Université de Montréal

Contrôle génétique de l'épissage alternatif dans le contexte de la réponse immunitaire innée

Par
Olivier TASTET

Département de Biochimie et Médecine Moléculaire
Faculté de Médecine

Mémoire présentée à la Faculté de Médecine
En vue de l'obtention du grade M.Sc. en Bio-informatique
Option Recherche

Résumé (*anglais*)

When triggered, the innate immune response initiates molecular changes in the cells that allow an appropriate defense to the detected threat. These changes will affect the transcription of immune related genes while other genes are turned off to efficiently let the organism fight the pathogen. Alternative splicing plays a major role in the fine-tuning of messenger RNA expression. Recent researches have shown that there were major shifts of RNA splicing following infection by two bacteria (*Listeria monocytogenes* and *Salmonella typhimurium*). This study aims to better define the role of inter-individual genetic variation on alternative splicing in the context of bacterial infection. We first defined a list of annotated alternative splicing events for which we could recover the proportion of exclusion and inclusion form (PSI-value). We used this phenotype as a proxy for isoform usage. Unlike researches focusing on the proportion of annotated transcripts, we could isolate certain types of events and analyze the processing of exon separately. A multiple linear regression approach was then used to map the psi-value on the genotyping data. We identified 1948 unique events significantly correlated to genotype (rpQTL) and hundreds that were specific to infected macrophages. In total, this set of events spanned 1 162 genes. These results show that the inclusion of certain exons is genetically controlled in macrophages and that some of this control only occurs in infected macrophages. We also showed that alternative-first-exon (AFE) seem to be the least affected by genotype while alternative-last-exon (ALE) and TandemUTR are the most affected. Further analysis of rpQTL SNP showed independence between eQTLs and rpQTLs. Overall our results propose that splicing-QTLs are an important force in gene regulation during the immune response.

Key words: Genetic variation, splicing, immune response, macrophages, QTL, RNA processing

Résumé (*français*)

Lorsqu'enclenchée, la réponse immunitaire innée permet l'initiation de changements moléculaires qui permettent une défense appropriée contre la menace détectée. Ces changements affectent la transcription de gènes impliqués dans l'immunité de l'organisme. L'épissage alternatif joue un rôle crucial dans la régulation de la transcription des ARN messagers. De récentes études ont mis en évidence les changements significatifs du traitement des exons dus à l'épissage alternatif lors de l'infection bactérienne de macrophages humains. Ce projet vise à caractériser l'impact de la variabilité génétique sur l'épissage alternatif dans le contexte de la réponse à un agent bactérien. Nous avons tout d'abord identifié un ensemble d'événements distincts de traitement d'exons pour lesquels nous pouvions obtenir le pourcentage d'inclusion (valeur PSI). Ce phénotype fût utilisé pour étudier le traitement de l'ARNm. Contrairement aux études qui traitent chaque isoforme séparément, notre approche isole les différentes catégories d'événements d'épissage. Une approche tirant profit de la régression linéaire a été utilisée pour associer la variation interindividuelle des valeurs PSI à la variabilité génétique de polymorphismes locaux. Nous avons pu identifier 1948 associations significatives impliquant 1 162 gènes uniques et plus d'une centaine associations spécifiques à l'infection. L'analyse des variants montre une indépendance entre l'effet du génotype sur l'expression (eQTL) et sur le traitement des exons lors de l'épissage d'un transcrit (rpQTL). Nos résultats montrent que l'épissage alternatif de gènes impliqués dans la médiation de la réponse immunitaire est affecté par la variabilité génétique.

Mots clés : Variation génétique, épissage alternatif, réponse immunitaire, macrophages, QTL, traitement de l'ARN

Table des matières

Index des Figures

Index des Figures Supplémentaires

Index des Tableaux

Liste des abréviations

TLR: Toll-like receptors/ Récepteurs de type *Toll*

PRR: Pattern recognition receptors/ Récepteurs de reconnaissance de motifs moléculaires

PAMP: Pathogen-associated molecular motifs/motifs moléculaires associés aux pathogènes

TF: Transcription factor/ Facteur de transcription

SNP: Single nucleotide polymorphism/ Polymorphisme nucléotidique

RNA/ARN: Ribonucleic acid/ Acide ribonucléique

NMD: Non-mediated decay/ Dégradation des ARNm non-sens

ATP: Adenosine tri-phosphate

PSI: Percent spliced in/ Pourcentage d'inclusion

MISO: Mixture of isoform

SRE: Splicing regulatory elements/ Éléments régulateurs d'épissage

NGS: Next generation sequencing/ Séquençage nouvelle-génération

ESS: Exonic splicing silencer/ Répresseur exonique

ESE: Exonic splicing enhancer/ Amplificateur exonique

ISS: Intronic splicing silencer/ Répresseur intronique

ISE: Intronic splicing enhancer/ Amplificateur intronique

RBP: RNA binding protein/ Protéine se liant à l'ARN

UTR: Untranslated Region/ Région non-traduite

GWAS: Genome wide association study/ Étude d'association pangénomique

QTL: Quantitative trait loci/ Locus de caractères quantitatifs

miRNA/miARN: micro RNA

AS: Alternative splicing/ Épissage alternatif

LCL: Lymphoblastoid cell line/ Ligné cellulaire lymphoblatoïde

PCA: Principal component analysis/ Analyse des composantes principales

GO: Gene ontology

OR: Odds ratio/ Rapport des cotes

AFE: Alternative first exon/ Premier exon alternatif

ALE: Alternative last exon/ Dernier exon alternatif

SE: Skipped exon/ exon intragénique alternatif

RI: Retained intron/ Rétention d'intron

TandemUTR: Tandem 3' untranslated region

FDR: False discovery rate/ Taux de faux positifs

PEER: Probabilistic estimation of expression residuals

LD: Linkage disequilibrium/ Déséquilibre de liaison

eQTL: expression quantitative trait loci/ QTL d'expression

asQTL: alternative splicing quantitative trait loci/ QTL d'isoforme alternative

rpQTL: RNA processing QTL/ QTL de traitement d'ARN

NIH: National Institute of Health

ENCODE: ENCyclopedia Of DNA Elements

Bp: Base pair/ Paires de bases

MCSF: Macrophage colony-stimulating factor/ Facteur stimulateur de colonie de macrophages

NGS : Next generation sequencing/ Séquençage nouvelle génération

À Madeleine Caron, source d’inspiration

Remerciements

J'aimerais remercier mon directeur de maitrise, Luis Bruno Barreiro, pour ses conseils et son support dans le déroulement de ce projet. J'aimerais également remercier l'ensemble du personnel du laboratoire Barreiro à Ste-Justine pour les nombreuses discussions, les rires et le soutien moral qu'ils m'ont amenés. De cette équipe je remercie spécialement Jean-Christophe Grenier pour son support et son aide mais surtout son écoute et la camaraderie que nous avons développée. Le temps passé à produire ce travail ne m'a pas simplement permis de développer mes compétences scientifiques mais également de tisser des liens non seulement professionnels mais également amicaux. Je remercie également Athma Pai, du Massachussett Instute of Technology pour son aide et sa collaboration dans l'écriture de l'article associé à ce mémoire. Je suis très reconnaissant du temps investi pour faire de cet article un travail scientifique élaboré et rigoureux.

Finalement j'aimerais remercier famille et amis pour avoir toléré mon stress dans les moments plus difficiles et d'avoir su m'encourager à persévérer. À Béatrice Trudel, merci de m'avoir soutenu dans les moments d'incertitudes et de remise en question. J'aimerais aussi spécialement remercier Marie-Claude Roy, Jean-François Pagé et Louise Roy pour leurs efforts à essayer de comprendre un sujet dans lequel ils n'ont aucune connaissance. Merci énormément d'avoir pris le temps de m'aiguiller dans la rédaction de ce mémoire.

.

Chapitre 1: INTRODUCTION

1. INTRODUCTION

Le système immunitaire

1.1.1 La réponse immunitaire innée

Le système immunitaire est la première ligne de défense contre les menaces extérieures aux organismes eucaryotes. Il permet à l'hôte de se défendre et de réagir lorsque l'organisme est attaqué par différents pathogènes comme les virus, les bactéries ou les parasites. Ainsi, la réponse immunitaire comprend l'ensemble des réactions biochimiques cellulaires qui se produisent au moment de la reconnaissance d'une menace externe, jusqu'au retour du système à son état basal. Ce phénotype est crucial à l'adaptation de l'organisme à son environnement. Alors que tous les organismes unicellulaires possèdent certains mécanismes de défense contre les pathogènes externes, les organismes multicellulaires ont développé une réponse aux pathogènes qui est plus structurée et organisée impliquant l'action de cellules spécialisées: *la réponse immunitaire innée* (1). Le principal avantage des organismes multicellulaires est l'établissement d'un système de cellules sentinelles dont le rôle est spécifiquement la reconnaissance et la réaction aux menaces extérieures (2). Chez les organismes unicellulaires, le processus de défense s'opère via certains processus moléculaires comme l'interférence ARN, la présence de peptides antimicrobiens ou d'enzymes de restriction mais, par définition, il n'existe pas de cellules immunitaires spécialisées. La réponse innée est dite non-spécifique car elle permet une réaction générique à un large ensemble de pathogènes. Face à un agent invasif inconnu, la réponse innée est d'abord rapidement initiée. Chez les vertébrés, une réponse plus spécifique et plus lente permet une défense additionnelle

contre les pathogènes. La *réponse immunitaire adaptative* permet de produire des anticorps contre certains microbes de manière à stimuler une réaction plus efficace dans l'éventualité d'une seconde rencontre de la menace extérieure.

1.1.2 Reconnaissance générique des pathogènes

La réponse innée est médiée par les cellules myéloïdes (**Figure supplémentaire 1;** page LXXI). Elle est initiée au moment ou un agent pathogène est détecté et phagocyté par une cellule sentinelle comme un macrophage ou un neutrophile (**Figure 1;** page 5). Chez les macrophages, la reconnaissance d'un microbe se fait par des récepteurs de reconnaissance de motifs moléculaires (PRRs; ex. TLRs, CLRs, NLRs). Les récepteurs de type *toll* (*toll-like receptors;* TLRs) sont une famille de PRR très bien caractérisée qui jouent un rôle majeur dans la réponse innée (3). Chez l'homme, les TLRs composent une famille de 10 protéines se regroupant sur la membrane cellulaire des macrophages ou sur les membranes d'organites internes comme les endosomes. Les TLRs internes reconnaissent le matériel ribonucléique viral alors que les TLRs externes sont activés par les bactéries et parasites (4).

La réponse immunitaire est déclenchée lorsque les PRR sont activés par la présence de motifs moléculaires associés aux pathogènes (PAMPs), situés sur les membranes bactériennes. Ces motifs sont conservés parmi les classes microbiennes, ce qui permet une réponse générique et rapide par l'activation de facteurs de transcription (ex. NKFB, iRF5, IRF3, IRF7) qui moduleront l'expression de gènes impliqués dans l'absorption et la destruction du pathogène (5). La réponse immunitaire innée implique donc des changements importants dans la régulation des gènes au moment de la détection et la réaction aux pathogènes.

En plus d'initier la réponse immunitaire innée, les macrophages ont un rôle communicateur avec la réponse adaptative. En effet, avec les cellules dendritiques, ils sont les principaux présentateurs d'antigènes aux lymphocytes T. Après la phagocytose du pathogène, les macrophages intègrent les antigènes à leurs membranes couplés aux protéines membranaires du complexe majeur d'histocompatibilité (MHC) (**Figure 1;** page 5). Ces protéines permettent d'identifier le macrophage comme du soi malgré la présence d'antigènes à sa surface (6).

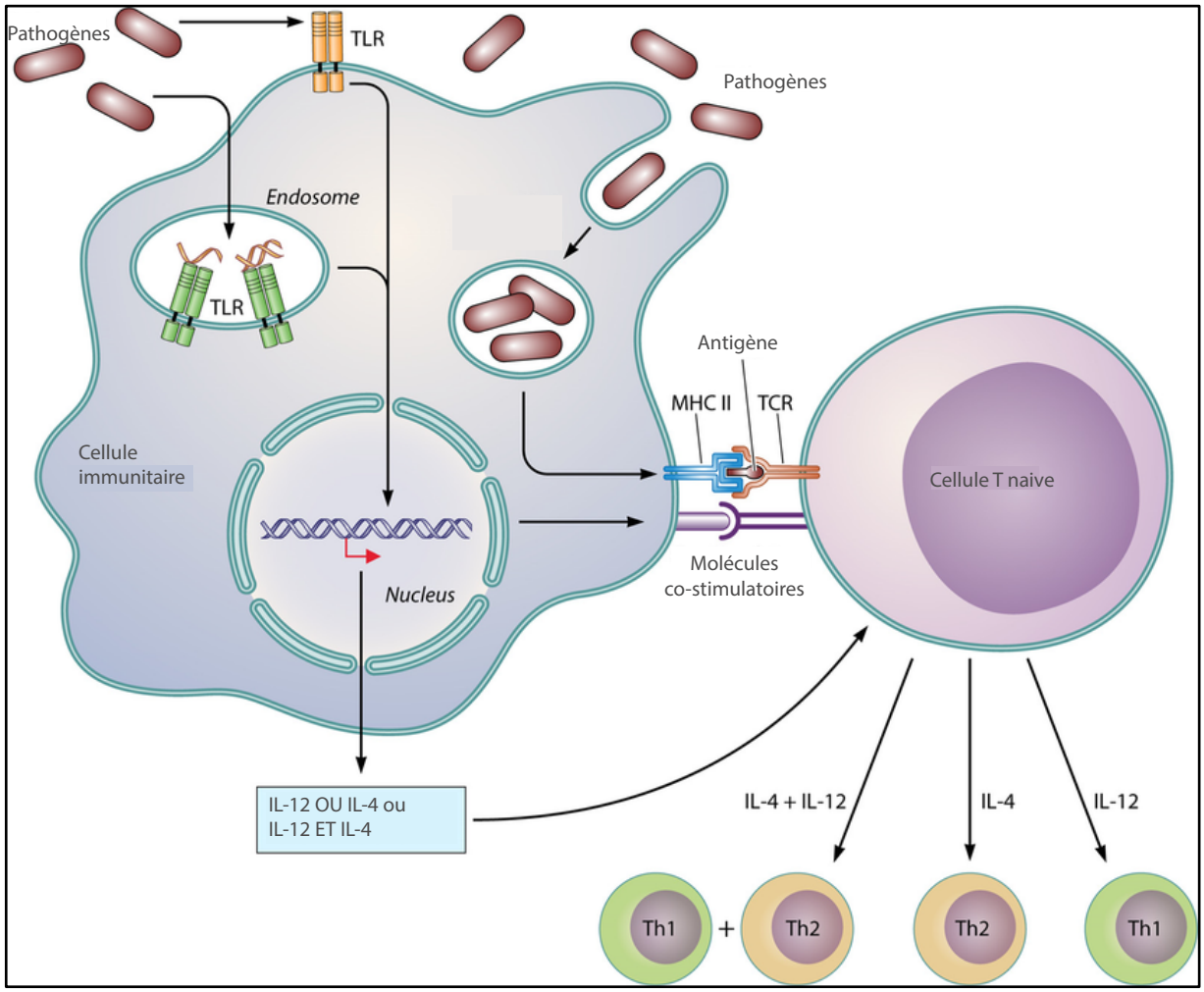


Figure 1 Déclenchement de la réponse immunitaire

Diagramme schématique représentant le déclenchement de la réponse immunitaire via la reconnaissance des pathogènes par une cellule effectrice du système immunitaire. La détection des pathogènes entraine une réponse transcriptionnelle qui permet le recrutement de la machinerie immunitaire. Les TLR sont un exemple important de l'intermédiaire par lequel les cellules immunitaires reconnaissent les pathogènes. Reproduit de (7).

1.1.3 Première ligne de défense contre l'environnement

La présence de pathogènes applique une pression évolutive importante sur l'organisme (8-10). L'interaction directe entre le système immunitaire et les pathogènes environnementaux force une course évolutive qui stimule la variabilité génétique chez l'hôte et le microbe. Les bactéries ont un taux mutationnel très élevé qui favorise la fixation rapide d'allèles bénéfiques. Elles ont également la capacité d'acquérir du nouveau matériel génétique par transferts horizontaux (11). La grande densité des populations bactériennes augmente aussi la probabilité d'une mutation bénéfique, permettant ainsi de contrer les défenses de l'hôte. L'évolution de ces systèmes est donc conjointe et dynamique (12). L'environnement pathogénique joue ainsi un rôle crucial dans l'évolution du système immunitaire.

1.1.4 Immunité et génétique des populations humaines

Le paysage génétique associé à l'immunité est donc constamment mis à l'épreuve par la présence des pathogènes. L'étude de la génétique des populations humaines permet d'observer cet effet sur la susceptibilité aux infections bactériennes. Il fut récemment démontré qu'il existait des différences immunitaires marquées entre des individus d'origines africaines et des individus d'origines européennes et que ces différences étaient en grande partie attribuables à la variabilité génétique interindividuelle (13, 14). L'hypothèse la plus répandue quant à la dispersion de l'homme est la théorie du berceau africain (**Figure Supplémentaire 2;** page LXXII) (15). En migrant vers des régions plus nordiques, les populations humaines ont fait face à un biodiversité pathogénique très différente. En effet, le milieu plus froid et ardu du nord ne favorise pas autant la virulence et la prolifération des agents infectieux

que le climat plus chaud de l'Afrique. L'ethnicité africaine étant associée à une réponse inflammatoire accrue, il est probable que cette différence soit arrivée lorsque les populations migrantes ont été soumises à une densité pathogénique moins importante, de par le climat plus froid et moins propice à la virulence microbienne. Ainsi, une réponse inflammatoire moins importante aurait défavorisé les désordres auto-immunitaires chez les populations européennes. Les polymorphismes sont des régions variables de l'ADN et incluent les changements de nucléotides uniques (SNP), les délétions, les insertions ou les duplications de séquences. Les SNP sont les polymorphismes les mieux catégorisés, principalement parce qu'ils n'impliquent qu'un seul changement moléculaire et qu'ils sont donc plus facile à identifier et plus constant chez une cohorte donnée. Ces derniers peuvent avoir des conséquences importantes s'ils modifient la séquence codante d'un gène mais peuvent également avoir un effet plus subtil, affectant plutôt les processus de régulation génique comme la transcription ou l'épissage alternatif des gènes. Les variants génétiques ont une contribution importante au niveau des différences transcriptionnelles entre les populations humaines (13, 14).

Épissage alternatif

1.2.1 Régulation génique

Dans les cellules eucaryotes et procaryotes, la régulation des gènes est initiée par la transcription. Les facteurs de transcriptions (TF) sont des protéines qui se lient à des motifs spécifiques qui permettent la modulation de la transcription d'un gène. Les TF peuvent agir en tant que promoteurs ou répresseurs de l'activité des ARN polymérases permettant ainsi une fine régulation de l'expression génique. Les organismes eucaryotes ont développé des mécanismes biochimiques qui permettent une régulation additionnelle de l'expression. Contrairement aux gènes procaryotes, les gènes eucaryotes sont composés de courtes régions codantes (exons) et de longues régions non-codantes (introns). Ces régions sont bordées par des jonctions portant des motifs de séquence spécifiques. La transcription produit initialement un transcrit immature comportant les régions exoniques et les régions introniques. L'épissage est le procédé biochimique qui retire les introns et assemble les exons pour former le transcrit mature. Ce processus permet de former plusieurs isoformes alternatives d'un même gène. Chez l'homme, il est aujourd'hui estimé que plus de 90% des gènes ont au moins une isoforme alternative (16). L'épissage alternatif est impliqué dans un grand nombre de fonctions cellulaires. Ce terme est utilisé lorsqu'un évènement d'épissage provoque la formation d'isoformes possédant un exon alternatif. Plusieurs études ciblées ont montré son rôle dans l'immunité par la modulation de la signalisation cellulaire des cytokines. Son implication dans le complexe majeur d'histocompatibilité (MHC) lui confère un intérêt particulier dans l'étude de la réponse immunitaire (17-19). Ces études ont cependant majoritairement portées sur des

gènes spécifiques et la caractérisation de l'épissage alternatif au niveau pan-génomique dans le contexte de l'immunité demeure un terrain peu exploré.

