo0[["Monitor"]][2:3]}

if(aiter %% Status == 0){
cat(", LP:", round(aMo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(aAcceptance/aiter, 2), "\n", sep = "", file =
     LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#MCMC trace plots
par(mfrow=c(2,1))
axticks <- seq(0, 30000, 10000)
axuse <-seq(0,30000,10000)

par(mar=c(2.5,2.5,2.5,2.5))
plot((aMon[1:30000,2]),type="l",main=expression(paste("Sample path for ",
    alpha)),ylab="",xlab="", xaxt = "n",ylim=c(0,0.02))
axis(side=1,at=axuse, labels=axticks)
plot((aMon[1:30000,3]),type="l",main=expression(paste("Sample path for ",
    beta)),ylab="",xlab="", xaxt = "n",ylim=c(0.015,0.15))
axis(side=1,at=axuse, labels=axticks)

#Summary statistics for multivariate estimation
aused = cbind(athinned[2000:6000,1:2])
atrue = c(ralphatil,rphitil)
amean = apply(aused,2,median,na.rm=T)
astd = apply(aused,2,sd,na.rm=T)
alCI = rep(0,2)
auCI = rep(0,2)
for (i in 1:2){
alCI[i] = quantile(aused[,i],p=0.05)
auCI[i] = quantile(aused[,i],p=0.95)
}
abothline = cbind(atrue,amean,astd,alCI,auCI)
```

A.1.3 Real data illustration

```
library(tweedie)
library(MASS)
library(statmod)
library(truncnorm)
library(rootSolve)
library(LaplacesDemon)
library(xtable)
library(ChainLadder)

###########Import data##########
pa = read.csv("pa.csv",header=TRUE)
ca = read.csv("ca.csv",header=TRUE)

st_pa = matrix(NA, nrow=10, ncol=10)
st_ca = matrix(NA, nrow=10, ncol=10)

#Standardise loss cells
for (i in 1:10){
for (j in 1:(10-i+1)){
st_pa[i,j] = pa[i,j+2]/pa[i,2]
st_ca[i,j] = ca[i,j+2]/ca[i,2]
}
}

###########Preliminary analysis - plot development###########

#Plot development trends
cst_pa = incr2cum(st_pa)
cst_ca = incr2cum(st_ca)
```

```
axticks <- seq(1, 10, 1)
axuse <-seq(1,10,1)
par(mfrow=c(1,2))

plot(cst_pa[1,], ylim=c(0,0.82),type = "l",lwd=1, main=expression(paste("
    Personal auto line")),xlab="Development year",ylab="Loss ratio",xaxt="n")
for(i in 2:10){
lines(cst_pa[i,],lwd=1)
}
for(i in 1:10){
points(cst_pa[i,],pch=20)
}
axis(side=1,at=axuse, labels=axticks)

plot(cst_ca[1,],ylim = c(0,0.76),type = "l",lwd=1, main=expression(paste("
    Commercial auto line")), xlab="Development year",ylab="Loss ratio",xaxt="n
    ")
for(i in 2:10){
lines(cst_ca[i,],lwd=1)
}
for(i in 1:10){
points(cst_ca[i,],pch=20)
}
axis(side=1,at=axuse, labels=axticks)

##########Preliminary analysis - analyse dependence using GLM (without
    calendar year factor)##########
stpa = as.vector(t(st_pa))
stca = as.vector(t(st_ca))

stpa = (stpa[!is.na(stpa)])
stca = stca[!is.na(stca)]

#Set up llh profile to find p
i = rep(1:10,10:1)
j <- sequence(10:1)

ci.vec = seq(1,4,by=0.01)

pallh = rep(0,length(ci.vec))
callh = pallh

for (t in 1:length(ci.vec)){
outpa <- glm(stpa~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[t
    ]))
disp1 <- summary(outpa)$dispersion
mu1 <- fitted(outpa)
den1 <- dtweedie(outpa$y, mu = mu1, phi = disp1, power = ci.vec[t])
pallh[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
outca <- glm(stca~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[t
    ]))
disp2 <- summary(outca)$dispersion
mu2 <- fitted(outca)
den2 <- dtweedie(outca$y, mu = mu2, phi = disp2, power = ci.vec[t])
callh[t] <- sum(log(den2))
}

i1 = c(i,i+10)
j1 = c(j,j+10)
cbine=c(stpa,stca)
allh = rep(0, length(ci.vec))

for (t in 1:length(ci.vec)){
outa <- glm(cbine~as.factor(i1) + as.factor(j1), fam=tweedie(var.power=ci.vec
    [t]))
disp <- summary(outa)$dispersion
mu <- fitted(outa)
```

```
den <- dtweedie(outa$y, mu = mu, phi = disp, power = ci.vec[t])
allh[t] <- sum(log(den))
}

paci = ci.vec[which.max(pallh)]
caci = ci.vec[which.max(callh)]
aci = ci.vec[which.max(allh)]

#Find CI for p
outpaci = rep(0,length(pallh))
for (k in 1:length(pallh)){
if (isTRUE(2*abs(pallh[k]-max(pallh)) <= 3.84))  {
outpaci[k] = ci.vec[k]
}
else{
outpaci[k] = 0
}
}

pacilow = outpaci[8]
pacihi = outpaci[41]

outcaci = rep(0,length(callh))
for (k in 1:length(callh)){
if (isTRUE(2*abs(callh[k]-max(callh)) <= 3.84))  {
outcaci[k] = ci.vec[k]
}
else{
outcaci[k] = 0
}
}

cacilow = outcaci[25]
cacihi = outcaci[64]

outaci = rep(0,length(allh))
for (k in 1:length(allh)){
if (isTRUE(2*abs(allh[k]-max(allh)) <= 3.84))  {
outaci[k] = ci.vec[k]
}
else{
outaci[k] = 0
}
}

acilow = outaci[22]
acihi = outaci[48]

#Ln L profile plot
plot(ci.vec,pallh, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(paci, max(pallh), pch=15, cex=1)
points(aci, pallh[26], pch=18, cex=1)
abline(v=pacilow)
abline(v=pacihi)

plot(ci.vec,callh, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(caci, max(callh), pch=15, cex=1)
points(aci, callh[26], pch=18, cex=1)
abline(v=cacilow)
abline(v=cacihi)

plot(ci.vec,allh, type="l", xlim = c(0.9,4.1), xlab="p", ylab="logL")
points(aci, max(allh), pch=15, cex=1)
abline(v=acilow)
abline(v=acihi)

#Results
pest = rbind(c(paci,pacilow,pacihi), c(caci,cacilow,cacihi), c(aci,acilow,
    acihi))

#GLM fitting with the best p
```

```
paglm <- glm(stpa ~ as.factor(i)+as.factor(j), family=tweedie(var.power=paci,
    link.power=0),control = list( epsilon=1e-09, trace=FALSE))
caglm <- glm(stca ~ as.factor(i)+as.factor(j), family=tweedie(var.power=caci,
    link.power=0),control = list( epsilon=1e-09, trace=FALSE))
allglm <-glm(cbine ~ factor(i1)+factor(j1), family=tweedie(var.power=aci,link
    .power=0),control = list( epsilon=1e-09, trace=FALSE))

#QQ plots
qqnorm(residuals(paglm, type="pearson"))
qqline(residuals(paglm, type="pearson"))

qqnorm(residuals(caglm, type="pearson"))
qqline(residuals(caglm, type="pearson"))

qqnorm(residuals(allglm, type="pearson"))
qqline(residuals(allglm, type="pearson"))

#Residuals analysis
res_pa = resid(paglm,"pearson")
res_ca = resid(caglm,"pearson")

peartest = cor.test(res_pa,res_ca,method=c("pearson"), conf.level=0.95)
speatest = cor.test(res_pa,res_ca,method=c("spearman"), conf.level=0.95)
kendtest = cor.test(res_pa,res_ca,method=c("kendall"), conf.level=0.95)
cort = rbind(c(peartest$estimate,speatest$estimate,kendtest$estimate),c(
    peartest$p.value,speatest$p.value,kendtest$p.value))

##########Preliminary analysis - analyse dependence using GLM (without
    calendar year factor)##########

#Set up llh profile to find p
k <- c(seq(1,10),seq(2,10),seq(3,10),seq(4,10), seq(5,10), seq(6,10), seq
    (7,10), seq(8,10), seq(9,10), 10)

pallh2 = rep(0,length(ci.vec))
callh2 = pallh2

for (t in 1:length(ci.vec)){
outpa2 <- glm(stpa~as.factor(i) + as.factor(j) + as.factor(k), fam=tweedie(
    var.power=ci.vec[t]))
disp12 <- summary(outpa2)$dispersion
mu12 <- fitted(outpa2)
den12 <- dtweedie(outpa2$y, mu = mu12, phi = disp12, power = ci.vec[t])
pallh2[t] <- sum(log(den12))
}

for (t in 1:length(ci.vec)){
outca2 <- glm(stca~as.factor(i) + as.factor(j)+ as.factor(k), fam=tweedie(var
    .power=ci.vec[t]))
disp22 <- summary(outca2)$dispersion
mu22 <- fitted(outca2)
den22 <- dtweedie(outca2$y, mu = mu22, phi = disp22, power = ci.vec[t])
callh2[t] <- sum(log(den22))
}

paci2 = ci.vec[which.max(pallh2)]
caci2 = ci.vec[which.max(callh2)]

#GLM fitting with the best p
paglm2 <- glm(stpa ~ as.factor(i)+as.factor(j)+ as.factor(k), family=tweedie(
    var.power=paci2,link.power=0),control = list( epsilon=1e-09, trace=FALSE))
caglm2 <- glm(stca ~ as.factor(i)+as.factor(j)+ as.factor(k), family=tweedie(
    var.power=caci2,link.power=0),control = list( epsilon=1e-09, trace=FALSE))

#Residual analysis
res_pa2 = resid(paglm2,"pearson")
res_ca2 = resid(caglm2,"pearson")

peartest2 = cor.test(res_pa2,res_ca2,method=c("pearson"), conf.level=0.95)
speatest2 = cor.test(res_pa2,res_ca2,method=c("spearman"), conf.level=0.95)
kendtest2 = cor.test(res_pa2,res_ca2,method=c("kendall"), conf.level=0.95)
```

```
cort2 = rbind(c(peartest2$estimate,speatest2$estimate,kendtest2$estimate),c(
    peartest2$p.value,speatest2$p.value,kendtest2$p.value))

#Heat maps of residuals - export to csv file
Var1 = j
Var2 = rep(10:1,10:1)
fitpa2 = stpa/fitted(paglm2)
fitca2 = stca/fitted(caglm2)

ratcalpa <- data.frame(Var1,Var2,fitpa2)
ratcalca <- data.frame(Var1,Var2,fitca2)

#write.csv(ratcalpa,"respaglm.csv")
#write.csv(ratcalca,"rescaglm.csv")

##########Choose initial values and get information for prior distributions
    selection##########

#Estimate alpha and beta
solvefn <- function(x,data){
F2 <- x[1]*sum(x[10:18]^(2-aci)) - as.numeric(crossprod(data[2,1:9],x
    [10:18]^(1-aci)))
F3 <- x[2]*sum(x[10:17]^(2-aci)) - as.numeric(crossprod(data[3,1:8],x
    [10:17]^(1-aci)))
F4 <- x[3]*sum(x[10:16]^(2-aci)) - as.numeric(crossprod(data[4,1:7],x
    [10:16]^(1-aci)))
F5 <- x[4]*sum(x[10:15]^(2-aci)) - as.numeric(crossprod(data[5,1:6],x
    [10:15]^(1-aci)))
F6 <- x[5]*sum(x[10:14]^(2-aci)) - as.numeric(crossprod(data[6,1:5],x
    [10:14]^(1-aci)))
F7 <- x[6]*sum(x[10:13]^(2-aci)) - as.numeric(crossprod(data[7,1:4],x
    [10:13]^(1-aci)))
F8 <- x[7]*sum(x[10:12]^(2-aci)) - as.numeric(crossprod(data[8,1:3],x
    [10:12]^(1-aci)))
F9 <- x[8]*sum(x[10:11]^(2-aci)) - as.numeric(crossprod(data[9,1:2],x
    [10:11]^(1-aci)))
F10 <- x[9]*sum(x[10:10]^(2-aci)) - as.numeric(crossprod(data[10,1],x
    [10:10]^(1-aci)))

F11 <- x[10]*sum(c(1,x[1:9])^(2-aci)) - as.numeric(crossprod(data[,1],c(1,x
    [1:9])^(1-aci)))
F12 <- x[11]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:9,2],c(1,
    x[1:8])^(1-aci)))
F13 <- x[12]*sum(c(1,x[1:7])^(2-aci)) - as.numeric(crossprod(data[1:8,3],c(1,
    x[1:7])^(1-aci)))
F14 <- x[13]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:7,4],c(1,
    x[1:6])^(1-aci)))
F15 <- x[14]*sum(c(1,x[1:5])^(2-aci)) - as.numeric(crossprod(data[1:6,5],c(1,
    x[1:5])^(1-aci)))
F16 <- x[15]*sum(c(1,x[1:4])^(2-aci)) - as.numeric(crossprod(data[1:5,6],c(1,
    x[1:4])^(1-aci)))
F17 <- x[16]*sum(c(1,x[1:3])^(2-aci)) - as.numeric(crossprod(data[1:4,7],c(1,
    x[1:3])^(1-aci)))
F18 <- x[17]*sum(c(1,x[1:2])^(2-aci)) - as.numeric(crossprod(data[1:3,8],c(1,
    x[1:2])^(1-aci)))
F19 <- x[18]*sum(c(1,x[1:1])^(2-aci)) - as.numeric(crossprod(data[1:2,9],c(1,
    x[1:1])^(1-aci)))
F20 <-x[19]*sum(1^(2-aci)) - as.numeric(crossprod(data[1,10],1^(1-aci)))

c(F2=F2,F3=F3,F4=F4,F5=F5,F6=F6,F7=F7,F8=F8,F9=F9,F10=F10,
F11=F11,F12=F12,F13=F13,F14=F14,F15=F15,F16=F16,F17=F17,F18=F18,F19=F19,F20=
    F20)
}

ss1 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(st_pa,na.rm=T)),
    data=st_pa)
ss2 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(st_ca,na.rm=T)),
    data=st_ca)

#Estimate phi1 and phi2
alphaest1 = c(1,ss1$root[1:9])
```

```
betaest1 = ss1$root[10:19]
resid1= matrix(NA,nrow=10,ncol=10)

for (i in 1:10){
for(j in 1:(10-i+1)){
resid1[i,j] = (st_pa[i,j]-alphaest1[i]*betaest1[j])^2/((alphaest1[i]*betaest1
    [j])^aci)
}
}
phi1est = sum(resid1,na.rm=T)/45

alphaest2 = c(1,ss2$root[1:9])
betaest2 = ss2$root[10:19]
resid2= matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for(j in 1:(10-i+1)){
resid2[i,j] = (st_ca[i,j]-alphaest2[i]*betaest2[j])^2/((alphaest2[i]*betaest2
    [j])^aci)
}
}
phi2est = sum(resid2,na.rm=T)/45

#Estimate alpha and beta
betatilest = 0.24
alphatilest=0.0005
minyest=matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for (j in 1:(10-i+1)){
minyest[i,j] <- min((((alphatilest/(alphaest1[i]*betaest1[j]))^(1-aci)*(
    phi1est/betatilest)))*alphatilest,(((alphatilest/(alphaest2[i]*betaest2[j
    ]))^(1-aci)*(phi2est/betatilest)))*alphatilest)
}}

lambdaest = alphatilest^(2-aci)/betatilest

###########Marginal estimatio###########

#Posterior function
model1 <- function(parm,data){
alpha1 <- exp(parm[1:9])
beta1 <-exp(parm[10:19])
alpha2<-exp(parm[20:28])
beta2<-exp(parm[29:38])
phi1 <- exp(parm[39])
phi2<-exp(parm[40])
lambda <- exp(parm[41])

alpha1.prior <- sum(dunif(log(alpha1),min=-0.5,max=1.5,log=T))
beta1.prior <- sum(dunif(log(beta1[1:2]),min=-2.5,max=-0.5,log=T)) +sum(dunif
    (log(beta1[3]),min=-3,max=-1,log=T)) +sum(dunif(log(beta1[4:5]),min=-4,max
    =-2,log=T)) +sum(dunif(log(beta1[6]),min=-5,max=-3,log=T))+sum(dunif(log(
    beta1[7]),min=-6,max=-4,log=T))+sum(dunif(log(beta1[8:9]),min=-6.5,max
    =-4.5,log=T)) +sum(dunif(log(beta1[10]),min=-8.5,max=-6,log=T))

alpha2.prior <- sum(dunif(log(alpha2),min=-0.5,max=1.5,log=T))
beta2.prior <- sum(dunif(log(beta2[1:2]),min=-3,max=-1,log=T))+sum(dunif(log(
    beta1[3]),min=-3.5,max=-1.5,log=T)) +sum(dunif(log(beta2[4:5]),min=-4,max
    =-2,log=T)) +sum(dunif(log(beta2[6:7]),min=-5,max=-3,log=T)) +sum(dunif(
    log(beta2[8]),min=-6,max=-4,log=T))+sum(dunif(log(beta2[9]),min=-6.7,max
    =-4.7,log=T))+sum(dunif(log(beta2[10]),min=-10,max=-7.8,log=T))

phi1.prior <- dunif(log(phi1),min=-7,max=-3,log=T)
phi2.prior <- dunif(log(phi2),min=-7,max=-3,log=T)

lambda.prior <- dunif(log(lambda),min=-3,max=1,log=T)

alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

l <- matrix(NA, nrow=10, ncol=10)
loss1<- data$loss1
```

```
loss2 <-data$loss2

for (i in 1:10){
for (j in 1:(10-i+1)){
A1=alphaf1[i]*beta1[j]*((phi1/((alphaf1[i]*beta1[j])^((2-aci))))*lambda+1)
B1 = phi1*((phi1/((alphaf1[i]*beta1[j])^((2-aci))))*lambda+1)^{(1-aci)}

A2=alphaf2[i]*beta2[j]*((phi2/((alphaf2[i]*beta2[j])^((2-aci))))*lambda+1)
B2 = phi2*((phi2/((alphaf2[i]*beta2[j])^((2-aci))))*lambda+1)^{(1-aci)}

l[i,j] = log(dtweedie(loss1[i,j],xi=aci,mu=A1,phi=B1))+log(dtweedie(loss2[i,j
    ],xi=aci,mu=A2,phi=B2))
}}

LL <- sum(l, na.rm=T)
LP <- LL + alpha1.prior+beta1.prior+alpha2.prior+beta2.prior+phi1.prior+phi2.
    prior+lambda.prior

list(LP = LP, Monitor = c(LP, alpha1, beta1,alpha2,beta2,phi1,phi2,lambda),
    parm=parm)
}

#Prepare for MCMC
data=list(loss1=st_pa,loss2=st_ca)
init <-  c(0.020, -0.084, 0.007, 0.036, 0.085, -0.002, -0.002, 0.183, 0.169,
    -1.436, -1.419, -1.987, -2.528, -3.168, -3.959, -4.956, -5.338, -5.390,
    -7.086, 0.168, 0.150, -0.064, 0.040, 0.155, 0.332, 0.359, 0.440, 0.510,
    -2.041, -1.795, -2.172, -2.386, -2.821, -3.757, -4.069, -4.970, -5.752,
    -8.762, -5.300, -5.050, -0.524)
lower = c(rep(-0.5,9),rep(-2.5,2),-3,rep(-4,2),-5,-6,rep(-6.5,2),-8.5,rep
    (-0.5,9),rep(-3,2),-3.5,rep(-4,2),rep(-5,2),-6,-6.7,-10,-7,-7,-3)
upper = c(rep(1.5,9),rep(-0.5,2),-1,rep(-2,2),-3, -4,rep(-4.5,2),-6, rep
    (1.5,9),rep(-1,2),-1.5,rep(-2,2),rep(-3,2),-4,-4.7,-7.8,-3,-3,1)
std = c( 0.0307, 0.0300, 0.0304, 0.030, 0.0307, 0.0338, 0.0331, 0.0321,
    0.0378, 0.0269, 0.0268,0.0283, 0.0317, 0.0345, 0.0410, 0.0558, 0.0657,
    0.0783, 0.0953, 0.0300, 0.0287, 0.0286,0.030, 0.0321, 0.0321, 0.0356,
    0.0400, 0.0425, 0.0300, 0.0308, 0.0336, 0.0334, 0.0432,0.0472, 0.0529,
    0.0795, 0.0784, 0.0859, 0.0623, 0.0625, 0.3035)-0.004

Iterations=300000
Status=1000
Thinning=5
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

Acceptance <- 0
Mo0 <- model1(init, data)
Mon <- matrix(NA, Iterations, length(Mo0[["Monitor"]]), byrow=TRUE)
dimension <- length(init)
thinned <- matrix(NA, floor(Iterations/Thinning)+1,length(init))
thinned[1,] <- init

#Run MCMC
set.seed(11)
for (iter in 1:Iterations) {

if(iter %% Status == 0) cat("Iteration: ", iter, sep="")

#Metropolis algorithm
prop <- rtruncnorm(1,mean = Mo0[["parm"]],sd = std, a = lower, b=upper)
Mo1 <- try(model1(prop, data), silent=TRUE)

log.u <- log(runif(1))
log.alpha <- Mo1[["LP"]] - Mo0[["LP"]] + log(dtruncnorm(Mo0[["parm"]], mean =
     prop, sd = std, a = lower, b =  upper)) - log(dtruncnorm(prop, mean=Mo0[[
    "parm"]], sd = std, a = lower, b =  upper))

if((is.finite(log.alpha)) && (!inherits(Mo1, "try-error")) &&  ((is.finite(
    Mo1[["Monitor"]]))) && (log.u < log.alpha)) {
```

```
Mo0 <- Mo1
Acceptance <- Acceptance + 1}

Mon[iter,] <- Mo0[["Monitor"]]

#Thin samples
if(iter %% Thinning == 0) {
t.iter <- floor(iter / Thinning) + 1
thinned[t.iter,] <- Mo0[["parm"]]}

#Show tracking
if(iter %% Status == 0){
cat(", LP:", round(Mo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(Acceptance/iter, 2), "\n", sep = "", file =
    LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#Plot sample paths
par(mfrow=c(3,2))
for (i in 2:42){
plot(Mon[2:Iterations,i],type="l")
}

#Summary statistics for marginal estimation
used = exp(thinned[30001:60000,])
mean = apply(used,2,median,na.rm=T)
st.dev = apply(used,2,sd,na.rm=T)
lCI = rep(0,41)
uCI = rep(0,41)
for (i in 1:41){
lCI[i] = quantile(used[,i],p=0.05)
uCI[i] = quantile(used[,i],p=0.95)
}
bothline = cbind(mean,st.dev,lCI,uCI)

##########Multivariate estimation (adaptive Metropolis)###########

#Use parameter estimates from marginal estimation
alpha1 <- mean[1:9]
beta1 <- mean[10:19]
alpha2<- mean[20:28]
beta2<- mean[29:38]
phi1 <- mean[39]
phi2<- mean[40]
lambda <- mean[41]
alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

#Posterior function
model2 <- function(parm,data){
alpha <- exp(parm)
beta <- alpha^(2-aci)/lambda

alpha.prior <- dunif(log(alpha),min=-7,max=-4,log=T)

l <- matrix(NA, nrow=10, ncol=10)
loss1<- data$loss1
loss2 <- data$loss2
miny <- matrix(NA,nrow=10,ncol=10)
prior = c(alpha.prior)

if(all(is.finite(prior))){
for (i in 1:10){
for (j in 1:(10-i+1)){
miny[i,j] <- min((loss1[i,j]/((alpha/(alphaf1[i]*beta1[j]))^(1-aci)*(phi1/
    beta))), (loss2[i,j]/((alpha/(alphaf2[i]*beta2[j]))^(1-aci)*(phi2/beta))))

if (miny[i,j]<=1e-200000000){
```

```
l[i,j]=log(dtweedie(loss1[i,j],xi=aci, mu=alphaf1[i]*beta1[j], phi = phi1))+
    log(dtweedie(loss2[i,j],xi=aci, mu=alphaf2[i]*beta2[j], phi = phi2))
}
else {
f<- function(z){
dtweedie(loss1[i,j]-(alpha/(alphaf1[i]*beta1[j]))^(1-aci)*(phi1/beta)*z,xi=
    aci, mu=alphaf1[i]*beta1[j], phi = phi1)*dtweedie(loss2[i,j]-(alpha/(
    alphaf2[i]*beta2[j]))^(1-aci)*(phi2/beta)*z,xi=aci, mu=alphaf2[i]*beta2[j
    ], phi = phi2)*dtweedie(z,xi=aci, mu=alpha, phi = beta)
}
llh<- try(integrate(f,lower=1e-2000000000, upper=miny[i,j]),silent=T)
if(inherits(llh, 'try-error') ){
l[i,j] <- log(0)
}
else{
l[i,j] <- log(llh$value)}}
}}}

LL <- sum(l, na.rm=T)
LP <- LL + alpha.prior

list(LP = LP, Monitor = c(LP, alpha,beta), parm=parm)
}

#Prepare for MCMC
ainit <-  -7
Iterations=100000
Status=100
Thinning=3
alpha.star=0.30
Periodicity=1
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

aAcceptance <- 0
aMo0 <- model2(ainit, data)
aMon <- matrix(NA,nrow=Iterations,ncol=length(aMo0[["Monitor"]]))
adimension <- length(ainit)
athinned <- matrix(NA, floor(Iterations/Thinning)+1,length(ainit)+1)
athinned[1,] <- ainit
aScaleF <- 0.0001/sqrt(adimension)
aVarCov <- matrix(0, adimension, adimension)
diag(aVarCov) <- rep(aScaleF, adimension)
aS <- t(chol(aVarCov))

#Run MCMC
set.seed(11)

for (aiter in 1:Iterations) {
if(aiter %% Status == 0) cat("Iteration: ", aiter, sep="")

#Adaptive Metropolis
aU <- rnorm(adimension)
aprop <- as.vector(aMo0[["parm"]] + aS %*% aU)
aMo1 <- try(model2(aprop, data), silent=TRUE)

alog.u <- log(runif(1))
alog.alpha <- aMo1[["LP"]] - aMo0[["LP"]]
if((is.finite(alog.alpha)) && (!inherits(aMo1, "try-error")) &&  ((is.finite(
    aMo1[["Monitor"]]))) && (alog.u < alog.alpha)) {
aMo0 <- aMo1
aAcceptance <- aAcceptance + 1}

aMon[aiter,] <- aMo0[["Monitor"]]

if({aiter >= 2} & {aiter %% Periodicity == 0}) {
aeta <- min(1, adimension*aiter^(-2/3))
aVarCov.test <- aS %*% (diag(adimension) + aeta*(min(1, exp(alog.alpha)) -
    alpha.star) * aU %*% t(aU) / sqrt(sum(aU^2))) %*% t(aS)
```

```
if(!all(is.finite(aVarCov.test))) {aVarCov.test <- aVarCov}
if(!is.symmetric.matrix(aVarCov.test)){aVarCov.test <- as.symmetric.matrix(
    aVarCov.test)}
if(is.positive.definite(aVarCov.test)){ aS.z <- try(t(chol(aVarCov)), silent=
    TRUE)
if(!inherits(aS.z, "try-error")) {
aVarCov <- aVarCov.test
aS <- aS.z}}}

#Thin samples
if(aiter %% Thinning == 0) {
at.iter <- floor(aiter / Thinning) + 1
athinned[at.iter,] <- aMo0[["Monitor"]][2:3]}

if(aiter %% Status == 0){
cat(", LP:", round(aMo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(aAcceptance/aiter, 2), "\n", sep = "", file =
     LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#MCMC trace plots
par(mfrow=c(2,1))
plot( log(aMon[2:(aiter-1),2]),type="l")
plot( log(aMon[2:(aiter-1),3]),type="l")

#Results
aused = (athinned[(nrow(athinned)-30000+1):nrow(athinned),])
amean = apply(aused,2,median,na.rm=T)
asd = apply(aused,2,sd,na.rm=T)
alCI = rep(0,2)
auCI = rep(0,2)
for (i in 1:2){
alCI[i] = quantile(aused[,i],p=0.05)
auCI[i] = quantile(aused[,i],p=0.95)

}
abothline = cbind(amean,asd,alCI,auCI)


##########Goodness of fit test##########

#Marginal fit
fittedmarginal1 = matrix(NA,nrow=10,ncol=10)
fittedmarginal2 = matrix(NA,nrow=10,ncol=10)
pres1 = matrix(NA,nrow=10,ncol=10)
pres2 = matrix(NA,nrow=10,ncol=10)

for (i in 1:10){
for (j in 1:(10-i+1)){
B1 = phi1*((phi1/((alphaf1[i]*beta1[j])^((2-aci))))*lambda+1)^{(1-aci)}
B2 = phi2*((phi2/((alphaf2[i]*beta2[j])^((2-aci))))*lambda+1)^{(1-aci)}

fittedmarginal1[i,j]=alphaf1[i]*beta1[j]*((phi1/((alphaf1[i]*beta1[j])^((2-
    aci))))*lambda+1)
pres1[i,j] = (st_pa[i,j]-fittedmarginal1[i,j])/sqrt(fittedmarginal1[i,j]^aci*
    B1)

fittedmarginal2[i,j]=alphaf2[i]*beta2[j]*((phi2/((alphaf2[i]*beta2[j])^((2-
    aci))))*lambda+1)
pres2[i,j] = (st_ca[i,j]-fittedmarginal2[i,j])/sqrt(fittedmarginal2[i,j]^aci*
    B2)
}
}

mtresid1=as.vector(pres1)
mtresid1=mtresid1[!is.na(mtresid1)]

mtresid2=as.vector(pres2)
```

```
mtresid2=mtresid2[!is.na(mtresid2)]

par(mfrow=c(1,2))
qqnorm(mtresid1,font.main = 1,main="Personal auto line")
qqline(mtresid1)

qqnorm(mtresid2,font.main = 1,main="Commercial auto line")
qqline(mtresid2)

#Multivariate fit
modelgof <- function(parm){
alpha1 <- parm[1:9]
beta1 <-parm[10:19]
alpha2<-parm[20:28]
beta2<-parm[29:38]
phi1 <- parm[39]
phi2<-parm[40]
alphatil <- parm[41]
betatil <- parm[42]
alphaf1 = c(1,alpha1)
alphaf2 = c(1,alpha2)

frandom = matrix(NA,ncol=10,nrow=10)
fpa = matrix(NA,ncol=10,nrow=10)
fca = matrix(NA,ncol=10,nrow=10)

for (i in 1:10){
for (j in 1:(10-i+1)){
frandom[i,j] = rtweedie(1,mu=alphatil,phi=betatil,xi=aci)
fpa[i,j] = ((alphatil/((alphaf1[i]*beta1[j])))^(1-aci)*(phi1/betatil)*frandom
    [i,j]+ rtweedie(1,mu=alphaf1[i]*beta1[j],phi=phi1,xi=aci))*pa[i,2]
fca[i,j] = ((alphatil/((alphaf2[i]*beta2[j])))^(1-aci)*(phi2/betatil)*frandom
    [i,j]+ rtweedie(1,mu=alphaf2[i]*beta2[j],phi=phi2,xi=aci))*ca[i,2]
}
}

list (vfpa = as.vector(t(fpa)), vfca = as.vector(t(fca)))
}

para = cbind(used[,1:40], aused)
N = nrow(para)
gof_sfpa = matrix(NA,nrow=N,ncol=100)
gof_sfca = matrix(NA,nrow=N,ncol=100)

set.seed(11)
for (n in 1:N){
mtest = modelgof(para[n,])
gof_sfpa[n,] = mtest[["vfpa"]]
gof_sfca[n,] = mtest[["vfca"]]
}

par(mfrow=c(1,2), mai = c(0.8, 0.75, 0.5, 0.1))
plot(ecdf(stpa)(stpa),ecdf(stca)(stca),xlab="Personal auto",ylab="Commercial
    auto",pch=1,main="Observed data",font.main = 1)
plot(ecdf(gof_sfpa[10000,])(gof_sfpa[10000,]),ecdf(gof_sfca[10000,])(gof_sfca
    [10000,]),pch=1,main="Fitted data",xlab="Personal auto",ylab="Commercial
    auto",font.main = 1)

##########Claims forecast##########

#Function to forecast future claims
modelp <- function(parm){
alpha1 <- (parm[1:9])
beta1 <-(parm[10:19])
alpha2<-(parm[20:28])
beta2<-(parm[29:38])
phi1 <- (parm[39])
phi2<-(parm[40])
alphatil <- (parm[41])
betatil <- (parm[42])
alphaf1 = c(1,alpha1)
```

```
alphaf2 = c(1,alpha2)

frandom = matrix(NA,ncol=10,nrow=10)
fpa = matrix(NA,ncol=10,nrow=10)
fca = matrix(NA,ncol=10,nrow=10)
ftotal = matrix(NA,ncol=10,nrow=10)

fpaacc = c(0,10)
fcaacc = c(0,10)
fttacc = c(0,10)

for (i in 2:10){
for (j in (10-i+2):10){
frandom[i,j] = rtweedie(1,mu=alphatil,phi=betatil,xi=aci)

fpa[i,j] = ((alphatil/((alphaf1[i]*beta1[j])))^(1-aci)*(phi1/betatil)*frandom
    [i,j]+ rtweedie(1,mu=alphaf1[i]*beta1[j],phi=phi1,xi=aci))*pa[i,2]
fca[i,j] = ((alphatil/((alphaf2[i]*beta2[j])))^(1-aci)*(phi2/betatil)*frandom
    [i,j]+ rtweedie(1,mu=alphaf2[i]*beta2[j],phi=phi2,xi=aci))*ca[i,2]
ftotal[i,j] = fpa[i,j] + fca[i,j]
}

fpaacc[i] = sum(fpa[i,], na.rm=T)
fcaacc[i] = sum(fca[i,],na.rm=T)
fttacc[i] = fpaacc[i]+fcaacc[i]
}

ttpa = sum(fpaacc, na.rm=T)
ttca = sum(fcaacc, na.rm=T)
tt = ttpa+ttca

list (t = c(ttpa, ttca, tt), vfpa = as.vector(t(fpa)), vfca = as.vector(t(fca
    )), vftotal = as.vector(t(ftotal)), fpaacc = fpaacc, fcaacc = fcaacc,
    fttacc = fttacc)
}

#Run simulation
para = cbind(used[,1:40], aused)
N = nrow(para)
stt = matrix(NA,nrow=N,ncol=15)

sfpa = matrix(NA,nrow=N,ncol=100)
sfca = matrix(NA,nrow=N,ncol=100)
sftotal = matrix(NA,nrow=N,ncol=100)

sfpaacc = matrix(NA,nrow=N,ncol=10)
sfcaacc = matrix(NA,nrow=N,ncol=10)
sfttacc = matrix(NA,nrow=N,ncol=10)

cat("Simulation started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

set.seed(11)
for (n in 1:N){
mpredict = modelp(para[n,])
stt[n,] = mpredict[["t"]]
sfpa[n,] = mpredict[["vfpa"]]
sfca[n,] = mpredict[["vfca"]]
sftotal[n,] = mpredict[["vftotal"]]
sfpaacc[n,] = mpredict[["fpaacc"]]
sfcaacc[n,] = mpredict[["fcaacc"]]
sfttacc[n,] = mpredict[["fttacc"]]
}

cat("Simulation ended on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

#summarise results
mean_stt = colMeans(stt[,1:3],na.rm=T)
```

```
sd_stt = apply(stt[,1:3],2,sd,na.rm=T)
var75 = c(0,0,0)
var95 = c(0,0,0)

for (i in 1:3){
var75[i] = quantile(stt[,i],p=0.75)
var95[i] = quantile(stt[,i],p=0.95)
}
summarytable = rbind(mean_stt,sd_stt,var75,var95)

paaccmean = colMeans(sfpaacc)
paaccsd = apply(sfpaacc,2,sd,na.rm=T)

caaccmean = colMeans(sfcaacc)
caaccsd = apply(sfcaacc,2,sd,na.rm=T)

ttaccmean = colMeans(sfttacc)
ttaccsd = apply(sfttacc,2,sd,na.rm=T)

accidentyrsummary = cbind(paaccmean,paaccsd,caaccmean,caaccsd,ttaccmean,
    ttaccsd)
xtable(accidentyrsummary)

library(EnvStats)

par(mfrow=c(1,1))
plot (density(stt[,1]), ylim=c(0,6e-05),xlim=c(55000,275000),xlab="Total
    unpaid losses (in 1,000's)",main="",lwd=3)
lines (density(stt[,2]), lty=2,lwd=3)
lines (density(stt[,3]), lty=3,lwd=3)
legend("top", legend = c("Personal auto", "Commercial auto", "Total"),
lty = 1:3,lwd=3, bty = "n",
title = "")
```

A.2 R codes for Chapter 5

A.2.1 Simulation illustration 1

```
library(tweedie)
library(MASS)
library(statmod)
library(truncnorm)
library(LaplacesDemon)
library(tmvtnorm)
library(MHadaptive)
##########Simulation##########
ralpha1 = c(1, 1.03, 1.19, 1.12, 1.15, 1.16, 1.12, 1.14, 1.21, 1.19)
ralpha2 = c(1, 1.19, 1.17, 1.15, 1.15, 1.20, 1.40, 1.45, 1.56, 1.66)

rbeta1 = c(60,20,10,5,2.5,1.25,0.6,0.3,0.15,0.15)
rbeta2 = c(10,20,25,20,15,8,3,2,1,1)

rphi1 = 0.5
rphi2 = 0.7

rzeta=0.5
ralphatil = sqrt(rbeta1[1:10]*rbeta2)*rzeta
rphitil = 0.6

sp=1.3

stau=(sp-2)/(sp-1)

#Generate loss triangles
random_w = matrix(NA, nrow=10,ncol=10)
usloss1 = matrix(NA, nrow=10,ncol=10)
usloss2 = matrix(NA, nrow=10, ncol=10)

set.seed(48)
for(i in 1:10){
for (j in 1:(10-i+1)){
random_w[i,j] = rtweedie(1,mu=ralphatil[j],phi=rphitil,xi=sp)
usloss1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
usloss2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}}

radj = 0.01
loss1=usloss1-radj
loss2 = usloss2

##########Premliminary analysis###########
plot(loss1[1,],type = "l",lwd=1,ylim=c(0,max(loss1,na.rm=T)*1.1))
for(i in 2:10){
lines(loss1[i,],lwd=1)
}

plot(loss2[1,],type = "l",lwd=1,ylim=c(0,max(loss2,na.rm=T)*1.1))
for(i in 2:10){
lines(loss2[i,],lwd=1)
}

#GLM analysis
vloss1 = as.vector(t(loss1))+radj
vloss2 = as.vector(t(loss2))

#Set up llh profile for each line
i = rep(1:10,each=10)
j = rep(1:10, 10)

ci.vec = seq(1,2,by=0.01)
```

```
llh1 = rep(0,length(ci.vec))
llh2 = llh1

for (t in 1:length(ci.vec)){
out1 <- glm(vloss1~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp1 <- summary(out1)$dispersion
mu1 <- fitted(out1)
den1 <- dtweedie(out1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2 <- glm(vloss2~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp2 <- summary(out2)$dispersion
mu2 <- fitted(out2)
den2 <- dtweedie(out2$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2[t] <- sum(log(den2))
}

#Set up llh profile for both lines combined
i1 = c(rep(1:10,each=10),rep(11:20, each=10))
j1 = c(rep(1:10,10),rep(11:20,10))
cbine=c(vloss1,vloss2)
allh = rep(0, length(ci.vec))

for (t in 1:length(ci.vec)){
outa <- glm(cbine~as.factor(i1) + as.factor(j1), fam=tweedie(var.power=ci.vec
    [t]))
disp <- summary(outa)$dispersion
mu <- fitted(outa)
den <- dtweedie(outa$y, mu = mu, phi = disp, power = ci.vec[t])
allh[t] <- sum(log(den))
}

#Find p that maximises llh
ci1 = ci.vec[which.max(llh1)]
ci2 = ci.vec[which.max(llh2)]
aci = ci.vec[which.max(allh)]

#power parameter to be used later
paglm <- glm(vloss1 ~ as.factor(i)+as.factor(j), family=tweedie(var.power=ci1
    ,link.power=0),control = list( epsilon=1e-09, trace=FALSE))
caglm <- glm(vloss2 ~ as.factor(i)+as.factor(j), family=tweedie(var.power=ci2
    ,link.power=0),control = list( epsilon=1e-09, trace=FALSE))

res1 = resid(paglm,"pearson")
res2 = resid(caglm,"pearson")

##########Marginal estimation##########

#Posterior function
llhfunc <- function(pars,data){
alpha1 <- exp(pars[1:9])
beta1 <-exp(pars[10:19])
alpha2<-exp(pars[20:28])
beta2<-exp(pars[29:38])
phi1 <- exp(pars[39])
phi2<-exp(pars[40])
lambda <- exp(pars[41])
adj <- exp(pars[42])
p <- exp(pars[43])

alpha1.prior <- sum(dunif(log(alpha1), min=lower[1:9], max=upper[1:9], log=T)
    )
beta1.prior <- sum(dunif(log(beta1), min=lower[10:19], max=upper[10:19], log=
    T))
alpha2.prior <- sum(dunif(log(alpha2), min=lower[20:28], max=upper[20:28],
    log=T))
beta2.prior <- sum(dunif(log(beta2), min=lower[29:38], max=upper[29:38], log=
```

```
    T))

phi1.prior <- dunif(log(phi1),min=lower[39],max=upper[39],log=T)
phi2.prior <- dunif(log(phi2),min=lower[40],max=upper[40],log=T)
lambda.prior <- dunif(log(lambda),min=lower[41],max=upper[41],log=T)
adj.prior <- dunif(log(adj),min=lower[42],max=upper[42],log=T)
p.prior <- dunif(log(p),min=lower[43],max=upper[43],log=T)

alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

loss1<- data[1:10,]
loss2 <-data[11:20,]

meancommon1 = matrix(0,nrow=10,ncol=10)
scaledcommon1 = matrix(0,nrow=10,ncol=10)
A1 = matrix(NA,nrow=10,ncol=10)
B1 = matrix(NA,nrow=10,ncol=10)

meancommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
A1 = meancommon1 + (alphaf1%o%beta1)
scaledcommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
B1 = phi1*(scaledcommon1 + 1)^(1-p)

meancommon2 = matrix(0,nrow=10,ncol=10)
scaledcommon2 = matrix(0,nrow=10,ncol=10)
A2 = matrix(NA,nrow=10,ncol=10)
B2 = matrix(NA,nrow=10,ncol=10)

meancommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
A2 = meancommon2 + (alphaf2%o%beta2)
scaledcommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
B2 = phi2*(scaledcommon2 + 1)^(1-p)

vecloss1na = c(t(loss1))
vecloss1 = vecloss1na[!is.na(vecloss1na)]
vecA1na = c(t(A1))
vecA1 = vecA1na[!is.na(vecloss1na)]
vecB1na = c(t(B1))
vecB1 = vecB1na[!is.na(vecloss1na)]

vecloss2na = c(t(loss2))
vecloss2 = vecloss2na[!is.na(vecloss2na)]
vecA2na = c(t(A2))
vecA2 = vecA2na[!is.na(vecloss2na)]
vecB2na = c(t(B2))
vecB2 = vecB2na[!is.na(vecloss2na)]

if ((any(vecloss1+adj<0)==T)|!is.finite(p.prior)){LL = -100000000}
else{
LL = sum(log(dtweedie(vecloss1+adj,xi=p,mu=vecA1,phi=vecB1)))+sum(log(
    dtweedie(vecloss2,xi=p, mu=vecA2, phi=vecB2)))}

LP <- LL + alpha1.prior+beta1.prior+alpha2.prior+beta2.prior+phi1.prior+phi2.
    prior+lambda.prior+p.prior+adj.prior

list(Monitor = c(LP, alpha1, beta1,alpha2,beta2,phi1,phi2,lambda,adj,p),pars=
    pars,LP=LP)
}

#Use adaptive MCMC to get estimates for covariance of proposal density
model1 <- function(pars,data){
LP = llhfunc(pars, data)$LP
return(LP)
}
```

```
data = rbind(loss1,loss2)
init <-  c(0.0296, 0.1740, 0.1133, 0.1398, 0.1484, 0.1133, 0.1310, 0.1906,
    0.1740, 4.0943, 2.9957, 2.3026, 1.6094,0.9163, 0.2231, -0.5108, -1.2040,
    -1.8971, -1.8971,  0.1740,  0.1570,  0.1398,  0.1398,  0.1823,  0.3365,
    0.3716,  0.4447,  0.5068,  2.3026,  2.9957, 3.2189,  2.9957,  2.7081,
    2.0794,  1.0986,  0.6931,  0.0000,  0.0000, -0.6931, -0.3567,  0.0118,
    -4.6052,  0.2624)
lower = c(rep(-0.2,9), 2.0000,  0.9957,  0.3026, -0.3906, -1.0837, -0.5000,
    -2.5108, -3.2040, -3.8971, -3.8971, rep(-0.2,9), 0.3026, 0.9957, 1.2189,
    0.9957,  0.7081,  1.0000, -0.9014, -1.3069, -2.0000, -2.0000, -2.6931,
    -1.0000, -2.0000, -4.6052,  0)
upper = c(rep(1.5,9), 6.0943, 4.9957, 4.3026, 3.6094, 2.9163, 2.2231, 1.4892,
     0.7960, 0.1029, 0.1029, rep(1.5,9), 4.3026, 4.9957, 5.2189, 4.9957,
    4.7081, 4.0794, 3.0986, 2.6931, 2.0000, 2.0000, 1.3069, 1.6433, 3.7000,
    3.0000, 0.6931)
std = (upper-lower)*0.01

par_names=paste(c(rep("alpha1",9),rep("beta1",10),rep("alpha2",9),rep("beta2"
    ,5),"phi1","phi2","phitil","adjustment","p"),c(seq(1:9),seq(1:10),seq(1:9)
    ,seq(1:5),0,0,0,0,0))
set.seed(1)
mh_test <- Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
    rbind(loss1,loss2),prop_sigma = diag(std),iterations = 70000,burn_in =
    30000)
set.seed(1)
mh_test1<-Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
    rbind(loss1,loss2),prop_sigma = mh_test$prop_sigma*0.001,iterations =
    50000,burn_in = 10000)

write.csv(mh_test1$prop_sigma, file="cov_matrix.csv")

par(mfrow=c(3,2))
for(i in 1:43){plot(mh_test1$trace[,i],type="l")}

#Posterior function 2
model1 <- function(pars,data){
llh = llhfunc(pars,data)
list(Monitor = c(llh$LP, llh$alpha1, llh$beta1,llh$alpha2,llh$beta2,llh$phi1,
    llh$phi2,llh$lambda,llh$adj,llh$p),pars=llh$pars,LP=llh$LP)
}

#Run MCMC
std = as.matrix(read.csv("cov_matrix.csv",header=T))[,2:44]*0.24

Iterations=200000
Status=1000
Thinning=5
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

Acceptance <- 0
Mo0 <- model1(init, data)
Mon <- matrix(NA, nrow=Iterations, ncol=length(Mo0[["Monitor"]]), byrow=TRUE)
dimension <- length(init)
thinned <- matrix(NA, floor(Iterations/Thinning)+1,length(init))
thinned[1,] <- init

set.seed(11)
for (iter in 1:Iterations) {

if(iter %% Status == 0) cat("Iteration: ", iter, sep="")

#Metropolis algorithm
prop <- mvrnorm(1,mu = Mo0[["pars"]],Sigma=std)
Mo1 <- try(model1(prop, data), silent=TRUE)

log.u <- log(runif(1))
log.alpha <- Mo1[["LP"]] - Mo0[["LP"]] + log(dmvnorm(Mo0[["pars"]], mean =
```

```
    prop, sigma=std)) - log(dmvnorm(prop, mean=Mo0[["pars"]], sigma=std))
if((is.finite(log.alpha)) && (!inherits(Mo1, "try-error")) &&  ((is.finite(
    Mo1[["Monitor"]]))) && (log.u < log.alpha)) {
Mo0 <- Mo1
Acceptance <- Acceptance + 1}

Mon[iter,] <- Mo0[["Monitor"]]

#Thin Samples
if(iter %% Thinning == 0) {
t.iter <- floor(iter / Thinning) + 1
thinned[t.iter,] <- Mo0[["pars"]]}

#Show tracking
if(iter %% Status == 0){
cat(", LP:", round(Mo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(Acceptance/iter, 2), "\n", sep = "", file =
    LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#Plot sample paths
thinnedplot = matrix(NA,nrow=5000,ncol=43)
freq = 8

for(n in 1:40000){
if(n%%freq == 0){
tp.iter <- floor(n / freq)
thinnedplot[tp.iter,] <- thinned[n,1:43]
}
}

xticks <- seq(0, 200000, 50000)
xuse <-seq(0,5000, 1250)
layout(matrix(c(1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,0,9,9,0), 5, 4, byrow=TRUE))
par(mar=c(2.5,2.5,2.5,2.5))

plot((thinnedplot[1:5000,1]),type="l",main=expression(paste(alphatil[2]^(1)))
    ,ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,10]),type="l",main=expression(paste(phitil[1]^(1))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,20]),type="l",main=expression(paste(alphatil[2]^(2))
    ),ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,29]),type="l",main=expression(paste(phitil[1]^(2))),
    ylab="",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,39]),type="l",main=expression(paste(phi^(1))),ylab="
    ",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,40]),type="l",main=expression(paste(phi^(2))),ylab="
    ",xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,41]),type="l",main=expression(paste(Lambda)),ylab=""
    ,xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,42]),type="l",main=expression(paste(xi^(1))),ylab=""
    ,xlab="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)
plot((thinnedplot[1:5000,43]),type="l",main=expression(paste(p)),ylab="",xlab
    ="Iteration", xaxt = "n")
axis(side=1,at=xuse, labels=xticks)

#Summary statistics for marginal estimation
true=exp(c(ralpha1[2:10],rbeta1,ralpha2[2:10],rbeta2,rphi1,rphi2,rzeta^(2-p)/
    rphitil,radj,sp))
used = exp(thinned[20001:40000,])
```

```
median = round(apply((used),2,median,na.rm = T),3)
st.dev = apply(used,2,sd,na.rm=T)
lCI = rep(0,43)
uCI = rep(0,43)
for (i in 1:43){
lCI[i] = round(quantile(used[,i],p=0.05),3)
uCI[i] = round(quantile(used[,i],p=0.95),3)
}
bothline = cbind((true),median,st.dev,(lCI),(uCI))

##########Multivariate estimation (adaptive Metropolis)###########

#Use parameter estimates from marginal estimation
alpha1 <-  median[1:9]
beta1 <- median[10:19]
alpha2<- median[20:28]
beta2<- median[29:38]
phi1 <-  median[39]
phi2<- median[40]
lambda <- median[41]
adj <- median[42]
p<- median[43]
alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)
alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

#Posterior function
model2 <- function(pars,data){
zeta <- exp(pars)
phitil<- zeta^(2-p)/lambda

alphatilmean = (sqrt(beta1*beta2))^(2-p)*zeta
zeta.prior <- dunif(log(zeta),min=-3,max=1,log=T)

l1 <- matrix(NA, nrow=10, ncol=10)
miny <- matrix(NA, nrow=10, ncol=10)
loss1<- data[1:10,]
loss2<- data[11:20,]

if(all(is.finite(zeta.prior))){
for (i in 1:10){
for (j in 1:(10-i+1)){
miny[i,j] <- min(((loss1[i,j]+adj)/((alphatilmean[j]/(alphaf1[i]*beta1[j]))
    ^(1-p)*(phi1/phitil))), (loss2[i,j]/((alphatilmean[j]/(alphaf2[i]*beta2[j
    ]))^(1-p)*(phi2/phitil))))
if (miny[i,j]<=1e-200000000){
l1[i,j]=log(dtweedie(loss1[i,j]+adj,xi=p, mu=alphaf1[i]*beta1[j], phi = phi1)
    )+log(dtweedie(loss2[i,j],xi=p, mu=alphaf2[i]*beta2[j], phi = phi2))
}
else {
f<- function(z){
dtweedie(loss1[i,j]+adj-(alphatilmean[j]/(alphaf1[i]*beta1[j]))^(1-p)*(phi1/
    phitil)*z,xi=p, mu=alphaf1[i]*beta1[j], phi = phi1)*dtweedie(loss2[i,j]-(
    alphatilmean[j]/(alphaf2[i]*beta2[j]))^(1-p)*(phi2/phitil)*z,xi=p, mu=
    alphaf2[i]*beta2[j], phi = phi2)*dtweedie(z,xi=p, mu=alphatilmean[j], phi
    = phitil)
}
llh<- try(integrate(f,lower=1e-2000000000, upper=miny[i,j]),silent=T)
if(inherits(llh, 'try-error')){
l1[i,j] <- log(0)
}
else{
l1[i,j] <- log(llh$value)}}
}}}

LL <- sum(l1, na.rm=T)
LP <- LL +  zeta.prior

list(LP = LP, Monitor = c(LP, zeta,phitil), pars=pars)
}
```

```
#Prepare for MCMC
ainit <-  -0.7

Iterations=200000
Status=1000
Thinning=5
alpha.star=0.3
Periodicity=1
LogFile=""

cat("MCMC started on ", date(), "\n", sep="")
time1 <- proc.time()

aAcceptance <- 0
aMo0 <- model2(ainit, data)
aMon <- matrix(NA,nrow=Iterations,ncol=length(aMo0[["Monitor"]]))
adimension <- length(ainit)
athinned <- matrix(NA, floor(Iterations/Thinning)+1,2)
athinned[1,] <- aMo0[["Monitor"]][2:3]
aScaleF <- 0.0001/sqrt(adimension)
aVarCov <- matrix(0, adimension, adimension)
diag(aVarCov) <- rep(aScaleF, adimension)
aS <- t(chol(aVarCov))

set.seed(11)
for (aiter in 1:Iterations) {
if(aiter %% Status == 0) cat("Iteration: ", aiter, sep="")