1.2.2 Le spliceosome

Comme dans le cas de la transcription, l'épissage alternatif requiert l'activité d'un complexe externe; le spliceosome. L'épissage alternatif est considéré comme un processus co-transcriptionnel c'est-à-dire qu'il se produit au même moment que la transcription d'un gène. Le spliceosome est un complexe d'ARN et de protéines qui se lie au transcrit immature pour en retirer les introns et assembler les exons. La fonction régulatrice du spliceosome est évolutivement conservée chez les eucaryotes. La formation de ce complexe nécessite la présence de cinq particules ribonucléiques (snRNPs : U1, U2, U4, U5, U6) ainsi qu'une multitude de facteurs protéiques (20) (**Figure 2;** page 10). Il existe deux grandes classes de spliceosome : la classe majeure, la plus commune, et la classe mineure, impliquée dans environ 0.5% des évènements d'épissage alternatif. Dans le cas de la classe majeure, l'épissage d'exon débute par l'association respective de U1 et U2 avec les jonctions 5' et 3' d'un intron. Une composante de U2 s'associe également avec une région intronique appelée le point de branchement (*branching point*). U4/U6 et U5 forment un complexe qui permet de réunir U1 et U2. En rassemblant les extrémité 3' et 5' des exons à assembler, l'intron forme un lasso, appelé un lariat. Un réarrangement dépendant de l'ATP survient et l'appariement de U1 avec la jonction d'épissage 5' est remplacé par un appariement avec U6 alors que U1 et U4 sont libérés du complexe. La jonction 5' est d'abord clivée puis U5 permet la juxtaposition des exons. Le lariat est finalement libéré, retirant définitivement la région intronique du transcrit

(21). Ainsi, l'épissage d'un exon requiert un grand nombre d'interactions entre les snRNPs et la séquence primaire du gène. La plupart des gènes humains étant multi-exoniques, l'épissage alternatif d'un gène comprend plusieurs évènements de traitement distincts impliquant ces interactions.

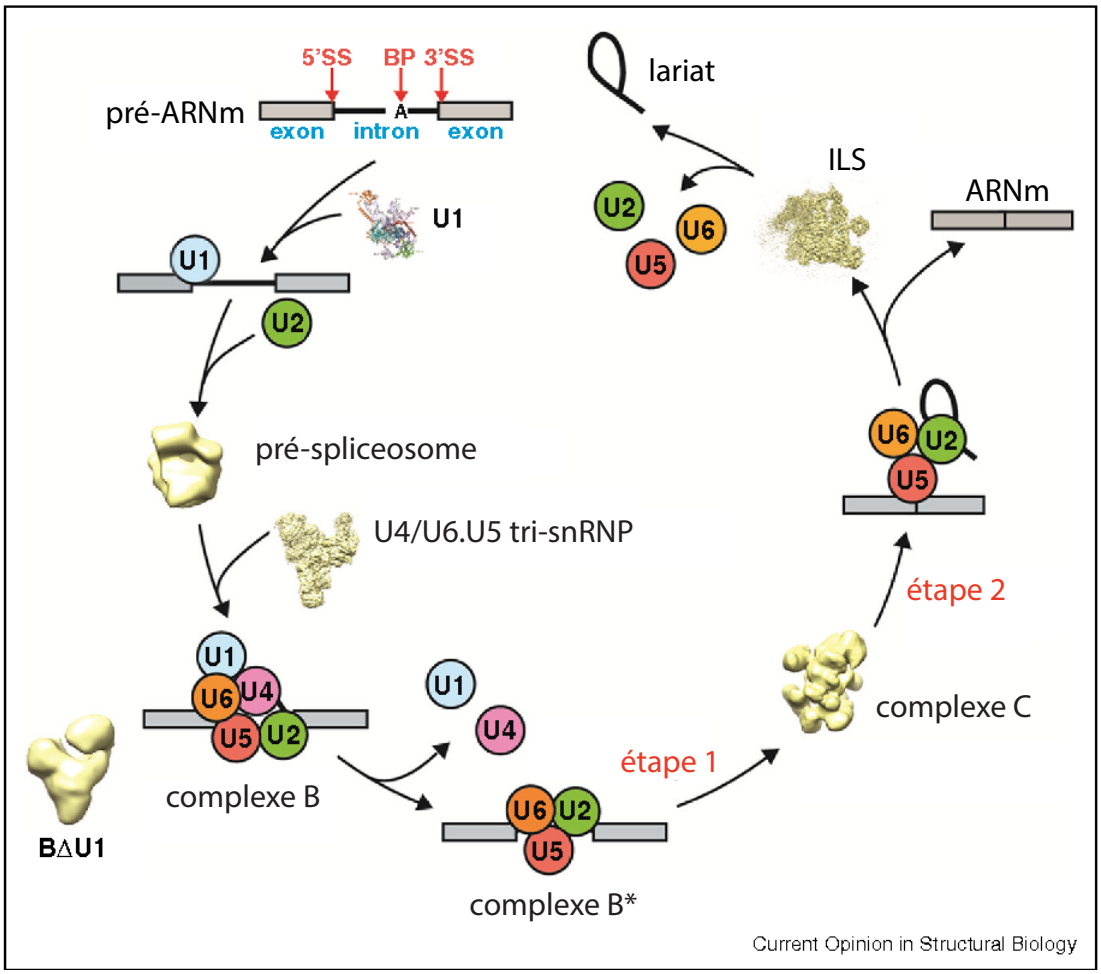


Figure 2 Formation du spliceosome

Schéma présentant les différentes étapes de la formation du spliceosome. Chaque étape est caractérisée par les interactions entre les différentes particules nucléaires (snRNPs). Ces particules sont représentées par ces cercles de couleur portant le nom de la particule impliquée. Reproduit de (22).

1.2.3 Différents mécanismes de traitement de l'ARN : conséquences sur l'isoforme finale

L'analyse approfondie de l'épissage alternatif permet de différencier certains types d'évènements de traitement d'ARN selon leur conséquence dans l'isoforme finale (**Figure 3;** page 12). La catégorie la plus commune est l'exclusion alternative d'un exon intragénique (23). Il existe plusieurs autres classes comme la rétention d'un intron, l'usage d'une jonction alternative (5' ou 3'), l'usage d'un premier ou dernier exon alternatif, des exons mutuellement exclusifs ainsi qu'une longueur variable de la région non-transcrite en 3' (3'UTR) (24). Il fût récemment démontré que ces classes d'évènements étaient affectées différemment lors d'une infection bactérienne (25). Par exemple, on observe un raccourcissement général des régions 3'UTR suite à l'activation de macrophages. Ce raccourcissement est lié à l'évasion du ciblage par les micro ARN (miRNA). Ce ciblage a une fonction régulatrice car les miRNA se fixent à des séquences spécifiques pour bloquer la maturation d'un transcrit (26). Ainsi, l'usage d'une région 3'UTR raccourcie minimise la disponibilité de ces sites de liaison et permet une augmentation de transcrits matures et fonctionnels. Les autres types d'évènements de traitement de l'ARN sont également associés à différentes fonctions biochimiques. En effet, l'usage alternatif d'un premier exon est associé avec l'usage de régions promotrices alternatives (27). La rétention d'intron est associée avec la régulation négative d'un gène. L'insertion d'une région intronique dans un transcrit intègre souvent un codon d'arrêt prématuré qui génère un produit dégénéré du gène qui sera ensuite dégradé (entres autres par NMD) (28). Le rôle de la rétention d'intron ne se limite évidemment pas qu'à la dégradation d'ARNm non-sens et peut aussi affecter l'efficacité de la traduction de l'ARNm ou même amener la production de protéines tronquées. Les différentes catégories d'évènement

d’épissage alternatif peuvent donc avoir des conséquences fonctionnelles distinctes sur la production et l’efficacité de l’ARNm.

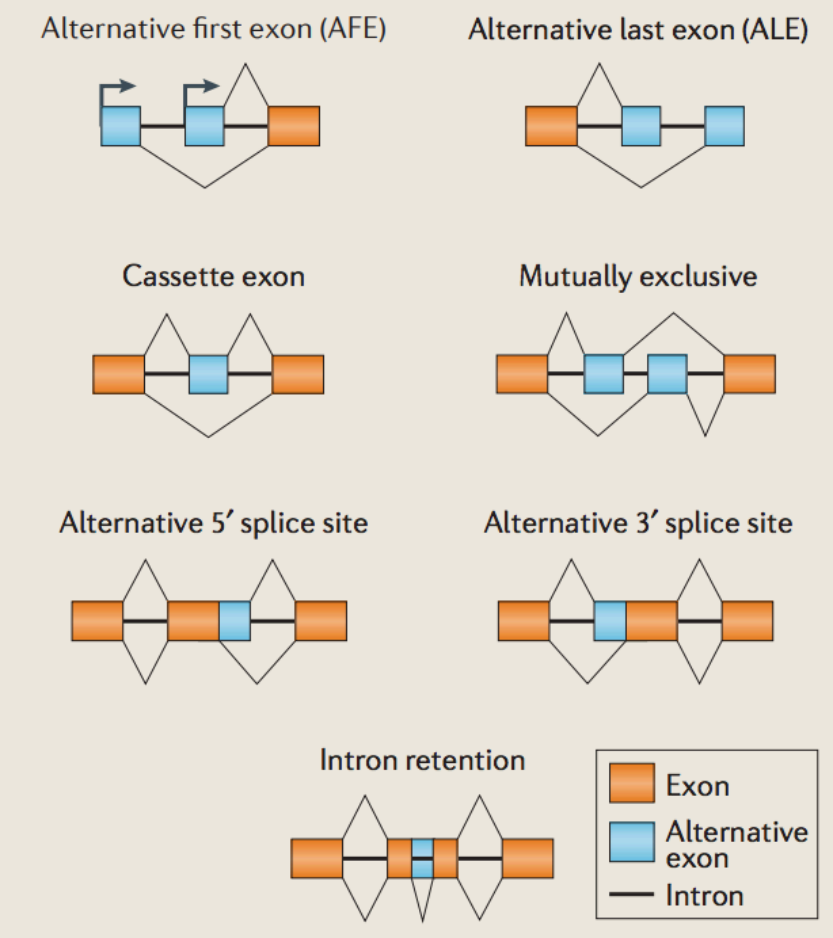


Figure 3 Types d'évènements

Exemples illustrant les différents types d’épissage alternatif possibles. Chaque évènement peut avoir des conséquences distinctes sur le transcrit mature. L’anglais est conservé pour faciliter la correspondance des acronymes. En francais (en ordre) : Premier exon alternatif (AFE), dernier exon alternatif (ALE), exon cassette, exons mutuellement exclusifs, site d’épissage 5’ alternatif, site d’épissage 3’ alternatif, rétention d’intron . Reproduit de (29).

1.2.4 Caractérisation bio-informatique

Les premières méthodes à haut débit pour caractériser l'épissage alternatif faisaient usage de micropuces adaptées. En effet, ces dernières sont spécialement manufacturées pour couvrir les jonctions entre exons ainsi que toute autre région subissant potentiellement de l'épissage alternatif. L'usage de ces micropuces a permis la découverte de nouvelles isoformes et l'identification de nombreux évènements de traitement d'ARN (30). Cependant, les avancées en séquençage de nouvelle génération (Next generation sequencing; NGS) ont permis le développement de méthodes plus précises qui permettent de quantifier les évènements d'épissage à travers un grand nombre d'échantillons. Une de ces méthodes, *RNA-Seq*, permet le séquençage de l'ensemble des transcrits matures présent dans un échantillon de cellules. Les ARNm sont d'abord capturés en visant la queue polyadénylée des transcrits matures. Ces derniers sont ensuite fragmentés en courtes séquences d'environ 100 paires de bases (bp) puis convertis en ADNc. Ces courtes séquences sont bordées d'amorces moléculaires qui permettent de les séquencer d'un sens (*single-end sequencing*) ou des deux sens (*paired-end sequencing*). L'alignement de ces courtes séquences sur le génome de référence permet d'avoir une vue globale sur le transcriptome (**Figure 4A;** page 15). En l'absence d'un génome de référence, ces séquences peuvent être assemblées selon leur similarité, et les transcrits reconstruits *de novo* (31).

Les avantages conférés par cette méthode de séquençage dans le cadre de l'analyse de l'épissage alternatif sont multiples. Contrairement aux micropuces, *RNA-Seq* est exempt de l'hybridation croisée qui se produit à cause d'appariements non-spécifiques aux sondes présentes sur la puce. Ainsi, les micropuces permettent difficilement de différencier des

isoformes très similaires (16). L'avantage principal de cette méthode NGS est de fournir une quantification fiable de l'expression des gènes et la proportion des différentes isoformes. L'analyse bio-informatique de l'épissage alternatif depuis le séquençage *RNA-Seq* se divise en deux catégories : 1) centré sur l'exon et 2) centré sur l'isoforme (32). La méthode centrée sur l'exon est basée sur l'estimation du taux d'épissage pour chaque évènement de traitement de l'ARN alors que la méthode centrée sur l'isoforme permet plutôt de considérer chaque isoforme séparément, en incluant tous les évènements qui permettent la formation de cette dernière. Ainsi, en considérant chaque exon séparément, il est possible d'analyser les différents types de traitement de l'ARN. L'avantage de l'analyse centrée sur l'exon permet d'analyser l'épissage alternatif de manière plus réaliste, en considérant la formation d'une isoforme comme un collectif d'évènements de traitement d'ARN messager. Ainsi plus d'un évènement pourrait expliquer la variabilité d'une isoforme données à travers la population. En considérant chaque exon séparément, il est possible de comprendre quels sont les mécanismes précis qui affectent cette variabilité. La métrique utilisée dans ce genre d'analyse est la valeur *Percent Spliced In* (PSI) (33). Plusieurs outils bio-informatiques permettent d'estimer cette valeur (ex. MISO, SpliceTrap (34), rMATS (35), SUPPA2(36)). Ces approches donnent des résultats assez concordants (37).

MISO (Mixture of ISOform) utilise une approche statistique pour estimer la valeur PSI (38). Cette mesure représente la proportion de *reads* qui appartiennent à la forme inclusive. Les exons alternatifs sont alternativement utilisés entre les différentes isoformes, c'est-à-dire qu'ils ne sont pas présents dans tous les transcrits. Les exons constitutifs sont, quant à eux, présents dans toutes les isoformes. Ainsi, les *reads* alignés à l'exon alternatif ou à ses jonctions

d'épissage (jonctions entre l'exon alternatif et les exons constitutifs) font partie de la forme inclusive et les *reads* se trouvant sur la jonction entre les exons constitutifs font partie de la forme exclusive (**Figure 4B;** page 15). La valeur PSI est donnée par le nombre de *reads* propre à la forme inclusive contre le nombre de *reads* propres à la forme exclusive. MISO utilise les reads constitutifs pour estimer le niveau d'expression total du gène, fixant ainsi, par inférence Bayésienne, un intervalle de confiance pour chaque valeur estimée. Généralement, la densité de reads des exons constitutifs sera supérieure à celle de l'exon alternatif car ils font partie des deux isoformes alternatives.

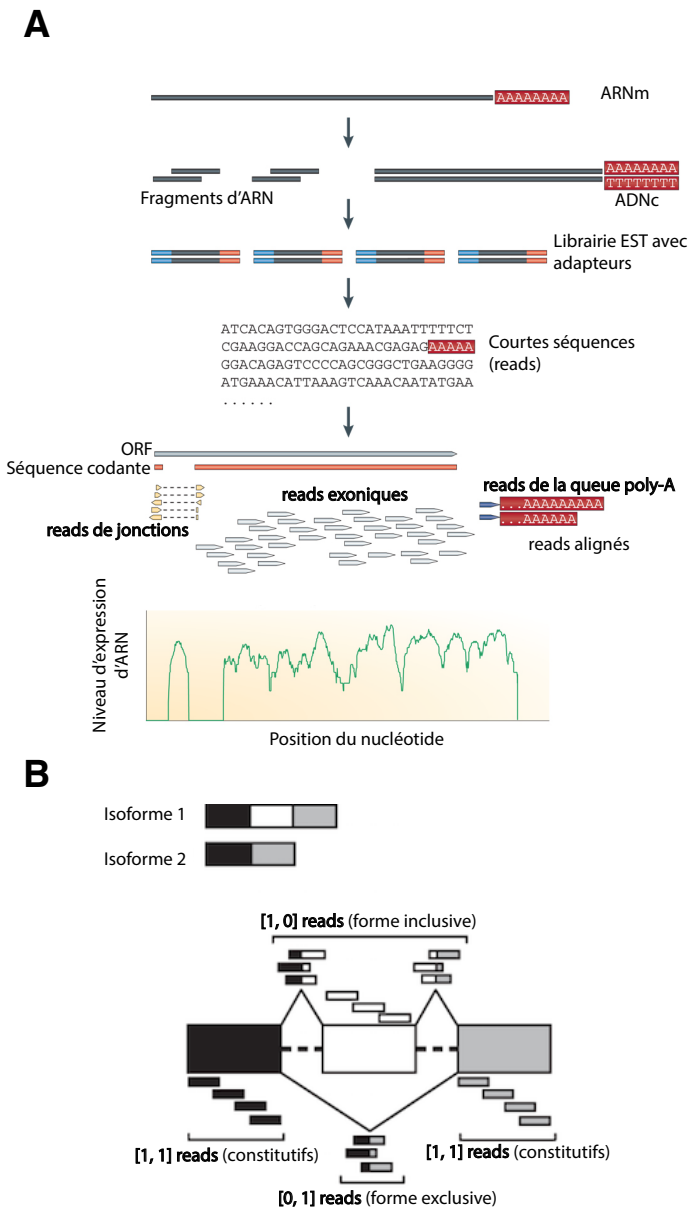


Figure 4 Caractérisation bio-informatique des évènements d'épissage

A) Schéma représentant la technique de séquençage de l'ARN *RNA-Seq*. Les différentes étapes de caractérisation sont montrées. Dans le cas d'analyses sur l'humain, l'accès au génome de référence permet une analyse précise des jonctions d'épissage (31) **B)** Diagramme présentant les reads utilisé pour estimer la valeur PSI par MISO. Les reads notés [1,1] sont dit constitutifs car ils appartiennent à toutes les isoformes connues. Les reads notés [1,0] sont inclusifs, c'est-à-dire qu'ils appartiennent à l'isoforme où l'exon alternatif est inclus. Les reads notés [0,1] font partie de l'isoforme exclusive, c'est-à-dire qui exclus l'exon alternatif (38).

1.2.5 Régulation cis/trans

L'épissage alternatif peut être affecté de manière directe par des éléments intrinsèques à la séquence du gène (interaction en *cis*) ou de manière indirecte par des facteurs distants (interaction en *trans*) (**Figure 5;** page 18). Comme mentionné précédemment, les jonctions d'épissage ont un effet direct sur le réarrangement du gène car ils en délimitent les points d'excision aux extrémités 5' et 3' des exons. Ces éléments sont facilement identifiables grâce à l'annotation disponible des gènes car ils bordent les régions transcrites. L'interaction entre le spliceosome et les jonctions d'épissage se fait par l'entremise de séquences spécifiques. En effet, la majorité des jonctions d'épissage présentes chez les gènes humains portent le motif canonique (5': GU---AG :3'). La liaison du spliceosome implique de plus longs motifs périphériques qui permettent une plus grande flexibilité de l'épissage en modulant la spécificité et l'efficacité du réarrangement. En plus des jonctions d'épissage, il existe également des éléments régulateurs d'épissage (*Splicing Regulatory Elements; SREs*) qui permettent de moduler l'activité du spliceosome par la liaison de facteurs distants (39). Ces éléments sont dits locaux car ils se situent dans les régions introniques et exonique des gènes. Bien que plusieurs efforts expérimentaux aient été déployés pour identifier ces régions, l'effort conjoint de la bio-informatique, par la comparaison de motifs, et de techniques de laboratoires ont permis l'identification de plusieurs centaines de SREs. Pour comprendre la fonction de ces régions, une approche bio-informatique utilisant un réseau Bayésien a permis de montrer la nature co-évolutive des SREs, soulignant ainsi leur importance fonctionnelle (39). Tout comme les jonctions d'épissage, les SREs sont liés par des facteurs distants. L'activité de ces facteurs est dite

indirecte (en *trans*). Les SREs peuvent se situer dans des régions exoniques ou introniques et agir comme amplificateurs ou inactivateurs des facteurs distants (*Exonic Splicing Enhancer*: ESE, *Exonic Splicing Silencer*: ESS, *Intronic Splicing Enhancer*: ISE, *Intronic Splicing Silencer*: ISS).