#Adaptive Metropolis
aU <- rnorm(adimension)
aprop <- as.vector(aMo0[["parm"]] + aS %*% aU)
aMo1 <- try(model2(aprop, data), silent=TRUE)

alog.u <- log(runif(1))
alog.alpha <- aMo1[["LP"]] - aMo0[["LP"]]
if((is.finite(alog.alpha)) && (!inherits(aMo1, "try-error")) &&  ((is.finite(
    aMo1[["Monitor"]]))) && (alog.u < alog.alpha)) {
aMo0 <- aMo1
aAcceptance <- aAcceptance + 1}

aMon[aiter,] <- aMo0[["Monitor"]]

if({aiter >= 2} & {aiter %% Periodicity == 0}) {
aeta <- min(1, adimension*aiter^(-2/3))
aVarCov.test <- aS %*% (diag(adimension) + aeta*(min(1, exp(alog.alpha)) -
    alpha.star) * aU %*% t(aU) / sqrt(sum(aU^2))) %*% t(aS)
if(!all(is.finite(aVarCov.test))) {aVarCov.test <- aVarCov}
if(!is.symmetric.matrix(aVarCov.test)){aVarCov.test <- as.symmetric.matrix(
    aVarCov.test)}
if(is.positive.definite(aVarCov.test)){ aS.z <- try(t(chol(aVarCov)), silent=
    TRUE)
if(!inherits(aS.z, "try-error")) {
aVarCov <- aVarCov.test
aS <- aS.z}}}

#Thin samples
if(aiter %% Thinning == 0) {
at.iter <- floor(aiter / Thinning) + 1
athinned[at.iter,] <- aMo0[["Monitor"]][2:3]}

if(aiter %% Status == 0){
cat(", LP:", round(aMo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(aAcceptance/aiter, 2), "\n", sep = "", file =
     LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#MCMC trace plots
par(mfrow=c(2,1))
```

```
plot(athinned[2:at.iter,1],type="l")
plot(athinned[2:at.iter,2],type="l")

#Summary statistics for multivariate estimation
aused = (athinned[20001:40000,])
amean = round(apply((aused),2,median,na.rm = T),3)
astd = apply(aused,2,sd,na.rm=T)
alCI = rep(0,2)
auCI = rep(0,2)
for (i in 1:2){
alCI[i] = round(quantile(aused[,i],p=0.05),3)
auCI[i] = round(quantile(aused[,i],p=0.95),3)
}
abothline = cbind(c(rzeta,rphitil),amean,astd,alCI,auCI)
```

A.2.2 Simulation illustration 2 (comparison)

A.2.2.1 Multivariate Tweedie model without treatment for unbalanced data

```
library(tweedie)
library(MASS)
library(statmod)
library(truncnorm)
library(tmvtnorm)
library(MHadaptive)

##########Simulation############
ralpha1 = c(1, 1.03, 1.19, 1.12, 1.15, 1.16, 1.12, 1.14, 1.21, 1.19)
ralpha2 = c(1, 1.19, 1.17, 1.15, 1.15, 1.20, 1.40, 1.45, 1.56, 1.66)

rbeta1 = c(60,20,10,5,2.5,1.25,0.6,0.3,0.15,0.15)
rbeta2 = c(10,20,25,20,15,8,3,2,1,1)

rphi1 = 0.9
rphi2 = 0.7

rzeta=0.5
ralphatil = c(sqrt(rbeta1[1:4]*rbeta2[1:4])*rzeta,sqrt(rbeta1[5:10]*rbeta2
    [5:10])*0.02)
rphitil = 0.6

sp=1.3

stau=(sp-2)/(sp-1)

#Generate loss triangles
set.seed(2)
random_w = matrix(NA, nrow=10,ncol=10)
loss1 = matrix(NA, nrow=10,ncol=10)
loss2 = matrix(NA, nrow=10, ncol=10)

for(i in 1:6){
for (j in 1:4){
random_w[i,j] = rtweedie(1,mu=ralphatil[j],phi=rphitil,xi=sp)
loss1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}
for (j in 5:(10-i+1)){
random_w[i,j] = rtweedie(1,mu=ralphatil[j],phi=rphitil,xi=sp)
loss1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}}

for(i in 7:10){
```

```
for (j in 1:(10-i+1)){
random_w[i,j] = rtweedie(1,mu=ralphatil[j],phi=rphitil,xi=sp)
loss1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}}

##########Premliminary analysis###########
plot(loss1[1,],type = "l",lwd=1,ylim=c(0,max(loss1,na.rm=T)*1.1))
for(i in 2:10){
lines(loss1[i,],lwd=1)
}

plot(loss2[1,],type = "l",lwd=1,ylim=c(0,max(loss2,na.rm=T)*1.1))
for(i in 2:10){
lines(loss2[i,],lwd=1)
}

#GLM analysis
vloss1 = as.vector(t(loss1))
vloss2 = as.vector(t(loss2))

#Set up llh profile for each line
i = rep(1:10,each=10)
j = rep(1:10, 10)

ci.vec = seq(1,2,by=0.01)

llh1 = rep(0,length(ci.vec))
llh2 = llh1

for (t in 1:length(ci.vec)){
out1 <- glm(vloss1~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp1 <- summary(out1)$dispersion
mu1 <- fitted(out1)
den1 <- dtweedie(out1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2 <- glm(vloss2~as.factor(i) + as.factor(j), fam=tweedie(var.power=ci.vec[
    t]))
disp2 <- summary(out2)$dispersion
mu2 <- fitted(out2)
den2 <- dtweedie(out2$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2[t] <- sum(log(den2))
}

#Set up llh profile for both lines combined
i1 = c(rep(1:10,each=10),rep(11:20, each=10))
j1 = c(rep(1:10,10),rep(11:20,10))
cbine=c(vloss1,vloss2)
allh = rep(0, length(ci.vec))

for (t in 1:length(ci.vec)){
outa <- glm(cbine~as.factor(i1) + as.factor(j1), fam=tweedie(var.power=ci.vec
    [t]))
disp <- summary(outa)$dispersion
mu <- fitted(outa)
den <- dtweedie(outa$y, mu = mu, phi = disp, power = ci.vec[t])
allh[t] <- sum(log(den))
}

#Find p that maximises llh
ci1 = ci.vec[which.max(llh1)]
ci2 = ci.vec[which.max(llh2)]
aci = ci.vec[which.max(allh)]

#power parameter to be used later
```

```
p = aci
paglm <- glm(vloss1 ~ as.factor(i)+as.factor(j), family=tweedie(var.power=ci1
    ,link.power=0),control = list( epsilon=1e-09, trace=FALSE))
caglm <- glm(vloss2 ~ as.factor(i)+as.factor(j), family=tweedie(var.power=ci2
    ,link.power=0),control = list( epsilon=1e-09, trace=FALSE))

res1 = resid(paglm,"pearson")
res2 = resid(caglm,"pearson")

##########Marginal estimation##########

#Posterior function
llhfunc <- function(pars,data){
alpha1 <- exp(pars[1:9])
beta1 <-exp(pars[10:19])
alpha2<-exp(pars[20:28])
beta2<-exp(pars[29:38])
phi1 <- exp(pars[39])
phi2<-exp(pars[40])
lambda <- exp(pars[41])
p <- exp(pars[42])

alpha1.prior <- sum(dunif(log(alpha1), min=lower[1:9], max=upper[1:9], log=T)
    )
beta1.prior <- sum(dunif(log(beta1), min=lower[10:19], max=upper[10:19], log=
    T))
alpha2.prior <- sum(dunif(log(alpha2), min=lower[20:28], max=upper[20:28],
    log=T))
beta2.prior <- sum(dunif(log(beta2), min=lower[29:38], max=upper[29:38], log=
    T))

phi1.prior <- dunif(log(phi1),min=lower[39],max=upper[39],log=T)
phi2.prior <- dunif(log(phi2),min=lower[40],max=upper[40],log=T)
lambda.prior <- dunif(log(lambda),min=lower[41],max=upper[41],log=T)
p.prior <- dunif(log(p),min=lower[42],max=upper[42],log=T)

alphatilmean = rep(lambda,10)

alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

loss1<- data[1:10,]
loss2 <-data[11:20,]

meancommon1 = matrix(0,nrow=10,ncol=10)
scalecommon1 = matrix(0,nrow=10,ncol=10)
A1 = matrix(NA,nrow=10,ncol=10)
B1 = matrix(NA,nrow=10,ncol=10)

meancommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
A1 = meancommon1 + (alphaf1%o%beta1)
scalecommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
B1 = phi1*(scalecommon1 + 1)^(1-p)

meancommon2 = matrix(0,nrow=10,ncol=10)
scalecommon2 = matrix(0,nrow=10,ncol=10)
A2 = matrix(NA,nrow=10,ncol=10)
B2 = matrix(NA,nrow=10,ncol=10)

meancommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
A2 = meancommon2 + (alphaf2%o%beta2)
scalecommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
B2 = phi2*(scalecommon2 + 1)^(1-p)

vecloss1na = c(t(loss1))
vecloss1 = vecloss1na[!is.na(vecloss1na)]
vecA1na = c(t(A1))
```

```
vecA1 = vecA1na[!is.na(vecloss1na)]
vecB1na = c(t(B1))
vecB1 = vecB1na[!is.na(vecloss1na)]

vecloss2na = c(t(loss2))
vecloss2 = vecloss2na[!is.na(vecloss2na)]
vecA2na = c(t(A2))
vecA2 = vecA2na[!is.na(vecloss2na)]
vecB2na = c(t(B2))
vecB2 = vecB2na[!is.na(vecloss2na)]

if (!is.finite(p.prior)){LL = -100000000}
else{
LL = sum(log(dtweedie(vecloss1,xi=p,mu=vecA1,phi=vecB1)))+sum(log(dtweedie(
    vecloss2,xi=p, mu=vecA2, phi=vecB2)))}

LP <- LL + alpha1.prior+beta1.prior+alpha2.prior+beta2.prior+phi1.prior+phi2.
    prior+lambda.prior+p.prior

list(Monitor = c(LP, alpha1, beta1,alpha2,beta2,phi1,phi2,lambda,p),pars=pars
    ,LP=LP)
}

#Use adaptive MCMC to get estimates for covariance of proposal density
model1 <- function(pars,data){
LP = llhfunc(pars, data)$LP
return(LP)
}

data = rbind(loss1,loss2)
init <-  c(0.0296,  0.1740,  0.1133,  0.1398,  0.1484,  0.1133,  0.1310,
    0.1906,  0.1740,  4.0943,  2.9957,  2.3026,  1.6094, 0.9163,  0.2231,
    -0.5108, -1.2040, -1.8971, -1.8971,  0.1740, 0.1570,  0.1398,  0.1398,
    0.1823,  0.3365,  0.3716, 0.4447,  0.5068,  2.3026,  2.9957,  3.2189,
    2.9957,  2.7081,  2.0794,  1.0986,  0.6931,  0.0000,  0.0000, -0.1054,
    -0.3567,  0.0256,  0.2624)
lower = c(rep(-0.5,9),2.0943,  0.9957,  0.3026, -0.3906, -1.0837, -1.7769,
    -2.5108, -3.2040, -3.8971, -3.8971, rep(-0.5,9),0.3026,  0.9957,  1.2189,
     0.9957,  0.7081,  0.0794, -0.9014, -1.3069, -2.0000, -2.0000, -2.1054,
    -2.1054, -2.0000,  0.00001)
upper = c(rep(1.5,9),6.0943, 4.9957, 4.3026, 3.6094, 2.9163, 2.2231, 1.4892,
    0.7960, 0.1029, 0.1029,rep(1.5,9),4.3026, 4.9957, 5.2189, 4.9957, 4.7081,
    4.0794, 3.0986, 2.6931, 2.0000, 2.0000, 1.8946, 1.6433, 4.0000, log
    (1.999999999))
std = (upper-lower)*0.007

par_names=paste(c(rep("alpha1",9),rep("beta1",10),rep("alpha2",9),rep("beta2"
    ,5),"phi1","phi2","phitil","p"),c(seq(1:9),seq(1:10),seq(1:9),seq(1:5)
    ,0,0,0,0))
set.seed(1)
mh_test <- Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
    rbind(loss1,loss2),prop_sigma = diag(std),iterations = 60000,burn_in =
    20000)
set.seed(1)
mh_test1<-Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
    rbind(loss1,loss2),prop_sigma = mh_test$prop_sigma*0.001,iterations =
    45000,burn_in = 15000)

write.csv(mh_test$prop_sigma, file="cov_matrix.csv")

par(mfrow=c(3,2))
for(i in 1:42){plot(mh_test1$trace[,i],type="l")}

#Posterior function 2
model1 <- function(pars,data){
llh = llhfunc(pars,data)
list(Monitor = c(llh$LP, llh$alpha1, llh$beta1,llh$alpha2,llh$beta2,llh$phi1,
    llh$phi2,llh$lambda,llh$p),pars=llh$pars,LP=llh$LP)
}

#Run MCMC
```

```
std = as.matrix(read.csv("cov_matrix.csv",header=T))[,2:43]*0.22

Iterations=200000
Status=1000
Thinning=5
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

Acceptance <- 0
Mo0 <- model1(init, data)
Mon <- matrix(NA, nrow=Iterations, ncol=length(Mo0[["Monitor"]]), byrow=TRUE)
dimension <- length(init)
thinned <- matrix(NA, floor(Iterations/Thinning)+1,length(init))
thinned[1,] <- exp(init)

set.seed(11)
for (iter in 1:Iterations) {

if(iter %% Status == 0) cat("Iteration: ", iter, sep="")

#Metropolis algorithm
prop <- mvrnorm(1,mu = Mo0[["pars"]],Sigma=std)
Mo1 <- try(model1(prop, data), silent=TRUE)

log.u <- log(runif(1))
log.alphatil <- Mo1[["LP"]] - Mo0[["LP"]] + log(dmvnorm(Mo0[["pars"]], mean =
     prop, sigma=std)) - log(dmvnorm(prop, mean=Mo0[["pars"]], sigma=std))
if((is.finite(log.alphatil)) && (!inherits(Mo1, "try-error")) &&  ((is.finite
    (Mo1[["Monitor"]]))) && (log.u < log.alphatil)) {
Mo0 <- Mo1
Acceptance <- Acceptance + 1}

Mon[iter,] <- Mo0[["Monitor"]]

#Thin Samples
if(iter %% Thinning == 0) {
t.iter <- floor(iter / Thinning) + 1
thinned[t.iter,] <- Mo0[["pars"]]}

#Show tracking
if(iter %% Status == 0){
cat(", LP:", round(Mo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(Acceptance/iter, 2), "\n", sep = "", file =
    LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()
save.image()

#MCMC trace plots
par(mfrow=c(3,2))
for (i in 1:42) {
plot(thinned[2:40000,i],type="l")
}

#Summary statistics for marginal estimation
true=exp(init)
used = exp(thinned[20001:40000,])
median = round(apply((used),2,median,na.rm = T),3)
st.dev = apply(used,2,sd,na.rm=T)
lCI = rep(0,42)
uCI = rep(0,42)
for (i in 1:42){
lCI[i] = round(quantile(used[,i],p=0.025),3)
uCI[i] = round(quantile(used[,i],p=0.975),3)
}

bothline = cbind((true),median,(lCI),(uCI))
```

```
#compare common shock proportion
alpha1 <- median[1:9]
beta1 <- median[10:19]
alpha2<- median[20:28]
beta2<- median[29:38]
phi1 <-  median[39]
phi2<- median[40]
lambda <- median[41]
p<- median[42]
alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)
alphatilmean = rep(lambda,10)

ratio1 = matrix(NA,nrow=10,ncol=10)
ratio2 = matrix(NA,nrow=10,ncol=10)

trueratio1 = matrix(NA,nrow=10,ncol=10)
trueratio2 = matrix(NA,nrow=10,ncol=10)

for(i in 1:10){
for(j in 1:(10-i+1)){
ratio1[i,j] = (alphatilmean[j]*(phi1)/((alphaf1[i]*beta1[j])^(1-p)))/((
    alphatilmean[j]*(phi1)/((alphaf1[i]*beta1[j])^(1-p)))+alphaf1[i]*beta1[j])
    *100
ratio2[i,j] = (alphatilmean[j]*(phi2)/((alphaf2[i]*beta2[j])^(1-p)))/((
    alphatilmean[j]*(phi2)/((alphaf2[i]*beta2[j])^(1-p)))+alphaf2[i]*beta2[j])
    *100

trueratio1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil
    )*ralphatil[j]/((ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/
    rphitil)*ralphatil[j]+ralpha1[i]*rbeta1[j])*100
trueratio2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil
    )*ralphatil[j]/((ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/
    rphitil)*ralphatil[j]+ralpha2[i]*rbeta2[j])*100
}
}
```

A.2.2.2 Multivariate Tweedie model with treatment for unbalanced data

```
library(tweedie)
library(MASS)
library(statmod)
library(truncnorm)
library(tmvtnorm)
library(MHadaptive)

##########Simulation############
ralpha1 = c(1, 1.03, 1.19, 1.12, 1.15, 1.16, 1.12, 1.14, 1.21, 1.19)
ralpha2 = c(1, 1.19, 1.17, 1.15, 1.15, 1.20, 1.40, 1.45, 1.56, 1.66)

rbeta1 = c(60,20,10,5,2.5,1.25,0.6,0.3,0.15,0.15)
rbeta2 = c(10,20,25,20,15,8,3,2,1,1)

rphi1 = 0.9
rphi2 = 0.7

rzeta=0.5
ralphatil = c(sqrt(rbeta1[1:4]*rbeta2[1:4])*rzeta,sqrt(rbeta1[5:10]*rbeta2
    [5:10])*0.02)
rphitil = 0.6

sp=1.3

stau=(sp-2)/(sp-1)

#Generate loss triangles
set.seed(2)
random_w = matrix(NA, nrow=10,ncol=10)
```

```
loss1 = matrix(NA, nrow=10,ncol=10)
loss2 = matrix(NA, nrow=10, ncol=10)

for(i in 1:6){
for (j in 1:4){
random_w[i,j] = rtweedie(1,mu=ralphatil[j],phi=rphitil,xi=sp)
loss1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}
for (j in 5:(10-i+1)){
random_w[i,j] = rtweedie(1,mu=ralphatil[j],phi=rphitil,xi=sp)
loss1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}}

for(i in 7:10){
for (j in 1:(10-i+1)){
random_w[i,j] = rtweedie(1,mu=ralphatil[j],phi=rphitil,xi=sp)
loss1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha1[i]*rbeta1[j],phi = rphi1,xi=sp)
loss2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil)*
    random_w[i,j]+ rtweedie(1,mu=ralpha2[i]*rbeta2[j],phi = rphi2,xi=sp)
}}

##########Marginal estimation##########

#Posterior function 1
llhfunc <- function(pars,data){
alpha1 <- exp(pars[1:9])
beta1 <-exp(pars[10:19])
alpha2<-exp(pars[20:28])
beta2<-exp(pars[29:38])
phi1 <- exp(pars[39])
phi2<-exp(pars[40])
lambda <- exp(pars[41])
p <- exp(pars[42])

alpha1.prior <- sum(dunif(log(alpha1), min=lower[1:9], max=upper[1:9], log=T)
    )
beta1.prior <- sum(dunif(log(beta1), min=lower[10:19], max=upper[10:19], log=
    T))
alpha2.prior <- sum(dunif(log(alpha2), min=lower[20:28], max=upper[20:28],
    log=T))
beta2.prior <- sum(dunif(log(beta2), min=lower[29:38], max=upper[29:38], log=
    T))

phi1.prior <- dunif(log(phi1),min=lower[39],max=upper[39],log=T)
phi2.prior <- dunif(log(phi2),min=lower[40],max=upper[40],log=T)
lambda.prior <- dunif(log(lambda),min=lower[41],max=upper[41],log=T)
p.prior <- dunif(log(p),min=lower[42],max=upper[42],log=T)

alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

loss1<- data[1:10,]
loss2 <-data[11:20,]

meancommon1 = matrix(0,nrow=10,ncol=10)
scalecommon1 = matrix(0,nrow=10,ncol=10)
A1 = matrix(NA,nrow=10,ncol=10)
B1 = matrix(NA,nrow=10,ncol=10)

meancommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
```

```
A1 = meancommon1 + (alphaf1%o%beta1)
scalecommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
B1 = phi1*(scalecommon1 + 1)^(1-p)

meancommon2 = matrix(0,nrow=10,ncol=10)
scalecommon2 = matrix(0,nrow=10,ncol=10)
A2 = matrix(NA,nrow=10,ncol=10)
B2 = matrix(NA,nrow=10,ncol=10)

meancommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
A2 = meancommon2 + (alphaf2%o%beta2)
scalecommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
B2 = phi2*(scalecommon2 + 1)^(1-p)

vecloss1na = c(t(loss1))
vecloss1 = vecloss1na[!is.na(vecloss1na)]
vecA1na = c(t(A1))
vecA1 = vecA1na[!is.na(vecloss1na)]
vecB1na = c(t(B1))
vecB1 = vecB1na[!is.na(vecloss1na)]

vecloss2na = c(t(loss2))
vecloss2 = vecloss2na[!is.na(vecloss2na)]
vecA2na = c(t(A2))
vecA2 = vecA2na[!is.na(vecloss2na)]
vecB2na = c(t(B2))
vecB2 = vecB2na[!is.na(vecloss2na)]

if (!is.finite(p.prior)){LL = -100000000}
else{
LL = sum(log(dtweedie(vecloss1,xi=p,mu=vecA1,phi=vecB1)))+sum(log(dtweedie(
    vecloss2,xi=p, mu=vecA2, phi=vecB2)))}

LP <- LL + alpha1.prior+beta1.prior+alpha2.prior+beta2.prior+phi1.prior+phi2.
    prior+lambda.prior+p.prior

list(Monitor = c(LP, alpha1, beta1,alpha2,beta2,phi1,phi2,lambda,p),pars=pars
    ,LP=LP)
}

#Use adaptive MCMC to get estimates for covariance of proposal density
model1 <- function(pars,data){
LP = llhfunc(pars, data)$LP
return(LP)
}

data = rbind(loss1,loss2)
init <-  c(0.0296,  0.1740,  0.1133,  0.1398,  0.1484,  0.1133,  0.1310,
    0.1906,  0.1740,  4.0943,  2.9957,  2.3026,  1.6094, 0.9163,  0.2231,
    -0.5108, -1.2040, -1.8971, -1.8971,  0.1740, 0.1570,  0.1398,  0.1398,
    0.1823,  0.3365,  0.3716, 0.4447,  0.5068,  2.3026,  2.9957,  3.2189,
    2.9957,  2.7081,  2.0794,  1.0986,  0.6931,  0.0000,  0.0000, -0.1054,
    -0.3567,  0.0256,  0.2624)
lower = c(rep(-0.5,9),2.0943,  0.9957,  0.3026, -0.3906, -1.0837, -1.7769,
    -2.5108, -3.2040, -3.8971, -3.8971, rep(-0.5,9),0.3026,  0.9957,  1.2189,
     0.9957,  0.7081,  0.0794, -0.9014, -1.3069, -2.0000, -2.0000, -2.1054,
    -2.1054, -2.0000,  0.00001)
upper = c(rep(1.5,9),6.0943, 4.9957, 4.3026, 3.6094, 2.9163, 2.2231, 1.4892,
    0.7960, 0.1029, 0.1029,rep(1.5,9),4.3026, 4.9957, 5.2189, 4.9957, 4.7081,
    4.0794, 3.0986, 2.6931, 2.0000, 2.0000, 1.8946, 1.6433, 4.0000, log
    (1.999999999))
std = (upper-lower)*0.01

par_names=paste(c(rep("alpha1",9),rep("beta1",10),rep("alpha2",9),rep("beta2"
    ,5),"phi1","phi2","phitil","p"),c(seq(1:9),seq(1:10),seq(1:9),seq(1:5)
    ,0,0,0,0))
set.seed(1)
mh_test <- Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
```

```
    rbind(loss1,loss2),prop_sigma = diag(std),iterations = 50000)
set.seed(1)
mh_test1<-Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
    rbind(loss1,loss2),prop_sigma = mh_test$prop_sigma*0.001,iterations =
    60000,burn_in = 10000)

write.csv(mh_test1$prop_sigma, file="cov_matrix.csv")

par(mfrow=c(3,2))
for(i in 1:42){plot(mh_test1$trace[,i],type="l")}

#Posterior function 2
model1 <- function(pars,data){
llh = llhfunc(pars,data)
list(Monitor = c(llh$LP, llh$alpha1, llh$beta1,llh$alpha2,llh$beta2,llh$phi1,
    llh$phi2,llh$lambda,llh$p),pars=llh$pars,LP=llh$LP)
}

#Run MCMC
std = as.matrix(read.csv("cov_matrix.csv",header=T))[,2:43]*0.25

Iterations=200000
Status=1000
Thinning=5
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

Acceptance <- 0
Mo0 <- model1(init, data)
Mon <- matrix(NA, nrow=Iterations, ncol=length(Mo0[["Monitor"]]), byrow=TRUE)
dimension <- length(init)
thinned <- matrix(NA, floor(Iterations/Thinning)+1,length(init))
thinned[1,] <- exp(init)

set.seed(11)
for (iter in 1:Iterations) {

if(iter %% Status == 0) cat("Iteration: ", iter, sep="")

#Metropolis algorithm
prop <- mvrnorm(1,mu = Mo0[["pars"]],Sigma=std)
Mo1 <- try(model1(prop, data), silent=TRUE)

log.u <- log(runif(1))
log.alphatiltil <- Mo1[["LP"]] - Mo0[["LP"]] + log(dmvnorm(Mo0[["pars"]],
    mean = prop, sigma=std)) - log(dmvnorm(prop, mean=Mo0[["pars"]], sigma=std
    ))
if((is.finite(log.alphatiltil)) && (!inherits(Mo1, "try-error")) &&  ((is.
    finite(Mo1[["Monitor"]]))) && (log.u < log.alphatiltil)) {
Mo0 <- Mo1
Acceptance <- Acceptance + 1}

Mon[iter,] <- Mo0[["Monitor"]]

#Thin Samples
if(iter %% Thinning == 0) {
t.iter <- floor(iter / Thinning) + 1
thinned[t.iter,] <- Mo0[["pars"]]}

#Show tracking
if(iter %% Status == 0){
cat(", LP:", round(Mo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(Acceptance/iter, 2), "\n", sep = "", file =
    LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()
save.image()
```

```
#MCMC trace plots
par(mfrow=c(3,2))
for (i in 1:42) {
plot(thinned[2:40000,i],type="l")
}

#Summary statistics for marginal estimation
true=exp(init)
used = exp(thinned[20001:40000,])
median = round(apply((used),2,median,na.rm = T),3)
st.dev = apply(used,2,sd,na.rm=T)
lCI = rep(0,42)
uCI = rep(0,42)
for (i in 1:42){
lCI[i] = round(quantile(used[,i],p=0.025),3)
uCI[i] = round(quantile(used[,i],p=0.975),3)
}

bothline = cbind((true),median,(lCI),(uCI))

#compare common shock proportion
alpha1 <-  (median[1:9])
beta1 <- (median[10:19])
alpha2<- (median[20:28])
beta2<- (median[29:38])
phi1 <-  (median[39])
phi2<- (median[40])
lambda <- median[41]
p<- median[42]
alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)
alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

ratio1 = matrix(NA,nrow=10,ncol=10)
ratio2 = matrix(NA,nrow=10,ncol=10)

trueratio1 = matrix(NA,nrow=10,ncol=10)
trueratio2 = matrix(NA,nrow=10,ncol=10)

for(i in 1:10){
for(j in 1:(10-i+1)){
ratio1[i,j] = (alphatilmean[j]*(phi1)/((alphaf1[i]*beta1[j])^(1-p)))/((
    alphatilmean[j]*(phi1)/((alphaf1[i]*beta1[j])^(1-p)))+alphaf1[i]*beta1[j])
    *100
ratio2[i,j] = (alphatilmean[j]*(phi2)/((alphaf2[i]*beta2[j])^(1-p)))/((
    alphatilmean[j]*(phi2)/((alphaf2[i]*beta2[j])^(1-p)))+alphaf2[i]*beta2[j])
    *100

trueratio1[i,j] = (ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/rphitil
    )*ralphatil[j]/((ralphatil[j]/(ralpha1[i]*rbeta1[j]))^(1-sp)*(rphi1/
    rphitil)*ralphatil[j]+ralpha1[i]*rbeta1[j])*100
trueratio2[i,j] = (ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/rphitil
    )*ralphatil[j]/((ralphatil[j]/(ralpha2[i]*rbeta2[j]))^(1-sp)*(rphi2/
    rphitil)*ralphatil[j]+ralpha2[i]*rbeta2[j])*100
}
}
```

A.2.3 Real data illustration

```
library(tweedie)
library(MASS)
library(statmod)
library(truncnorm)
library(rootSolve)
library(LaplacesDemon)
library(xtable)
library(ChainLadder)
```

```
library(tmvtnorm)
library(MHadaptive)

##########Import data##########
dloss1 = as.matrix(read.csv("line1.csv",header=TRUE))
dloss2 = as.matrix(read.csv("line2.csv",header=TRUE))

loss1=matrix(NA,nrow=10,ncol=10)
loss2=matrix(NA,nrow=10,ncol=10)

for (i in 1:10){
for(j in 1:(10-i+1)){

loss1[i,j] = dloss1[i,j+1]
loss2[i,j] = dloss2[i,j+1]
}
}

prem1 = c(85421,98579,103062,108412,111176,112050,112577,113707,126442,
    130484)
prem2 = c(116491,111467,107241,105687,105923,111487,113268,21606,110610,
    104304)

##########Preliminary analysis - plot development##########

#Plot development trends
par(mar=c(4,4,1,1))
axticks <- seq(1, 10, 1)
axuse <-seq(1,10,1)

plot(loss1[1,],type = "l",lwd=1,lty=1,ylim=c(0,max(loss1,na.rm=T)*1.1),xlab="
    Development year",xaxt="n", ylab="Loss ratio")
points(loss1[1,], pch=1)
lines(loss2[1,],type = "l",lty=2)
points(loss2[1,], pch=2)

axis(side=1,at=axuse, labels=axticks)
legend("topright", legend = c( "Bodily Injury", "Accident Benefits"),
lty = 1:2,lwd=1, bty = "n",pch=1:2,
title = "")

##########Preliminary analysis - analyse dependence using GLM (without
    calendar year factor)##########
vloss1 = as.vector(t(loss1))
vloss2 = as.vector(t(loss2))

vloss1 = vloss1[!is.na(vloss1)]
vloss2 = vloss2[!is.na(vloss2)]

#Set up llh profile to find p
i = rep(1:10,10:1)
j <- sequence(10:1)

ci.vec = seq(1.01,1.99,by=0.01)

llh1 = rep(0,length(ci.vec))
llh2 = llh1

for (t in 1:length(ci.vec)){
out1 <- glm(vloss1~as.factor(j) + as.factor(i)-1, fam=tweedie(var.power=ci.
    vec[t]))
disp1 <- summary(out1)$dispersion
mu1 <- fitted(out1)
den1 <- dtweedie(out1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2 <- glm(vloss2~as.factor(j) + as.factor(i)-1, fam=tweedie(var.power=ci.
    vec[t]))
disp2 <- summary(out2)$dispersion
```

```
mu2 <- fitted(out2)
den2 <- dtweedie(out2$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2[t] <- sum(log(den2))
}

i1 = c(i,i+10)
j1 = c(j,j+10)
cbine=c(vloss1,vloss2)
allh = rep(0, length(ci.vec))

for (t in 1:length(ci.vec)){
outa <- glm(cbine~as.factor(j1) + as.factor(i1)-1, fam=tweedie(var.power=ci.
    vec[t]))
disp <- summary(outa)$dispersion
mu <- fitted(outa)
den <- dtweedie(outa$y, mu = mu, phi = disp, power = ci.vec[t])
allh[t] <- sum(log(den))
}

ci1 = ci.vec[which.max(llh1)]
ci2 = ci.vec[which.max(llh2)]
aci = ci.vec[which.max(allh)]
p=aci

#CI for p
outci1 = rep(0,length(llh1))
for (k in 1:length(llh1)){
if (isTRUE(2*abs(llh1[k]-max(llh1)) <= 3.84))  {
outci1[k] = ci.vec[k]
}
else{
outci1[k] = 0
}
}

ci1low = outci1[2]
ci1hi = outci1[36]

outci2 = rep(0,length(llh2))
for (k in 1:length(llh2)){
if (isTRUE(2*abs(llh2[k]-max(llh2)) <= 3.84))  {
outci2[k] = ci.vec[k]
}
else{
outci2[k] = 0
}
}

ci2low = outci2[5]
ci2hi = outci2[41]

outaci = rep(0,length(allh))
for (k in 1:length(allh)){
if (isTRUE(2*abs(allh[k]-max(allh)) <= 3.84))  {
outaci[k] = ci.vec[k]
}
else{
outaci[k] = 0
}
}

acilow = outaci[8]
acihi = outaci[27]

##Ln L profile plot
plot(ci.vec,llh1, type="l", xlim = c(1,2), xlab="p", ylab="logL")
points(ci1, max(llh1), pch=15, cex=1)
points(aci, llh1[which.max(allh)], pch=18, cex=1)
abline(v=ci1low)
abline(v=ci1hi)
```

```
plot(ci.vec,llh2, type="l", xlim = c(1,2), xlab="p", ylab="logL")
points(ci2, max(llh2), pch=15, cex=1)
points(aci, llh2[which.max(allh)], pch=18, cex=1)
abline(v=ci2low)
abline(v=ci2hi)

plot(ci.vec,allh, type="l", xlim = c(1,2), xlab="p", ylab="logL")
points(aci, max(allh), pch=15, cex=1)
abline(v=acilow)
abline(v=acihi)

#GLM fitting with the best p
paglm <- glm(vloss1 ~ as.factor(j)+as.factor(i), family=tweedie(var.power=ci1
    ,link.power=0),control = list( epsilon=1e-09, trace=FALSE))
caglm <- glm(vloss2 ~ as.factor(j)+as.factor(i), family=tweedie(var.power=ci2
    ,link.power=0),control = list( epsilon=1e-09, trace=FALSE))
allglm <-glm(cbine ~ factor(i1)+factor(j1)-1, family=tweedie(var.power=aci,
    link.power=0),control = list( epsilon=1e-09, trace=FALSE))

#QQ plots
qqnorm(residuals(paglm, type="pearson"))
qqline(residuals(paglm, type="pearson"))

qqnorm(residuals(caglm, type="pearson"))
qqline(residuals(caglm, type="pearson"))

qqnorm(residuals(allglm, type="pearson"))
qqline(residuals(allglm, type="pearson"))

#Residuals analysis
res1 = resid(paglm,"pearson")
res2 = resid(caglm,"pearson")

peartest = cor.test(res1,res2,method=c("pearson"), conf.level=0.95)
speatest = cor.test(res1,res2,method=c("spearman"), conf.level=0.95)
kendtest = cor.test(res1,res2,method=c("kendall"), conf.level=0.95)
cort = rbind(c(peartest$estimate,speatest$estimate,kendtest$estimate),c(
    peartest$p.value,speatest$p.value,kendtest$p.value))

##########Preliminary analysis - analyse dependence using GLM (without
    calendar year factor)##########

#Set up llh profile to find p
k <- c(seq(1,10),seq(2,10),seq(3,10),seq(4,10), seq(5,10), seq(6,10), seq
    (7,10), seq(8,10), seq(9,10), 10)

llh1ca = rep(0,length(ci.vec))
llh2ca = llh1ca

for (t in 1:length(ci.vec)){
outpa2 <- glm(vloss1~as.factor(i) + as.factor(j) + as.factor(k), fam=tweedie(
    var.power=ci.vec[t]))
disp12 <- summary(outpa2)$dispersion
mu12 <- fitted(outpa2)
den12 <- dtweedie(outpa2$y, mu = mu12, phi = disp12, power = ci.vec[t])
llh1ca[t] <- sum(log(den12))
}


for (t in 1:length(ci.vec)){
outca2 <- glm(vloss2~as.factor(i) + as.factor(j)+ as.factor(k), fam=tweedie(
    var.power=ci.vec[t]))
disp22 <- summary(outca2)$dispersion
mu22 <- fitted(outca2)
den22 <- dtweedie(outca2$y, mu = mu22, phi = disp22, power = ci.vec[t])
llh2ca[t] <- sum(log(den22))
}

ci1ca = ci.vec[which.max(llh1ca)]
ci2ca = ci.vec[which.max(llh2ca)]
```

```
#GLM fitting with the best p
paglm2 <- glm(vloss1 ~ as.factor(i)+as.factor(j)+ as.factor(k), family=
    tweedie(var.power=ci1ca,link.power=0),control = list( epsilon=1e-09, trace
    =FALSE))
caglm2 <- glm(vloss2 ~ as.factor(i)+as.factor(j)+ as.factor(k), family=
    tweedie(var.power=ci2ca,link.power=0),control = list( epsilon=1e-09, trace
    =FALSE))

#Residual analysis
res_pa2 = resid(paglm2,"pearson")
res_ca2 = resid(caglm2,"pearson")

peartest2 = cor.test(res_pa2,res_ca2,method=c("pearson"), conf.level=0.95)
speatest2 = cor.test(res_pa2,res_ca2,method=c("spearman"), conf.level=0.95)
kendtest2 = cor.test(res_pa2,res_ca2,method=c("kendall"), conf.level=0.95)
cort2 = rbind(c(peartest2$estimate,speatest2$estimate,kendtest2$estimate),c(
    peartest2$p.value,speatest2$p.value,kendtest2$p.value))

#Heat maps of residuals - export to csv file
Var1 = j
Var2 = rep(10:1,10:1)
fitpa2 = vloss1/fitted(paglm2)
fitca2 = vloss2/fitted(caglm2)

ratcalpa <- data.frame(Var1,Var2,fitpa2)
ratcalca <- data.frame(Var1,Var2,fitca2)

#write.csv(ratcalpa,"respaglm.csv")
#write.csv(ratcalca,"rescaglm.csv")

##########Choose initial values and get information for prior distributions
    selection##########

#Estimate alpha and beta
solvefn <- function(x,data){
F2 <- x[1]*sum(x[10:18]^(2-aci)) - as.numeric(crossprod(data[2,1:9],x
    [10:18]^(1-aci)))
F3 <- x[2]*sum(x[10:17]^(2-aci)) - as.numeric(crossprod(data[3,1:8],x
    [10:17]^(1-aci)))
F4 <- x[3]*sum(x[10:16]^(2-aci)) - as.numeric(crossprod(data[4,1:7],x
    [10:16]^(1-aci)))
F5 <- x[4]*sum(x[10:15]^(2-aci)) - as.numeric(crossprod(data[5,1:6],x
    [10:15]^(1-aci)))
F6 <- x[5]*sum(x[10:14]^(2-aci)) - as.numeric(crossprod(data[6,1:5],x
    [10:14]^(1-aci)))
F7 <- x[6]*sum(x[10:13]^(2-aci)) - as.numeric(crossprod(data[7,1:4],x
    [10:13]^(1-aci)))
F8 <- x[7]*sum(x[10:12]^(2-aci)) - as.numeric(crossprod(data[8,1:3],x
    [10:12]^(1-aci)))
F9 <- x[8]*sum(x[10:11]^(2-aci)) - as.numeric(crossprod(data[9,1:2],x
    [10:11]^(1-aci)))
F10 <- x[9]*sum(x[10:10]^(2-aci)) - as.numeric(crossprod(data[10,1],x
    [10:10]^(1-aci)))

F11 <- x[10]*sum(c(1,x[1:9])^(2-aci)) - as.numeric(crossprod(data[,1],c(1,x
    [1:9])^(1-aci)))
F12 <- x[11]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:9,2],c(1,
    x[1:8])^(1-aci)))
F13 <- x[12]*sum(c(1,x[1:7])^(2-aci)) - as.numeric(crossprod(data[1:8,3],c(1,
    x[1:7])^(1-aci)))
F14 <- x[13]*sum(c(1,x[1:8])^(2-aci)) - as.numeric(crossprod(data[1:7,4],c(1,
    x[1:6])^(1-aci)))
F15 <- x[14]*sum(c(1,x[1:5])^(2-aci)) - as.numeric(crossprod(data[1:6,5],c(1,
    x[1:5])^(1-aci)))
F16 <- x[15]*sum(c(1,x[1:4])^(2-aci)) - as.numeric(crossprod(data[1:5,6],c(1,
    x[1:4])^(1-aci)))
F17 <- x[16]*sum(c(1,x[1:3])^(2-aci)) - as.numeric(crossprod(data[1:4,7],c(1,
    x[1:3])^(1-aci)))
F18 <- x[17]*sum(c(1,x[1:2])^(2-aci)) - as.numeric(crossprod(data[1:3,8],c(1,
    x[1:2])^(1-aci)))
F19 <- x[18]*sum(c(1,x[1:1])^(2-aci)) - as.numeric(crossprod(data[1:2,9],c(1,
```

```
    x[1:1])^(1-aci)))
F20 <-x[19]*sum(1^(2-aci)) - as.numeric(crossprod(data[1,10],1^(1-aci)))

c(F2=F2,F3=F3,F4=F4,F5=F5,F6=F6,F7=F7,F8=F8,F9=F9,F10=F10,
F11=F11,F12=F12,F13=F13,F14=F14,F15=F15,F16=F16,F17=F17,F18=F18,F19=F19,F20=
    F20)
}

ss1 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(loss1,na.rm=T)),
    data=loss1)
ss2 <- multiroot(f = solvefn, start = c(rep(1,9),colMeans(loss2,na.rm=T)),
    data=loss2)

#Estimate phi1 and phi2
alphaest1 = c(1,ss1$root[1:9])
betaest1 = ss1$root[10:19]
resid1= matrix(NA,nrow=10,ncol=10)

for (i in 1:10){
for(j in 1:(10-i+1)){
resid1[i,j] = (loss1[i,j]-alphaest1[i]*betaest1[j])^2/((alphaest1[i]*betaest1
    [j])^aci)
}
}
phi1est = sum(resid1,na.rm=T)/45

alphaest2 = c(1,ss2$root[1:9])
betaest2 = ss2$root[10:19]
resid2= matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for(j in 1:(10-i+1)){
resid2[i,j] = (loss2[i,j]-alphaest2[i]*betaest2[j])^2/((alphaest2[i]*betaest2
    [j])^aci)
}
}
phi2est = sum(resid2,na.rm=T)/45

#Estimate alpha and beta
betatilest = 0.24
alphatilest=0.0005
minyest=matrix(NA,nrow=10,ncol=10)
for (i in 1:10){
for (j in 1:(10-i+1)){
minyest[i,j] <- min((((alphatilest/(alphaest1[i]*betaest1[j]))^(1-aci)*(
    phi1est/betatilest)))*alphatilest,(((alphatilest/(alphaest2[i]*betaest2[j
    ]))^(1-aci)*(phi2est/betatilest)))*alphatilest)
}}

############Marginal estimatio############

#Posterior function
llhfunc <- function(pars,data){
alpha1 <- exp(pars[1:9])
beta1 <-exp(pars[10:19])
alpha2<-exp(pars[20:28])
beta2<-exp(pars[29:38])
phi1 <- exp(pars[39])
phi2<-exp(pars[40])
lambda <- exp(pars[41])
p <- exp(pars[42])

alpha1.prior <- sum(dunif(log(alpha1), min=lower[1:9], max=upper[1:9], log=T)
    )
beta1.prior <- sum(dunif(log(beta1), min=lower[10:19], max=upper[10:19], log=
    T))
alpha2.prior <- sum(dunif(log(alpha2), min=lower[20:28], max=upper[20:28],
    log=T))
beta2.prior <- sum(dunif(log(beta2), min=lower[29:38], max=upper[29:38], log=
    T))

phi1.prior <- dunif(log(phi1),min=lower[39],max=upper[39],log=T)
```

```
phi2.prior <- dunif(log(phi2),min=lower[40],max=upper[40],log=T)
lambda.prior <- dunif(log(lambda),min=lower[41],max=upper[41],log=T)
p.prior <- dunif(log(p),min=lower[42],max=upper[42],log=T)

alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

loss1<- data[1:10,]
loss2 <-data[11:20,]

meancommon1 = matrix(0,nrow=10,ncol=10)
scalecommon1 = matrix(0,nrow=10,ncol=10)
A1 = matrix(NA,nrow=10,ncol=10)
B1 = matrix(NA,nrow=10,ncol=10)

meancommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
A1 = meancommon1 + (alphaf1%o%beta1)
scalecommon1[1:10,1:10] = ((alphaf1%o%beta1[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi1,10),nrow=10,byrow=T)
B1 = phi1*(scalecommon1 + 1)^(1-p)

meancommon2 = matrix(0,nrow=10,ncol=10)
scalecommon2 = matrix(0,nrow=10,ncol=10)
A2 = matrix(NA,nrow=10,ncol=10)
B2 = matrix(NA,nrow=10,ncol=10)

meancommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-1))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
A2 = meancommon2 + (alphaf2%o%beta2)
scalecommon2[1:10,1:10] = ((alphaf2%o%beta2[1:10])^(p-2))*matrix(rep(
    alphatilmean*phi2,10),nrow=10,byrow=T)
B2 = phi2*(scalecommon2 + 1)^(1-p)

vecloss1na = c(t(loss1))
vecloss1 = vecloss1na[!is.na(vecloss1na)]
vecA1na = c(t(A1))
vecA1 = vecA1na[!is.na(vecloss1na)]
vecB1na = c(t(B1))
vecB1 = vecB1na[!is.na(vecloss1na)]

vecloss2na = c(t(loss2))
vecloss2 = vecloss2na[!is.na(vecloss2na)]
vecA2na = c(t(A2))
vecA2 = vecA2na[!is.na(vecloss2na)]
vecB2na = c(t(B2))
vecB2 = vecB2na[!is.na(vecloss2na)]

if (!is.finite(p.prior)){LL = -100000000}
else{
LL = sum(log(dtweedie(vecloss1,xi=p,mu=vecA1,phi=vecB1)))+sum(log(dtweedie(
    vecloss2,xi=p, mu=vecA2, phi=vecB2)))}

LP <- LL + alpha1.prior+beta1.prior+alpha2.prior+beta2.prior+phi1.prior+phi2.
    prior+lambda.prior+p.prior

list(Monitor = c(LP, alpha1, beta1,alpha2,beta2,phi1,phi2,lambda,p),pars=pars
    ,LP=LP)
}

#Use adaptive MCMC to get estimates for covariance of proposal density
model1 <- function(pars,data){
LP = llhfunc(pars, data)$LP
return(LP)
}

data = rbind(loss1,loss2)
lower = c(log(ss1$root)[1:9]-1.2,log(ss1$root)[10:19]-1,log(ss2$root)
    [1:9]-1.2,log(ss2$root)[10:19]-1.5,rep(-5,3),log(1.1))
```

```
upper = c(log(ss1$root)[1:9]+1.5,log(ss1$root)[10:19]+1,log(ss2$root)
    [1:9]+1.5,log(ss2$root)[10:19]+1.2,0,1,0,log(1.95))
init=c((lower[1:38]+upper[1:38])/2,-1,-1,0,log(1.2))
std = (upper-lower)/50

par_names=paste(c(rep("alpha1",9),rep("beta1",10),rep("alpha2",9),rep("beta2"
    ,5),"phi1","phi2","phitil","p"),c(seq(1:9),seq(1:10),seq(1:9),seq(1:5)
    ,0,0,0,0))
set.seed(1)
mh_test <- Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
    rbind(loss1,loss2),prop_sigma = diag(std),iterations = 60000,burn_in =
    20000)
set.seed(1)
mh_test1<-Metro_Hastings(li_func=model1, pars=init,par_names=list(),data=
    rbind(loss1,loss2),prop_sigma = mh_test$prop_sigma*0.00001,iterations =
    70000,burn_in = 40000)

write.csv(mh_test1$prop_sigma, file="cov_matrix.csv")

par(mfrow=c(3,2))
for(i in 1:42){plot(mh_test1$trace[,i],type="l")}

#Posterior function
model1 <- function(pars,data){
llh = llhfunc(pars,data)
list(Monitor = c(llh$LP, llh$alpha1, llh$beta1,llh$alpha2,llh$beta2,llh$phi1,
    llh$phi2,llh$lambda,llh$p),pars=llh$pars,LP=llh$LP)
}

#Run MCMC
lower = c(log(ss1$root)[1:9]-1.2,log(ss1$root)[10:19]-1,log(ss2$root)
    [1:9]-1.2,log(ss2$root)[10:19]-1.5,rep(-5,3),log(1.1))
upper = c(log(ss1$root)[1:9]+1.5,log(ss1$root)[10:19]+1,log(ss2$root)
    [1:9]+1.5,log(ss2$root)[10:19]+1.2,0,1,0,log(1.95))
init=c((lower[1:38]+upper[1:38])/2,-1,-1,-1.5,log(1.2))
std = as.matrix(read.csv("cov_matrix.csv",header=T))[,2:43]*0.15

Iterations=400000
Status=1000
Thinning=5
LogFile=""

cat("MCMC started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

Acceptance <- 0
Mo0 <- model1(init, data)
Mon <- matrix(NA, nrow=Iterations, ncol=length(Mo0[["Monitor"]]), byrow=TRUE)
dimension <- length(init)
thinned <- matrix(NA, floor(Iterations/Thinning)+1,length(init))
thinned[1,] <- exp(init)

set.seed(11)
for (iter in 1:Iterations) {

if(iter %% Status == 0) cat("Iteration: ", iter, sep="")

#Metropolis algorithm
prop <- mvrnorm(1,mu = Mo0[["pars"]],Sigma=std)
Mo1 <- try(model1(prop, data), silent=TRUE)

log.u <- log(runif(1))
log.alphatiltil <- Mo1[["LP"]] - Mo0[["LP"]] + log(dmvnorm(Mo0[["pars"]],
    mean = prop, sigma=std)) - log(dmvnorm(prop, mean=Mo0[["pars"]], sigma=std
    ))
if((is.finite(log.alphatiltil)) && (!inherits(Mo1, "try-error")) &&  ((is.
    finite(Mo1[["Monitor"]])))) && (log.u < log.alphatiltil)) {
Mo0 <- Mo1
Acceptance <- Acceptance + 1}
```

```
Mon[iter,] <- Mo0[["Monitor"]]

#Thin Samples
if(iter %% Thinning == 0) {
t.iter <- floor(iter / Thinning) + 1
thinned[t.iter,] <- Mo0[["pars"]]}

#Show tracking
if(iter %% Status == 0){
cat(", LP:", round(Mo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(Acceptance/iter, 2), "\n", sep = "", file =
    LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()
save.image()

#MCMC trace plots
par(mfrow=c(3,2))
for (i in 1:42) {
plot(thinned[2:t.iter,i],type="l")
}

#Summary statistics for marginal estimation
used = exp(thinned[60001:80000,])
median = round(apply((used),2,median,na.rm = T),3)
st.dev = apply(used,2,sd,na.rm=T)
lCI = rep(0,42)
uCI = rep(0,42)
for (i in 1:42){
lCI[i] = round(quantile(used[,i],p=0.05),3)
uCI[i] = round(quantile(used[,i],p=0.95),3)
}

bothline = cbind(median,st.dev,(lCI),(uCI))

##########Multivariate estimation (adaptive Metropolis)############

#Use parameter estimates from marginal estimation
alpha1 <-  median[1:9]
beta1 <- median[10:19]
alpha2<- median[20:28]
beta2<- median[29:38]
phi1 <-  median[39]
phi2<- median[40]
lambda <- median[41]
p<- median[42]
alphaf1<-c(1,alpha1)
alphaf2<-c(1,alpha2)

#Posterior function
model2 <- function(pars,data){
zeta <- exp(pars)
phitil<- zeta^(2-p)/lambda

alphatilmean = (sqrt(beta1*beta2))^(2-p)*zeta
zeta.prior <- dunif(log(zeta),min=-3,max=1,log=T)

l1 <- matrix(NA, nrow=10, ncol=10)
miny <- matrix(NA, nrow=10, ncol=10)
loss1<- data[1:10,]
loss2<- data[11:20,]

if(all(is.finite(zeta.prior))){
for (i in 1:10){
for (j in 1:(10-i+1)){
miny[i,j] <- min((loss1[i,j]/((alphatilmean[j]/(alphaf1[i]*beta1[j]))^(1-p)*(
    phi1/phitil))), (loss2[i,j]/((alphatilmean[j]/(alphaf2[i]*beta2[j]))^(1-p)
    *(phi2/phitil))))
if (miny[i,j]<=1e-200000000){
```

```
l1[i,j]=log(dtweedie(loss1[i,j],xi=p, mu=alphaf1[i]*beta1[j], phi = phi1))+
    log(dtweedie(loss2[i,j],xi=p, mu=alphaf2[i]*beta2[j], phi = phi2))
}
else {
f<- function(z){
dtweedie(loss1[i,j]-(alphatilmean[j]/(alphaf1[i]*beta1[j]))^(1-p)*(phi1/
    phitil)*z,xi=p, mu=alphaf1[i]*beta1[j], phi = phi1)*dtweedie(loss2[i,j]-(
    alphatilmean[j]/(alphaf2[i]*beta2[j]))^(1-p)*(phi2/phitil)*z,xi=p, mu=
    alphaf2[i]*beta2[j], phi = phi2)*dtweedie(z,xi=p, mu=alphatilmean[j], phi
    = phitil)
}
llh<- try(integrate(f,lower=1e-2000000000, upper=miny[i,j]),silent=T)
if(inherits(llh, 'try-error')){
l1[i,j] <- log(0)
}
else{
l1[i,j] <- log(llh$value)}}
}}}