Les facteurs d'épissage sont généralement des protéines se liant à l'ARN (*RNA binding proteins; RBPs*). Les RBPs vont généralement se lier aux SREs pour amplifier ou diminuer l'efficacité de l'épissage par le spliceosome (40). Plusieurs méthodes ont été développées pour tenter d'identifier la relation entre ces deux éléments. RNA Bind-N-Seq est une méthode in vitro qui permet non seulement d'interroger les motifs canoniques connus mais également d'identifier des motifs presqu'optimaux qui ont aussi une importance fonctionnelle (41). Ainsi, la régulation de l'épissage alternatif des gènes passe par un réseaux complexe d'interactions qui dépendent de l'identité des séquences des transcrits.

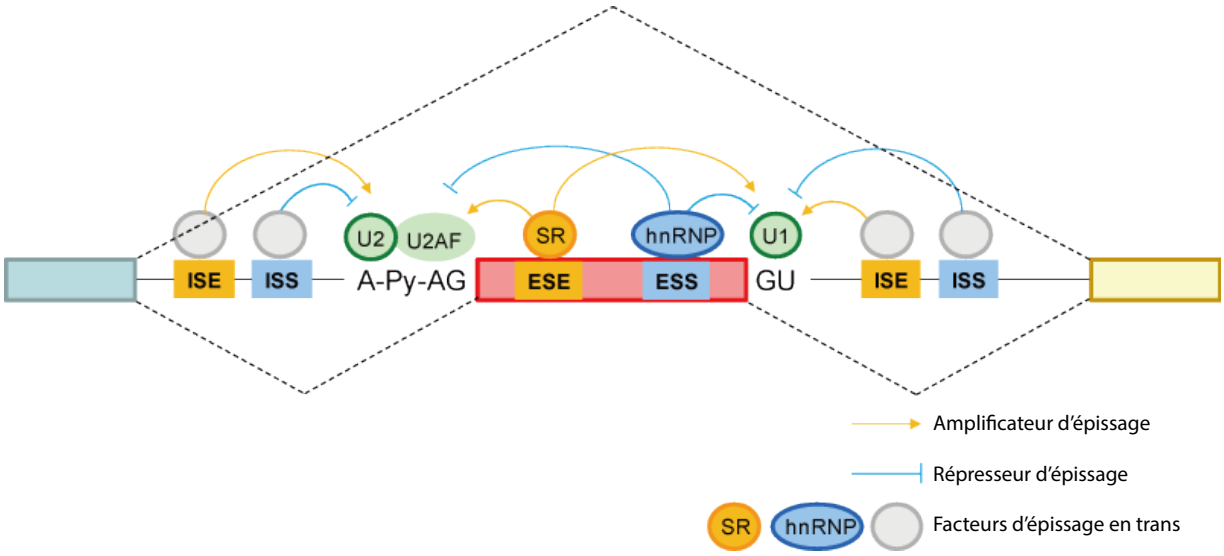


Figure 5 Régulation de l'épissage alternatif

Schéma montrant les différents éléments qui participent à la régulation du processus d'épissage. Les régions notées ISS, ISE, ESE, ESS sont des motifs séquentiels spécifiques où se lient les facteurs d'épissage. Les jonctions d'épissages portent les motifs spécifiques GU (jonction 5') et AG (jonction 3'). Ces éléments sont dits régulateurs en cis car ils affectent directement l'épissage du gène dans lequel ils se trouvent. Les éléments notés SR (protéines riches en Serine-Arginine) et hnRNP (Ribonucleoprotéines nucléaires hétérogènes) sont des facteurs d'épissage pouvant se lier aux régulateurs cis. Leur action est dite en trans car ils proviennent généralement de régions génomiques éloignées du gène
(Human-transcriptome database for alternative splicing: http://h-invitational.jp/h-dbas/as_mechanism.jsp)

Études d'association

1.3.1 Genome-wide association studies (GWAS)

Au cours de la dernière décennie, de multiples efforts ont été entrepris pour expliquer et comprendre les facteurs qui influencent la susceptibilité aux maladies. Entre autres, certains facteurs environnementaux comme l'âge, le genre ou la période saisonnière peuvent grandement influencer la production de cytokines, importants médiateurs du signalement cellulaire lors de la réponse immunitaire, affectant ainsi notre habileté à combattre un pathogène donné (42). En plus des facteurs environnementaux, la variabilité génétique joue un rôle crucial dans la susceptibilité aux maladies (**Figure 6;** page 20) (43). Plusieurs études ont établi le lien entre la variabilité génétique et la proportion de types cellulaires présents dans le sang (44-47). De manière générale, la variabilité génétique peut directement influencer la fonction des protéines en modulant des domaines intragéniques transcrits ou avoir un effet plus subtil en affectant des régions régulatrices amplificatrices ou inactivatrices. Une approche pour tenter d'identifier des variants génétiques associés aux maladies est l'étude d'association pan-génomique (GWAS). La technologie de micropuces de génotypage, développée dans les années 90, permet de caractériser un grand nombre de variants à travers l'ensemble du génome (48). L'information de génotypage est ensuite utilisée pour déterminer si certains variants sont statistiquement associés à divers traits discrets. Cette approche est abondamment utilisée dans l'optique d'identifier les facteurs génétiques liés aux maladies complexes. Ainsi, une base de données a été déployée pour cataloguer les différentes associations (49). La plupart des signaux GWAS se trouvent dans des régions non-codantes du génome ce qui porte à croire que cette variation génétique influence plutôt la régulation des

gènes (50, 51). Une des limitations de l'approche GWAS est qu'il est difficile de lier les variations associées aux maladies à des mécanismes précis et soulèvent donc souvent de nouvelles questions. En effet, il est plus difficile d'inférer la conséquence d'un variant lorsque ce dernier n'affecte pas la région codante du gène (52). De plus, l'ensemble des SNP ne sont pas systématiquement interrogés dans une étude GWAS. Ainsi, il est possible que le variant causal soit en fait un SNP codant qui n'ait pas été interrogé.

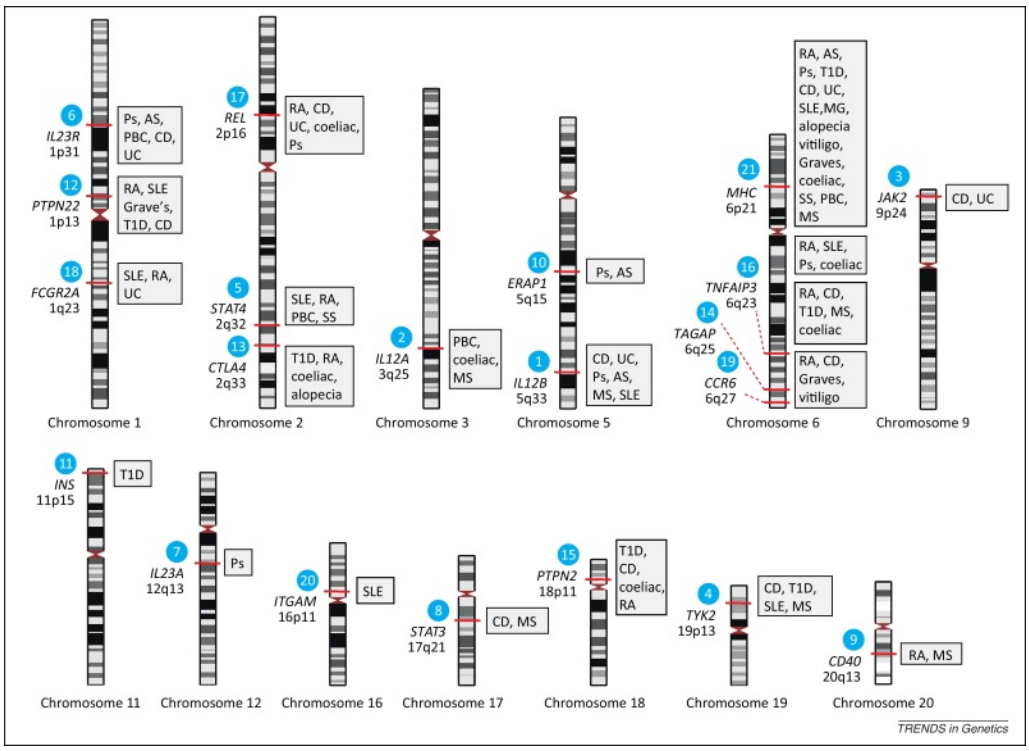


Figure 6 Loci associés à des désordres immunitaires

Carte génomique indiquant des loci associés à des désordres immunitaires par une approche GWAS. Bien que la variation génétique associée aux désordres soit identifiée, la conséquence physiologique de cette variation reste inexpliquée. Une approche QTL pourrait identifier l'impact de cette variation sur la transcription ou l'épissage alternatif d'un gène, proposant ainsi un mécanisme biochimique liant le désordre avec le polymorphisme (53).

1.3.2 Quantification de phénotype et association génétique (QTL)

Les mutations génétiques peuvent provoquer des changements dramatiques dans la régulation des gènes car elles peuvent entrainer des conséquences importantes aux différentes étapes de la transcription et de la traduction d'un gène. En affectant les régions promotrices, elles peuvent moduler l'affinité des facteurs de transcription, ce qui peut directement affecter le niveau d'ARNm produit. Elles peuvent également incorporer un codon stop dans une séquence exonique du gène, ce qui provoquera une terminaison précoce de la transcription. La variation génétique interindividuelle peut cependant avoir un effet plus subtil et plutôt altérer les motifs locaux se trouvant dans les gènes. Ces modifications peuvent ainsi affecter le processus d'épissage alternatif en modifiant l'affinité du spliceosome pour les jonctions d'épissage ou en modulant l'affinité des facteurs d'épissage pour les SREs. Finalement, un polymorphisme se situant dans la région 3' UTR peut, sans affecter la région traduite, moduler l'interaction entre le transcrit mature et la machinerie cellulaire. En effet, la région 3' UTR est associé au transport et à la survie du transcrit.

En quantifiant un phénotype donné, il est possible d'associer la variabilité génétique interindividuelle à la variabilité de ce phénotype. En effet, dans le cas d'un variant se trouvant dans une jonction d'épissage ou sur un motif de liaison de facteur d'épissage, le traitement des exons de l'ARN immature peut potentiellement être altéré. En général, on nomme cette association quantitative entre un phénotype et le génotype *Quantitative trait loci* (QTL). Le niveau d'expression des gènes est un phénotype très étudié avec cette approche analytique. Lorsque la variation génétique est associée au niveau d'expression d'un gène, on parle d'eQTL (*expression quantitative trait loci*) (**Figure 7;** page 23). Il est aussi possible de considérer la

proportion d'isoformes d'un gène ou le taux d'épissage d'un exon donné (valeur PSI) comme un trait quantitatif (**Figure 7;** page 23). L'approche centrée sur l'exon permet de considérer les différents évènements de traitement de l'ARN comme des évènements distincts, pouvant être individuellement affectés par différentes variations génétiques.

Il existe plusieurs méthodes pour identifier les cis-QTL ; la plus répandue repose sur le concept mathématique de régression linéaire (54, 55). En effet, cette approche assume un effet additif du génotype. Un individu hétérozygote n'aura qu'une des deux copies du gène qui sera affecté par le variant alors que les deux copies du gènes d'un individu homozygote pour l'allèle mutée seront affectés par le polymorphisme. On parle d'un effet en *cis* lorsque le variant affecte localement le gène se trouvant sur le même brin. Par exemple, la modulation du site d'initiation de la transcription ou de la région de liaison du spliceosome serait considéré comme une association en *cis*. Il existe aussi des relations indirectes (ou en *trans*) où, par exemple, un variant pourrait affecter l'expression d'un facteur d'épissage, ce qui aurait comme conséquence de moduler la régulation de gènes éloignés. Statistiquement, il est plus difficile d'identifier les association *trans* car l'analyse requiert un très grand nombre de tests (56). En effet, le phénotype d'intérêt doit être testé contre l'ensemble des variants disponibles. Un plus grand nombre de test augmente la probabilité d'obtenir un signal statistique significatif par chance, ce qui rend plus compliqué l'identification de signal statistique véritable. Les analyses en *trans* requièrent généralement un seuil de significativité beaucoup plus astringent. Dans le cas d'une association en *cis*, on ne teste généralement que les variants situés dans le voisinage du gène, ce qui réduit énormément la quantité de tests effectués.

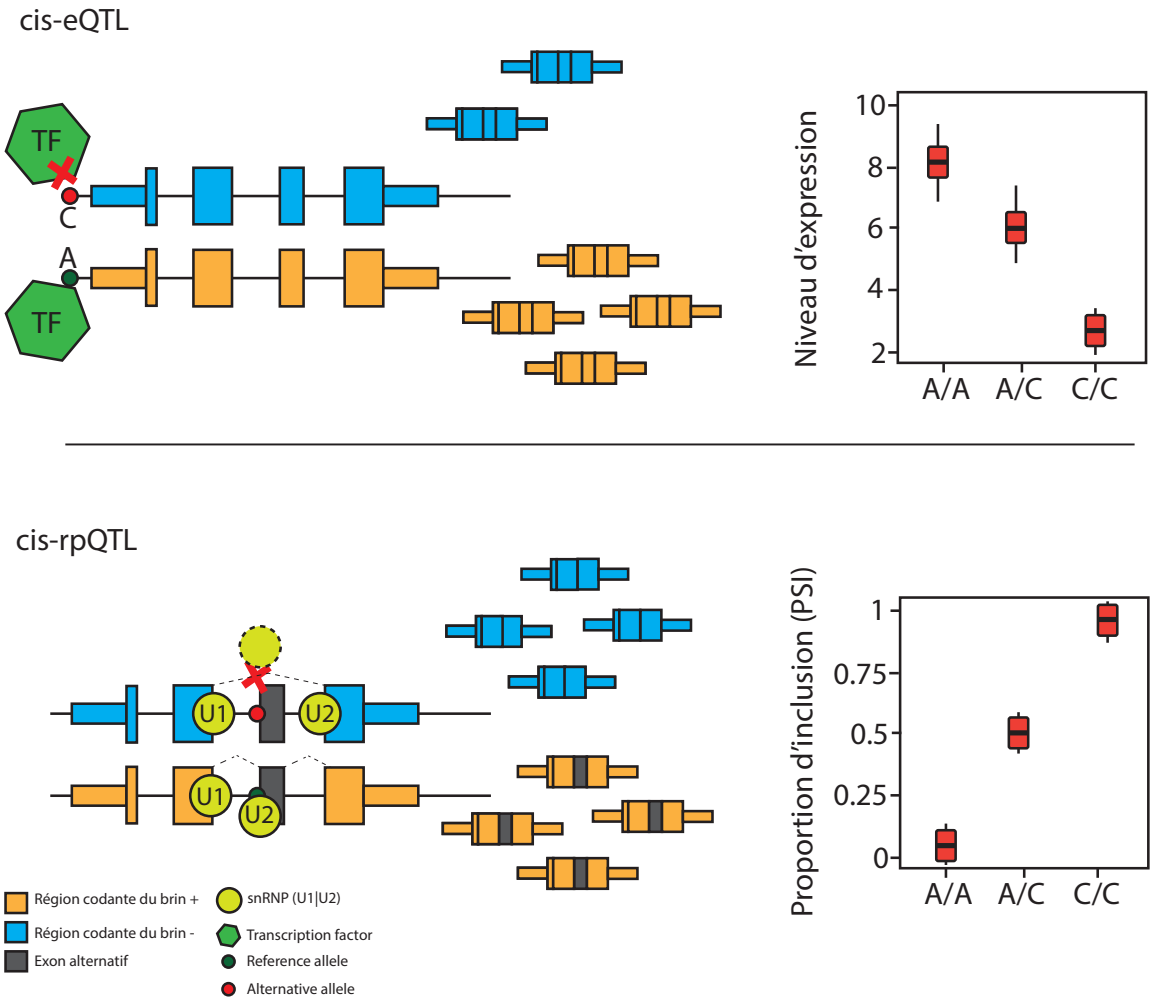


Figure 7 cis-QTL: eQTL vs rpQTL

Schéma résumant le concept biologique et mathématique d'un QTL. Les panneaux du haut et du bas permettent d'identifier la différence entre un eQTL et un rpQTL. Dans le cas du eQTL, le SNP (ref=A, alt=C) est situé sur le site de liaison d'un facteur de transcription TF (en vert). L'allèle alternative diminue l'affinité de TF pour l'initiation de la transcription du gène. L'effet de cette variation est intrinsèque au brin sur lequel elle se trouve. Ainsi, les individus homozygotes pour l'allèle de référence ont une activité transcriptionnelle diminuée pour les deux copies du gène. Un individu hétérozygote, n'aura cependant qu'une copie du gène qui sera affectée. Ceci est représenté par le graphique hypothétique à droite. On voit que le niveau d'expression est directement affecté par l'affinité décroissante de TF provoqué par le SNP. Dans le cas du rpQTL (en bas), la mutation se trouve dans la région 3' d'un intron, affectant l'épissage de ce dernier en défavorisant la liaison d'une des composantes du spliceosome. Ceci a pour effet de retirer un exon du transcrit mature sans affecter le nombre total d'ARN. Les individus homozygotes pour l'allèle alternative ont donc 100% des transcrits mature sans l'exon 3 alors que les individus hétérozygotes ont 50% de transcrits sous la forme inclusive et 50% sous la forme exclusive.

1.3.3 QTL d'épissage et maladies

La régulation qu'apporte l'épissage alternatif des gènes en fait un bon candidat pour l'étude des maladies. En effet, plusieurs études ont identifié des désordres d'épissage comme le point central de plusieurs maladies (29, 57-63). Les mutations qui perturbent le traitement adéquat des transcrits provoquent la production de protéines défectueuses, affectant ainsi les propriétés fonctionnelles des gènes. La majorité des mutations qui affectent le processus d'épissage sont des SNP se situant dans les jonctions introniques et exoniques d'épissage. L'épissage alternatif joue un rôle central dans diverses maladies comme la dystrophie musculaire de Duchesne (64) et le développement précoce du Parkinson (65) ainsi que plusieurs désordres immunitaires (62, 66). Les mutations les plus communes provoquant un épissage aberrant impliqué dans les maladies se trouvent dans les régions de régulation en *cis*. Ainsi, la perturbation de l'épissage alternatif peut provoquer des transcrits non-fonctionnels qui vont participer au développement des maladies. Plusieurs efforts ont donc été initiés pour tenter de lier la variabilité interindividuelle de l'épissage alternatif avec la variabilité génétique. L'accès public de données de séquençage a permis à plusieurs groupes de caractériser le contrôle de l'épissage alternatif à travers différents tissus. En utilisant les profils *RNA-Seq* de tissus cérébraux de 285 individus, *Takata et al* ont pu identifier 1 341 gènes présentant au moins un QTL d'épissage d'exon (67). L'équipe a démontré un enrichissement significatif de variants associés à la schizophrénie par des études GWAS. Une autre étude conduite par Ma *et al* a permis la caractérisation des QTL d'épissage dans des tissus adipeux et la mise en évidence de quatre polymorphismes affectant l'épissage alternatif de quatre gènes distincts (68). Selon la base de données GWAS, ces SNP sont associés à la distribution corporelle du gras.

L'analyse de l'impact de la variation génétique sur la régulation de l'épissage des gènes a également été effectuée sur des populations de cellules immunitaires. Les cellules les plus étudiées dans ce contexte sont les cellules lymphoblastoïdes (LCLs). Une étude conduite par Li *et al* fait état de l'analyse de l'effet des variants génétiques sur les différentes étapes de la régulation génique dans les LCLs (69). Ils ont pu identifier 2 893 sQTL (splicing QTL) impliquant 2 313 gènes uniques. Leurs résultats présentent les sQTL comme indépendants des eQTLs. Ces derniers ont d'ailleurs un apport comparable aux risques liés aux maladies complexes relevés par les études GWAS. Bien que les QTLs d'épissage n'aient pas bien été caractérisés dans des cellules immunitaires activées, une étude collaborative récente a permis leur identification dans différents types cellulaires liés à l'immunité : monocytes, neutrophiles et cellules T (70). Les auteurs ont pu identifier plusieurs milliers de gènes dont l'épissage était affecté par la variation génétique interindividuelle. Ils ont observé que le contrôle de l'épissage semblait avoir un caractère spécifique au type cellulaire, ce qui renforce le rôle fonctionnel des QTL d'épissage. Bien qu'une étude ait procédé à l'analyse du contrôle génétique de l'épissage dans des macrophages murins (71), aucune étude ne porte sur les macrophages humains stimulés. L'implication de l'épissage alternatif dans une multitudes de désordres auto-immunitaires en fait un bon candidat pour comprendre la régulation génétique de la réponse immunitaire.

En résumé…

- La réponse immunitaire est un phénotype en constante évolution et qui implique des changements dans la régulation des gènes.
- La réponse est initiée par la rencontre de pathogènes par les cellules sentinelles du système immunitaire inné (ex. macrophages).
- Il existe des différences immunitaires notoires dans la régulation transcriptionnelle des gènes entre les populations humaines et ces différences sont liées à la variabilité génétique.
- L'épissage alternatif est un aspect important de la régulation des gènes.
- Ce processus implique plusieurs interactions entre des facteurs externes et la séquence primaire des gènes.
- L'avènement des technologies de séquençage de nouvelle génération et le développement d'outils statistiques bio-informatiques permettent la caractérisation des différents types d'évènements d'épissage alternatif.
- Les études d'association montrent l'implication des variants génétiques dans le risque aux maladies complexe.
- Les analyses QTL permettent d'associer la variation génétique à la variabilité de phénotypes quantifiable (ie expression [eQTL], épissage [sQTL]).
- L'identification des sQTLs dans plusieurs types de tissus cellulaires soulève l'importance de ce type de contrôle génétique sur la régulation des gènes.

1.4 Objectifs du projet

1.4.1 *Background* : Caractérisation du transcriptome de macrophages activés

Ce projet de maitrise s'insère dans une plus large initiative de caractérisation du transcriptome des macrophages lors de la réponse immunitaire innée. En 2016, Nédélec *et al* ont publié un projet qui fait principalement état des changements d'expression au moment où les macrophages sont stimulés par une présence bactérienne (13). Pour simuler une réponse immunitaire adéquate, deux espèces bactériennes ont été utilisées. De manière à avoir une vue globale et générale de la réponse immunitaire, les deux espèces ont été choisies en fonction de leur composition membranaire. En effet, les bactéries peuvent être classifiées selon deux grand groupes : Les gram- et gram+. Dans le cadre du projet, *Listeria monocytogenes* (gram+) et *Salmonella typhimirium* (gram-) ont été utilisés pour provoquer deux conditions distinctes d'infection chez les macrophages humains. L'objectif du projet visait à caractériser les différences immunitaires en comparant les profils transcriptomiques d'individus d'origines africaines et d'origines européennes. En utilisant la technologie NGS *RNA-Seq*, l'équipe de recherche a pu identifier des milliers de gènes montrant des différences populationnelles lors de l'infection bactérienne. Considérant la forte pression évolutive exercée par les pathogènes sur le système immunitaire, les auteurs ont voulu identifier l'apport génétique contribuant à ces différences. Une analyse eQTL a permis d'identifier la variabilité génétique comme source majeure des différences populationnelles observées pour 804 gènes. En plus de l'influence de la variabilité génétique sur l'expression des gènes, les auteurs ont également identifié 1 120 gènes dont la proportion d'isoformes était lié à des polymorphismes locaux. Cependant, ils n'ont pas considéré l'épissage alternatif comme une

mosaïque d'évènements distincts. Comme mentionné plus haut, la variabilité génétique peut affecter le traitement d'un exon particulier.

En utilisant un ensemble réduit des échantillons du projet de Nédélec *et al*, Pai et al ont conduit une recherche visant à caractériser les changements de traitement de l'ARN suite à l'infection bactérienne en considérant chaque exon épissé séparément (25). Ainsi, en utilisant le profil transcriptomique des macrophages de 60 individus dans les 3 conditions mentionnés ci-haut, plus de 1 000 évènements d'épissage montrent des différences significatives suite à l'activation des macrophages. De plus, cette étude a permis d'observer une directivité pour les évènements situés dans la régions 3' des transcrits. En effet, ils ont pu mettre en évidence un raccourcissement généralisé de la région 3' UTR, associé à une diminution des sites de ciblage par les miARN. Bien que ce projet ait pu mettre en évidence le rôle clé que joue le traitement de l'ARN dans la régulation de la réponse immunitaire innée, le nombre réduit d'échantillons amenait certaines limitations dont l'impossibilité d'associer statistiquement la proportion d'inclusion des exons à la variabilité génétique.

1.4.2 Projet de maitrise : rpQTL *mapping*

La réponse immunitaire induit des changements importants dans la régulation des gènes des macrophages lors de la rencontre d'un pathogène. Plusieurs études ont mis en évidence le rôle de la variabilité génétique sur le niveau d'expression des gènes dans un contexte immunitaire. L'épissage est un processus faisant intervenir un grand nombre d'interactions entre la machinerie de la cellule et la séquence primaire de l'ADN. L'implication de ce contrôle génétique dans le contexte de l'activation des gènes lors de la réponse

immunitaire demeure peu caractérisée. Dans le cadre de ce mémoire, nous faisons références aux rpQTL (RNA processing/ traitement de l'ARN). Ce terme est utilisé pour faire référence aux différents évènements qui mènent à la maturation de l'ARN, par opposition à l'étude de la proportion des isoformes (sQTL).

1.4.2.1 Objectif principal

L'objectif principal de ce projet de maitrise était d'analyser l'association entre la variabilité génétique interindividuelle et le traitement de l'ARN dans le contexte de la réponse immunitaire innée chez les macrophages humains. Les objectifs spécifiques du projet étaient :

1.4.2.2 Objectifs spécifiques

1) Caractériser l'épissage alternatif centré sur l'exon suite à une infection bactérienne des macrophages.

2) Associer les variations génétiques locales aux évènements d'épissage alternatif.

3) Intégrer l'influence des variations génétique sur l'expression à notre modèle.

4) Caractériser les variants responsables d'associations significatives avec le traitement de l'ARN.

Chapitre 2: *RNA processing QTLs modulate human macrophage responses to bacterial infection*

Article en preparation (EP)

2.1 Title and author list

RNA processing QTLs modulate human macrophage responses to bacterial infection

Olivier Tastet[1,2], Athma Pai[3], Ariane Pagé Sabourin[1], Yohann Nédélec[1,2], Jean-Christophe Grenier[1], Alain Pacis[2], Anne Dumaine[1], Vania Yotova[1], Luis B Barreiro[1,4,5*].

[1]CHU Sainte-Justine Research Center, Department of Genetics, Montreal, H3T1C5, Canada; [2]University of Montreal, Department of Biochemistry, Montreal, H3T1J4, Canada; [3]Department of Biology, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA; [4]University of Montreal, Department of Pediatrics, Montreal, H3T1J4, Canada; [5]University of Chicago, Department of Medicine, Genetics Section, Chicago, Illinois, USA.

2.2 Author contribution

Dr Barreiro is the principal investigator for the project and takes primary responsibility for the manuscript. Ariane Pagé Sabourin, Anne Dumaine et Vania Yotova worked on sample and library preparation as well as sequencing. Jean-Christophe Grenier and Yohann Nédélec worked on the processing and reception of sequencing data. Athma Pai has been a collaborator for the conception of the project as well a counselor for drafting the manuscript. Olivier Tastet performed all bioinformatic analysis, writing of the article and production of all figures.

Lors de ce projet, mon rôle fût de:

- Analyser les valeurs PSI estimées depuis le séquençage *RNA-Seq*.
- Associer la variabilité génétique à la variabilité de ce phénotype dans notre cohorte.
- Caractériser ces les associations identifiées.
- Interpréter les résultats.
- Écrire l'article scientifique présenté ci-contre.

2.3 Abstract

When triggered, the innate immune response initiates molecular changes in the cells that allows an appropriate defense to the detected threat. These changes will affect the transcription of immune related genes while other genes are turned off to efficiently let the organism fight the pathogen. Alternative splicing plays a major role in the fine-tuning of messenger RNA expression. Recent studies have shown that there were major shifts of RNA splicing following infection by two bacteria (*Listeria monocytogenes* and *Salmonella typhimurium*). This study aims to better define the role of inter-individual genetic variation on alternative splicing in the context of bacterial infection. Using 420 *RNA-Seq* samples sequenced in a previous study from the lab (13), we first defined a list of annotated alternative splicing events for which we could recover the proportion of the inclusion form (PSI-value). We used this phenotype as a proxy for isoform usage. Unlike studies focusing on the proportion of annotated transcripts, we isolated distinct categories of events and analyze the processing of exon. A multiple linear regression approach was then used to map the PSI-value on the genotyping data. We identified 1948 unique events significantly correlated to genotype (rpQTL) and hundreds that were specific to infected macrophages. In total, this set of events spanned 1 162 genes. These results show that the inclusion of certain exons is genetically controlled in macrophages and that some of this control only occur in infected macrophages This control affects genes implicated in the immune response and activation of the adaptive response. We also showed that alternative-first-exon (AFE) seem to be the least affected by genotype while alternative-last-exon (ALE) and TandemUTR are the most affected. Further analysis of rpQTL SNP showed independence between eQTLs and rpQTLs. Overall our results propose that rpQTL are an important force in gene regulation during the immune response.

2.4 Background

Innate immune cells, such as dendritic cells, natural killer cells, and macrophages, are the first actors recruited to respond to an invading pathogen. These cells are equipped with various pattern recognition receptors, which recognize a wide array of conserved pathogen-associated molecular patterns and discriminate between self and non-self molecules (72). Recognition of immune stimuli activates downstream molecular signaling pathways that culminate in the rapid induction of sophisticated transcriptional programs involving the regulation of thousands of genes (73). This cascade starts the process of pathogen clearance and the subsequent initiation of appropriate adaptive immune responses. Substantial inter-individual transcriptional variations are observed in the context of the innate immune system, including differences in gene splicing (74, 75).

It has been shown that differences in immune response across individuals is likely due to genetic variants affecting the expression levels of key immune-related genes (76-78). Changes in alternative pre-mRNA processing are key determinants in the modulation of gene expression. Post-transcriptional mechanisms, such as alternative splicing (AS), could lead to different functional mRNA/transcript variants from the same gene. Consequently, changes to the AS landscape are expected to have a significant impact on innate immune response (79). For example, pathogen stimulation of human macrophages has been shown to induce important changes in RNA splicing, specifically a directional shift tandem in 3'UTR regions (25, 80). In addition, a recent study by *Nédélec et al* (13) has suggested that isoform abundance is controlled by genetic variants in macrophages, referred to as alternative splicing quantitative

trait loci (asQTLs) (13). While asQTLs have been investigated in several studies before, the mechanisms involved haven't been examined in depth in the context of the immune response in macrophages.

Because of its potential to affect isoform abundance and modify functional domain of proteins, RNA splicing carries an interest in the context of many different diseases. Therefore, this process is at the center of several studies trying to elucidate the molecular basis of human health disorders. The QTL approach allows to identify sets of polymorphisms that affect specific events of RNA splicing and highlight variations that could play a role in disease susceptibility. Therefore, splicing QTL have been identified in multiple different tissues and conditions and the relevance of genetic variation in RNA splicing is well established. A recent study reviewed the involvement of polymorphisms in epigenetic and transcriptional variation in human immune cell lines and showed that thousands of genes undergo genetic control of RNA processing (70). This is concordant with multiple other studies conducted in brain and lymphoblastoid cell lines (LCLs) (67, 69, 81, 82). While there usually is an overlap between genetic-linked variation of expression and alternative splicing, splicing QTLs have been shown to be generally independent from eQTLs by looking at linkage between lead SNP and distribution of the physical location of the SNP relative to the genes (83, 84). While the impact of genetic variation on RNA processing in the context of the immune response is still poorly understood, there is strong evidence for the implication of alternative splicing in immunity.

Here, we analyzed *RNA-seq* data of non-infected and infected macrophages (with either *Listeria monocytogenes* and *Salmonella typhimurium*) from 140 individuals in combination with their microarray-based genotype data. We observed the influence of thousands of genetic variants in local RNA processing events (rpQTLs) and that infection-specific rpQTLs are independent from the effect of mapped eQTLs. We refer to these associations as RNA processing rather than splicing because alternative length of polyadenylated tail of transcripts isn't necessarily yielded by the splicing machinery. Together, our results support the idea that genetic control of RNA processing plays an important role in the innate immune system.

2.5 Material and Methods

2.5.0 Sample Set. Samples consisted of buffy coats from 175 healthy male donors obtained from the Indiana Blood Center. The population of individuals was composed of African-Americans (n=76) and European-Americans (n=99). The choice of studying only male subjects allows to avoid the confounding effect of sex-specific differences in immune response. Monocytes were isolated from peripheral blood mononuclear cells and derived into macrophages with MCSF (*Macrophage colony-stimulating factor*) (20ng/mL; R&D systems). Macrophages were then infected with *Listeria monocytogenes* and *Salmonella typhimirium*. DNA extraction of samples was done with the PureGene DNA extraction kit (Gentra Systems). All samples were genotyped using the Illumina HumanOmni5Exome BeadChip which allows to characterize > 4.3 millions of variants. Genotype calling was done using Genome Studio v2010. Only samples with genotype calling rate > 98% were kept (2 samples rejected). SNP with > 5% missing data and violating Hardy-Weinberg equilibrium were excluded from the analysis ($P < 10^{-5}$). In total, 4 452 246 SNP passed the quality control. Imputation was done with Impute2 (v 2.3.0). After keeping SNP with minor allele frequency >5% and applying the mentioned filters, we used 6 943 841 SNP.

2.5.1 *RNA-Seq*. Total RNA was extracted from non-infected and infected macrophages using the miRNeasy kit (QIAGEN). Samples that did not show RNA degradation were kept. The Illumina TruSeq protocol was used to prepare RNA-sequencing libraries. The cDNA libraries were next sequenced with single-end 100bp reads on the Illumina HiSeq2500 sequencing machine. Sequencing was done in CHU Ste-Justine (315) and McGill's Innovation Center (166)

2.5.2 Estimation of PSI-values. PSI-values were estimated using the MISO software (v0.4.9) using default settings and hg19 version 1 annotation (from MISO website, http://miso.readthedocs.org/en/fastmiso/annotation.html). The software provides one PSI-value per annotated RNA processing event per sample. We could accurately estimate PSI-values for 140 of the 175 individuals. Indeed, 35 samples presented a high rate of missing data. Events were used in downstream analysis if enough informative reads detected in both infected and non-infected macrophages across at least 90% of samples. These events were further classified into 5 major categories based on their relative location to the gene: alternative first exon (AFE; n = 8 547), alternative last exon (ALE; n = 5 823), skipped exon (SE; n = 17 943), retained intron (RI; n = 3 417), and alternative polyadenylation sites, leading to tandem untranslated region (TandemUTR; n = 1 859).

2.5.3 Statistical testing. Statistical testing was done using the free software environment for statistical computing and graphics R (85). In this study, we used Fisher test and T-test when suited.

2.5.4 Principal component analysis. To explore the variation in the PSI values, we used *prcomp*, an R package made to conduct a PCA analysis. Batch and Individual effects are regressed out of the values. Preliminary analysis showed sequencing center clustering (**Supplemental Figure 3;** page LXXIII). Therefore, we removed this technical bias in PSI variability by keeping the residuals of linear regression model where PSI ~ Sequencing_center. After considering pairing in the samples, we observed clear condition clusters alongside PC1.

2.5.5 Gene ontology enrichment of genes driving the first principal component. To identify the set of genes that are driving the clustering of samples by infection status, we calculated the correlation between the splicing ratios and the dispersion across the PC1 axis. Only genes with spearman coefficient greater than 75% were kept. RecursiveGO was then used to remove redundant GO enriched terms. The implementation of recursiveGO is not yet published (**Supplementary Materials**; p. LXXXV-CX). The algorithm uses an iterative approach which set genes aside as soon as they are used for a given term. This avoid the redundancy of several GO categories bearing the same set of genes . However, to compare different GO ontology across conditions, a more traditional approach (GOrilla; http://cbl-gorilla.cs.technion.ac.il/) was used (see section 2.5.11).

2.5.6 Detecting significant changes in RNA processing in response to infection. To characterize the changes in splicing following infection, we used a linear regression model where we corrected for the covariates regressed out in the PCA (Sequencing center and Individual effect). For each category of splicing events and infection condition, we used a linear regression model (*lm* function in R) with the following design: *PSI values ~ Sequencing_center + Individual_effect + Condition*.

2.5.7 FDR correction. Throughout the analysis, FDRs were calculated following the idea proposed by Storey et Tibshirani (86). Our approach is a two-component modelling of the distribution of p-values where the observed values are compared to the null expectation, which is given by permutation. Unlike the original method, our implementation allows the null

expectation to follow a non-uniform trend. We used the same implementation as in Nédélec et al. The implementation was done under R (**Supplementary Materials**; p. XCII-CXXXIII).

2.5.8 PSI filtering and normalization. With the PSI estimates obtained from MISO, we first removed all events that had no variance across samples. We then needed to filter out events that were wrongly annotated. PEER was used to correct for surrogate variables and batch effect (70, 87).

2.5.9 RNA processing QTL mapping. rpQTL mapping was done using the *matrix eQTL* package in R (88). This package allows a time-efficient mapping of local polymorphisms onto a given phenotype (PSI). We tested SNP within 100-kb flanking the gene containing the event. We conditioned the window according to the gene instead of the splicing event itself so that multiple events within a given gene were tested against the same set of SNP. We also used the first 5 principal components of the genotyping data to account for population structures in the samples. To facilitate downstream analysis, we kept the most significant association per splicing event. An FDR correction was then obtained using the method mentioned above. Null expectation was set by permuting (10 times) the genotype-individual relationship but keeping the SNP-wise distribution of genotype. The QTL mapping was performed for each permutation and these P-values were used as the null distribution. We used this approach to maintain LD blocks which plays a role in the background signal.

2.5.10 Estimating condition-specificity. To determine whether an rpQTL is specific to infection, i.e. the association is only significant following bacterial infection of the macrophages, we used a relaxed FDR-threshold approach as proposed in *Nédélec et al* (13). To assert the certainty that the association is not significant in each condition, we verify that the FDR is greater than 30% in the other conditions. For instance, an rpQTL is infection-specific if it has a FDR<5% in at least one of the infected conditions and has FDR>30% in non-infected samples.

2.5.11 Comparing Pathway enrichment across conditions. To characterize the pathway enrichments of rpQTLs we performed a GO enrichment analysis using the online tool GOrilla (89). Unlike recursiveGO, this approach yields more redundant categories which allows to better compare the enriched terms across conditions as genes are not dismissed if used in a given GO term. Therefore, the overall set of GO categories may contain redundant terms, but this feature allows to better define overlap of pathways between two distinct analysis. An enrichment is called as significant if it has an FDR<10%.

2.5.12 Overlap between splicing categories. Finding the overlap between categories of splicing event is done using a random sampling approach. We first determine how many genes are overlapping between 2 categories (separating between infection-specific and shared rpQTLs as explained in 2.5.10). We then randomly sampled the same number of genes among all tested genes within categories to establish if the initial overlap is greater than what would be expected by chance.

2.5.13 Integrating eQTLs to rpQTL mapping. To integrate eQTLs into our rpQTL modelling, we used the best association found in each respective set and a linear regression approach using R. For every significant rpQTL, we find the associated eQTL SNP which we incorporate into the model. A basic model is made by using the same corrected values as with *matrix eQTL*. By comparing the effect sizes associated with the rpQTL SNP before and after integrating the eQTL SNP, we can quantify the effect of eQTL on the rpQTL. To determine if the change in effect sizes is significant, we use a permutation approach where the integration is also done using a set of 1000 SNP with allele frequency similar to the eQTL SNP (+/- 2%). To quantify the effect of integrating eQTL into rpQTL mapping, we simply calculate the difference in effect sizes before and after integrating the eQTL SNP and divide it by the initial effect size found without considering the eQTL. This allows us to determine a percentage of change in effect size in respect to initial effect of the rpQTL SNP.