LL <- sum(l1, na.rm=T)
LP <- LL +  zeta.prior

list(LP = LP, Monitor = c(LP, zeta,phitil), pars=pars)
}

#Prepare for MCMC
ainit <-  log(1)

Iterations=90000
Status=1000
Thinning=3
alpha.star=0.44
Periodicity=1
LogFile=""

cat("MCMC started on ", date(), "\n", sep="")
time1 <- proc.time()

aAcceptance <- 0
aMo0 <- model2(ainit, data)
aMon <- matrix(NA,nrow=Iterations,ncol=length(aMo0[["Monitor"]]))
adimension <- length(ainit)
athinned <- matrix(NA, floor(Iterations/Thinning)+1,2)
athinned[1,] <- aMo0[["Monitor"]][2:3]
aScaleF <- 0.0001/sqrt(adimension)
aVarCov <- matrix(0, adimension, adimension)
diag(aVarCov) <- rep(aScaleF, adimension)
aS <- t(chol(aVarCov))

set.seed(11)
for (aiter in 1:Iterations) {
if(aiter %% Status == 0) cat("Iteration: ", aiter, sep="")

#Adaptive Metropolis
aU <- rnorm(adimension)
aprop <- as.vector(aMo0[["pars"]] + aS %*% aU)
aMo1 <- try(model2(aprop, data), silent=TRUE)

alog.u <- log(runif(1))
alog.alpha <- aMo1[["LP"]] - aMo0[["LP"]]
if((is.finite(alog.alpha)) && (!inherits(aMo1, "try-error")) &&  ((is.finite(
    aMo1[["Monitor"]]))) && (alog.u < alog.alpha)) {
aMo0 <- aMo1
aAcceptance <- aAcceptance + 1}

aMon[aiter,] <- aMo0[["Monitor"]]

if({aiter >= 2} & {aiter %% Periodicity == 0}) {
aeta <- min(1, adimension*aiter^(-2/3))
aVarCov.test <- aS %*% (diag(adimension) + aeta*(min(1, exp(alog.alpha)) -
    alpha.star) * aU %*% t(aU) / sqrt(sum(aU^2))) %*% t(aS)
```

```
if(!all(is.finite(aVarCov.test))) {aVarCov.test <- aVarCov}
if(!is.symmetric.matrix(aVarCov.test)){aVarCov.test <- as.symmetric.matrix(
    aVarCov.test)}
if(is.positive.definite(aVarCov.test)){ aS.z <- try(t(chol(aVarCov)), silent=
    TRUE)
if(!inherits(aS.z, "try-error")) {
aVarCov <- aVarCov.test
aS <- aS.z}}}

#Thin samples
if(aiter %% Thinning == 0) {
at.iter <- floor(aiter / Thinning) + 1
athinned[at.iter,] <- aMo0[["Monitor"]][2:3]}

if(aiter %% Status == 0){
cat(", LP:", round(aMo0[["LP"]], 2), sep = "", file = LogFile, append = TRUE)
cat(", Acceptance rate:", round(aAcceptance/aiter, 2), "\n", sep = "", file =
     LogFile, append = TRUE)}
}

cat("MCMC ended on ", date(), "\n", sep="")
time1 <- proc.time()

#MCMC trace plots
par(mfrow=c(2,1))
plot(athinned[2:at.iter,1],type="l")
plot(athinned[2:at.iter,2],type="l")

#Summary statistics for multivariate estimation
aused = (athinned[10002:30001,])
amean = round(apply((aused),2,median,na.rm = T),3)
astd = apply(aused,2,sd,na.rm=T)
alCI = rep(0,2)
auCI = rep(0,2)
for (i in 1:2){
alCI[i] = round(quantile(aused[,i],p=0.05),3)
auCI[i] = round(quantile(aused[,i],p=0.95),3)
}
abothline = cbind(amean,astd,alCI,auCI)

##########Goodness of fit test##########

#Marginal fit
fittedmarginal1 = matrix(NA,nrow=10,ncol=10)
fittedmarginal2 = matrix(NA,nrow=10,ncol=10)
pres1 = matrix(NA,nrow=10,ncol=10)
pres2 = matrix(NA,nrow=10,ncol=10)

alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

for (i in 1:10){
for (j in 1:(10-i+1)){
B1 = phi1*((phi1/((alphaf1[i]*beta1[j])^(2-p)))*alphatilmean[j]+1)^{1-p}

B2 = phi2*((phi2/((alphaf2[i]*beta2[j])^(2-p)))*alphatilmean[j]+1)^{1-p}

fittedmarginal1[i,j]=alphaf1[i]*beta1[j]*((phi1/((alphaf1[i]*beta1[j])^(2-p))
    )*alphatilmean[j]+1)
pres1[i,j] = (loss1[i,j]-fittedmarginal1[i,j])/sqrt(fittedmarginal1[i,j]^p*B1
    )


fittedmarginal2[i,j]=alphaf2[i]*beta2[j]*((phi2/((alphaf2[i]*beta2[j])^(2-p))
    )*alphatilmean[j]+1)
pres2[i,j] = (loss2[i,j]-fittedmarginal2[i,j])/sqrt(fittedmarginal2[i,j]^p*B2
    )
}
}

mtresid1=as.vector(pres1)
mtresid1=mtresid1[!is.na(mtresid1)]
```

```
mtresid2=as.vector(pres2)
mtresid2=mtresid2[!is.na(mtresid2)]

par(mfrow=c(1,2))
qqnorm(mtresid1,font.main = 1,main="Bodily Injury line")
qqline(mtresid1)

qqnorm(mtresid2,font.main = 1,main="Accident Benefits line")
qqline(mtresid2)

#Multivariate fit
modelgof <- function(parm){
alpha1 <- parm[1:9]
beta1 <-parm[10:19]
alpha2<-parm[20:28]
beta2<-parm[29:38]
phi1 <- parm[39]
phi2<-parm[40]
p<- parm[41]
lambda <- parm[42]
phitil <- parm[43]
alphaf1 = c(1,alpha1)
alphaf2 = c(1,alpha2)
alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

frandom = matrix(NA,ncol=10,nrow=10)
fpa = matrix(NA,ncol=10,nrow=10)
fca = matrix(NA,ncol=10,nrow=10)

for (i in 1:10){
for (j in 1:(10-i+1)){
frandom[i,j] = rtweedie(1,mu=alphatilmean[j],phi=phitil,xi=p)
fpa[i,j] = ((alphatilmean[j]/((alphaf1[i]*beta1[j])))^(1-p)*(phi1/phitil)*
    frandom[i,j]+ rtweedie(1,mu=alphaf1[i]*beta1[j],phi=phi1,xi=p))
fca[i,j] = ((alphatilmean[j]/((alphaf2[i]*beta2[j])))^(1-p)*(phi2/phitil)*
    frandom[i,j]+ rtweedie(1,mu=alphaf2[i]*beta2[j],phi=phi2,xi=p))
}}

list (t = c(ttpa,ttca,tt), vfpa = as.vector(t(fpa)), vfca = as.vector(t(fca))
    , vftotal = as.vector(t(ftotal)))
}

para = cbind(used[,1:40],used[,42],aused)
N = nrow(para)
gof_sfpa = matrix(NA,nrow=N,ncol=100)
gof_sfca = matrix(NA,nrow=N,ncol=100)

set.seed(11)
for (n in 1:N){
mtest = modelgof(para[n,])
gof_sfpa[n,] = mtest[["vfpa"]]
gof_sfca[n,] = mtest[["vfca"]]
}

par(mfrow=c(1,2), mai = c(0.8, 0.75, 0.5, 0.1))
plot(ecdf(vloss1)(vloss1),ecdf(vloss2)(vloss2),xlab="Bodily Injury",ylab="
    Accident Benefits",main="Observed data",font.main = 1)
plot(ecdf(gof_sfpa[10000,])(gof_sfpa[10000,]),ecdf(gof_sfca[10000,])(gof_sfca
    [10000,]),main="Fitted data",xlab="Bodily Injury",ylab="Accident Benefits"
    ,font.main = 1)

##########Claims forecast##########

#Function to forecast future claims
modelp <- function(parm){
alpha1 <- parm[1:9]
beta1 <-parm[10:19]
alpha2<-parm[20:28]
beta2<-parm[29:38]
phi1 <- parm[39]
```

```
phi2<-parm[40]
p<- parm[41]
lambda <- parm[42]
phitil <- parm[43]
alphaf1 = c(1,alpha1)
alphaf2 = c(1,alpha2)
alphatilmean = (sqrt(beta1*beta2))^(2-p)*lambda

frandom = matrix(NA,ncol=10,nrow=10)
fpa = matrix(NA,ncol=10,nrow=10)
fca = matrix(NA,ncol=10,nrow=10)
ftotal = matrix(NA,ncol=10,nrow=10)

fpaacc = c(0,10)
fcaacc = c(0,10)
fttacc = c(0,10)

for (i in 2:10){
for (j in (10-i+2):10){
frandom[i,j] = rtweedie(1,mu=alphatilmean[j],phi=phitil,xi=p)

fpa[i,j] = (((alphatilmean[j]/((alphaf1[i]*beta1[j])))^(1-p)*(phi1/phitil)*
    frandom[i,j]+ rtweedie(1,mu=alphaf1[i]*beta1[j],phi=phi1,xi=p)))*prem1[i]
fca[i,j] = (((alphatilmean[j]/((alphaf2[i]*beta2[j])))^(1-p)*(phi2/phitil)*
    frandom[i,j]+ rtweedie(1,mu=alphaf2[i]*beta2[j],phi=phi2,xi=p)))*prem2[i]
ftotal[i,j] = fpa[i,j] + fca[i,j]
}

fpaacc[i] = sum(fpa[i,], na.rm=T)
fcaacc[i] = sum(fca[i,],na.rm=T)
fttacc[i] = fpaacc[i]+fcaacc[i]
}

ttpa = sum(fpaacc, na.rm=T)
ttca = sum(fcaacc, na.rm=T)
tt = ttpa+ttca

list (t = c(ttpa, ttca, tt), vfpa = as.vector(t(fpa)), vfca = as.vector(t(fca
    )), vftotal = as.vector(t(ftotal)), fpaacc = fpaacc, fcaacc = fcaacc,
    fttacc = fttacc)
}

#Run simulation
para = cbind(used[,1:40],used[,42],aused)
N = nrow(para)
stt = matrix(NA,nrow=N,ncol=15)

sfpa = matrix(NA,nrow=N,ncol=100)
sfca = matrix(NA,nrow=N,ncol=100)
sftotal = matrix(NA,nrow=N,ncol=100)

sfpaacc = matrix(NA,nrow=N,ncol=10)
sfcaacc = matrix(NA,nrow=N,ncol=10)
sfttacc = matrix(NA,nrow=N,ncol=10)

cat("Simulation started on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

set.seed(11)
for (n in 1:N){
mpredict = modelp(para[n,])
stt[n,] = mpredict[["t"]]
sfpa[n,] = mpredict[["vfpa"]]
sfca[n,] = mpredict[["vfca"]]
sftotal[n,] = mpredict[["vftotal"]]
sfpaacc[n,] = mpredict[["fpaacc"]]
sfcaacc[n,] = mpredict[["fcaacc"]]
sfttacc[n,] = mpredict[["fttacc"]]
}
```

```
cat("Simulation ended on", date(), "\n", sep="")
time1 <- proc.time()
LDcall <- match.call()

#summarise results
mean_stt = colMeans(stt[,1:3],na.rm=T)
sd_stt = apply(stt[,1:3],2,sd,na.rm=T)
var75 = c(0,0,0)
var95 = c(0,0,0)

for (i in 1:3){
var75[i] = quantile(stt[,i],p=0.75)
var95[i] = quantile(stt[,i],p=0.95)
}

paaccmean = colMeans(sfpaacc)
paaccsd = apply(sfpaacc,2,sd,na.rm=T)

caaccmean = colMeans(sfcaacc)
caaccsd = apply(sfcaacc,2,sd,na.rm=T)

ttaccmean = colMeans(sfttacc)
ttaccsd = apply(sfttacc,2,sd,na.rm=T)

accidentyrsummary = cbind(paaccmean,paaccsd,caaccmean,caaccsd,ttaccmean,
    ttaccsd)
xtable(accidentyrsummary)

summarytable = rbind(mean_stt,sd_stt,var75,var95)

library(EnvStats)

par(mfrow=c(1,1))
plot (density(stt[,1]), ylim=c(0,2.5e-05),xlim=c(40000,500000),xlab="Total
    unpaid losses",main="",lwd=3)
lines (density(stt[,2]), lty=2,lwd=3)
lines (density(stt[,3]), lty=3,lwd=3)
legend("top", legend = c("Bodily Injury", "Accient Benefits", "Total"),
lty = 1:3,lwd=3, bty = "n",
title = "")

#Common shock proportions
ratio1 = matrix(NA,nrow=10,ncol=10)
ratio2 = matrix(NA,nrow=10,ncol=10)

for(i in 1:10){
for(j in 1:10){
ratio1[i,j] = (alphatilmean[j]*(phi1)/((alphaf1[i]*beta1[j])^(1-p)))/((
    alphatilmean[j]*(phi1)/((alphaf1[i]*beta1[j])^(1-p)))+alphaf1[i]*beta1[j])
    *100
ratio2[i,j] = (alphatilmean[j]*(phi2)/((alphaf2[i]*beta2[j])^(1-p)))/((
    alphatilmean[j]*(phi2)/((alphaf2[i]*beta2[j])^(1-p)))+alphaf2[i]*beta2[j])
    *100
}
}
```

A.3 R codes for Chapter 6

A.3.1 Gaussian model illustration

```
library(magic)
library(MASS)
library(matlib)

##########Simulate triangles##########
I=15

#Parameters
lambda1 = 0.6
lambda2 = 0.8

phi1= 0.02
phi2= 0.02

gamma1sd = c(0.01,0.005,0.001)
gamma2sd = c(0.005,0.002,0.0005)
psisd = c(0.005,0.005,0.005)

coeff=1

#Initialisation
set.seed(4175)
y_kf = list()
a_kf = list()
h_kf = list()
e_kf = list()
gamma_kf=list()
psi_kf = list()

vector.h1 = rep(0,I)
vector.h2 = rep(0,I)
vector.h1[1] = 0.5
vector.h2[1] = 0.5

commonshock = rnorm(I-1,mean=0,sd=psisd[1])
for (i in 2:I){
vector.h1[i] =  vector.h1[i-1]+lambda1*commonshock[i-1] + rnorm(1,mean=0,sd=
    psisd[2])
vector.h2[i] =  vector.h2[i-1]+lambda2*commonshock[i-1] + rnorm(1,mean=0,sd=
    psisd[3])
}

a1_0 = 7
r1_0 = 1.5
s1_0 = -0.8

a2_0 = 7
r2_0 = 2
s2_0 = -0.4

gamma.0 = c(a1_0,r1_0,s1_0,a2_0,r2_0,s2_0)
q_kf = diag(c(gamma1sd,gamma2sd))

psi.1 = c(vector.h1,vector.h2)

gamma = gamma.0+c(mvrnorm(1,mu=rep(0,3),Sigma=q_kf[1:3,1:3]),mvrnorm(1,mu=rep
    (0,3),Sigma=q_kf[4:6,4:6]))
gamma_kf[[1]] = gamma

an = matrix(0,nrow=(I),ncol=3)
an[1:nrow(an),1] = rep(1,nrow(an))
an[,2] = c(log(seq(from=1,to=I)))
an[,3] = c((seq(from=1,to=(I))))

A = cbind(adiag(an,an))
```

```
a_kf[[1]] = A

E = diag(2*I)
e_kf[[1]] = E

H = diag(c(rep(phi1,I),rep(phi2,I)))
h_kf[[1]] = H

y1 = A%*%(gamma)+ E%*%psi.1 + as.matrix(mvrnorm(1,mu = rep(0,2*I),Sigma=H))
y_kf[[1]] = y1

line1 = matrix(NA,nrow=I,ncol=I)
line2 = matrix(NA,nrow=I,ncol=I)
line1[1,] = y1[1:I]
line2[1,] = y1[(I+1):(2*I)]

vector.a1 = rep(NA,I)
vector.a1[1] = gamma[1]
vector.r1 = rep(NA,I)
vector.r1[1] = gamma[2]
vector.s1 = rep(NA,I)
vector.s1[1] = gamma[3]
vector.a2 = rep(NA,I)
vector.a2[1] = gamma[4]
vector.r2 = rep(NA,I)
vector.r2[1] = gamma[5]
vector.s2 = rep(NA,I)
vector.s2[1] = gamma[6]

for (i in 2:I) {
#State simulation
gamma=gamma+as.matrix(mvrnorm(n=1,mu=rep(0,6),Sigma = q_kf))
gamma_kf[[i]] = gamma

#Observation simulation
an = matrix(0,nrow=(I+1-i),ncol=3)
an[1:nrow(an),1] = rep(1,nrow(an))
an[,2] = c(log(seq(from=1,to=(I+1-i))))
an[,3] = c((seq(from=1,to=(I-i+1))))

A = adiag(an,an)

H = diag(c(rep(phi1,(I-i+1)),rep(phi2,(I-i+1))))
en = matrix(0,nrow=(15-i+1),ncol=i-1)
en = cbind(en,diag(15-i+1))
E = adiag(en,en)

yn = A%*%gamma+ E%*%psi.1 + as.matrix(mvrnorm(n=1,mu=rep(0,(I+1-i)*2),Sigma =
     H))

line1[i,1:(I-i+1)] = yn[1:(I-i+1)]
line2[i,1:(I-i+1)] = yn[(I-i+2):length(yn)]

y_kf[[i]] = yn
a_kf[[i]] = A
e_kf[[i]] = E
h_kf[[i]] = H

vector.a1[i] = gamma[1]
vector.r1[i] = gamma[2]
vector.s1[i] = gamma[3]
vector.a2[i] = gamma[4]
vector.r2[i] = gamma[5]
vector.s2[i] = gamma[6]
}

i=I+1

an = matrix(0,nrow=(I+1-i),ncol=3)
an[1:nrow(an),1] = rep(1,nrow(an))
an[,2] = c(log(seq(from=1,to=(I+1-i))))
```

```
an[,3] = c((seq(from=1,to=(I-i+1))))
A = adiag(an,an)

en = matrix(0,nrow=(15-i+1),ncol=i-1)
en = cbind(en,diag(15-i+1))
E = adiag(en,en)

a_kf[[i]] = A
e_kf[[i]] = E

par(mfrow=c(4,2))
plot(vector.a1,type="l")
plot(vector.a2,type="l")
plot(vector.r1,type="l")
plot(vector.s1,type="l")
plot(vector.r2,type="l")
plot(vector.s2,type="l")
plot(vector.h1,type="l")
plot(vector.h2,type="l")

##########Preliminary analysis - development plots##########
line1exp = exp(line1)
line2exp = exp(line2)

par(mfrow=c(1,1))
plot(line1exp[1,],type = "l",lwd=1, ylim=c(min(line1exp,na.rm=T),max(line1exp
    ,na.rm=T)))
for (i in 2:I){
lines(line1exp[i,],lwd=1)
}

plot(line2exp[1,],type = "l",lwd=1, ylim=c(min(line2exp,na.rm=T),max(line2exp
    ,na.rm=T)))
for (i in 2:I){
lines(line2exp[i,],lwd=1)
}

##########Preliminary analysis - GLM analysis to pick initial values
    ##########
vline1 = as.vector(t(line1))
vline1 = vline1[!is.na(vline1)]
vline2 = as.vector(t(line2))
vline2 = vline2[!is.na(vline2)]

#First accident year
j= seq(1:I)
i=rep(1,I)
out1 <- lm(line1[1,]~ i + log(j) + (j))
out2 <- lm(line2[1,]~ i + log(j) + (j))

#All accident years
j1= c(c(1:15),c(1:14),c(1:13),c(1:12),c(1:11),c(1:10),c(1:9),c(1:8),c(1:7),c
    (1:6),c(1:5),c(1:4),c(1:3),c(1:2),1)
i1=rep(1:I,I:1)

out1all <- lm(vline1~ as.factor(i1) + log(j1) + (j1))
out2all <- lm(vline2~ as.factor(i1) + log(j1) + (j1))

length = seq(from=I,to=1)
Var1 = sequence(length)
Var2 = rep(I:1,I:1)
line1frame <- data.frame(Var1,Var2,vline1)
line2frame <- data.frame(Var1,Var2,vline2)

##########Parameter estimation-MLE with Kalman filtering##########
#Likelihood function
llhfunction <- function(parsq){
par = exp(parsq)

h_kf_e = list()
llh_v = rep(NA,I)
```

```
f.t = list()
v.t = list()
q.t = list()
gamma.t = list()
gamma.filter = list()
q.filter = list()
psi.t = list()
psi.filter = list()
qpsi.t = list()
qpsi.filter = list()
g.t = list()
gpsi.t = list()
y.fit = list()

q.1 = diag(c(par[3:8]))
q.t[[1]] = q.1

sim.cal = matrix(NA,nrow=10000,ncol=I*2)
sim.cal[,1] = rep(0.45,10000)
sim.cal[,16] = rep(0.45,10000)

set.seed(4178)
for(n in 1:10000){
commonshock.e = rnorm(I-1,mean=0,sd=par[9])
for (i in 2:I){
sim.cal[n,i] = sim.cal[n,i-1]+par[12]*commonshock.e[i-1] + rnorm(1,mean=0,sd=
    par[10])
sim.cal[n,I+i] = sim.cal[n,I+i-1]+par[13]*commonshock.e[i-1] + rnorm(1,mean
    =0,sd=par[11])
}
}

gamma.t[[1]] = c(out1all$coefficients[1]-0.45,out1all$coefficients[16],
    out1all$coefficients[17],out2all$coefficients[1]-0.45,out2all$coefficients
    [16],out2all$coefficients[17])
psi.t[[1]] = colMeans(sim.cal)
qpsi.t[[1]] = cov(sim.cal)

gamma.vector=matrix(NA,nrow=I,ncol=6)
psi.vector = matrix(NA,nrow=I,ncol=2)

for (i in 1:(I)){
h_kf_e[[i]] = diag(c(rep(par[1],(I-i+1)),rep(par[2],(I-i+1))))

#Measurement update for calendar factors
gpsi.t[[i]] = qpsi.t[[i]]%*%t(e_kf[[i]])%*%inv(e_kf[[i]]%*%qpsi.t[[i]]%*%t(e_
    kf[[i]])+h_kf_e[[i]])
if (i==1){
psi.filter[[i]] = psi.t[[i]] + gpsi.t[[i]]%*%(y_kf[[i]]-a_kf[[i]]%*%gamma.t[[
    i]]-e_kf[[i]]%*%psi.t[[i]])}
else{
psi.filter[[i]] = psi.t[[i]] + gpsi.t[[i]]%*%(y_kf[[i]]-a_kf[[i]]%*%gamma.
    filter[[i-1]]-e_kf[[i]]%*%psi.t[[i]])}
qpsi.filter[[i]] = qpsi.t[[i]] - gpsi.t[[i]]%*%e_kf[[i]]%*%qpsi.t[[i]]

#Measurement update for states
g.t[[i]] = q.t[[i]]%*%t(a_kf[[i]])%*%inv(a_kf[[i]]%*%q.t[[i]]%*%t(a_kf[[i]])+
    h_kf_e[[i]])
gamma.filter[[i]] = gamma.t[[i]] + g.t[[i]]%*%(y_kf[[i]]-a_kf[[i]]%*%gamma.t
    [[i]]-e_kf[[i]]%*%psi.filter[[i]])
q.filter[[i]] = q.t[[i]] - g.t[[i]]%*%a_kf[[i]]%*%q.t[[i]]

#Time update for calendar factors
psi.t[[i+1]] = psi.filter[[i]]
qpsi.t[[i+1]] = qpsi.filter[[i]] + qpsi.t[[1]]*0.5

#Time update for states
gamma.t[[i+1]] = gamma.filter[[i]]
q.t[[i+1]] = q.filter[[i]]  + q.1
f.t[[i]] = a_kf[[i]]%*%q.t[[i]]%*%t(a_kf[[i]]) + e_kf[[i]]%*%qpsi.t[[i]]%*%t(
    e_kf[[i]]) + h_kf_e[[i]]
```

```
v.t[[i]] = y_kf[[i]] - a_kf[[i]]%*%gamma.t[[i]] - e_kf[[i]]%*%psi.t[[i]]

y.fit[[i]] = a_kf[[i]]%*%gamma.filter[[i]] + e_kf[[i]]%*%psi.filter[[i]]

gamma.vector[i,1] = gamma.filter[[i]][1]
gamma.vector[i,2] = gamma.filter[[i]][2]
gamma.vector[i,3] = gamma.filter[[i]][3]
gamma.vector[i,4] = gamma.filter[[i]][4]
gamma.vector[i,5] = gamma.filter[[i]][5]
gamma.vector[i,6] = gamma.filter[[i]][6]
psi.vector[i,1] = psi.filter[[i]][i]
psi.vector[i,2] = psi.filter[[i]][I+i]

llh_v[i] = -length(y_kf[[i]])*0.5*(log(2*pi)) - 0.5*(log(det(f.t[[i]]))+t(v.t
    [[i]])%*%inv(f.t[[i]])%*%v.t[[i]])}

llh = -sum(llh_v)
list(llh = llh, y.fit=y.fit,gamma.t=gamma.t,q.t=q.t,qpsi.filter=qpsi.filter,
    qpsi.t=qpsi.t,gamma.filter = gamma.filter, q.filter=q.filter,psi.t = psi.t
    , psi.filter=psi.filter,gamma.vector=gamma.vector,psi.vector=psi.vector)
}

#Estimate parameters
llh_func <- function(parsq){
llh = llhfunction(parsq)$llh
return(llh)
}

par_est_init = c(-3.91, -3.91, -4.61, -5.30, -6.91, -5.30, -6.21, -7.60,
    -5.30, -5.30, -5.30, -0.51, -0.22)
set.seed(4157)
mstep = optim(par_est_init*1.2,llh_func,method="BFGS",hessian = T,control=
    list(trace=T,abstol=1e-5))
fisher_info <- try(solve(-mstep$hessian),silent=T)
prop_sigma <-try((sqrt(diag(fisher_info))),silent=T)

#Results
upper <- try(mstep$par + 1.96*prop_sigma,silent=T)
lower <- try(mstep$par - 1.96*prop_sigma,silent=T)
true = log(c(phi1,phi2,gamma1sd,gamma2sd,psisd,lambda1,lambda2))
results = cbind(true,mstep$par,lower,upper)

##########States estimation using Kalman filtering##########
function_fit <- function(parsq){
fit = llhfunction(parsq)
list(y.fit=fit$y.fit,gamma.t=fit$gamma.t,q.t=fit$q.t,qpsi.filter=fit$qpsi.
    filter,qpsi.t=fit$qpsi.t,gamma.filter =fit$gamma.filter, q.filter=fit$q.
    filter,psi.t = fit$psi.t, psi.filter=fit$psi.filter,gamma.vector=fit$gamma
    .vector,psi.vector=fit$psi.vector)}

par_fit = function_fit(mstep$par)