2.5.14 Comparing rpQTL-SNP locations. Since we are as interested in SNP falling in the 3' downstream region as in SNP falling in the 5' upstream, we used a binning approach to compare the location of SNP associated with splicing. This approach was used so that we can easily compare 3' SNP versus 5' SNP. Gene bodies and 20-kb regions upstream and downstream were each divided into 200 intervals. We gathered data from windows within each of these intervals and plotted the density of SNP for all windows overlapping each position.

2.5.15 Comparing associated SNP between event categories. Comparison of rpQTL-SNP is done by comparing the polymorphisms affecting different events within the same gene. Linkage disequilibrium is calculated for all pairs of SNP using plink v2.0 beta (90). To compare the classes of events, we calculated the median LD of all pairs of SNP for a given pairwise comparison of event category. When the same SNP was involved in different events within the same gene, the LD was fixed at 1.

2.5.16 Enrichments. All following enrichment analysis are done using a multiple logistic regression approach. Enrichments were used to find statistical overrepresentation of VEP terms, eQTL genes, TF binding sites and RBP binding sites among the set of rpQTL-SNP. Multiple logistic regressions can be used with binomial nominal variables to predict the probability of a value of the Y variables as a function of the X variables. Using binary annotation of SNP and genes, we can calculate odds ratios of the likelihood of the X variable co-occurring with the Y variable. The function was implemented in R (**Supplementary Materials**; p. XCI). All enrichments of rpQTL SNP consider ties as it is very hard to assume the best SNP when more than one polymorphism has the lowest P-value.

2.5.17 Transcription factor (TF) binding sites. To determine if specific TF binding sites are more likely to harbor causal SNP for an rpQTL, we obtained TF footprints in non-infected, *Salmonella*-infected and *Listeria*-infected macrophages, using previously published ATAC-seq data (13). The enrichment of TF binding locations among rpQTL was estimated using multiple logistic regression.

2.5.18 RNA Binding Proteins (RBP) binding sites. To consider motif enrichment related to RBPs we used the online tool MEME (http://meme-suite.org/tools/meme-chip). The MEME-Chip tool was used with the *Differential Enrichment mode.* The reference motifs were from the Ray2003 Homo sapiens (DNA-encoded) database under the RNA (DNA-encoded) category. Regions of 10 bp around the rpQTL SNP (+/- 5bp) were extracted from the fasta reference genome. To identify a set of RBP bound region of the genome, we used data provided by the ENCODE project (https://www.encodeproject.org/). Using BEDtools, we only kept the regions overlapping between the two cell-types available (K562, HepG2) as these regions were less likely to be undergoing cell-type specific interactions (91).

2.5.19 GWAS Enrichment Analyses. We downloaded the NIH GWAS catalog and performed enrichment as described in 2.5.16 (https://www.ebi.ac.uk/gwas/docs/file-downloads). To consider the enrichment for specific GWAS-identified traits, we only kept these traits that had more than 20 associated SNP to lighten computational burden of the enrichment analysis.

2.6 Results

2.6.1 Bacterial infection induces extensive changes in RNA processing events

In order to identify infection-induced changes in RNA processing, we utilized the probabilistic framework implemented in the software Mixture of Isoforms (MISO), which characterizes changes in exon usage from *RNA-seq* data. Specifically, we calculated a "percent spliced in" (PSI) value for each of the transcript variants, across all the macrophage samples. This measure quantifies the ratio of inclusion for each RNA processing event. We identified 40 335 individual RNA processing events within 8 829 unique genes across all samples. After filtering of the PSI-values we kept 37 470 events spanning 8 370 unique genes. Principal component analysis using PSI values from the identified RNA processing events revealed sample clustering by infection status (**Figure 8A,** page 47**; Supplemental Figure 4**, page LXXIV), which indicates that the establishment of the transcriptional program engaged upon infection is likely regulated at the level of specific exon usage rather than full isoforms. Indeed, we found that genes containing exons whose usage is highly correlated ($R^2 > 0.75$) with the first principal component were strongly enriched among functional GO terms directly related to immune function, including defense response to other organisms (GO:0098542; $P = 5.62 \times 10^{-6}$, OR=3.55) and cytoskeleton organization (GO:0007010; P = 5.52e-5, OR=1.93).

We next used a linear model framework to identify RNA processing events that significantly differ between *Listeria*- or *Salmonella*-infected macrophages versus non-infected controls. We found evidence for substantial differential RNA processing in immune-related genes after infection (**Figure 8C,** page 47**;** FDR < 1% and $|\beta| > 0.05$), corroborating our previous findings using a smaller sample size (25). Moreover, we found infection-specific changes in

RNA processing, with a more extensive shift in response to *Salmonella* (21.83% (8 166) versus 9.77% (3 618) of all tested events (37 470) after infection with *Listeria*) (**Table I**, page LXXXIII). This observation is more pronounced for retained introns and tandem 3' UTRs (RI: 26% versus 14% [P=9.1 x 10^{-25} Fisher], and TandemUTR: 28% versus 12% [P=7.62e x 10^{-31} Fisher] differential usage, after infection with *Salmonella* and *Listeria*, respectively). Finally, we observed widespread changes in polyadenylation site usage following infection. For both ALEs and TandemUTRs, we see significant shifts towards usage of upstream polyadenylation sites, represented by mean ΔPSI value < 0 or $Beta_{NI/(S|L)} < 0$ in both exon categories (72.1% and 74.5% of ALEs, P < 2.2 x 10^{-16} [Fisher test] and 90% and 95.4% of TandemUTRs, P < 2.2 x 10^{-16} [Fisher test]; **Figure 8E**, page 47; **Supplemental Figure 5**, page LXXV).

While there are substantial RNA processing changes inherent to the immune response after infection with bacterial pathogens, it is less clear which transcriptome changes might underlie inter-individual variation in susceptibility to infection and the resulting immune response. Indeed, there is substantial variability in exon usage across individuals, in all conditions (**Figure 8F**, page 47). This is particularly true for differentially processed events; whose mean usage significantly changes following infection. Furthermore, there is significantly more variability in exon usage after both *Listeria* and *Salmonella* infections, suggesting that RNA processing changes might be associated with differences in immune responses.

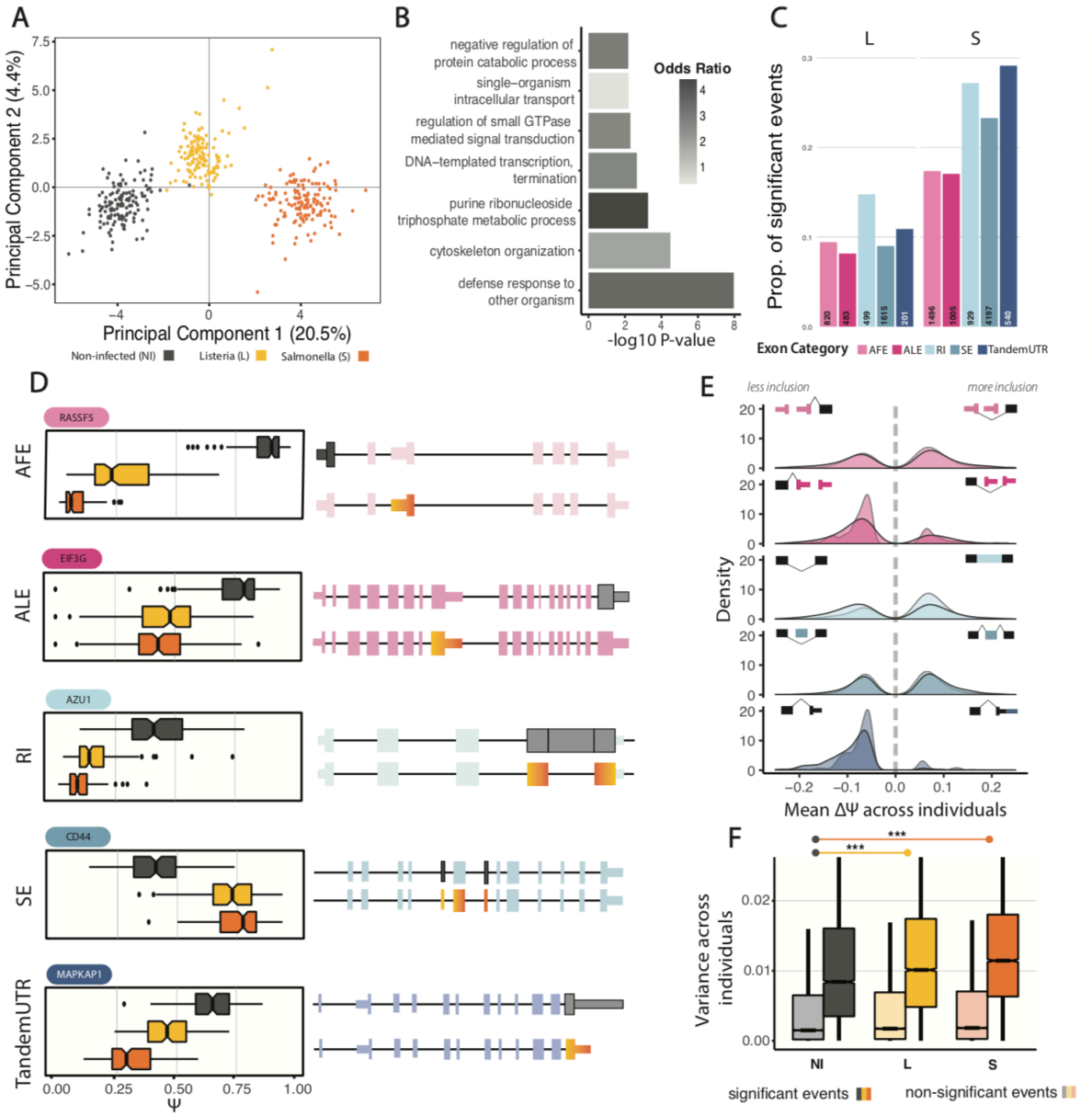


Figure 8 Bacterial infection induces extensive changes in RNA processing events

A) First 2 principal components of corrected PSI-values, colored by condition. **B)** Gene Ontology enrichment using a recursive foreground-background approach. Foreground is defined as events/gene with > 75% correlation with PC1**. C)** Proportion of splicing events significantly changing following infection (FDR<1%, | Beta | > 0.05). **D)** Examples of differentially spliced exons for all categories. On the left side, boxplots showing the PSI-values, separated by condition; on the right side, schematic representation of the genes, showing the preferred splicing conformation for infection (orange/yellow) and non-infection (grey). **E)** Distribution of ΔPSI values for the previously identified differentially spliced events. The left side represents less inclusion/use of most proximal/shortening of 3' UTR region. **F)** Distribution of variance across all individuals for DS events (darker) and non-DS events (lighter). Significance is shown for comparisons between whole distributions between conditions. (NI>S and NI>L)

2.6.2 Genetic control of RNA splicing in macrophages

To better understand the genetic effects on RNA processing, particularly in response to an immune stimulus, we mapped genetic loci associated with variation in exon usage. After filtering and normalization of the PSI values (92), we used *matrix eQTL* to test for associations between PSI values for individual events and all single nucleotide polymorphisms (SNP) within a 100 kb cis-window around the gene containing the event. Significant associations were independently assessed within each event type-condition context (**Figure 9A**, page 51). Across all conditions and events, we identified 1 860 significant RNA processing QTLs (rpQTLs; FDR < 5%), which represent 5% of the 37 470 events that were tested. The number of rpQTLs identified across infection conditions were approximately equivalent, but there were large differences in the number of rpQTLs identified across event types (**Table II**; page LXXXIII). ALEs were the most affected by genetic variation (438 ALE-QTLs representing 8% of ALEs tested), while AFEs have the lowest proportion of significant rpQTLs (**Figure 9B**, page 51; 152 AFE-QTLs representing 2% of AFEs tested).

In order to identify genetic variants that may be specifically responsible for the regulation of RNA processing upon immune response, we looked for the overlaps between rpQTLs occurring in non-infected and infected conditions (**Figure 9B**, page 51). To account for differences in statistical power between conditions, shared rpQTLs were defined as those significant in at least one condition (FDR < 5%) and significant with a relaxed threshold in the other conditions (FDR < 30%). Using this conservative approach, approximately 30% of rpQTLs are infection-specific (343 genes), which is similar to the proportion of infection-specific eQTLs in this dataset (13). There is variability in infection-specificity across event types, where AFE-

QTLs have the greatest amount of overlap across conditions, while ALEs and SEs have the greatest proportion of infection-specific QTLs. Gene ontology analysis of rpQTLs within each condition indicate that rpQTLs in the *Listeria* and *Salmonella* infection conditions are more enriched in genes with immune activation and immune-cell specific functions than in *Non-infected* (**Figure 9C**; page 51). In line with this, there are many infection-specific rpQTLs in various immune-related genes implicated in cell differentiation (*CD151*, *CD36*, *CD44*, *CD55*, *CD82*, *CD86*), transcriptional response to infection (*IRF4*, *IRF5*, *IRF7*, *IRF8*, *NFKB1*) and pattern-recognition receptors that allow initiation of inflammation (*NOD1*). Many other key immune genes (*i.e.*, *TLR2, IL32, CD46, TNIP3, HLA-F)*), may be identified as infection-specific QTLs with relaxed thresholds (FDR INF <10% and FDR NI >30%).

An alternative approach for identifying genetic variants specifically relevant to immune response is to map response QTLs, where the genetic variant is associated specifically with the change in regulatory activity. We identified 36 events with response rpQTLs by mapping ΔPSI values representing the change in exon usage between the non-infected and Listeria or Salmonella conditions. These represented a very low proportion of all tested events (0.15%; FDR < 5%; **Supplementary Figure 7**, page LXXVI). Of these events, 21 were also had significant differential RNA processing between non-infected and infected conditions and 11 had infection-specific rpQTLs (e.g. TNFSF13B; **Supplementary Figure 8**, page LXXVII).

Lastly, we compared every splicing class with one another to test if different types of rpQTL were more likely to co-occur on the same gene than expected by chance alone. We observed that SEs overlapped significantly with all types of events and that genes associated with AFEs rpQTL were also more likely to have ALEs rpQTL than expected by chance (**Figure**

9D; page 51). Interestingly, even though TandemUTRs significantly overlapped with SEs, the median LD of all pairs of SNP of SE and TandemUTR occurring in the same genes is lower than the rest of the overlaps. This indicates that while SEs and TandemUTRs are likely to occur together, they seem to be governed by unrelated SNP.

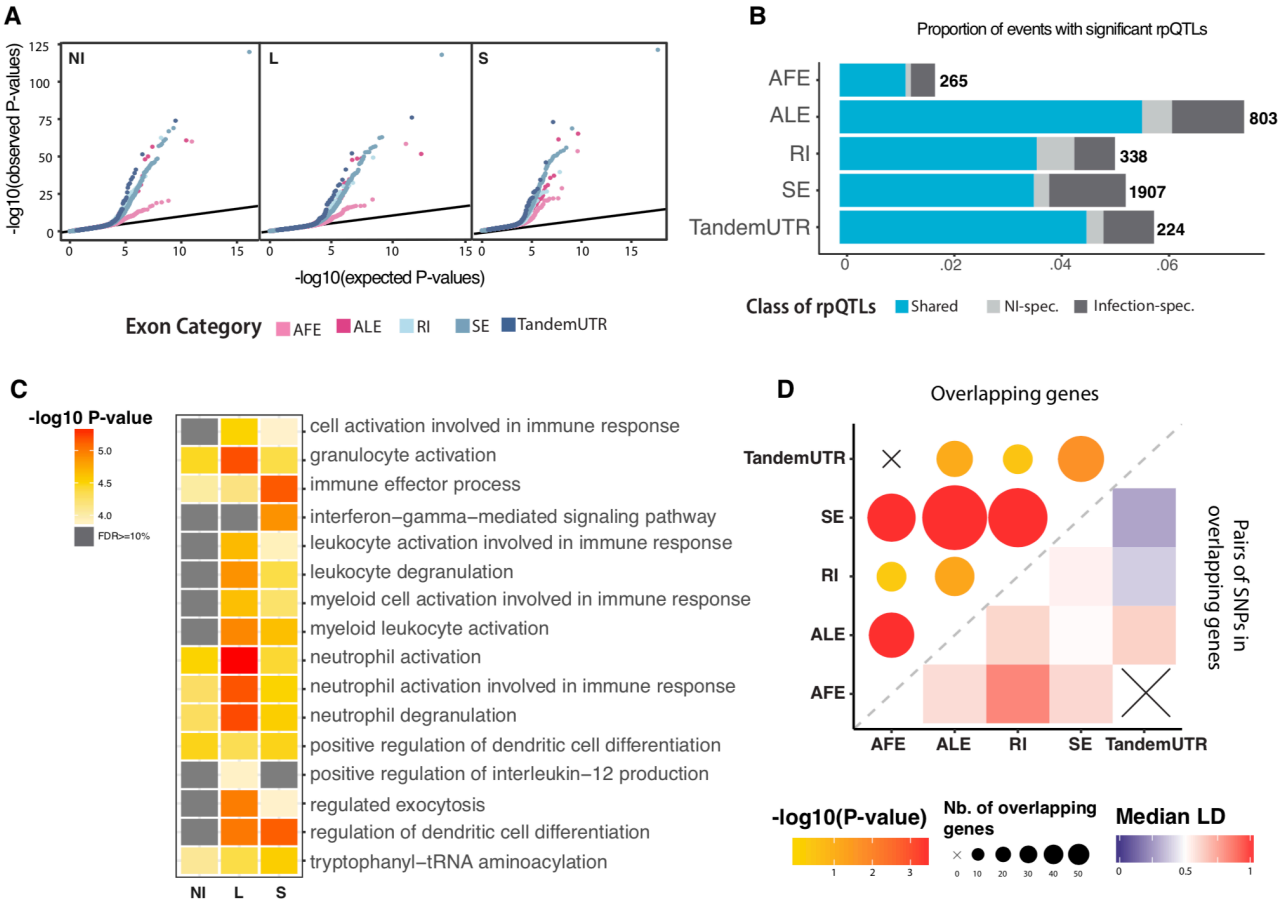


Figure 9 Genetic control of RNA splicing in human macrophages

A) QQ-plots of the mapped rpQTLs. Expected P-values are obtained using permutations of the genotype data. **B)** Proportion of rpQTLs across all categories of splicing. Colors correspond to condition specificity. **C)** GO enrichment for rpQTL genes across all 3 conditions. GO analysis is performed using GOrilla to allow a better comparison across conditions. **D)** Overlap between categories of event. Top left part shows the significance of gene overlap between pairwise comparison. Size of dots is proportional to the number of genes overlapping and colour represents significance of overlap. “X" marks 0 overlap. Bottom right shows median LD when considering all pairs of rpQTL SNP associated with events occurring within the same gene.

2.6.3 Comparing splicing QTLs and expression QTLs

It is unclear how much of the observed rpQTLs might be linked to genetic variation influencing expression levels. To answer this question, we first used a logistic regression approach to test the enrichment of rpQTL genes for eQTL genes (**Figure 10A**, page 54). Generally, we see a significant enrichment of eQTL genes among rpQTL genes suggesting that genetic cis-regulation of RNA processing is linked with genetic cis-regulation of expression. We then sought to quantify the influence of the best eQTL SNP on the effect of the rpQTL association. To do so, we used a linear regression model that recalculates the association between the best rpQTL SNP and the PSI-value while accounting for the effects of the best eQTL SNP associated with the same gene as identified in *Nédélec et al (13)*. Comparing the effect sizes of the rpQTL before and after accounting for the eQTL SNP allows to identify events where the variation in splicing associated with genotype is linked to a similar variation in expression (**Figure 10B**, page 54). Overall, the effect size of 63% of rpQTLs located in an eQTL gene are significantly affected by the strongest eQTL (1000 permutations, $P<0.01$). However, the distribution of the percentage of change is bimodal and suggest a minor cluster of genes strongly affected by eQTLs (change > 40%) (**Supplementary Figure 9**, page LXXIX). When considering this threshold, we see that only 13% of rpQTLs have a significant impact of the eQTL on the genetic regulation of RNA processing. Our results are consistent with other studies that suggested that rpQTLs has an independent effect from eQTLs but there still is a small fraction of rpQTL where the variation of expression and the variation of splicing is linked to similar genetic variation. By stratifying our model by event type, we see that AFEs have a higher proportion of rpQTL with an eQTL effect (AFE:21.9%, ALE:14.6%, SE:13.2%, RI: 14%,

TandemUTR:11.9%). We also observed that the effect of the best eQTL SNP had a stronger effect on AFE rpQTL SNP compared to all other events ($P=1.899 \times 10^{-5}$ [T-test], **Figure 10C**, page 54). Gene ontology analysis showed that rpQTL without eQTL effects are strongly enriched for T cell mediated immunity (GO:0002456; $P=6.03 \times 10^{-6}$, OR=13.2) and other immune functions (e.g. type I interferon signaling pathway (GO:0060337; $P=1.12 \times 10^{-4}$, OR=4.32), negative regulation of leukocyte activation (GO:1902106; $P=8.12 \times 10^{-4}$, OR=4.21) (**Figure 10D**, page 54). These results suggest that rpQTLs are functionally relevant to the immune response and that this relevance is independent from the effect of eQTLs. Further investigating the genetic variation involved in these associations will shed light on the mechanisms involved in the rpQTL associations.

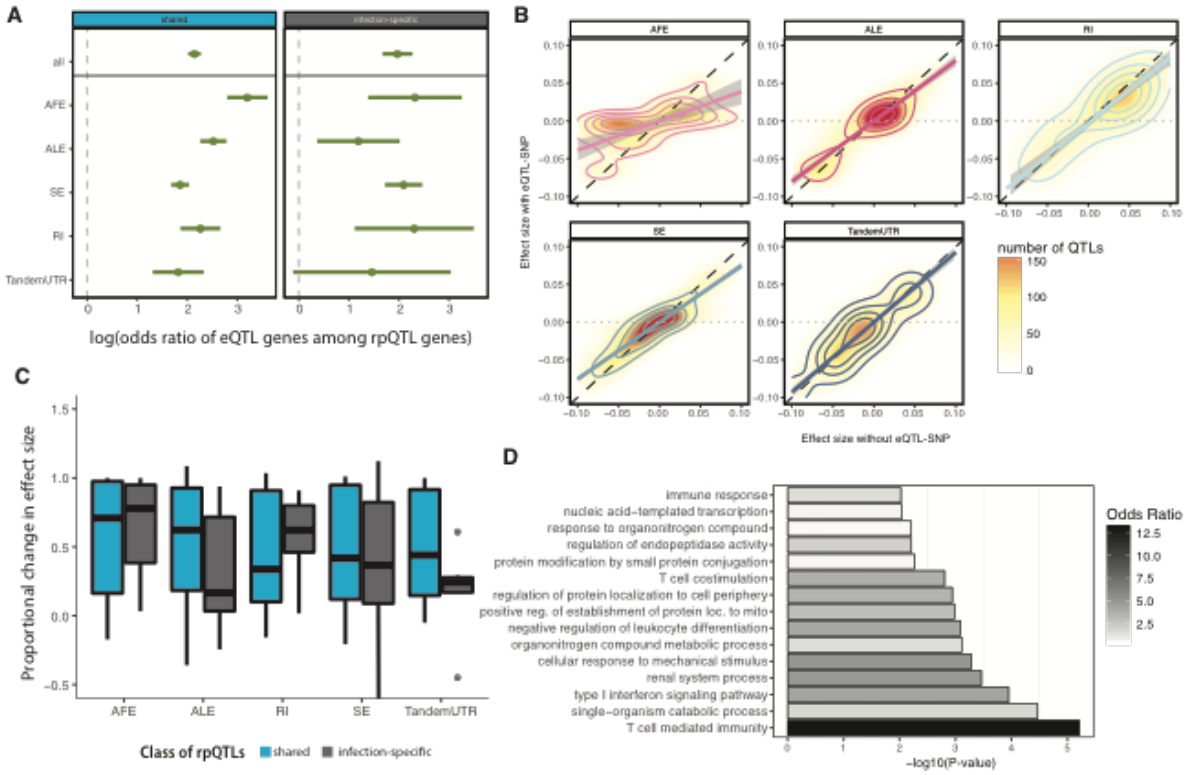


Figure 10 Integrating eQTL to rpQTL mapping

A) Enrichment of eQTL genes among genes with at least one rpQTL mapped. Enrichment is calculated using a logistic regression approach and error bars are calculated using standard deviation. Enrichment is displayed as log2 odds ratio. **B)** 2-dimensional density plots showing the effect of integrating the best eQTL SNP while mapping infection-specific rpQTLs. On the X-axis, the effect size attributable to the rpQTL SNP without considering the best eQTL SNP; on the Y-axis, the effect size attributable to the rpQTL SNP after considering the variation attributable to the eQTL SNP. **C)** For genes that are both eQTL and rpQTL, the distribution of differences in effect size given by (*beta-beta')* / *beta*. **D)** GO enrichment done with the recursive approach for rpQTL genes that are not significantly affected by taking the eQTL effects into account.

2.6.4 Functional annotation of rpQTLs

To gain a better understanding of the functional relevance of rpQTLs, we next characterized the genetic variations involved in these associations. We first investigated the location of the QTL SNP. As expected, eQTL SNP are primarily located close to the transcription start site (TSS) whereas rpQTLs are enriched for more intragenic locations (**Figure 11A**, page 58). Interestingly, eQTLs implicating genes that are also rpQTLs and where genetic effects are confounded have a high density of SNP located in the gene body, suggesting the expression of this set of genes may be linked with transcript processing rather than transcription initiation (**Supplemental Figure 10**, page LXXX). When the RNA processing events are broken down into categories, we observe a clear enrichment for transcription end site located SNP in TandemUTRs rpQTL (**Supplemental Figure 11**, page LXXXI). This result indicates that, unlike eQTLs, rpQTLs involving the 3' end of the gene generally work through genetic variation that affects the RNA processing event itself. Interestingly, AFE SNP are not necessary located near the TSS. As shown in **Figure 9D**, AFEs and ALEs overlap significantly, therefore it is possible that the AFEs are sometimes associated with ALEs involving an expression-altering feature (i.e. miRNA, RBP binding site, etc). Our observations enforce the idea that genetic control of RNA splicing works through modulation of the processing events themselves and involve variation at the regions implicated in the splicing of the gene, further supporting the idea that rpQTLs are independent from eQTLs.

We next used the Variant Effect Predictor (VEP) annotation to characterize the potential consequence of SNP involved in QTL associations (**Figure 11B**, page 58). VEP is an online resource released by *Ensembl* that provide annotation for genomic variants in coding

and non-coding regions (93). As expected, non-ambiguous rpQTLs were enriched for splicing junctions and non-ambiguous eQTLs were rather enriched for regulatory elements and 5' UTR variants. These results highlight the genetic differences in the eQTL polymorphisms and rpQTL polymorphisms. When looking at the ambiguous set, we see that eQTLs are enriched for splicing junctions, which might indicate that some of the variation found in expression among eQTLs is potentially due to alteration of the RNA processing mechanisms. Interestingly, ambiguous rpQTLs are mostly enriched for stop lost and miRNA binding sites. These consequences are likely to affect the overall level of transcripts, indicating that splicing regulating the inclusion of these features might often be the cause of variation in expression.

We next considered TF binding by using ATAC-Seq foot-printing (94). Further differentiating rpQTLs from eQTLs, we observed that the latter was significantly enriched for TF binding sites ($P = 2.40 \times 10^{-118}$; OR=2.25) while the former was not (P-value=0.31; OR=0.8) (**Figure 11C**, page 58). Since RNA splicing is dependent on the binding of several RNA-Binding proteins (RBP), we also considered the relationship between RBP binding sites and rpQTLs. We first considered the enrichment for known motifs around the QTL SNP. Using the online tool MEME, we found enrichments for several known RBP motifs in rpQTLs but did not find any in the set of eQTLs (**Table III**, page LXXXIV). We also used the eCLIP RBP peaks data identified in the project ENCODE to identify if rpQTL-SNP are more likely to fall in these regions of binding that expected by chance (95). We identified that rpQTLs ($P = 2.76 \times 10^{-52}$; OR=1.69) were more enriched than eQTLs ($P = 8.34 \times 10^{-6}$; OR=1.12) for RBP binding sites (**Figure 11C**, page 58). Interestingly, the comparison is not as striking as when looking at TF binding sites. This might be explained by the role of RBPs in expression. Indeed, some of these factors are involved in

localization and transport of the mRNA, therefore affecting the overall level of transcripts without interfering with initiation of the transcriptional program. These analyses help to highlight the differences between the mechanisms of eQTLs and rpQTLs. Indeed, while the eQTLs seem to generally be due to variation that affect the transcription process, rpQTLs are linked to variation in splicing junctions and RBP binding sites. Interestingly, a set of eQTLs are related to splicing variation that potentially affect the overall levels of transcripts. As shown, the splicing mechanism is affected by genetic variability in the recognition of splicing junctions by the spliceosome as well as trans factor binding. These influence the transcriptional landscape of immune genes and therefore, certainly play a role in inter-individual variability of the macrophage response to pathogens.

Finally, we used the GWAS catalog to test if our set of rpQTLs was enriched among SNP associated by GWAS with any disease of quantitative trait. We found that rpQTLs were strongly enriched among GWAS-associated SNP (P-value=2.43 x 10^{-6}, OR=1.55; **Figure 11D**, page 58). The ambiguous group shows a more significant enrichment (P-value=5.08 x 10^{-7}, OR=1.94) than the non-ambiguous one (P-value=0.025, OR=1.34), indicating that the genetic control of RNA processing associated with genetic control of expression is likely involved in the variation of GWAS-trait (**Supplementary Figure 12**, page LXXXIV). We also separated the analysis by GWAS trait and only kept those that had more that 20 SNP associated (657 traits). Strikingly, we found several enrichments for blood-related traits linked to variation in cellular counts (e.g. white blood cell count), indicating that rpQTLs might represent a putative mechanism for the extensive level of interindividual variation in the cellular composition of whole blood samples (**Figure 11D**, page 58).

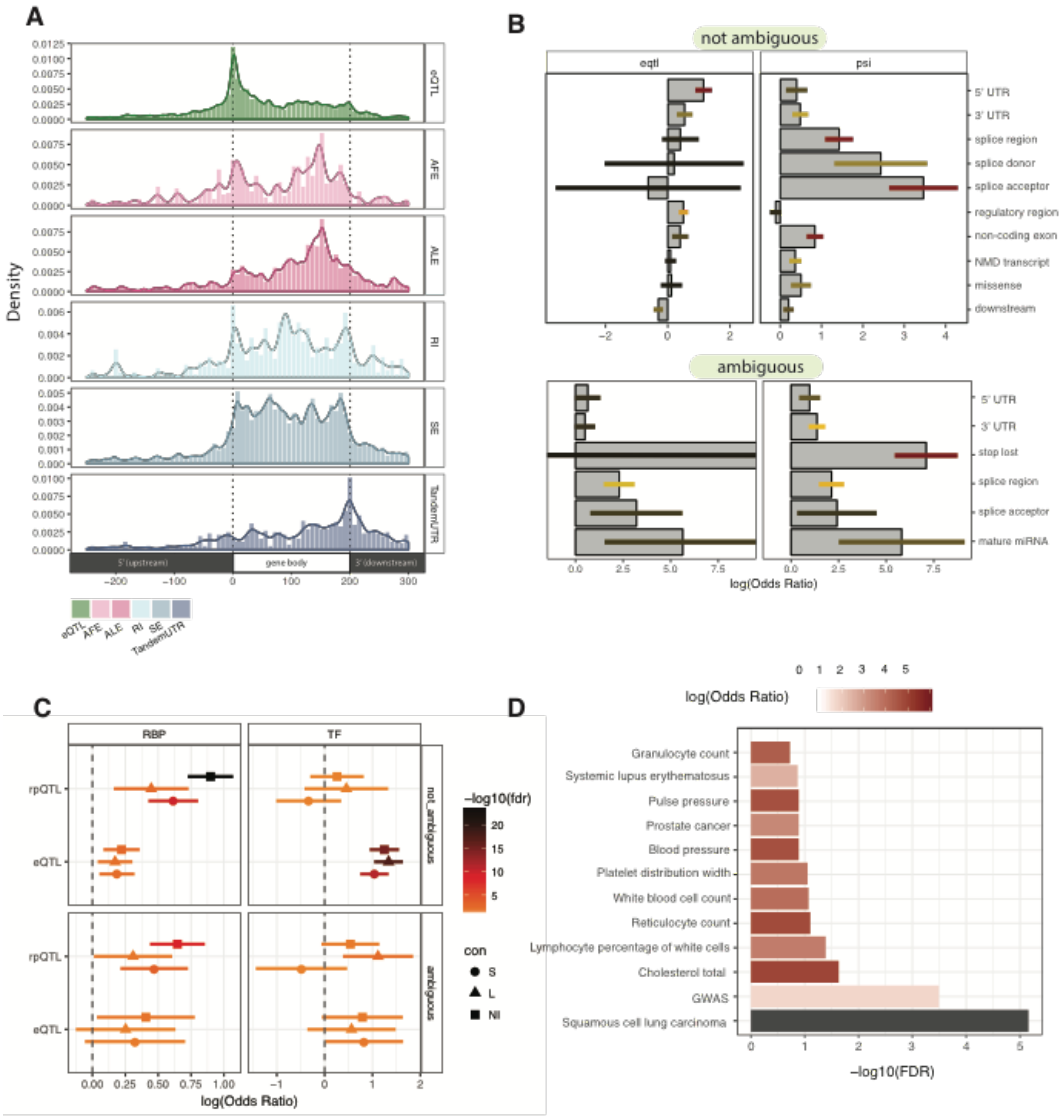


Figure 11 Annotation of rpQTL genetic variation

A) SNP location of significant QTLs divided in 6 categories: eQTLs, AFE-rpQTLs, ALE-rpQTLs, RI-rpQTLs, SE-rpQTLs, TandemUTR-rpQTLs. The distinction between rpQTL significantly affected by eQTL genetic variation and those that are not is found in the supplementary section. Dotted lines represent start and end of the QTL gene. We used a binning approach to consider both the distance from the TSS and the TES at the same time where every gene is divided in 200 bins, on the Y axis. **B)** VEP enrichment plots for the different significant VEP annotation terms. Colored bars are the 95% confidence. eQTLs (left) and rpQTLs (right) are shown to allow comparison. Associations with expression and RNA processing linked to similar genetic variation are shown as the ambiguous group. **C)** Enrichment for RBP binding sites and TF binding sites among rpQTLs and eQTLs, separated between ambiguous and non-ambiguous QTLs. **D)** Enrichment of rpQTL SNP for GWAS-identified genetic variation. The GWAS term refers to all GWAS SNP together.

2.7 Discussion and conclusions

In this study, we examined RNA processing events during bacterial infection using *RNA-Seq* data on non-infected and infected macrophages from 140 individuals. The considerable number of available samples allowed to confirm the consistent changes that occurs in the alternative splicing landscape of macrophages when stimulated with bacterial agents. We used MISO to analyze different classes of events separately and found that TandemUTRs and retained introns are the most affected by infection. While these changes have already been shown in previous publications, the association of genetic variation and variation in splicing in human macrophages in the context of bacterial infection was not well characterized. We therefore used genotyping to identify 1 890 significant associations (FDR<5%) spanning 1 162 unique genes. Furthermore, we achieved to identify 427 events that were only significantly control in at least one of the infection condition but not in the non-infected cells. These rpQTLs involved sets of genes enriched in immune related functions, suggesting genetic variation plays a role in the regulation of the immune response through splicing.

The integration of the eQTL mapping done by *Nédélec et al* allowed us to show that while rpQTL genes were likely to also be eQTL genes, these two effects are generally distinct from one another. Indeed, only 13% of all rpQTLs have an eQTL effect. When looking at the location of the eQTL-SNP involved in these ambiguous associations, we still observe a peak near the TSS but also a considerable density of SNP falling in the gene body (**Supplemental Figure 10**, page LXXX). Polymorphisms associated both in eQTLs and rpQTLs were significantly enriched for splicing junctions, suggesting that the overlapping genetic variation between eQTLs and rpQTLs is rather due to alteration of the splicing machinery. We extended our

analysis to the characterization of the SNP involved in the splicing QTLs. Further differentiating eQTLs from rpQTLs, we showed that while the former is consistently enriched for TF binding sites, the latter is not at all. Therefore, the effect of genetic variation that we observed on RNA splicing is independent from the initiation of the transcription process. As the RBP enrichment analysis showed, rpQTLs are more enriched for post-transcriptional interactions, suggesting that the regulation and maturation of the transcript is modulated by genetic variation. The weak enrichment for RBPs among eQTLs is explained by the fact that multiple RBPs influence transcription, RNA transport and localization. Alteration of the binding sites of those trans-acting factors is likely to affect the overall quantity of mature transcripts.

Stratifying the analysis by categories of splicing events highlights the combinatory process of RNA splicing as downstream events show significant differences with upstream events in the context of genetic association. Indeed, we showed that inter-individual variation in AFEs was proportionally less prone to SNP-related effects than ALEs and TandemUTRs. Interestingly, we observed that AFEs were the most eQTL-driven events. This might be linked to alternative promoter usage, which would explain the strong enrichment of eQTL genes among AFE-rpQTLs.

Overall, our results suggest that genetic variation plays an important role in the regulation of the immune response. Indeed, we also showed that polymorphisms associated in rpQTLs were enriched for loci affecting blood physiology as identified by GWAS. This highlights the functional relevance of genetic control of splicing. We've shown that the effect of rpQTLs and some eQTLs is due to variation in splicing junctions and in RBP binding regions which shows that disruption of splicing may affect overall transcript levels. This study allows a

better characterization of RNA splicing in immunity. Given the growing understanding that RNA splicing is at the center of many diseases, identifying the relationship between genetic variation and RNA processing will allow a more complete comprehension of the aberrant genic mechanisms involved in disease susceptibility.

Chapitre 3: DISCUSSION

3 Discussion

3.1 Synthèse du projet

Dans le cadre de ce projet de maitrise, nous avons ainsi pu mettre en évidence les changements qui se produisent au niveau de l'épissage alternatif lors d'une activation des macrophages. En effet, la principale source de variation des proportions d'inclusion exonique était la stimulation des macrophages. Plus de 10% des évènements testés montraient des changements significatifs. Ces changements se produisent dans des gènes liés à la défense de l'organisme et au réarrangement du cytosquelette impliqué dans les fonctions phagocytaires. Nos résultats démontrent également une tendance au raccourcissement de la région 3'UTR ainsi qu'un usage préférentiel du dernier exon le plus proximal suite à l'infection bactérienne. Ces résultats concordent avec les travaux montrant la tendance des transcrits à limiter le ciblage par les miARN. La première partie du projet nous a permis de confirmer les résultats obtenus par *Pai et al*. En bénéficiant du nombre accru d'échantillons, nous avons ainsi pu utiliser une approche alternative, soit la régression linéaire, pour modéliser la variation des valeurs PSI selon la condition d'infection.

Pour tenter d'expliquer la variance interindividuelle du traitement de l'ARN, nous avons utilisé l'approche QTL pour associer les valeurs PSI avec la variabilité génétique. Nous avons utilisé les puces Illumina Omni5 pour caractériser 6 943 841 SNP (incluant imputation) et avons ainsi pu identifier 1 860 rpQTL (*RNA processing* QTL) unique à travers les trois conditions étudiées. Nous avons également identifié plusieurs QTLs spécifiques à l'infection bactérienne (427 rpQTL). Ces associations impliquent des gènes importants dans le cadre de la réponse

immunitaire. De plus, en considérant l'épissage alternatif comme une mosaïque d'évènements de traitement de l'ARN, nous avons pu démontré que l'usage alternatif d'un premier exon étaient moins sujet au contrôle génétique que les évènements impliquant la région 3' des transcrits dans les macrophages humains.