##########States estimation using Kalman smoothing##########
psi.smooth = matrix(NA,I,28)
psi.smooth[I,] = par_fit$psi.filter[[I]][c(-1,-16)]

gamma.smooth = matrix(NA,I,6)
gamma.smooth[I,] = par_fit$gamma.filter[[I]]
y.smooth = list()

y.smooth.fit =a_kf[[I]]%*%gamma.smooth[I,] + e_kf[[I]]%*%c(0.45,psi.smooth[I
    ,1:14],0.45,psi.smooth[I,15:28])
y.smooth[[I]] = y.smooth.fit
for (i in (I-1):1){
psi.smooth[i,] = par_fit$psi.filter[[i]][c(-1,-16)] + par_fit$qpsi.filter[[i
    ]][c(-1,-16),c(-1,-16)]%*%inv(par_fit$qpsi.t[[i+1]][c(-1,-16),c(-1,-16)])%
    *%(psi.smooth[i+1,]-par_fit$psi.t[[i+1]][c(-1,-16)])
gamma.smooth[i,] = par_fit$gamma.filter[[i]] + par_fit$q.filter[[i]]%*%inv(
    par_fit$q.t[[i+1]])%*%(gamma.smooth[i+1,]-par_fit$gamma.t[[i+1]])
y.smooth.fit =a_kf[[i]]%*%gamma.smooth[i,] + e_kf[[i]]%*%c(0.45,psi.smooth[i
    ,1:14],0.45,psi.smooth[i,15:28])
```

```
y.smooth[[i]] = y.smooth.fit
}

##########Summarise results##########
state_est = array(0,c(12,I,3))
state_est[1,,] = cbind(par_fit$gamma.vector[1:I,1],vector.a1[1:I],gamma.
    smooth[,1])
state_est[2,,] = cbind(par_fit$gamma.vector[1:I,2],vector.r1[1:I],gamma.
    smooth[,2])
state_est[3,,] = cbind(par_fit$gamma.vector[1:I,3],vector.s1[1:I],gamma.
    smooth[,3])
state_est[4,,] = cbind(par_fit$gamma.vector[1:I,4],vector.a1[1:I],gamma.
    smooth[,4])
state_est[5,,] = cbind(par_fit$gamma.vector[1:I,5],vector.r2[1:I],gamma.
    smooth[,5])
state_est[6,,] = cbind(par_fit$gamma.vector[1:I,6],vector.s2[1:I],gamma.
    smooth[,6])
state_est[7,,] = cbind(par_fit$psi.vector[1:I,1],vector.h1[1:I],c(0.45,psi.
    smooth[1,1:14]))
state_est[8,,] = cbind(par_fit$psi.vector[1:I,2],vector.h2[1:I],c(0.45,psi.
    smooth[1,15:28]))
#Mean of Hoerl curve
state_est[9,,] = cbind((par_fit$gamma.vector[,2]-1)/(-par_fit$gamma.vector
    [,3]),(vector.r1-1)/(-vector.s1),(gamma.smooth[,2]-1)/(-gamma.smooth[,3]))
state_est[10,,] = cbind((par_fit$gamma.vector[,5]-1)/(-par_fit$gamma.vector
    [,6]),(vector.r2-1)/(-vector.s2),(gamma.smooth[,5]-1)/(-gamma.smooth[,6]))
#Variance of Hoerl curve
state_est[11,,] = cbind((par_fit$gamma.vector[,2]-1)/(par_fit$gamma.vector
    [,3]^2),(vector.r1-1)/(vector.s1^2),(gamma.smooth[,2]-1)/(gamma.smooth
    [,3]^2))
state_est[12,,] = cbind((par_fit$gamma.vector[,5]-1)/(par_fit$gamma.vector
    [,6]^2),(vector.r2-1)/(vector.s2^2),(gamma.smooth[,5]-1)/(gamma.smooth
    [,6]^2))

#Plot the difference ratio
difference_plot = matrix(NA,I,12)
for(i in 1:12){
difference_plot[,i] = (state_est[i,,3])/state_est[i,,2]
}

par(mar=c(2.5,2.5,2.5,2.5))
par(mfrow=c(4,2))
axticks <- seq(1, 15, 1)
axuse <-seq(1,15,1)
plot(difference_plot[,1],type="l",lty=1,main=expression(paste(a[i]^(1))),ylim
    =c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,1],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,2],type="l",lty=1,main=expression(paste(a[i]^(2))),ylim
    =c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,2],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,9],type="l",lty=1,main=expression(paste((r[i]^(1)-1)/(-
    s[i]^(1)))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,9],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,10],type="l",lty=1,main=expression(paste((r[i]^(2)-1)/
    (-s[i]^(2)))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,10],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,11],type="l",lty=1,main=expression(paste((r[i]^(1)-1)/
    ((s[i]^(1))^2))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,11],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,12],type="l",lty=1,main=expression(paste((r[i]^(2)-1)/
    ((s[i]^(2))^2))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,12],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,7],type="l",lty=1,main=expression(paste(h[t]^(1))),ylim
    =c(0.5,1.5),xlab="i",ylab='\n')
```

```
points(difference_plot[,7],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,8],type="l",lty=1,main=expression(paste(h[t]^(2))),ylim
    =c(0.5,1.5),xlab="t",ylab='\n')
points(difference_plot[,8],pch=19)
axis(side=1,at=axuse, labels=axticks)

#Prepare data for heatmap
res1 = matrix(NA,nrow=I,ncol=I)
res2 = matrix(NA,nrow=I,ncol=I)
for (i in 1:I){
for (m in 1:(I-i+1)){
res1[i,m] = y_kf[[i]][m]/y.smooth[[i]][m]
res2[i,m] = y_kf[[i]][(I-i+1)+m]/y.smooth[[i]][(I-i+1)+m]
}}

res1 = as.vector(res1)
res1 = res1[!is.na(res1)]
res2 = as.vector(res2)
res2 = res2[!is.na(res2)]

length = seq(from=I,to=1)
Var2 = c(c(15:1),c(15:2),c(15:3),c(15:4),c(15:5),c(15:6),c(15:7),c(15:8),c
    (15:9),c(15:10),c(15:11),c(15:12),c(15:13),c(15:14),15)
Var1 = rep(1:I,I:1)

heatmap1 <- data.frame(Var1,Var2,res1)
heatmap2 <- data.frame(Var1,Var2,res2)

#Plot the tracking of development pattern
par(mar=c(4,4,1,1))
par(mfrow=c(2,2))

plot(exp(y.smooth[[2]][1:(I-2+1)]),type="l",main=expression("Triangle 1"),lty
    =2,ylim=c(min(exp(y.smooth[[2]][1:(I-2+1)]),line1exp[2,],exp(y.smooth
    [[2-1]][1:(I-2+2)]),na.rm=T),max(exp(y.smooth[[2]][1:(I-2+1)]),line1exp
    [2,],exp(y.smooth[[2-1]][1:(I-2+2)]),na.rm=T)))
lines(line1exp[2,],lty=1)
lines(exp(y.smooth[[2-1]][1:(I-2+2)]),lty=3)
points(exp(y.smooth[[2]][1:(I-2+1)]),pch=2)
points(line1exp[2,],pch=1)
points(exp(y.smooth[[2-1]][1:(I-2+2)]),pch=3)
legend("topright", legend = c( "Year 2 observed", "Year 2 smoothed","Year 1
    smoothed"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")

plot(exp(y.smooth[[2]][(I-2+2):(2*(I-2+1))]),lty=2,type="l",main=expression("
    Triangle 2"),ylim=c(min(exp(y.smooth[[2]][(I-2+2):(2*(I-2+1))]),line2exp
    [2,],exp(y.smooth[[2-1]][(I-2+3):(2*(I-2+2))]),na.rm=T),max(exp(y.smooth
    [[2]][(I-2+2):(2*(I-2+1))]),line2exp[2,],exp(y.smooth[[2-1]][(I-2+3):(2*(I
    -2+2))]),na.rm=T)))
lines(line2exp[2,],lty=1)
lines(exp(y.smooth[[2-1]][(I-2+3):(2*(I-2+2))]),lty=3)
points(exp(y.smooth[[2]][(I-2+2):(2*(I-2+1))]),pch=2)
points(line2exp[2,],pch=1)
points(exp(y.smooth[[2-1]][(I-2+3):(2*(I-2+2))]),pch=3)
legend("bottom", legend = c( "Year 2 observed", "Year 2 smoothed","Year 1
    smoothed"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")

plot(exp(y.smooth[[3]][1:(I-3+1)]),type="l",main=expression("Triangle 1"),lty
    =2,ylim=c(min(exp(y.smooth[[3]][1:(I-3+1)]),line1exp[3,],exp(y.smooth
    [[3-1]][1:(I-3+2)]),na.rm=T),max(exp(y.smooth[[3]][1:(I-3+1)]),line1exp
    [3,],exp(y.smooth[[3-1]][1:(I-3+2)]),na.rm=T)))
lines(line1exp[3,],lty=1)
lines(exp(y.smooth[[3-1]][1:(I-3+2)]),lty=3)
points(exp(y.smooth[[3]][1:(I-3+1)]),pch=2)
points(line1exp[3,],pch=1)
```

```
points(exp(y.smooth[[3-1]][1:(I-3+2)]),pch=3)
legend("topright", legend = c( "Year 3 observed", "Year 3 smoothed","Year 2
    smoothed"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")

plot(exp(y.smooth[[3]][(I-3+2):(2*(I-3+1))]),lty=2,type="l",main=expression("
    Triangle 2"),ylim=c(min(exp(y.smooth[[3]][(I-3+2):(2*(I-3+1))]),line2exp
    [3,],exp(y.smooth[[3-1]][(I-3+3):(2*(I-3+2))]),na.rm=T),max(exp(y.smooth
    [[3]][(I-3+2):(2*(I-3+1))]),line2exp[3,],exp(y.smooth[[3-1]][(I-3+3):(2*(I
    -3+2))]),na.rm=T)))
lines(line2exp[3,],lty=1)
lines(exp(y.smooth[[3-1]][(I-3+3):(2*(I-3+2))]),lty=3)
points(exp(y.smooth[[3]][(I-3+2):(2*(I-3+1))]),pch=2)
points(line2exp[3,],pch=1)
points(exp(y.smooth[[3-1]][(I-3+3):(2*(I-3+2))]),pch=3)
legend("bottom", legend = c( "Year 3 observed", "Year 3 smoothed","Year 2
    smoothed"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")

#Check fitted correlation
cor.test(vector.h1,vector.h2,method="pearson",conf.level = 0.95)
cor.test(par_fit$psi.vector[,1],par_fit$psi.vector[,2],method="pearson",conf.
    level = 0.95)
```

A.3.2 Tweedie model illustration

```
library(magic)
library(MASS)
library(matlib)
library(tweedie)
library(statmod)

##########Simulate triangles##########
I=15

#Parameters
lambda1 = 0.6
lambda2 = 0.8

phi1= 0.4
phi2= 0.5

p1= 1.27
p2= 1.35

gamma1sd = c(0.01,0.005,0.001)
gamma2sd = c(0.005,0.002,0.0005)
psisd = c(0.005,0.005,0.005)

coeff=1

#Initialisation
set.seed(1030)
y_kf = list()
a_kf = list()
h_kf = list()
e_kf = list()
gamma_kf=list()
psi_kf = list()
p_kf = list()

vector.h1 = rep(0,I)
vector.h2 = rep(0,I)
vector.h1[1] = 0.5
vector.h2[1] = 0.5

commonshock = rnorm(I-1,mean=0,sd=psisd[1])
for (i in 2:I){
```

```
vector.h1[i] =  vector.h1[i-1]+lambda1*commonshock[i-1] + rnorm(1,mean=0,sd=
    psisd[2])
vector.h2[i] =  vector.h2[i-1]+lambda2*commonshock[i-1] + rnorm(1,mean=0,sd=
    psisd[3])
}

a1_0 = 7
r1_0 = 1.5
s1_0 = -0.8

a2_0 = 7
r2_0 = 2
s2_0 = -0.4

gamma.0 = c(a1_0,r1_0,s1_0,a2_0,r2_0,s2_0)
q_kf = diag(c(gamma1sd,gamma2sd))

psi.1 = c(vector.h1,vector.h2)

gamma = gamma.0+c(mvrnorm(1,mu=rep(0,3),Sigma=q_kf[1:3,1:3]),mvrnorm(1,mu=rep
    (0,3),Sigma=q_kf[4:6,4:6]))
gamma_kf[[1]] = gamma

an = matrix(0,nrow=(I),ncol=3)
an[1:nrow(an),1] = rep(1,nrow(an))
an[,2] = c(log(seq(from=1,to=I)))
an[,3] = c((seq(from=1,to=(I))))

A = cbind(adiag(an,an))
a_kf[[1]] = A

E = diag(2*I)
e_kf[[1]] = E

H = c(rep(phi1,I),rep(phi2,I))
h_kf[[1]] = H

p_kf[[1]] =c(rep(power1,I),rep(power2,I))

y1 = rep(NA,2*I)
for (i in 1:(2*I)){
y1[i] = rtweedie(1,mu = exp(A[i,]%*%(gamma)+ E[i,]%*%psi.1),phi=H[i],xi = p_
    kf[[1]][i])}

y_kf[[1]] = y1

line1 = matrix(NA,nrow=I,ncol=I)
line2 = matrix(NA,nrow=I,ncol=I)
line1[1,] = y1[1:I]
line2[1,] = y1[(I+1):(2*I)]

vector.a1 = rep(NA,I)
vector.a1[1] = gamma[1]
vector.r1 = rep(NA,I)
vector.r1[1] = gamma[2]
vector.s1 = rep(NA,I)
vector.s1[1] = gamma[3]
vector.a2 = rep(NA,I)
vector.a2[1] = gamma[4]
vector.r2 = rep(NA,I)
vector.r2[1] = gamma[5]
vector.s2 = rep(NA,I)
vector.s2[1] = gamma[6]

for (i in 2:I) {
#State simulation
gamma=gamma+as.matrix(mvrnorm(n=1,mu=rep(0,6),Sigma = q_kf))
gamma_kf[[i]] = gamma

#Observation simulation
an = matrix(0,nrow=(I+1-i),ncol=3)
an[1:nrow(an),1] = rep(1,nrow(an))
```

```
an[,2] = c(log(seq(from=1,to=(I+1-i))))
an[,3] = c((seq(from=1,to=(I-i+1))))

A = adiag(an,an)

H = c(rep(phi1,(I-i+1)),rep(phi2,(I-i+1)))
en = matrix(0,nrow=(15-i+1),ncol=i-1)
en = cbind(en,diag(15-i+1))
E = adiag(en,en)
power = c(rep(power1,(I-i+1)),rep(power2,(I-i+1)))

yn = rep(NA,2*(I-i+1))

for (n in 1:length(yn)){
yn[n] = rtweedie(1,mu = exp(A[n,]%*%(gamma)+ E[n,]%*%psi.1),phi=H[n],xi =
    power[n])}

line1[i,1:(I-i+1)] = yn[1:(I-i+1)]
line2[i,1:(I-i+1)] = yn[(I-i+2):length(yn)]

y_kf[[i]] = yn
a_kf[[i]] = A
e_kf[[i]] = E
h_kf[[i]] = H
p_kf[[i]] = power

vector.a1[i] = gamma[1]
vector.r1[i] = gamma[2]
vector.s1[i] = gamma[3]
vector.a2[i] = gamma[4]
vector.r2[i] = gamma[5]
vector.s2[i] = gamma[6]
}

par(mfrow=c(4,2))
plot(vector.a1,type="l")
plot(vector.a2,type="l")
plot(vector.r1,type="l")
plot(vector.s1,type="l")
plot(vector.r2,type="l")
plot(vector.s2,type="l")
plot(vector.h1,type="l")
plot(vector.h2,type="l")

##########Preliminary analysis - development plots##########
par(mfrow=c(1,1))
plot(line1[1,],type = "l",lwd=1, ylim=c(min(line1,na.rm=T),max(line1,na.rm=T)
    ))
for (i in 2:I){
lines(line1[i,],lwd=1)
}

plot(line2[1,],type = "l",lwd=1, ylim=c(min(line2,na.rm=T),max(line2,na.rm=T)
    ))
for (i in 2:I){
lines(line2[i,],lwd=1)
}

##########Preliminary analysis - GLM analysis to pick initial values
    ##########
vline1 = as.vector(t(line1))
vline1 = vline1[!is.na(vline1)]
vline2 = as.vector(t(line2))
vline2 = vline2[!is.na(vline2)]

#All accident years - set up llh profile to find p
j= c(c(1:15),c(1:14),c(1:13),c(1:12),c(1:11),c(1:10),c(1:9),c(1:8),c(1:7),c
    (1:6),c(1:5),c(1:4),c(1:3),c(1:2),1)
i=rep(1:I,I:1)
ci.vec = seq(1.05,1.95,by=0.01)
```

```
llh1 = rep(0,length(ci.vec))
llh2 = llh1

for (t in 1:length(ci.vec)){
out1 <- glm(vline1~(as.factor(i) + log(j) + (j)), fam=tweedie(var.power=ci.
    vec[t],link.power=0))
disp1 <- summary(out1)$dispersion
mu1 <- fitted(out1)
den1 <- dtweedie(out1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2 <-  glm(vline2~(as.factor(i) + log(j) + (j)), fam=tweedie(var.power=ci.
    vec[t],link.power=0))
disp2 <- summary(out2)$dispersion
mu2 <- fitted(out2)
den2 <- dtweedie(out2$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2[t] <- sum(log(den2))
}

ci1 = ci.vec[which.max(llh1)]
ci2 = ci.vec[which.max(llh2)]

#First accident year
i = rep(1,15)
j = c(1:15)
out1 <- glm(vline1[1:15]~log(j) + (j), fam=tweedie(var.power=ci1,link.power
    =0))
out2 <- glm(vline2[1:15]~log(j) + (j), fam=tweedie(var.power=ci2,link.power
    =0))

#######Particle filtering and parameter estimation
    ##############################
#Initialise filter
power1 = atanh(1.2-2.01)
power2 = atanh(1.3-2.01)
powerfunc = function(x){
tanh(x)+2.01
}

set.seed(1030)
N  = 50000
theta_init = c(-0.916, -0.693, -4.605, -5.298, -6.908, -5.298, -6.215,
    -7.601, -5.298, -5.298, -5.298, -0.511, -0.223, -1.127, -0.887)
a = 0.95
h  = sqrt(1-a^2)
gamma.vector=matrix(NA,nrow=I,ncol=6)
psi.vector = matrix(NA,nrow=I,ncol=2)

theta = matrix(NA,nrow=N,ncol=15)
theta= mvrnorm(N,theta_init,Sigma = diag(abs(theta_init))*0.05)
thetaexp = exp(theta)
thetaexp[,14:15] = powerfunc(theta[,14:15])

gamma.filter = matrix(0,nrow=N,ncol=6)
psi.filter = matrix(0,nrow=N,ncol=30)
psi.filter[,1] = rep(0.45,N)
psi.filter[,16] = rep(0.45,N)

w1 = rep(0,N)

for (i in 1:N){
gamma.filter[i,] = mvrnorm(1,mu=c(out1$coefficients[1]-0.45,out1$coefficients
    [2],out1$coefficients[3],out2$coefficients[1]-0.45,out2$coefficients[2],
    out2$coefficients[3]),Sigma = diag(c(thetaexp[i,3:8])))
for (j in 2:15){
commonshock.e = rnorm(1,mean=0,sd=thetaexp[i,9])
psi.filter[i,j] = psi.filter[i,j-1]+thetaexp[i,12]*commonshock.e + rnorm(1,
    mean=0,sd=thetaexp[i,10])
psi.filter[i,j+15] = psi.filter[i,15+j-1]+thetaexp[i,13]*commonshock.e +
```

```
    rnorm(1,mean=0,sd=thetaexp[i,11])    }

logmu1 = (exp(a_kf[[1]]%*%gamma.filter[i,] + e_kf[[1]]%*%psi.filter[i,]))

if (any(logmu1==0)|any(!is.finite(logmu1))) { w1[i] = -1e10}
else {
w1[i] = sum(log(dtweedie(y_kf[[1]][1:I],mu = logmu1[1:I],phi = (thetaexp[i
    ,1]),power=thetaexp[i,14])))+sum(log(dtweedie(y_kf[[1]][(I+1):(2*I)],mu =
    logmu1[(I+1):(2*I)],phi = (thetaexp[i,2]),power=thetaexp[i,15])))
}}

w_norm = exp(w1)/sum(exp(w1))

k = sample(1:N, replace=T,size=N,prob=w_norm)
nlevels(as.factor(k))

gamma.filter = gamma.filter[k,1:ncol(gamma.filter)]
psi.filter = psi.filter[k,1:ncol(psi.filter)]
theta = theta[k,1:ncol(theta)]
thetaexp = exp(theta)
thetaexp[,14:15] = powerfunc(theta[,14:15])

gamma.vector[1,1] = colMeans(gamma.filter)[1]
gamma.vector[1,2] = colMeans(gamma.filter)[2]
gamma.vector[1,3] = colMeans(gamma.filter)[3]
gamma.vector[1,4] = colMeans(gamma.filter)[4]
gamma.vector[1,5] = colMeans(gamma.filter)[5]
gamma.vector[1,6] = colMeans(gamma.filter)[6]
psi.vector[1,1] = colMeans(psi.filter)[1]
psi.vector[1,2] = colMeans(psi.filter)[16]

y.fit = exp(a_kf[[1]]%*%colMeans(gamma.filter) + e_kf[[1]]%*%colMeans(psi.
    filter))

line1fit = matrix(NA,nrow=I,ncol=I)
line2fit = matrix(NA,nrow=I,ncol=I)
line1fit[1,] = y.fit[1:I]
line2fit[1,] = y.fit[(I+1):(2*I)]

plot(line1fit[1,],type="l")
lines(line1[1,])
plot(line2fit[1,],type="l")
lines(line2[1,])

LogFile=""

#Run filter
for (i in 2:I){
cat("Time: ", i, "\n", sep = "")

#Project parameter and calendar factor values
thetahat = a*theta + rep((1-a)*apply(theta,2,mean),each=nrow(theta))
thetahatexp = exp(thetahat)
thetahatexp[,14:15] = powerfunc(thetahat[,14:15])

psihat = a*psi.filter + rep((1-a)*apply(psi.filter,2,mean),each=nrow(psi.
    filter))

#Compute look-ahead importance weights
w = rep(0,N)
w_e = rep(0,N)

for (j in 1:N){
logmu = exp(a_kf[[i]]%*%gamma.filter[j,] + e_kf[[i]]%*%psihat[j,])
if (any(logmu==0)|any(!is.finite(logmu))) {w[j] = -1e10}
else {
w_e[j] =  sum(log(dtweedie(y_kf[[i]][1:(I-i+1)],mu = logmu[1:(I-i+1)],phi = (
    thetahatexp[j,1]),power=thetahatexp[j,14])))+sum(log(dtweedie(y_kf[[i]][(I
    -i+2):length(y_kf[[i]])],mu = logmu[(I-i+2):length(y_kf[[i]])],phi =
    thetahatexp[j,2],power=thetahatexp[j,15])))
w[j] = w_e[j]+ log(w_norm[j])}}
```

```
w_norm = exp(w)/sum(exp(w))

#Resample
k = sample(1:N, replace=T,size=N,prob=w_norm)

#Draw parameter and calendar factors with the resampled index
varest=apply(theta,2,var)
theta1 = t(apply(thetahat[k,1:15],1,function(x){mvrnorm(1,x,diag(varest*(h^2)
    ))}))
thetaexp1 = exp(theta1)
thetaexp1[,14:15] = powerfunc(theta1[,14:15])

psivarest = apply(psi.filter,2,var)
psi.filter1 = t(apply(psihat[k,1:30],1,function(x){mvrnorm(1,x,diag(psivarest
    *(h^2)))}))

gamma.filter = gamma.filter[k,1:ncol(gamma.filter)]
gamma.filter1 = matrix(NA,nrow=N,ncol = length(gamma_kf[[i]]))

#Calculate importance weights
w1 = rep(0,N)

for (j in 1:N){
gamma.filter1[j,] =  gamma.filter[j,] + mvrnorm(n=1,mu=rep(0,6),Sigma = diag(
    thetaexp1[j,3:8]))
logmu1 = exp(a_kf[[i]]%*% gamma.filter1[j,]+ e_kf[[i]]%*%psi.filter1[j,])
if (any(logmu1==0)|any(!is.finite(logmu1))) { w1[j] = -1e10}
else {
w1[j] =sum(log(dtweedie(y_kf[[i]][1:(I-i+1)],mu = logmu1[1:(I-i+1)],phi = (
    thetaexp1[j,1]),power=thetaexp1[j,14])))+sum(log(prod(dtweedie(y_kf[[i]][(
    I-i+2):length(y_kf[[i]])],mu = logmu1[(I-i+2):length(y_kf[[i]])],phi =
    thetaexp1[j,2],power=thetaexp1[j,15]))))-w_e[k][j]
}}

w_norm = exp(w1)/sum(exp(w1))

#Calculate filtered statistics and resample for next period
k = sample(1:N, replace=T,size=N,prob=w_norm)
cat(", level=",nlevels(as.factor(k)), "\n", sep = "", file = LogFile, append
    = TRUE)

gamma.filter = gamma.filter1[k,1:ncol(gamma.filter1)]
psi.filter = psi.filter1[k,1:ncol(psi.filter1)]
theta = theta1[k,1:ncol(theta1)]
thetaexp = exp(theta)
thetaexp[,14:15] = powerfunc(theta[,14:15])

gamma.vector[i,1] = colMeans(gamma.filter)[1]
gamma.vector[i,2] = colMeans(gamma.filter)[2]
gamma.vector[i,3] = colMeans(gamma.filter)[3]
gamma.vector[i,4] = colMeans(gamma.filter)[4]
gamma.vector[i,5] = colMeans(gamma.filter)[5]
gamma.vector[i,6] = colMeans(gamma.filter)[6]
psi.vector[i,1] = colMeans(psi.filter)[i]
psi.vector[i,2] = colMeans(psi.filter)[15+i]

y.fit = exp(a_kf[[i]]%*%colMeans(gamma.filter)+ e_kf[[i]]%*%colMeans(psi.
    filter))

line1fit[i,1:(I-i+1)] = y.fit[1:(I-i+1)]
line2fit[i,1:(I-i+1)] = y.fit[(I-i+2):length(y.fit)]

plot(line1fit[i,],type="l")
lines(line1[i,])
plot(line2fit[i,],type="l")
lines(line2[i,])
}