Après avoir identifié un enrichissement significatif de gènes eQTLs parmi les gènes rpQTLs nous avons modélisé l'association rpQTL en intégrant la variation du SNP eQTL. Nous avons ainsi pu démontrer l'indépendance du contrôle génétique de l'épissage alternatif par rapport au contrôle génétique de l'expression génique. Nous observons cependant que l'usage alternatif du premier exon est généralement plus associé au contrôle de l'expression que la longueur variable de la région 3'UTR. Ceci peut potentiellement être lié à l'usage de régions promotrices alternatives qui auraient comme effet la modulation différentielle des TFs. L'intégration des eQTL dans l'identification des rpQTL permet d'identifier un ensemble de SNP dont l'effet sur l'épissage est indépendant de l'effet de SNP locaux sur l'expression. Les gènes affectés par cet ensemble sont enrichis pour des fonctions cellulaires impliquées dans la réponse immunitaire, comme l'activation des cellules T. Nous pouvons également identifier un ensemble de gènes pour lesquels l'effet des eQTL et des rpQTL est ambiguë. C'est-à-dire qu'il est difficile de déterminer si la variation génétique affecte principalement la transcription, ce qui entraine un effet sur l'épissage ou si la régulation de l'épissage alternatif affecte la production de transcrits mature, affectant ainsi le niveau d'expression. Il est également possible qu'un évènement d'épissage alternatif (ex. rétention d'un intron contenant un codon STOP, usage alternatif d'une région 3' avec site de liaison de miARN) affecte le niveau de transcrits matures dans la cellule.

Nous avons donc caractérisé les polymorphismes impliqués dans ces associations en utilisant des banques de données publiques (i.e. VEP, ENCODE, GWAS-*catalog*). Les SNP eQTL se trouvent généralement près du TSS ou en amont de ce dernier alors que les SNP rpQTL se trouvent plutôt dans le corps du gène où près du TES. En utilisant l'annotation VEP, nous avons pu caractériser les polymorphismes impliqués dans les associations ambiguës. Nos résultats montrent que les rpQTL dont l'effet est significativement associé à l'effet d'un eQTL sont liés à des SNP impliquant un codon stop ou un site de liaison de miARN. Il est intéressant de noter que les eQTL impliquant un gène contenant un rpQTL significativement affecté par l'effet du eQTL sont enrichis pour des variants situés dans les jonctions d'épissage, suggérant que les mécanismes génétiques responsables de la variation de l'expression sont plutôt liés à la régulation post-transcriptionnelle de l'ARN qu'à la transcription du gène en soi. De plus, la banque de données ENCODE a permis d'investiguer le rôle des RBPs dans le contrôle génétique de l'épissage alternatif. Nous remarquons que les eQTLs sont faiblement enrichis pour les sites de liaisons des RBPs. Plusieurs de ces facteurs sont impliqués dans le transport des transcrits où même dans l'initiation de la transcription, ce qui peut affecter le niveau de transcrits dans la cellule. Bien que les eQTLs soient significativement enrichis pour les sites de liaison des TF, ce n'est pas du tout le cas des rpQTLs. Nos résultats indiquent donc que les rpQTLs sont dus à l'altération des processus de maturation des transcrits d'ARN immatures. Les eQTLs sont généralement dus à l'altération de la transcription en soi mais également de ce processus de maturation via la liaison de RBPs ou l'incorporation de caractéristiques pouvant affecter le nombre total de transcrits aboutissant à une forme fonctionnelle.

Finalement, nous avons tenté de lier les rpQTL identifiés avec des SNP relevés par des études GWAS. En utilisant la base de données du NIH, nous avons pu identifier un enrichissement significatif de SNP-GWAS parmi notre ensemble de rpQTL. Cet enrichissement implique l'ensemble des variants associés à des traits GWAS. Nous avons ensuite différencié les différents traits GWAS. Les variants rpQTL sont enrichis pour des traits liés à la composition cellulaire du sang. Nos résultats suggèrent que le contrôle génétique de l'épissage alternative pourrait jouer un rôle important dans la variabilité interindividuelle lié à la présence de cellules immunitaires dans le sang. Ce phénotype pourrait expliquer une partie de la variation de la susceptibilité aux infections bactériennes.

3.2 Impacts et perspectives

De plus en plus d'évidences placent la régulation des gènes au cœur de la compréhension des maladies. L'épissage alternatif est un aspect majeur de la régulation des gènes eucaryotes et son implication dans la défense de l'organisme contre les pathogènes est cruciale. Les avancées technologiques et le développement d'outils bio-informatiques permettent aujourd'hui de caractériser ce phénotype avec précision. Les nombreuses études GWAS ont souligné l'importance de la variation génétique dans les maladies et notre approche permet de cataloguer son influence sur la régulation des gènes dans le contexte de l'activation de la réponse immunitaire innée. L'analyse QTL de phénotypes impliqués dans la régulation des gènes est une approche puissante pour identifier les sources potentielles de variabilité impliquées dans la susceptibilité aux maladies. L'identification de rpQTLs propres aux conditions d'infection des macrophages souligne l'importance du contrôle génétique du

traitement de l'ARN dans le contexte de la réponse immunitaire. De plus, comme nous l'avons démontré, ce contrôle est majoritairement distinct de celui exercé sur l'expression des gènes (eQTLs). De plus, lorsque l'effet des rpQTL n'est pas distinct de l'effet des eQTLs, nous avons montré que les polymorphismes impliqués semblaient plutôt affecter la régulation en cis de l'épissage alternatif ou, plus généralement, la maturation des transcrits.

L'enrichissement de gènes rpQTL indépendants des eQTLs pour les fonctions de présentation d'antigènes et de communication avec les cellules T suggèrent l'importance des rpQTLs au niveau de la communication entre la réponse innée et la réponse adaptative. Les travaux présentés dans ce mémoire soulignent l'implication de la variabilité génétique dans la régulation de la maturation des transcrits d'ARN. Bien que plusieurs études aient déjà relevé cette relation, ce projet de mémoire est la première initiative d'identification de rpQTL dans des macrophages humains soumis à un stress bactérien, mettant ainsi en évidence le rôle de la variabilité génétique dans le traitement post-transcriptionnel de l'ARN lors de la réponse immunitaire. Nous avons choisi une modélisation linéaire pour associer le génotype à la variation d'épissage des exons pour faciliter la comparaison avec les résultats publiés par *Nédélec et al.* Bien que plusieurs autres études aient utilisé cette approche, de récentes recherches proposent des méthodes alternatives qui prennent en compte la nature bornée des valeurs PSI (valeurs entre 0 et 1) (96). Notre analyse nous a cependant permis d'identifier des associations impliquant des gènes montrant des variants d'épissage connus (97-99) (ex. IRF5, OAS1, LST1, CD44 : **Figure Supplémentaire 6**, page LXXVI).

Une autre limitation potentielle du projet se manifeste dans la méthode de séquençage du transcriptome. En effet, la fragmentation en court *reads* peut amener un biais d'alignement

où un fragment est assigné à plus d'une région du génome. Ces variants sont alors rejetés de l'analyse. De nouvelles techniques permettent l'utilisation de *reads* beaucoup plus long qui permettent un alignement présentant une précision accrue. Moins de *reads* sont rejetés car il y a moins d'appariements multiples (100). Cette technique permettrait donc d'avoir une estimation plus rigoureuse des valeurs PSI.

Finalement, considérant l'importance de la régulation en trans de l'épissage alternatif par les différentes composantes du spliceosome et la diversité des facteurs d'épissage, un plus grand échantillonnage permettrait d'identifier des *trans*-QTL d'épissage. En effet, l'identification de *trans*-QTL demeure un défi de par le grand nombre de tests requis. De plus, l'effet indirect des variants a généralement un effet moins important sur le phénotype qu'un variant en cis (101). Peu d'études se sont penchées sur la caractérisation des trans-QTL d'épissage vu les défis statistiques que cette analyse soulève. Les facteurs d'épissages ont cependant un rôle primordial dans la maturation des ARN messagers. Des polymorphismes affectant la production de ces facteurs auraient un effet potentiellement important sur l'activité de ces derniers. Ceci entrainerait un effet généralisé sur l'épissage d'un ensemble de gènes comportant des sites de liaison pour ces facteurs.

Ce projet a permis la caractérisation de l'effet en *cis* des SNP sur l'épissage alternatif se produisant dans les macrophages humains. Nos résultats présentent un portrait de ce phénotype dans le cadre de l'infection bactérienne *in vitro* et permettent de comprendre quels mécanismes sont les plus sujets à la variabilité génétique. Les analyses présentées dans ce mémoire mettent en évidence le pouvoir d'une approche quantitative des sciences de la vie pour comprendre les mécanismes sous-jacents aux maladies et à l'habileté de combattre les

infections bactériennes. Notre époque est témoin de changements magistraux au niveau de la compréhension de vivant et la génétique est déjà au cœur du traitement d'un grand nombre de désordres de santé. L'épissage alternatif est un aspect régulateur des gènes central au développement de la thérapie génique (102, 103). De manière générale, nous proposons un catalogue d'évènements d'épissage alternatif dont la variabilité interindividuelle est liée à la variabilité génétique, ce qui permet d'aiguiser notre compréhension de la dynamique de ce phénotype.

References

1. Cooper MD, Alder MN. The evolution of adaptive immune systems. Cell. 2006;124(4):815-22.
2. Danilova N. The evolution of immune mechanisms. J Exp Zool B Mol Dev Evol. 2006;306(6):496-520.
3. Janeway CA, Jr., Medzhitov R. Innate immune recognition. Annu Rev Immunol. 2002;20:197-216.
4. Barreiro LB, Ben-Ali M, Quach H, Laval G, Patin E, Pickrell JK, et al. Evolutionary dynamics of human Toll-like receptors and their different contributions to host defense. PLoS Genet. 2009;5(7):e1000562.
5. Mogensen TH. Pathogen recognition and inflammatory signaling in innate immune defenses. Clin Microbiol Rev. 2009;22(2):240-73, Table of Contents.
6. Mantegazza AR, Magalhaes JG, Amigorena S, Marks MS. Presentation of phagocytosed antigens by MHC class I and II. Traffic. 2013;14(2):135-52.
7. Gupta SK, Bajwa P, Deb R, Chellappa MM, Dey S. Flagellin a toll-like receptor 5 agonist as an adjuvant in chicken vaccines. Clin Vaccine Immunol. 2014;21(3):261-70.
8. Barreiro LB, Patin E, Neyrolles O, Cann HM, Gicquel B, Quintana-Murci L. The heritage of pathogen pressures and ancient demography in the human innate-immunity CD209/CD209L region. Am J Hum Genet. 2005;77(5):869-86.
9. Karlsson EK, Kwiatkowski DP, Sabeti PC. Natural selection and infectious disease in human populations. Nature Reviews Genetics. 2014;15(6):379-93.
10. Cagliani R, Sironi M. Pathogen-driven selection in the human genome. Int J Evol Biol. 2013;2013:204240.
11. de la Casa-Esperon E. Horizontal transfer and the evolution of host-pathogen interactions. Int J Evol Biol. 2012;2012:679045.
12. Brunham RC, Plummer FA, Stephens RS. Bacterial antigenic variation, host immune response, and pathogen-host coevolution. American Society for Microbiology. 1993;61(6):2273-6.
13. Nedelec Y, Sanz J, Baharian G, Szpiech ZA, Pacis A, Dumaine A, et al. Genetic Ancestry and Natural Selection Drive Population Differences in Immune Responses to Pathogens. Cell. 2016;167(3):657-69 e21.
14. Quach H, Rotival M, Pothlichet J, Loh YE, Dannemann M, Zidane N, et al. Genetic Adaptation and Neandertal Admixture Shaped the Immune System of Human Populations. Cell. 2016;167(3):643-56 e17.
15. Lopez S, van Dorp L, Hellenthal G. Human Dispersal Out of Africa: A Lasting Debate. Evol Bioinform Online. 2015;11(Suppl 2):57-68.
16. Wang ET, Sandberg R, Luo S, Khrebtukova I, Zhang L, Mayr C, et al. Alternative isoform regulation in human tissue transcriptomes. Nature. 2008;456(7221):470-6.
17. Belicha-Villanueva A, Blickwedehl J, McEvoy S, Golding M, Gollnick SO, Bangia N. What is the role of alternate splicing in antigen presentation by major histocompatibility complex class I molecules? Immunol Res. 2010;46(1-3):32-44.

18. Ishitani A, Geraghty DE. Alternative splicing of HLA-G transcripts yields proteins with primary structures resembling both class I and class II antigens. Proc Natl Acad Sci U S A. 1992;89:3947-51.
19. Lynch KW. Consequences of regulated pre-mRNA splicing in the immune system. Nat Rev Immunol. 2004;4(12):931-40.
20. Will CL, Luhrmann R. Spliceosome structure and function. Cold Spring Harb Perspect Biol. 2011;3(7).
21. Black DL. Mechanisms of alternative pre-messenger RNA splicing. Annu Rev Biochem. 2003;72:291-336.
22. Nguyen TH, Galej WP, Fica SM, Lin PC, Newman AJ, Nagai K. CryoEM structures of two spliceosomal complexes: starter and dessert at the spliceosome feast. Curr Opin Struct Biol. 2016;36:48-57.
23. Wang Y, Liu J, Huang BO, Xu YM, Li J, Huang LF, et al. Mechanism of alternative splicing and its regulation. Biomed Rep. 2015;3(2):152-8.
24. Park E, Pan Z, Zhang Z, Lin L, Xing Y. The Expanding Landscape of Alternative Splicing Variation in Human Populations. Am J Hum Genet. 2018;102(1):11-26.
25. Pai AA, Baharian G, Page Sabourin A, Brinkworth JF, Nedelec Y, Foley JW, et al. Widespread Shortening of 3' Untranslated Regions and Increased Exon Inclusion Are Evolutionarily Conserved Features of Innate Immune Responses to Infection. PLoS Genet. 2016;12(9):e1006338.
26. Lianoglou S, Garg V, Yang JL, Leslie CS, Mayr C. Ubiquitously transcribed genes use alternative polyadenylation to achieve tissue-specific expression. Genes Dev. 2013;27(21):2380-96.
27. Landry JR, Mager DL, Wilhelm BT. Complex controls: the role of alternative promoters in mammalian genomes. Trends Genet. 2003;19(11):640-8.
28. Ge Y, Porse BT. The functional consequences of intron retention: alternative splicing coupled to NMD as a regulator of gene expression. Bioessays. 2014;36(3):236-43.
29. Scotti MM, Swanson MS. RNA mis-splicing in disease. Nat Rev Genet. 2016;17(1):19-32.
30. Lee C, Roy M. Analysis of alternative splicing with microarrays: successes and challenges. Genome Biology. 2004;5(231).
31. Wang Z, Gerstein M, Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics. 2009;10:57-63.
32. Chen L. Statistical and Computational Methods for High-Throughput Sequencing Data Analysis of Alternative Splicing. Stat Biosci. 2013;5(1):138-55.
33. Kakaradov B, Xiong HY, Lee LJ, Jojic N, Frey BJ. Challenges in estimating percent inclusion of alternatively spliced junctions from RNA-seq data. BMC Bioinformatics. 2012;13.
34. Wu J, Akerman M, Sun S, McCombie WR, Krainer AR, Zhang MQ. SpliceTrap: a method to quantify alternative splicing under single cellular conditions. Bioinformatics. 2011;27(21):3010-6.
35. Shen S, Park JW, Lu ZX, Lin L, Henry MD, Wu YN, et al. rMATS: robust and flexible detection of differential alternative splicing from replicate RNA-Seq data. Proc Natl Acad Sci U S A. 2014;111(51):E5593-601.

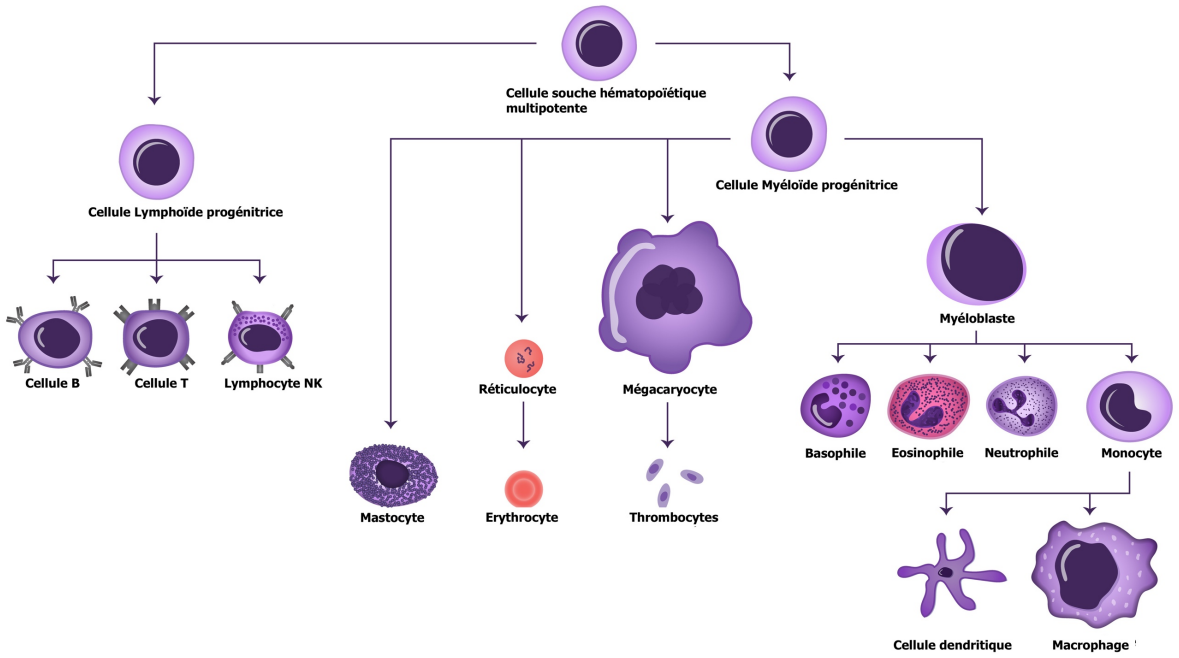
36. Trincado JL, Entizne JC, Hysenaj G, Singh B, Skalic M, Elliott DJ, et al. SUPPA2: fast, accurate, and uncertainty-aware differential splicing analysis across multiple conditions. Genome Biol. 2018;19(1):40.
37. Alamancos GP, Pages A, Trincado JL, Bellora N, Eyras E. Leveraging transcript quantification for fast computation of alternative splicing profiles. RNA. 2015;21(9):1521-31.
38. Katz Y, Wang ET, Airoldi EM, Burge CB. Analysis and design of RNA sequencing experiments for identifying isoform regulation. Nature Methods. 2010;7:1009-15.
39. Wang Z, Burge CB. Splicing regulation: from a parts list of regulatory elements to an integrated splicing code. RNA. 2008;14(5):802-13.
40. Fu XD, Ares M, Jr. Context-dependent control of alternative splicing by RNA-binding proteins. Nat Rev Genet. 2014;15(10):689-701.
41. Lambert N, Robertson A, Jangi M, McGeary S, Sharp PA, Burge CB. RNA Bind-n-Seq: quantitative assessment of the sequence and structural binding specificity of RNA binding proteins. Mol Cell. 2014;54(5):887-900.
42. Ter Horst R, Jaeger M, Smeekens SP, Oosting M, Swertz MA, Li Y, et al. Host and Environmental Factors Influencing Individual Human Cytokine Responses. Cell. 2016;167(4):1111-24 e13.
43. Chapman SJ, Hill AV. Human genetic susceptibility to infectious disease. Nat Rev Genet. 2012;13(3):175-88.
44. Astle WJ, Elding H, Jiang T, Allen D, Ruklisa D, Mann AL, et al. The Allelic Landscape of Human Blood Cell Trait Variation and Links to Common Complex Disease. Cell. 2016;167(5):1415-29 e19.
45. Jain D, Hodonsky CJ, Schick UM, Morrison JV, Minnerath S, Brown L, et al. Genome-wide association of white blood cell counts in Hispanic/Latino Americans: the Hispanic Community Health Study/Study of Latinos. Hum Mol Genet. 2017;26(6):1193-204.
46. Nalls MA, Couper DJ, Tanaka T, van Rooij FJ, Chen MH, Smith AV, et al. Multiple loci are associated with white blood cell phenotypes. PLoS Genet. 2011;7(6):e1002113.
47. Crosslin DR, McDavid A, Weston N, Nelson SC, Zheng X, Hart E, et al. Genetic variants associated with the white blood cell count in 13,923 subjects in the eMERGE Network. Hum Genet. 2012;131(4):639-52.
48. Lamy P, Grove J, Wiuf C. A review of software for microarray genotyping. Human Genomics 2011;5(4):304-9.
49. MacArthur J, Bowler E, Cerezo M, Gil L, Hall P, Hastings E, et al. The new NHGRI-EBI Catalog of published genome-wide association studies (GWAS Catalog). Nucleic Acids Res. 2017;45(D1):D896-D901.
50. Hrdlickova B, de Almeida RC, Borek Z, Withoff S. Genetic variation in the non-coding genome: Involvement of micro-RNAs and long non-coding RNAs in disease. Biochim Biophys Acta. 2014;1842(10):1910-22.
51. Ricano-Ponce I, Wijmenga C. Mapping of immune-mediated disease genes. Annu Rev Genomics Hum Genet. 2013;14:325-53.
52. van der Sijde MR, Ng A, Fu J. Systems genetics: From GWAS to disease pathways. Biochim Biophys Acta. 2014;1842(10):1903-9.