##########Summarise results##########
state_est = array(0,c(12,I,2))
```

```
state_est[1,,] = cbind(gamma.vector[1:I,1],vector.a1[1:I])
state_est[2,,] = cbind(gamma.vector[1:I,2],vector.r1[1:I])
state_est[3,,] = cbind(gamma.vector[1:I,3],vector.s1[1:I])
state_est[4,,] = cbind(gamma.vector[1:I,4],vector.a1[1:I])
state_est[5,,] = cbind(gamma.vector[1:I,5],vector.r2[1:I])
state_est[6,,] = cbind(gamma.vector[1:I,6],vector.s2[1:I])
state_est[7,,] = cbind(psi.vector[1:I,1],vector.h1[1:I])
state_est[8,,] = cbind(psi.vector[1:I,2],vector.h2[1:I])
#Mean of Hoerl curve
state_est[9,,] = cbind((gamma.vector[,2]-1)/(-gamma.vector[,3]),(vector.r1-1)
    /(-vector.s1))
state_est[10,,] = cbind((gamma.vector[,5]-1)/(-gamma.vector[,6]),(vector.r2
    -1)/(-vector.s2))
#Variance of Hoerl curve
state_est[11,,] = cbind((gamma.vector[,2]-1)/(gamma.vector[,3]^2),(vector.r1
    -1)/(vector.s1^2))
state_est[12,,] = cbind((gamma.vector[,5]-1)/(gamma.vector[,6]^2),(vector.r2
    -1)/(vector.s2^2))

#Parameter estimates
true_par = c(phi1,phi2,gamma1sd,gamma2sd,psisd,lambda1,lambda2,p1,p2)
parameterest = cbind(true_par,apply(thetaexp,2,mean),apply(thetaexp,2,
    quantile,probs=0.05),apply(thetaexp,2,quantile,probs=0.95))

#Plot the difference ratio of states
difference_plot = matrix(NA,I,12)
for(i in 1:12){
difference_plot[,i] = (state_est[i,,1])/state_est[i,,2]
}

par(mar=c(2.5,2.5,2.5,2.5))
par(mfrow=c(4,2))
axticks <- seq(1, 15, 1)
axuse <-seq(1,15,1)
plot(difference_plot[,1],type="l",lty=1,main=expression(paste(a[i]^(1))),ylim
    =c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,1],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,2],type="l",lty=1,main=expression(paste(a[i]^(2))),ylim
    =c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,2],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,3],type="l",lty=1,main=expression(paste((r[i]^(1)-1)/(-
    s[i]^(1)))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,3],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,4],type="l",lty=1,main=expression(paste((r[i]^(2)-1)/(-
    s[i]^(2)))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,4],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,5],type="l",lty=1,main=expression(paste((r[i]^(1)-1)/((
    s[i]^(1))^2))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,5],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,6],type="l",lty=1,main=expression(paste((r[i]^(2)-1)/((
    s[i]^(2))^2))),ylim=c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,6],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,7],type="l",lty=1,main=expression(paste(h[t]^(1))),ylim
    =c(0.5,1.5),xlab="i",ylab='\n')
points(difference_plot[,7],pch=19)
axis(side=1,at=axuse, labels=axticks)
plot(difference_plot[,8],type="l",lty=1,main=expression(paste(h[t]^(2))),ylim
    =c(0.5,1.5),xlab="t",ylab='\n')
points(difference_plot[,8],pch=19)
axis(side=1,at=axuse, labels=axticks)

#Prepare data for heatmap
res1 = line1/line1fit
res2 = line2/line2fit
```

```
res1 = as.vector((res1))
res2 = as.vector((res2))
res1 = res1[!is.na(res1)]
res2 = res2[!is.na(res2)]

Var2 = c(c(15:1),c(15:2),c(15:3),c(15:4),c(15:5),c(15:6),c(15:7),c(15:8),c
    (15:9),c(15:10),c(15:11),c(15:12),c(15:13),c(15:14),15)
Var1 = rep(1:I,I:1)

heatmap1 <- data.frame(Var1,Var2,res1)
heatmap2 <- data.frame(Var1,Var2,res2)

#Plot the tracking of development patterns
par(mar=c(2.5,2.5,2.5,2.5))
par(mfrow=c(2,2))
plot(line1[7,],type="l",main=expression("Triangle 1"),ylim=c(min(line1[7,],
    line1fit[7,],line1fit[7-1,],na.rm=T),max(line1[7,],line1fit[7,],line1fit
    [7-1,],na.rm=T)))
lines(line1fit[7,],lty=2)
lines(line1fit[7-1,],lty=3)
points(line1[7,],pch=1)
points(line1fit[7,],pch=2)
points(line1fit[7-1,],pch=3)
legend("topright", legend = c( "Year 7 observed", "Year 7 filtered","Year 6
    filtered"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")
plot(line2[7,],type="l",main=expression("Triangle 2"),ylim=c(min(line2[7,],
    line2fit[7,],line2fit[7-1,],na.rm=T),max(line2[7,],line2fit[7,],line2fit
    [7-1,],na.rm=T)))
lines(line2fit[7,],lty=2)
lines(line2fit[7-1,],lty=3)
points(line2[7,],pch=1)
points(line2fit[7,],pch=2)
points(line2fit[7-1,],pch=3)
legend("topright", legend = c( "Year 7 observed", "Year 7 filtered","Year 6
    filtered"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")

plot(line1[12,],type="l",main=expression("Triangle 1"),ylim=c(min(line1[12,],
    line1fit[12,],line1fit[12-1,],na.rm=T),max(line1[12,],line1fit[12,],
    line1fit[12-1,],na.rm=T)))
lines(line1fit[12,],lty=2)
lines(line1fit[12-1,],lty=3)
points(line1[12,],pch=1)
points(line1fit[12,],pch=2)
points(line1fit[12-1,],pch=3)
legend("topright", legend = c( "Year 12 observed", "Year 12 filtered","Year
    11 filtered"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")
plot(line2[12,],type="l",main=expression("Triangle 2"),ylim=c(min(line2[12,],
    line2fit[12,],line2fit[12-1,],na.rm=T),max(line2[12,],line2fit[12,],
    line2fit[12-1,],na.rm=T)))
lines(line2fit[12,],lty=2)
lines(line2fit[12-1,],lty=3)
points(line2[12,],pch=1)
points(line2fit[12,],pch=2)
points(line2fit[12-1,],pch=3)
legend("topright", legend = c( "Year 12 observed", "Year 12 filtered","Year
    11 filtered"),
lty = 1:3,lwd=1, bty = "n",pch=1:3,
title = "")

#Check fitted correlation
cor.test(vector.h1,vector.h2,method="pearson",conf.level = 0.95)
cor.test(psi.vector[,1],psi.vector[,2],method="pearson",conf.level = 0.95)
```

A.3.3 Real data illustration

```
library(magic)
library(MASS)
library(matlib)
library(tweedie)
library(statmod)

# library(nlme)
# library(ggplot2)
# library(xtable)
# library(numDeriv)
# library(dplyr)
# library(truncnorm)
# library(xtable)
# library(ChainLadder)
# library(base)
# library(settings)

###########Import data##########
dline1 = as.matrix(read.csv("AB.csv",header=TRUE))
dline2 = as.matrix(read.csv("NonAB.csv",header=TRUE))

line1=matrix(NA,nrow=10,ncol=10)
line2=matrix(NA,nrow=10,ncol=10)

for (i in 1:10){
for(j in 1:(10-i+1)){
line1[i,j] = dline1[i,j+1]
line2[i,j] = dline2[i,j+1]
}
}

I=10

###########Preliminary analysis - plot development###########
axticks <- seq(1, 10, 1)
axuse <-seq(1,10,1)
par(mfrow=c(1,2))

plot(line1[1,],type = "l",lwd=1,ylim=c(0,max(line1,na.rm=T)*1.1),xlab="
    Development year",xaxt="n", main=expression(paste("Accident Benefits (
    excluding DI)")),ylab="Loss ratio")
for(i in 2:10){
lines(line1[i,],lwd=1)
}
for(i in 1:10){
points(line1[i,],pch=20)
}
axis(side=1,at=axuse, labels=axticks)

plot(line2[1,],type = "l",lwd=1,ylim=c(0,max(line2,na.rm=T)*1.1),xlab="
    Development year",xaxt="n", main=expression(paste("Accident Benefits (DI
    only)")),ylab="Loss ratio")
for(i in 2:10){
lines(line2[i,],lwd=1)
}
for(i in 1:10){
points(line2[i,],pch=20)
}
axis(side=1,at=axuse, labels=axticks)

##########Preliminary analysis - analyse dependence using GLM##########
vline1 = c(t(line1))
vline2 = c(t(line2))

#Set up llh profile
ci.vec = seq(1.01,2.99,by=0.01)
```

```
llh1 = rep(0,length(ci.vec))
llh2 = llh1

i = rep(1:10,each=10)
j = rep(1:10,10)
j1 = rep(c(1,2,rep(0,8)),10)

for (t in 1:length(ci.vec)){
out1 <- glm(vline1~as.factor(i) + as.factor(j1) + log(j) + j-1, fam=tweedie(
    var.power=ci.vec[t],link.power=0))
disp1 <- summary(out1)$dispersion
mu1 <- fitted(out1)
den1 <- dtweedie(out1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2 <-  glm(vline2~as.factor(i) + as.factor(j1)+ log(j) + j-1, fam=tweedie(
    var.power=ci.vec[t],link.power=0))
disp2 <- summary(out2)$dispersion
mu2 <- fitted(out2)
den2 <- dtweedie(out2$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2[t] <- sum(log(den2))
}

ci1 = ci.vec[which.max(llh1)]
ci2 = ci.vec[which.max(llh2)]

#Analyse residuals
out1 <- glm(vline1~as.factor(i) + as.factor(j1)+ log(j) + j -1, fam=tweedie(
    var.power=ci1,link.power=0))
out2 <- glm(vline2~as.factor(i) + as.factor(j1)+ log(j) + j -1, fam=tweedie(
    var.power=ci2,link.power=0))

residuals1 = residuals(out1,type="pearson")
residuals2 = residuals(out2,type="pearson")

tglmpearsoncoef = cor.test(residuals1,residuals2,method="pearson",conf.level
    = 0.95)
tglmkendallcoef = cor.test(residuals1,residuals2,method="kendall",conf.level
    = 0.95)
tglmspearmancoef = cor.test(residuals1,residuals2,method="spearman",conf.
    level = 0.95)

#Prepare for heat maps of residuals
Var1 = c(seq(1:10), seq(1:9),seq(1:8),seq(1:7),seq(1:6),seq(1:5),seq(1:4),seq
    (1:3),seq(1:2),1)
Var2 = rep(10:1,10:1)
fitglm1 = vline1[!is.na(vline1)]/fitted(out1)
fitglm2 = vline2[!is.na(vline2)]/fitted(out2)

heatmap1 <- data.frame(Var1,Var2,fitglm1)
heatmap2 <- data.frame(Var1,Var2,fitglm1)

#Residuals by calendar years (all calendar years combined)
residuals1 = fitted(out1)
residuals2 = fitted(out2)

residuals1_cal = matrix(NA,nrow=10, ncol=10)
residuals1_cal[1,1] = residuals1[1]
residuals1_cal[2,1:2] = c(residuals1[2],residuals1[11])
residuals1_cal[3,1:3] = c(residuals1[3],residuals1[12],residuals1[20])
residuals1_cal[4,1:4] = c(residuals1[4],residuals1[13],residuals1[21],
    residuals1[28])
residuals1_cal[5,1:5] = c(residuals1[5],residuals1[14],residuals1[22],
    residuals1[29],residuals1[35])
residuals1_cal[6,1:6] = c(residuals1[6],residuals1[15],residuals1[23],
    residuals1[30],residuals1[36],residuals1[41])
residuals1_cal[7,1:7] = c(residuals1[7],residuals1[16],residuals1[24],
    residuals1[31],residuals1[37],residuals1[42],residuals1[46])
residuals1_cal[8,1:8] = c(residuals1[8],residuals1[17],residuals1[25],
```

```
    residuals1[32],residuals1[38],residuals1[43],residuals1[47],residuals1
    [50])
residuals1_cal[9,1:9] = c(residuals1[9],residuals1[18],residuals1[26],
    residuals1[33],residuals1[39],residuals1[44],residuals1[48],residuals1
    [51],residuals1[53])
residuals1_cal[10,1:10] = c(residuals1[10],residuals1[19],residuals1[27],
    residuals1[34],residuals1[40],residuals1[45],residuals1[49],residuals1
    [52],residuals1[54],residuals1[55])

residuals2_cal = matrix(NA,nrow=10, ncol=10)
residuals2_cal[1,1] = residuals2[1]
residuals2_cal[2,1:2] = c(residuals2[2],residuals2[11])
residuals2_cal[3,1:3] = c(residuals2[3],residuals2[12],residuals2[20])
residuals2_cal[4,1:4] = c(residuals2[4],residuals2[13],residuals2[21],
    residuals2[28])
residuals2_cal[5,1:5] = c(residuals2[5],residuals2[14],residuals2[22],
    residuals2[29],residuals2[35])
residuals2_cal[6,1:6] = c(residuals2[6],residuals2[15],residuals2[23],
    residuals2[30],residuals2[36],residuals2[41])
residuals2_cal[7,1:7] = c(residuals2[7],residuals2[16],residuals2[24],
    residuals2[31],residuals2[37],residuals2[42],residuals2[46])
residuals2_cal[8,1:8] = c(residuals2[8],residuals2[17],residuals2[25],
    residuals2[32],residuals2[38],residuals2[43],residuals2[47],residuals2
    [50])
residuals2_cal[9,1:9] = c(residuals2[9],residuals2[18],residuals2[26],
    residuals2[33],residuals2[39],residuals2[44],residuals2[48],residuals2
    [51],residuals2[53])
residuals2_cal[10,1:10] = c(residuals2[10],residuals2[19],residuals2[27],
    residuals2[34],residuals2[40],residuals2[45],residuals2[49],residuals2
    [52],residuals2[54],residuals2[55])

sumcal1fit = rowSums(residuals1_cal,na.rm=T)
sumcal2fit = rowSums(residuals2_cal,na.rm=T)

calline1 = vline1[!is.na(vline1)]
calline2 = vline2[!is.na(vline2)]

line1_cal = matrix(NA,nrow=10, ncol=10)
line1_cal[1,1] = calline1[1]
line1_cal[2,1:2] = c(calline1[2],calline1[11])
line1_cal[3,1:3] = c(calline1[3],calline1[12],calline1[20])
line1_cal[4,1:4] = c(calline1[4],calline1[13],calline1[21],calline1[28])
line1_cal[5,1:5] = c(calline1[5],calline1[14],calline1[22],calline1[29],
    calline1[35])
line1_cal[6,1:6] = c(calline1[6],calline1[15],calline1[23],calline1[30],
    calline1[36],calline1[41])
line1_cal[7,1:7] = c(calline1[7],calline1[16],calline1[24],calline1[31],
    calline1[37],calline1[42],calline1[46])
line1_cal[8,1:8] = c(calline1[8],calline1[17],calline1[25],calline1[32],
    calline1[38],calline1[43],calline1[47],calline1[50])
line1_cal[9,1:9] = c(calline1[9],calline1[18],calline1[26],calline1[33],
    calline1[39],calline1[44],calline1[48],calline1[51],calline1[53])
line1_cal[10,1:10] = c(calline1[10],calline1[19],calline1[27],calline1[34],
    calline1[40],calline1[45],calline1[49],calline1[52],calline1[54],calline1
    [55])

line2_cal = matrix(NA,nrow=10, ncol=10)
line2_cal[1,1] = calline2[1]
line2_cal[2,1:2] = c(calline2[2],calline2[11])
line2_cal[3,1:3] = c(calline2[3],calline2[12],calline2[20])
line2_cal[4,1:4] = c(calline2[4],calline2[13],calline2[21],calline2[28])
line2_cal[5,1:5] = c(calline2[5],calline2[14],calline2[22],calline2[29],
    calline2[35])
line2_cal[6,1:6] = c(calline2[6],calline2[15],calline2[23],calline2[30],
    calline2[36],calline2[41])
line2_cal[7,1:7] = c(calline2[7],calline2[16],calline2[24],calline2[31],
    calline2[37],calline2[42],calline2[46])
line2_cal[8,1:8] = c(calline2[8],calline2[17],calline2[25],calline2[32],
    calline2[38],calline2[43],calline2[47],calline2[50])
line2_cal[9,1:9] = c(calline2[9],calline2[18],calline2[26],calline2[33],
```

```
    calline2[39],calline2[44],calline2[48],calline2[51],calline2[53])
line2_cal[10,1:10] = c(calline2[10],calline2[19],calline2[27],calline2[34],
    calline2[40],calline2[45],calline2[49],calline2[52],calline2[54],calline2
    [55])

sumcal1obs = rowSums(line1_cal,na.rm=T)
sumcal2obs = rowSums(line2_cal,na.rm=T)

diff1nocal = (sumcal1obs-sumcal1fit)/sumcal1fit
diff2nocal = (sumcal2obs-sumcal2fit)/sumcal2fit

par(mar=c(4,4,1,1))
plot(diff1nocal,type="l",ylim=c(min(diff1nocal,diff2nocal),max(diff1nocal,
    diff2nocal)),ylab="Residuals",xlab="Calendar year")
lines(diff2nocal,lty=2)
points(diff1nocal,pch=20)
points(diff2nocal,pch=20)
legend("top", legend = c("Accident Benefits (excluding DI)", "Accident
    Benefits (DI only)"),
lty = 1:2,lwd=1, bty = "n",
title = "")

#Plot the last 3 CYs
residualsglm1_cal = matrix(NA,nrow=10, ncol=10)
residualsglm1_cal[1,1] = fitglm1[1]
residualsglm1_cal[2,1:2] = c(fitglm1[2],fitglm1[11])
residualsglm1_cal[3,1:3] = c(fitglm1[3],fitglm1[12],fitglm1[20])
residualsglm1_cal[4,1:4] = c(fitglm1[4],fitglm1[13],fitglm1[21],fitglm1[28])
residualsglm1_cal[5,1:5] = c(fitglm1[5],fitglm1[14],fitglm1[22],fitglm1[29],
    fitglm1[35])
residualsglm1_cal[6,1:6] = c(fitglm1[6],fitglm1[15],fitglm1[23],fitglm1[30],
    fitglm1[36],fitglm1[41])
residualsglm1_cal[7,1:7] = c(fitglm1[7],fitglm1[16],fitglm1[24],fitglm1[31],
    fitglm1[37],fitglm1[42],fitglm1[46])
residualsglm1_cal[8,1:8] = c(fitglm1[8],fitglm1[17],fitglm1[25],fitglm1[32],
    fitglm1[38],fitglm1[43],fitglm1[47],fitglm1[50])
residualsglm1_cal[9,1:9] = c(fitglm1[9],fitglm1[18],fitglm1[26],fitglm1[33],
    fitglm1[39],fitglm1[44],fitglm1[48],fitglm1[51],fitglm1[53])
residualsglm1_cal[10,1:10] = c(fitglm1[10],fitglm1[19],fitglm1[27],fitglm1
    [34],fitglm1[40],fitglm1[45],fitglm1[49],fitglm1[52],fitglm1[54],fitglm1
    [55])

residualsglm2_cal = matrix(NA,nrow=10, ncol=10)
residualsglm2_cal[1,1] = fitglm2[1]
residualsglm2_cal[2,1:2] = c(fitglm2[2],fitglm2[11])
residualsglm2_cal[3,1:3] = c(fitglm2[3],fitglm2[12],fitglm2[20])
residualsglm2_cal[4,1:4] = c(fitglm2[4],fitglm2[13],fitglm2[21],fitglm2[28])
residualsglm2_cal[5,1:5] = c(fitglm2[5],fitglm2[14],fitglm2[22],fitglm2[29],
    fitglm2[35])
residualsglm2_cal[6,1:6] = c(fitglm2[6],fitglm2[15],fitglm2[23],fitglm2[30],
    fitglm2[36],fitglm2[41])
residualsglm2_cal[7,1:7] = c(fitglm2[7],fitglm2[16],fitglm2[24],fitglm2[31],
    fitglm2[37],fitglm2[42],fitglm2[46])
residualsglm2_cal[8,1:8] = c(fitglm2[8],fitglm2[17],fitglm2[25],fitglm2[32],
    fitglm2[38],fitglm2[43],fitglm2[47],fitglm2[50])
residualsglm2_cal[9,1:9] = c(fitglm2[9],fitglm2[18],fitglm2[26],fitglm2[33],
    fitglm2[39],fitglm2[44],fitglm2[48],fitglm2[51],fitglm2[53])
residualsglm2_cal[10,1:10] = c(fitglm2[10],fitglm2[19],fitglm2[27],fitglm2
    [34],fitglm2[40],fitglm2[45],fitglm2[49],fitglm2[52],fitglm2[54],fitglm2
    [55])

par(mfrow=c(3,1))
par(mar=c(2.5,2.5,2.5,2.5))
axticks <- seq(1, 10, 1)
axuse <-seq(1,10,1)
for(i in 8:10){
plot(residualsglm1_cal[i,1:i],type="l",ylim=c(min(residualsglm1_cal[i,1:i],
    residualsglm2_cal[i,1:i],na.rm=T),max(residualsglm1_cal[i,1:i],
    residualsglm2_cal[i,1:i],na.rm=T)),ylab="Residuals",xlab="Accident year")
lines(residualsglm2_cal[i,1:i],type="l",lty=2)
```

```
points(residualsglm1_cal[i,1:i],pch=19)
points(residualsglm2_cal[i,1:i],pch=19)
legend("top", legend = c("Accident Benefits (excluding DI)", "Accident
    Benefits (DI only)"),
lty = 1:2,lwd=1, bty = "n",
title = "")
axis(side=1,at=axuse, labels=axticks)
}

##########Additional preliminary analysis to choose initial values#########

#Year 1 glm
j = c(1:10)
j1 = c(1,2,rep(0,8))

ci.vec = seq(1.01,2.99,by=0.01)

llh1yr1 = rep(0,length(ci.vec))
llh2yr1 = llh1yr1

for (t in 1:length(ci.vec)){
out1yr1 <- glm(vline1[1:10]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci.vec[t],link.power=0))
disp1 <- summary(out1yr1)$dispersion
mu1 <- fitted(out1yr1)
den1 <- dtweedie(out1yr1$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1yr1[t] <- sum(log(den1))
}


for (t in 1:length(ci.vec)){
out2yr1 <-  glm(vline2[1:10]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci.vec[t],link.power=0))
disp2 <- summary(out2yr1)$dispersion
mu2 <- fitted(out2yr1)
den2 <- dtweedie(out2yr1$y, mu = mu2, phi = disp2, power = ci.vec[t])
llh2yr1[t] <- sum(log(den2))
}

ci1yr1 = ci.vec[which.max(llh1yr1)]
ci2yr1 = ci.vec[which.max(llh2yr1)]

out1yr1 <- glm(vline1[1:10]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci1yr1,link.power=0))
out2yr1 <- glm(vline2[1:10]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci2yr1,link.power=0))

#Year 2 glm
j = c(1:9)
j1 = c(1,2,rep(0,7))

ci.vec = seq(1.01,2.99,by=0.01)

llh1yr2 = rep(0,length(ci.vec))
llh2yr2 = llh1yr2

for (t in 1:length(ci.vec)){
out1yr2 <- glm(line1[2,1:9]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci.vec[t],link.power=0))
disp1 <- summary(out1yr2)$dispersion
mu1 <- fitted(out1yr2)
den1 <- dtweedie(out1yr2$y, mu = mu1, phi = disp1, power = ci.vec[t])
llh1yr2[t] <- sum(log(den1))
}

for (t in 1:length(ci.vec)){
out2yr2 <-  glm(line2[2,1:9]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci.vec[t],link.power=0))
disp2 <- summary(out2yr2)$dispersion
mu2 <- fitted(out2yr2)
den2 <- dtweedie(out2yr2$y, mu = mu2, phi = disp2, power = ci.vec[t])
```

```
llh2yr2[t] <- sum(log(den2))
}

ci1yr2 = ci.vec[which.max(llh1yr2)]
ci2yr2 = ci.vec[which.max(llh2yr2)]

out1yr2 <- glm(line1[2,1:9]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci1yr2,link.power=0))
out2yr2 <- glm(line2[2,1:9]~ log(j) + j + as.factor(j1), fam=tweedie(var.
    power=ci2yr2,link.power=0))

##########Run particle filter##########

#Set up matrices
y_kf = list()
a_kf = list()
h_kf = list()
e_kf = list()

e_kf[[1]] = diag(2*I)
y_kf[[1]] = c(line1[1,],line2[1,])

an = matrix(0,nrow=(I),ncol=5)
an[1:nrow(an),1] = rep(1,nrow(an))
an[,2] = c(log(seq(from=1,to=I)))
an[,3] = c((seq(from=1,to=(I))))
an[1:2,4:5] = diag(2)
a_kf[[1]] = cbind(adiag(an,an))

for(i in 2:I){
en = matrix(0,nrow=(I-i+1),ncol=i-1)
en = cbind(en,diag(I-i+1))
e_kf[[i]] = adiag(en,en)

y_kf[[i]] = c(line1[i,1:(I-i+1)],line2[i,1:(I-i+1)])

if(i<I){
an = matrix(0,nrow=(I+1-i),ncol=5)
an[1:nrow(an),1] = rep(1,nrow(an))
an[,2] = c(log(seq(from=1,to=(I+1-i))))
an[,3] = c((seq(from=1,to=(I-i+1))))
an[1:2,4:5] = diag(2)
A = adiag(an,an)}

if(i==I){
A = matrix(NA,nrow=2,ncol=10)
A[1,] = c(1,0,1,1,0,rep(0,5))
A[2,] = c(rep(0,5),1,0,1,1,0)
}
a_kf[[i]] = A}

#Initialise filter
a = 0.95
h  = sqrt(1-a^2)
set.seed(2000)
N  = 50000
gamma_list = list()
gamma.vector=matrix(NA,nrow=I,ncol=10)
psi.vector = matrix(NA,nrow=I,ncol=2)

phi1 = 0.005
phi2 = 0.005
a1sd = 0.004
r1sd = 0.05
s1sd = 0.01
a2sd =0.005
r2sd = 0.08
s2sd = 0.02
b11sd = 0.2
b21sd = 0.15
b12sd = 0.2
b22sd = 0.1
```

```
commonsd = 0.12
h1sd = 0.08
h2sd = 0.07
lambda1 = 0.6
lambda2 = 0.6
power1 = atanh(1.2-2.01)
power2 = atanh(1.3-2.01)

powerfunc = function(x){
tanh(x)+2.01
}
theta_init = c(log(c(phi1,phi2,a1sd,r1sd,s1sd,b11sd,b21sd,a2sd,r2sd,s2sd,
    b12sd,b22sd,commonsd,h1sd,h2sd)),log(c(lambda1,lambda2)),power1,power2)
varest_init = abs(c(log(c(phi1,phi2,a1sd,r1sd,s1sd,b11sd,b21sd,a2sd,r2sd,s2sd
    ,b12sd,b22sd,commonsd,h1sd,h2sd))*0.01,log(c(lambda1,lambda2))*0.15,power1
    *0.2,power2*0.2))
state_init = c(-2,1.3,-0.7,0.56,0.15,-3.7,2,-0.88,0.93,0.28)

theta = matrix(NA,nrow=N,ncol=19)
theta= mvrnorm(N,theta_init,Sigma = diag(varest_init))
thetaexp = exp(theta)
thetaexp[,18:19] = powerfunc(theta[,18:19])

gamma.filter = matrix(0,nrow=N,ncol=10)
psi.filter = matrix(0,nrow=N,ncol=20)
psi.filter[,1] = rep(0,N)
psi.filter[,11] = rep(0,N)

w1 = rep(0,N)

for (i in 1:N){
gamma.filter[i,] = mvrnorm(1,mu=state_init,Sigma = diag(thetaexp[i,3:12]))
for (j in 2:10){
commonshock.e = rnorm(1,mean=0,sd=thetaexp[i,13])
psi.filter[i,j] = psi.filter[i,j-1]+thetaexp[i,16]*commonshock.e + rnorm(1,
    mean=0,sd=thetaexp[i,14])
psi.filter[i,j+10] = psi.filter[i,I+j-1]+thetaexp[i,17]*commonshock.e + rnorm
    (1,mean=0,sd=thetaexp[i,15])    }

logmu1 = (exp(a_kf[[1]]%*%gamma.filter[i,] + e_kf[[1]]%*%psi.filter[i,]))

if (any(logmu1==0)|any(!is.finite(logmu1))) { w1[i] = -1e10}
else {
w1[i] = sum(log(dtweedie(y_kf[[1]][1:I],mu = logmu1[1:I],phi = (thetaexp[i
    ,1]),power=thetaexp[i,18])))+sum(log(dtweedie(y_kf[[1]][(I+1):(2*I)],mu =
    logmu1[(I+1):(2*I)],phi = (thetaexp[i,2]),power=thetaexp[i,19])))
}}

w_norm = exp(w1)/sum(exp(w1))

k = sample(1:N, replace=T,size=N,prob=w_norm)
nlevels(as.factor(k))

gamma.filter = gamma.filter[k,1:ncol(gamma.filter)]
gamma_list[[1]] = gamma.filter
psi.filter = psi.filter[k,1:ncol(psi.filter)]
theta = theta[k,1:ncol(theta)]
thetaexp = exp(theta)
thetaexp[,18:19] = powerfunc(theta[,18:19])

gamma.vector[1,1] = colMeans(gamma.filter)[1]
gamma.vector[1,2] = colMeans(gamma.filter)[2]
gamma.vector[1,3] = colMeans(gamma.filter)[3]
gamma.vector[1,4] = colMeans(gamma.filter)[4]
gamma.vector[1,5] = colMeans(gamma.filter)[5]
gamma.vector[1,6] = colMeans(gamma.filter)[6]
gamma.vector[1,7] = colMeans(gamma.filter)[7]
gamma.vector[1,8] = colMeans(gamma.filter)[8]
gamma.vector[1,9] = colMeans(gamma.filter)[9]
gamma.vector[1,10] = colMeans(gamma.filter)[10]
```

```
psi.vector[1,1] = colMeans(psi.filter)[1]
psi.vector[1,2] = colMeans(psi.filter)[11]

y.fit = exp(a_kf[[1]]%*%colMeans(gamma.filter) + e_kf[[1]]%*%colMeans(psi.
    filter))

line1fit = matrix(NA,nrow=I,ncol=I)
line2fit = matrix(NA,nrow=I,ncol=I)
line1fit[1,] = y.fit[1:I]
line2fit[1,] = y.fit[(I+1):(2*I)]

par(mfrow=c(2,1))
plot(line1fit[1,],type="l")
lines(line1[1,],lty=2)
plot(line2fit[1,],type="l")
lines(line2[1,],lty=2)

LogFile=""

#Run filter
for (i in 2:I){
cat(i, sep=" ")

#Project parameter and calendar factor values
thetahat = a*theta + rep((1-a)*apply(theta,2,mean),each=nrow(theta))
thetahatexp = exp(thetahat)
thetahatexp[,18:19] = powerfunc(thetahat[,18:19])

psihat = a*psi.filter + rep((1-a)*apply(psi.filter,2,mean),each=nrow(psi.
    filter))

#Compute look-ahead importance weights
w = rep(0,N)
w_e = rep(0,N)

for (j in 1:N){
logmu = exp(a_kf[[i]]%*% ( gamma.filter[j,]) + e_kf[[i]]%*%psihat[j,])
if (any(logmu==0)|any(!is.finite(logmu))) { w[j] = -1e10}
else {
w_e[j] =  sum(log(dtweedie(y_kf[[i]][1:(I-i+1)],mu = logmu[1:(I-i+1)],phi = (
    thetahatexp[j,1]),power=thetahatexp[j,18])))+sum(log(dtweedie(y_kf[[i]][(I
    -i+2):length(y_kf[[i]])],mu = logmu[(I-i+2):length(y_kf[[i]])],phi =
    thetahatexp[j,2],power=thetahatexp[j,19])))
w[j] = w_e[j]+ log(w_norm[j])}}

w_norm = exp(w)/sum(exp(w))

#Resample
k = sample(1:N, replace=T,size=N,prob=w_norm)

#Draw parameter and calendar factors with the resampled index
varest=apply(theta,2,var)
theta1 = t(apply(thetahat[k,1:19],1,function(x){mvrnorm(1,x,diag(varest*(h^2)
    ))}))
thetaexp1 = exp(theta1)
thetaexp1[,18:19] = powerfunc(theta1[,18:19])

psivarest = apply(psi.filter,2,var)
psi.filter1 = t(apply(psihat[k,1:20],1,function(x){mvrnorm(1,x,diag(psivarest
    *(h^2)))}))

gamma.filter = gamma.filter[k,1:ncol(gamma.filter)]
gamma.filter1 = matrix(NA,nrow=N,ncol = 10)

#Calculate importance weights
w1 = rep(0,N)

for (j in 1:N){
gamma.filter1[j,] =  gamma.filter[j,] +mvrnorm(n=1,mu=rep(0,10),Sigma = diag(
    thetaexp1[j,3:12]))
logmu1 = exp(a_kf[[i]]%*% gamma.filter1[j,]+ e_kf[[i]]%*%psi.filter1[j,])
if (any(logmu1==0)|any(!is.finite(logmu1))) { w1[j] = -1e10}
```

```
else {
w1[j] =sum(log(dtweedie(y_kf[[i]][1:(I-i+1)],mu = logmu1[1:(I-i+1)],phi = (
    thetaexp1[j,1]),power=thetaexp1[j,18])))+sum(log(prod(dtweedie(y_kf[[i]][(
    I-i+2):length(y_kf[[i]])],mu = logmu1[(I-i+2):length(y_kf[[i]])],phi =
    thetaexp1[j,2],power=thetaexp1[j,19])))-w_e[k][j]
}}

w_norm = exp(w1)/sum(exp(w1))

#Calculate filtered statistics and resample for next period
k = sample(1:N, replace=T,size=N,prob=w_norm)
cat(", level=",nlevels(as.factor(k)), "\n", sep = "", file = LogFile, append
    = TRUE)

gamma.filter = gamma.filter1[k,1:ncol(gamma.filter1)]
psi.filter = psi.filter1[k,1:ncol(psi.filter1)]
theta = theta1[k,1:ncol(theta1)]
thetaexp = exp(theta)
thetaexp[,18:19] = powerfunc(theta[,18:19])

gamma_list[[i]] = gamma.filter
gamma.vector[i,1] = colMeans(gamma.filter)[1]
gamma.vector[i,2] = colMeans(gamma.filter)[2]
gamma.vector[i,3] = colMeans(gamma.filter)[3]
gamma.vector[i,4] = colMeans(gamma.filter)[4]
gamma.vector[i,5] = colMeans(gamma.filter)[5]
gamma.vector[i,6] = colMeans(gamma.filter)[6]
gamma.vector[i,7] = colMeans(gamma.filter)[7]
gamma.vector[i,8] = colMeans(gamma.filter)[8]
gamma.vector[i,9] = colMeans(gamma.filter)[9]
gamma.vector[i,10] = colMeans(gamma.filter)[10]
psi.vector[i,1] = colMeans(psi.filter)[1]
psi.vector[i,2] = colMeans(psi.filter)[10+i]

y.fit = exp(a_kf[[i]]%*%colMeans(gamma.filter)+ e_kf[[i]]%*%colMeans(psi.
    filter))

line1fit[i,1:(I-i+1)] = y.fit[1:(I-i+1)]
line2fit[i,1:(I-i+1)] = y.fit[(I-i+2):length(y.fit)]

plot(line1fit[i,],type="l",ylim=c(min(line1fit[i,],line1[i,],line1fit[i-1,],
    na.rm=T),max(line1fit[i,],line1fit[i-1,],line1[i,],na.rm=T)))
lines(line1[i,],lty=2)
lines(line1fit[i-1,],lty=3)

plot(line2fit[i,],type="l",ylim=c(min(line2fit[i,],line2[i,],line2fit[i-1,],
    na.rm=T),max(line2fit[i,],line2fit[i-1,],line2[i,],na.rm=T)))
lines(line2[i,],lty=2)
lines(line2fit[i-1,],lty=3)
}

##########Summarise results and check goodness of fit##########
state_est = cbind(gamma.vector,psi.vector)

par(mfrow=c(3,2))
for(i in 1:12){
plot(state_est[,i],type="l",lty=1)
}

parameterest = cbind(apply(thetaexp,2,mean),apply(thetaexp,2,quantile,probs
    =0.05),apply(thetaexp,2,quantile,probs=0.95))

#Prepare data for heatmap
res1 = line1/line1fit
res2 = line2/line2fit
res1 = as.vector((res1))
res2 = as.vector((res2))
res1 = res1[!is.na(res1)]
res2 = res2[!is.na(res2)]

Var2 = c(c(10:1),c(10:2),c(10:3),c(10:4),c(10:5),c(10:6),c(10:7),c(10:8),c
```

```
    (10:9),10)
Var1 = rep(1:I,I:1)

heatmap1 <- data.frame(Var1,Var2,res1)
heatmap2 <- data.frame(Var1,Var2,res2)

#Plot the tracking of development patterns
par(mar=c(2.5,2.5,2.5,2.5))
par(mfrow=c(4,2))
plot(line1[1,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[1,],line1fit[1,],na.rm=T),max(line1[1,],line1fit[1,],na.
    rm=T)))
lines(line1fit[1,],lty=2)
points(line1[1,],pch=1)
points(line1fit[1,],pch=2)
legend("topright", legend = c( "Year 1 observed", "Year 1 filtered"), lty =
    1:2,lwd=1, bty = "n",pch=1:2, title = "")

plot(line2[1,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[1,],line2fit[1,],na.rm=T),max(line2[1,],line2fit[1,],na.rm=T)))
lines(line2fit[1,],lty=2)
points(line2[1,],pch=1)
points(line2fit[1,],pch=2)
legend("topright", legend = c( "Year 1 observed", "Year 1 filtered"), lty =
    1:2,lwd=1, bty = "n",pch=1:2,title = "")

plot(line1[2,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[2,],line1fit[2,],line1fit[2-1,],na.rm=T),max(line1[2,],
    line1fit[2,],line1fit[2-1,],na.rm=T)))
lines(line1fit[2,],lty=2)
lines(line1fit[2-1,],lty=3)
points(line1[2,],pch=1)
points(line1fit[2,],pch=2)
points(line1fit[2-1,],pch=3)
legend("topright", legend = c( "Year 2 observed", "Year 2 filtered","Year 1
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

plot(line2[2,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[2,],line2fit[2,],line2fit[2-1,],na.rm=T),max(line2[2,],line2fit
    [2,],line2fit[2-1,],na.rm=T)))
lines(line2fit[2,],lty=2)
lines(line2fit[2-1,],lty=3)
points(line2[2,],pch=1)
points(line2fit[2,],pch=2)
points(line2fit[2-1,],pch=3)
legend("topright", legend = c( "Year 2 observed", "Year 2 filtered","Year 1
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

plot(line1[3,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[3,],line1fit[3,],line1fit[3-1,],na.rm=T),max(line1[3,],
    line1fit[3,],line1fit[3-1,],na.rm=T)))
lines(line1fit[3,],lty=2)
lines(line1fit[3-1,],lty=3)
points(line1[3,],pch=1)
points(line1fit[3,],pch=2)
points(line1fit[3-1,],pch=3)
legend("topright", legend = c( "Year 3 observed", "Year 3 filtered","Year 2
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

plot(line2[3,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[3,],line2fit[3,],line2fit[3-1,],na.rm=T),max(line2[3,],line2fit
    [3,],line2fit[3-1,],na.rm=T)))
lines(line2fit[3,],lty=2)
lines(line2fit[3-1,],lty=3)
points(line2[3,],pch=1)
points(line2fit[3,],pch=2)
points(line2fit[3-1,],pch=3)
legend("topright", legend = c( "Year 3 observed", "Year 3 filtered","Year 2
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")
```

```
plot(line1[4,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[4,],line1fit[4,],line1fit[4-1,],na.rm=T),max(line1[4,],
    line1fit[4,],line1fit[4-1,],na.rm=T)))
lines(line1fit[4,],lty=2)
lines(line1fit[4-1,],lty=3)
points(line1[4,],pch=1)
points(line1fit[4,],pch=2)
points(line1fit[4-1,],pch=3)
legend("topright", legend = c( "Year 4 observed", "Year 4 filtered","Year 3
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

plot(line2[4,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[4,],line2fit[4,],line2fit[4-1,],na.rm=T),max(line2[4,],line2fit
    [4,],line2fit[4-1,],na.rm=T)))
lines(line2fit[4,],lty=2)
lines(line2fit[4-1,],lty=3)
points(line2[4,],pch=1)
points(line2fit[4,],pch=2)
points(line2fit[4-1,],pch=3)
legend("topright", legend = c( "Year 4 observed", "Year 4 filtered","Year 3
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

par(mfrow=c(5,2))
plot(line1[5,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[5,],line1fit[5,],line1fit[5-1,],na.rm=T),max(line1[5,],
    line1fit[5,],line1fit[5-1,],na.rm=T)))
lines(line1fit[5,],lty=2)
lines(line1fit[5-1,],lty=3)
points(line1[5,],pch=1)
points(line1fit[5,],pch=2)
points(line1fit[5-1,],pch=3)
legend("topright", legend = c( "Year 5 observed", "Year 5 filtered","Year 4
    filtered"), lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

plot(line2[5,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[5,],line2fit[5,],line2fit[5-1,],na.rm=T),max(line2[5,],line2fit
    [5,],line2fit[5-1,],na.rm=T)))
lines(line2fit[5,],lty=2)
lines(line2fit[5-1,],lty=3)
points(line2[5,],pch=1)
points(line2fit[5,],pch=2)
points(line2fit[5-1,],pch=3)
legend("topright", legend = c( "Year 5 observed", "Year 5 filtered","Year 4
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

plot(line1[6,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[6,],line1fit[6,],line1fit[6-1,],na.rm=T),max(line1[6,],
    line1fit[6,],line1fit[6-1,],na.rm=T)))
lines(line1fit[6,],lty=2)
lines(line1fit[6-1,],lty=3)
points(line1[6,],pch=1)
points(line1fit[6,],pch=2)
points(line1fit[6-1,],pch=3)
legend("topright", legend = c( "Year 6 observed", "Year 6 filtered","Year 5
    filtered"), lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

plot(line2[6,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[6,],line2fit[6,],line2fit[6-1,],na.rm=T),max(line2[6,],line2fit
    [6,],line2fit[6-1,],na.rm=T)))
lines(line2fit[6,],lty=2)
lines(line2fit[6-1,],lty=3)
points(line2[6,],pch=1)
points(line2fit[6,],pch=2)
points(line2fit[6-1,],pch=3)
legend("topright", legend = c( "Year 6 observed", "Year 6 filtered","Year 5
    filtered"), lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

plot(line1[7,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[7,],line1fit[7,],line1fit[7-1,],na.rm=T),max(line1[7,],
    line1fit[7,],line1fit[7-1,],na.rm=T)))
```

```
lines(line1fit[7,],lty=2)
lines(line1fit[7-1,],lty=3)
points(line1[7,],pch=1)
points(line1fit[7,],pch=2)
points(line1fit[7-1,],pch=3)
legend("topright", legend = c( "Year 7 observed", "Year 7 filtered","Year 6
    filtered"), lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

plot(line2[7,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[7,],line2fit[7,],line2fit[7-1,],na.rm=T),max(line2[7,],line2fit
    [7,],line2fit[7-1,],na.rm=T)))
lines(line2fit[7,],lty=2)
lines(line2fit[7-1,],lty=3)
points(line2[7,],pch=1)
points(line2fit[7,],pch=2)
points(line2fit[7-1,],pch=3)
legend("topright", legend = c( "Year 7 observed", "Year 7 filtered","Year 6
    filtered"), lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

plot(line1[8,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[8,],line1fit[8,],line1fit[8-1,],na.rm=T),max(line1[8,],
    line1fit[8,],line1fit[8-1,],na.rm=T)))
lines(line1fit[8,],lty=2)
lines(line1fit[8-1,],lty=3)
points(line1[8,],pch=1)
points(line1fit[8,],pch=2)
points(line1fit[8-1,],pch=3)
legend("topright", legend = c( "Year 8 observed", "Year 8 filtered","Year 7
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

plot(line2[8,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[8,],line2fit[8,],line2fit[8-1,],na.rm=T),max(line2[8,],line2fit
    [8,],line2fit[8-1,],na.rm=T)))
lines(line2fit[8,],lty=2)
lines(line2fit[8-1,],lty=3)
points(line2[8,],pch=1)
points(line2fit[8,],pch=2)
points(line2fit[8-1,],pch=3)
legend("topright", legend = c( "Year 8 observed", "Year 8 filtered","Year 7
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

plot(line1[9,],type="l",main=expression("Accident Benefits (excluding DI)"),
    ylim=c(min(line1[9,],line1fit[9,],line1fit[9-1,],na.rm=T),max(line1[9,],
    line1fit[9,],line1fit[9-1,],na.rm=T)))
lines(line1fit[9,],lty=2)
lines(line1fit[9-1,],lty=3)
points(line1[9,],pch=1)
points(line1fit[9,],pch=2)
points(line1fit[9-1,],pch=3)
legend("topright", legend = c( "Year 9 observed", "Year 9 filtered","Year 8
    filtered"),lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

plot(line2[9,],type="l",main=expression("Accident Benefits (DI only)"),ylim=c
    (min(line2[9,],line2fit[9,],line2fit[9-1,],na.rm=T),max(line2[9,],line2fit
    [9,],line2fit[9-1,],na.rm=T)))
lines(line2fit[9,],lty=2)
lines(line2fit[9-1,],lty=3)
points(line2[9,],pch=1)
points(line2fit[9,],pch=2)
points(line2fit[9-1,],pch=3)
legend("topright", legend = c( "Year 9 observed", "Year 9 filtered","Year 8
    filtered"), lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

#Plot residuals by accident year, development year and calendar year
devratio1 = colMeans(line1/line1fit,na.rm=T)
devratio2 = colMeans(line2/line2fit,na.rm=T)
accratio1 = rowMeans(line1/line1fit,na.rm=T)
accratio2 = rowMeans(line2/line2fit,na.rm=T)

fitratio1_cal = matrix(NA,nrow=10, ncol=10)
```

```
fitratio2_cal = matrix(NA,nrow=10, ncol=10)
for (t in 1:10){
for (i in 1:t){
fitratio1_cal[t,i] = line1[i,(t+1-i)]/line1fit[i,(t+1-i)]
fitratio2_cal[t,i] = line2[i,(t+1-i)]/line2fit[i,(t+1-i)]}}
calratio1 = rowMeans(fitratio1_cal,na.rm=T)
calratio2 = rowMeans(fitratio2_cal,na.rm=T)

par(mfrow=c(1,2))
plot(devratio1,ylim=c(0.4,1.8),type="l",xlab="Year", main=expression(paste("
    Accident Benefits (excluding DI)")),ylab="Residuals")
lines(accratio1,lty=2)
lines(calratio1,lty=3)
points(devratio1,pch=1)
points(accratio1,pch=2)
points(calratio1,pch=3)
legend("topleft", legend = c( "Development year", "Accident year","Calendar
    year"), lty = 1:3,lwd=1, bty = "n",pch=1:3,title = "")

plot(devratio2,ylim=c(0.05,1.4),type="l",xlab="Year", main=expression(paste("
    Accident Benefits (DI only)")),ylab="Residuals")
lines(accratio2,lty=2)
lines(calratio2,lty=3)
points(devratio2,pch=1)
points(accratio2,pch=2)
points(calratio2,pch=3)
legend("bottomleft", legend = c( "Development year", "Accident year","
    Calendar year"), lty = 1:3,lwd=1, bty = "n",pch=1:3, title = "")

#Check fitted correlation coefficients
theoreticalcor = (0.7135*0.6614*0.1114)/(sqrt(0.7135*0.1114+0.0755)*sqrt
    (0.6614*0.1114+0.1079))
samplecor = cor.test(psi.vector[,1],psi.vector[,2])

#Check residuals by calendar year
obscal1 = matrix(NA,nrow=10, ncol=10)
obscal2 = matrix(NA,nrow=10, ncol=10)
fitcal1 = matrix(NA,nrow=10, ncol=10)
fitcal2 = matrix(NA,nrow=10, ncol=10)

for (t in 1:10){
for (i in 1:t){
obscal1[t,i] = line1[i,(t+1-i)]
obscal2[t,i] = line2[i,(t+1-i)]
fitcal1[t,i] = line1fit[i,(t+1-i)]
fitcal2[t,i] = line2fit[i,(t+1-i)]
}
}

calratiodev1 = (rowSums(obscal1,na.rm=T)-rowSums(fitcal1,na.rm=T))/rowSums(
    fitcal1,na.rm=T)
calratiodev2 = (rowSums(obscal2,na.rm=T)-rowSums(fitcal2,na.rm=T))/rowSums(
    fitcal2,na.rm=T)

par(mar=c(4,4,1,1))
par(mfrow=c(1,1))
plot(calratiodev1,type="l",ylim=c(min(calratiodev1,calratiodev2),max(
    calratiodev1,calratiodev2)),ylab="Residuals",xlab="Calendar year")
lines(calratiodev2,lty=2)
points(calratiodev1,pch=20)
points(calratiodev2,pch=20)
legend("topleft", legend = c("Accident Benefits (excluding DI)", "Accident
    Benefits (DI only)"),lty = 1:2,lwd=1, bty = "n",title = "")
abline(h=0)

##########Forecast future outstanding claims##########
#Set up matrices and prepare relevant information
prem = c(116491,    111467, 107241, 105687, 105923, 111487, 113268, 121606,
    110610, 104304)
proj_theta = thetaexp
```

```
proj_psi = matrix(NA,nrow=N,ncol=18)
est_cal = psi.filter
proj_gamma = gamma_list

proj_y = array(NA,c(N,18,9))
proj_a_kf = list()
proj_ay = array(NA,c(N,3,9))
proj_e_kf = list()
total = matrix(NA,N,3)

for(i in 2:I){
en = matrix(0,nrow=(I-(I-i+1)),ncol=I-1)
en[1:nrow(en),1:nrow(en)] = diag(nrow(en))
E = adiag(en,en)
proj_e_kf[[i]] = E

an = matrix(0,nrow=(I-(I-i+1)),ncol=5)
an[1:nrow(an),1] = rep(1,nrow(an))
an[,2] = c(log(seq(from=(I-i+2),to=I)))
an[,3] = c((seq(from=(I-i+2),to=I)))
if(i==10){
an[1,5] = 1}
A = adiag(an,an)
proj_a_kf[[i]] = A}

#Project future calendar factors
set.seed(2000)
for (t in 1:N){
commonshock.e = rnorm(1,mean=0,sd=proj_theta[t,13])
proj_psi[t,1] = est_cal[t,10]+proj_theta[t,16]*commonshock.e + rnorm(1,mean
    =0,sd=proj_theta[t,14])
proj_psi[t,10] = est_cal[t,20]+proj_theta[t,17]*commonshock.e + rnorm(1,mean
    =0,sd=proj_theta[t,15])
for(i in 2:9){
commonshock.e = rnorm(1,mean=0,sd=proj_theta[t,13])
proj_psi[t,i] = proj_psi[t,i-1]+proj_theta[i,16]*commonshock.e + rnorm(1,mean
    =0,sd=proj_theta[i,14])
proj_psi[t,9+i] = proj_psi[t,i+8]+proj_theta[i,17]*commonshock.e + rnorm(1,
    mean=0,sd=proj_theta[i,15])}

#Project future claims
y.forecast = matrix(NA,nrow=18,ncol=9)
for(i in 2:I){
logmu1 = (exp(proj_a_kf[[i]]%*%proj_gamma[[i]][t,] + proj_e_kf[[i]]%*%proj_
    psi[t,]))
y.forecast1 = rep(NA,2*(i-1))
phivector = c(rep(proj_theta[t,1],i-1),rep(proj_theta[t,2],i-1))
pvector = c(rep(proj_theta[t,18],i-1),rep(proj_theta[t,19],i-1))
for (n in 1:length(y.forecast1)){
y.forecast1[n] = rtweedie(1,mu = logmu1[n],phi=phivector[n],xi = pvector[n])}

y.forecast[1:(i-1),i-1] = y.forecast1[1:(length(y.forecast1)/2)]*prem[i]
y.forecast[I:(I+i-2),i-1] = y.forecast1[((length(y.forecast1)/2)+1):length(y.
    forecast1)]*prem[i]
}

proj_y[t,,] = y.forecast
proj_ay[t,1,] = apply(y.forecast[1:9,],2,sum,na.rm=T)
proj_ay[t,2,] = apply(y.forecast[10:18,],2,sum,na.rm=T)
proj_ay[t,3,] = apply(y.forecast,2,sum,na.rm=T)
total[t,1] = sum(y.forecast[1:9,],na.rm=T)
total[t,2] = sum(y.forecast[10:18,],na.rm=T)
total[t,3] = sum(y.forecast,na.rm=T)}

#Predictive statistics
aytable = cbind(apply(proj_ay[,1,],2,mean),apply(proj_ay[,1,],2,sd),apply(
    proj_ay[,2,],2,mean),apply(proj_ay[,2,],2,sd),apply(proj_ay[,3,],2,mean),
    apply(proj_ay[,3,],2,sd))
totaltable = rbind(apply(total,2,mean),apply(total,2,sd),apply(total,2,
    quantile,prob=c(0.75,0.95)))
```

```
riskmargin = cbind(totaltable[3:4,1] - totaltable[1,1],totaltable[3:4,2] -
    totaltable[1,2],totaltable[3:4,3] - totaltable[1,3])

#Distribution of total claims
library(EnvStats)
par(mfrow=c(1,1))
plot (density(total[-2044,1]), ylim=c(0,3.6e-05),xlim=c(8500,360000),xlab="
    Total unpaid losses",main="",lwd=3)
lines (density(total[-2044,2]), lty=2,lwd=3)
lines (density(total[-2044,3]), lty=3,lwd=3)
legend("top", legend = c("Accident Benefits (excluding DI)", "Accident
    Benefits (DI only)", "Total"),lty = 1:3,lwd=3, bty = "n",title = "")
```