53. Knight JC. Genomic modulators of the immune response. Trends Genet. 2013;29(2):74-83.
54. Legarra A, Fernando RL. Linear models for joint association and linkage QTL mapping. Genet Sel Evol. 2009;41:43.
55. Knott SA. Regression-based quantitative trait loci mapping: robust, efficient and effective. Philos Trans R Soc Lond B Biol Sci. 2005;360(1459):1435-42.
56. Rakitsch B, Stegle O. Modelling local gene networks increases power to detect trans-acting genetic effects on gene expression. Genome Biol. 2016;17:33.
57. Lu ZX, Jiang P, Xing Y. Genetic variation of pre-mRNA alternative splicing in human populations. Wiley Interdiscip Rev RNA. 2012;3(4):581-92.
58. Doerks T, Copley RR, Schultz J, Ponting CP, Bork P. Systematic identification of novel protein domain families associated with nuclear functions. Genome Res. 2002;12(1):47-56.
59. Tazi J, Bakkour N, Stamm S. Alternative splicing and disease. Biochim Biophys Acta. 2009;1792(1):14-26.
60. Newman JRB, Conesa A, Mika M, New FN, Onengut-Gumuscu S, Atkinson MA, et al. Disease-specific biases in alternative splicing and tissue-specific dysregulation revealed by multitissue profiling of lymphocyte gene expression in type 1 diabetes. Genome Res. 2017;27(11):1807-15.
61. Robert C, Watson M. The incredible complexity of RNA splicing. Genome Biol. 2016;17(1):265.
62. Evsyukova I, Somarelli JA, Gregory SG, Garcia-Blanco MA. Alternative splicing in multiple sclerosis and other autoimmune diseases. RNA Biology. 2014;7(4):462-73.
63. Faustino NA, Cooper TA. Pre-mRNA splicing and human disease Genes Dev. 2003;17:419-37.
64. Meregalli M, Maciotta S, Angeloni V, Torrente Y. Duchenne muscular dystrophy caused by a frame-shift mutation in the acceptor splice site of intron 26. BMC Med Genet. 2016;17(1):55.
65. Fu RH, Liu SP, Huang SJ, Chen HJ, Chen PR, Lin YH, et al. Aberrant alternative splicing events in Parkinson's disease. Cell Transplant. 2013;22(4):653-61.
66. Yabas M, Elliott H, Hoyne GF. The Role of Alternative Splicing in the Control of Immune Homeostasis and Cellular Differentiation. Int J Mol Sci. 2015;17(1).
67. Takata A, Matsumoto N, Kato T. Genome-wide identification of splicing QTLs in the human brain and their enrichment among schizophrenia-associated loci. Nat Commun. 2017;8:14519.
68. Ma L, Jia P, Zhao Z. Splicing QTL of human adipose-related traits. Sci Rep. 2018;8(1):318.
69. Li YI, van de Geijn B, Raj A, Knowles DA, Petti AA, Golan D, et al. RNA splicing is a primary link between genetic variation and disease. Science. 2016;352(6285):600-4.
70. Chen L, Ge B, Casale FP, Vasquez L, Kwan T, Garrido-Martin D, et al. Genetic Drivers of Epigenetic and Transcriptional Variation in Human Immune Cells. Cell. 2016;167(5):1398-414 e24.
71. Hassan MA, Butty V, Jensen KD, Saeij JP. The genetic basis for individual differences in mRNA splicing and APOBEC1 editing activity in murine macrophages. Genome Res. 2014;24(3):377-89.

72. Kumar H, Kawai T, Akira S. Pathogen recognition by the innate immune system. Int Rev Immunol. 2011;30(1):16-34.
73. Aderem A, Ulevitch RJ. Toll-like receptors in the induction of the innate immune response. Nature. 2000;406.
74. Carpenter S, Ricci EP, Mercier BC, Moore MJ, Fitzgerald KA. Post-transcriptional regulation of gene expression in innate immunity. Nat Rev Immunol. 2014;14(6):361-76.
75. Schaub A, Glasmacher E. Splicing in immune cells-mechanistic insights and emerging topics. Int Immunol. 2017;29(4):173-81.
76. Sanders MS, van Well GT, Ouburg S, Morre SA, van Furth AM. Genetic variation of innate immune response genes in invasive pneumococcal and meningococcal disease applied to the pathogenesis of meningitis. Genes Immun. 2011;12(5):321-34.
77. Liston A, Carr EJ, Linterman MA. Shaping Variation in the Human Immune System. Trends Immunol. 2016;37(10):637-46.
78. Medina-Acosta E, Nakaya HT, Pontillo A, de Souza Campos Fernandes RC. Genetic control of immune response and susceptibility to infectious diseases. Biomed Res Int. 2014;2014:796073.
79. Martinez NM, Lynch KW. Control of alternative splicing in immune responses: many regulators, many predictions, much still to learn. Immunol Rev. 2013;253(1):216-36.
80. Sandberg R, Neilson JR, Sarma A, Sharp PA, Burge CB. Proliferating cells express mRNAs with shortened 3' untranslated regions and fewer microRNA target sites. Science. 2008;320(5883):1643-7.
81. Ongen H, Dermitzakis ET. Alternative Splicing QTLs in European and African Populations. Am J Hum Genet. 2015;97(4):567-75.
82. Li YI, Knowles DA, Humphrey J, Barbeira AN, Dickinson SP, Im HK, et al. Annotation-free quantification of RNA splicing using LeafCutter. Nat Genet. 2018;50(1):151-8.
83. Westra HJ, Franke L. From genome to function by studying eQTLs. Biochim Biophys Acta. 2014;1842(10):1896-902.
84. Lappalainen T, Sammeth M, Friedlander MR, t Hoen PA, Monlong J, Rivas MA, et al. Transcriptome and genome sequencing uncovers functional variation in humans. Nature. 2013;501(7468):506-11.
85. Team Rc. R: A language and environment for statistical computing. R Foundation for Statistical Computing. 3.3.3 ed2013.
86. Storey JD, Tibshirani R. Statistical significance for genomwide studies. Proc Natl Acad Sci U S A. 2003;100(16):9440-5.
87. Stegle O, Parts L, Piipari M, Winn J, Durbin R. Using probabilistic estimation of expression residuals (PEER) to obtain increased power and interpretability of gene expression analyses. Nat Protoc. 2012;7(3):500-7.
88. Shabalin AA. Matrix eQTL: ultra fast eQTL analysis via large matrix operations. Bioinformatics. 2012;28(10):1353-8.
89. Eden E, Navon R, Steinfeld I, Lipson D, Yakhini Z. GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists. BMC Bioinformatics. 2009;10:48.
90. Chang CC, Chow CC, Tellier LC, Vattikuti S, Purcell SM, Lee JJ. Second-generation PLINK: rising to the challenge of larger and richer datasets. Gigascience. 2015;4:7.

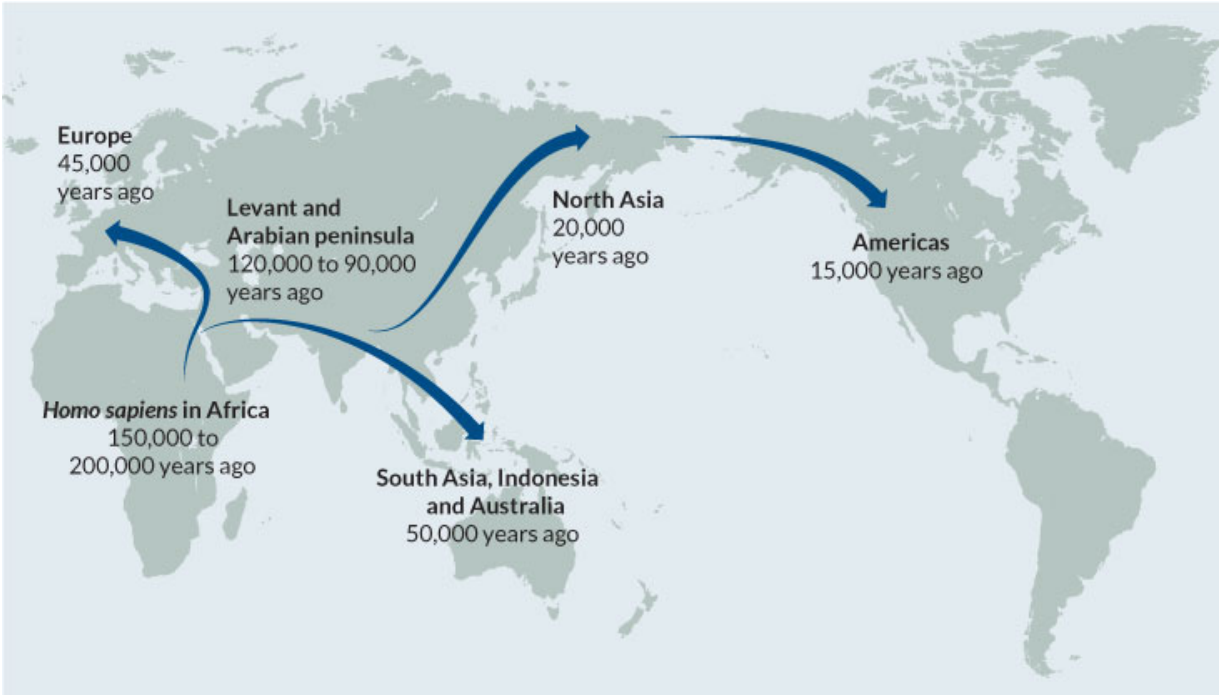
91. Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010;26(6):841-2.
92. Zhang X, Du R, Li S, Zhang F, Jin L, Wang H. Evaluation of copy number variation detection for a SNP array platform. BMC Bioinformatics. 2014;15:50.
93. McLaren W, Gil L, Hunt SE, Riat HS, Ritchie GR, Thormann A, et al. The Ensembl Variant Effect Predictor. Genome Biol. 2016;17(1):122.
94. Buenrostro JD, Giresi PG, Zaba LC, Chang HY, Greenleaf WJ. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. Nat Methods. 2013;10(12):1213-8.
95. Van Nostrand EL, Pratt GA, Shishkin AA, Gelboin-Burkhart C, Fang MY, Sundararaman B, et al. Robust transcriptome-wide discovery of RNA-binding protein binding sites with enhanced CLIP (eCLIP). Nat Methods. 2016;13(6):508-14.
96. Jia C, Hu Y, Liu Y, Li M. Mapping Splicing Quantitative Trait Loci in RNA-Seq. Cancer Inform. 2015;14(Suppl 1):45-53.
97. Garcia-Bermudez M, Lopez-Mejias R, Genre F, Castaneda S, Llorca J, Gonzalez-Gay MA. Interferon regulatory factor 5 genetic variants are
associated with cardiovascular disease in patients
with rheumatoid arthritis. Arthritis research and therapy 2014;16(146).
98. Deng FY, Lei SF, Zhang YH, Zhang ZL, Guo YF. Functional relevance for associations between genetic variants and systemic lupus erythematosus. PLoS One. 2013;8(1):e53037.
99. Lalonde E, Ha KC, Wang Z, Bemmo A, Kleinman CL, Kwan T, et al. RNA sequencing reveals the role of splicing polymorphisms in regulating human gene expression. Genome Res. 2011;21(4):545-54.
100. Krizanovic K, Echchiki A, Roux J, Sikic M. Evaluation of tools for long read RNA-seq splice-aware alignment. Bioinformatics. 2018;34(5):748-54.
101. Petretto E, Mangion J, Dickens NJ, Cook SA, Kumaran MK, Lu H, et al. Heritability and tissue specificity of expression quantitative trait loci. PLoS Genet. 2006;2(10):e172.
102. Berger A, Maire S, Gaillard MC, Sahel JA, Hantraye P, Bemelmans AP. mRNA trans-splicing in gene therapy for genetic diseases. Wiley Interdiscip Rev RNA. 2016;7(4):487-98.
103. Garcia-Blanco MA, Baraniak AP, Lasda EL. Alternative splicing in disease and therapy. Nat Biotechnol. 2004;22(5).
104. Demenocal PB, Stringer C. Climate and the peopling of the world J Hum Evol. 2016;538:49-50.

ANNEXE: Figures Supplémentaires



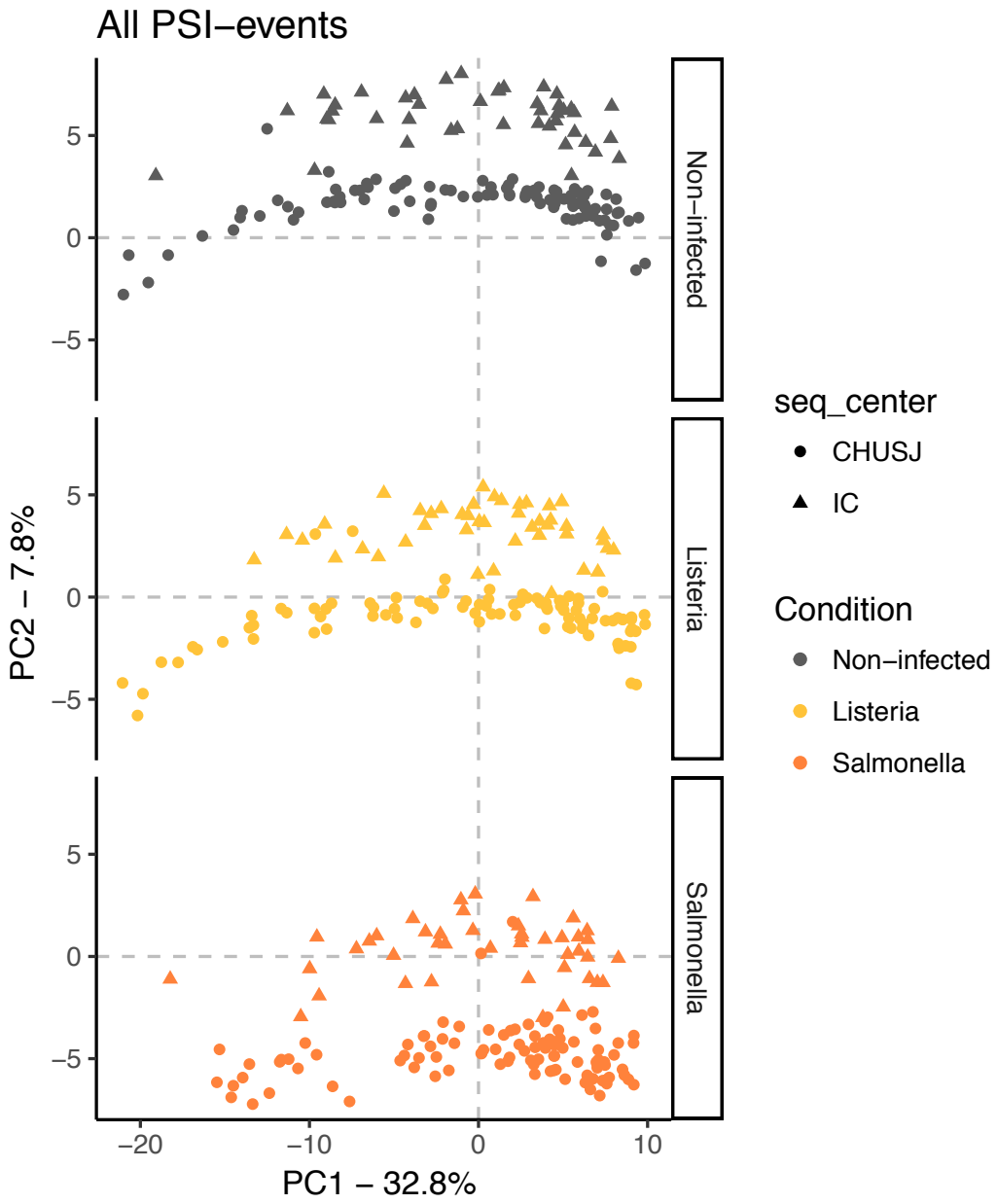
Supplemental Figure 1 Lignées cellulaires hématopoïétiques

Schéma présentant les différents types cellulaires découlant de la lignée hématopoïétique. Les macrophages font partie de la lignée myéloïde et sont dérivés des monocytes. Ces différents types cellulaires communiquent entre eux pour déclencher, maintenir et réguler la réponse immunitaire à des pathogènes.



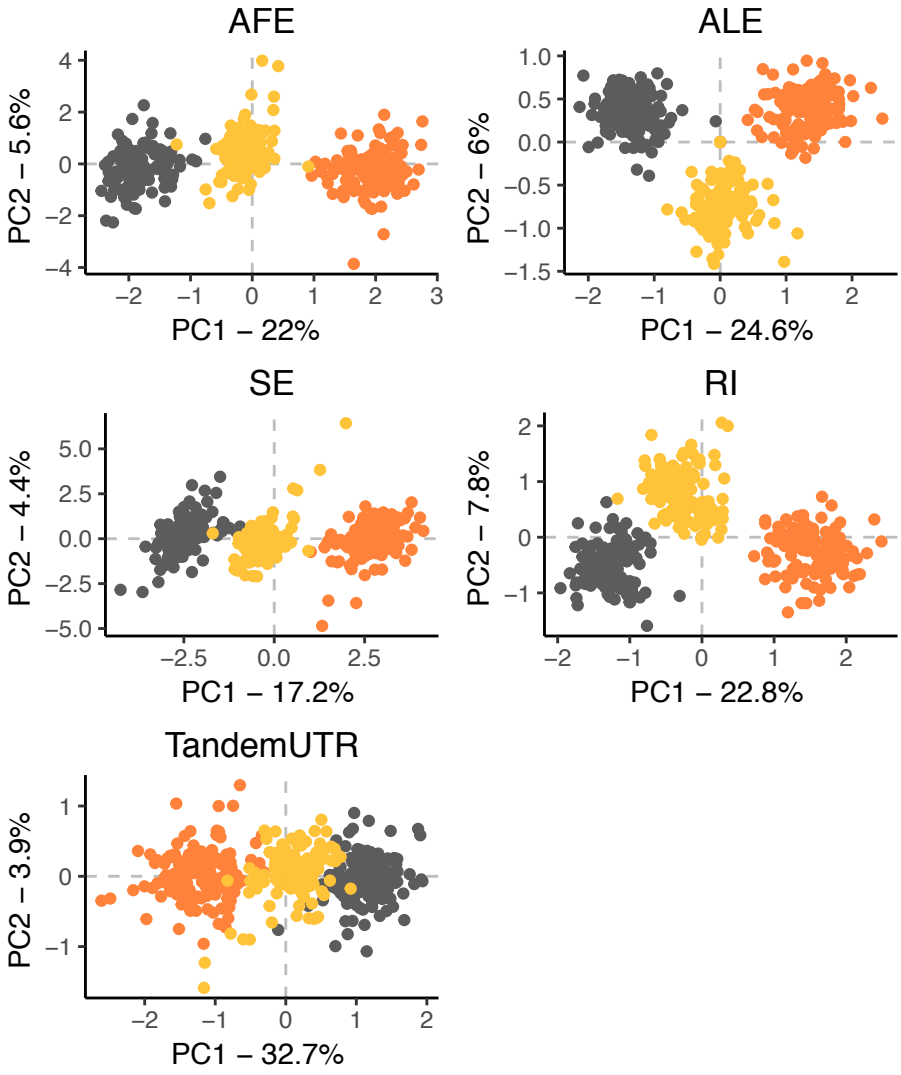
Supplemental Figure 2 Migrations des premières populations humaines

Carte géographique illustrant les migrations hypothétiques de l'homme lors de la dispersion d'Homo Sapiens sur le globe. Ces migrations ont provoqué des changements drastiques dans l'environnement pathogénique des populations, ce qui a, entre autres, façonné les différences génétiques responsables de la susceptibilité variable aux infections (104).



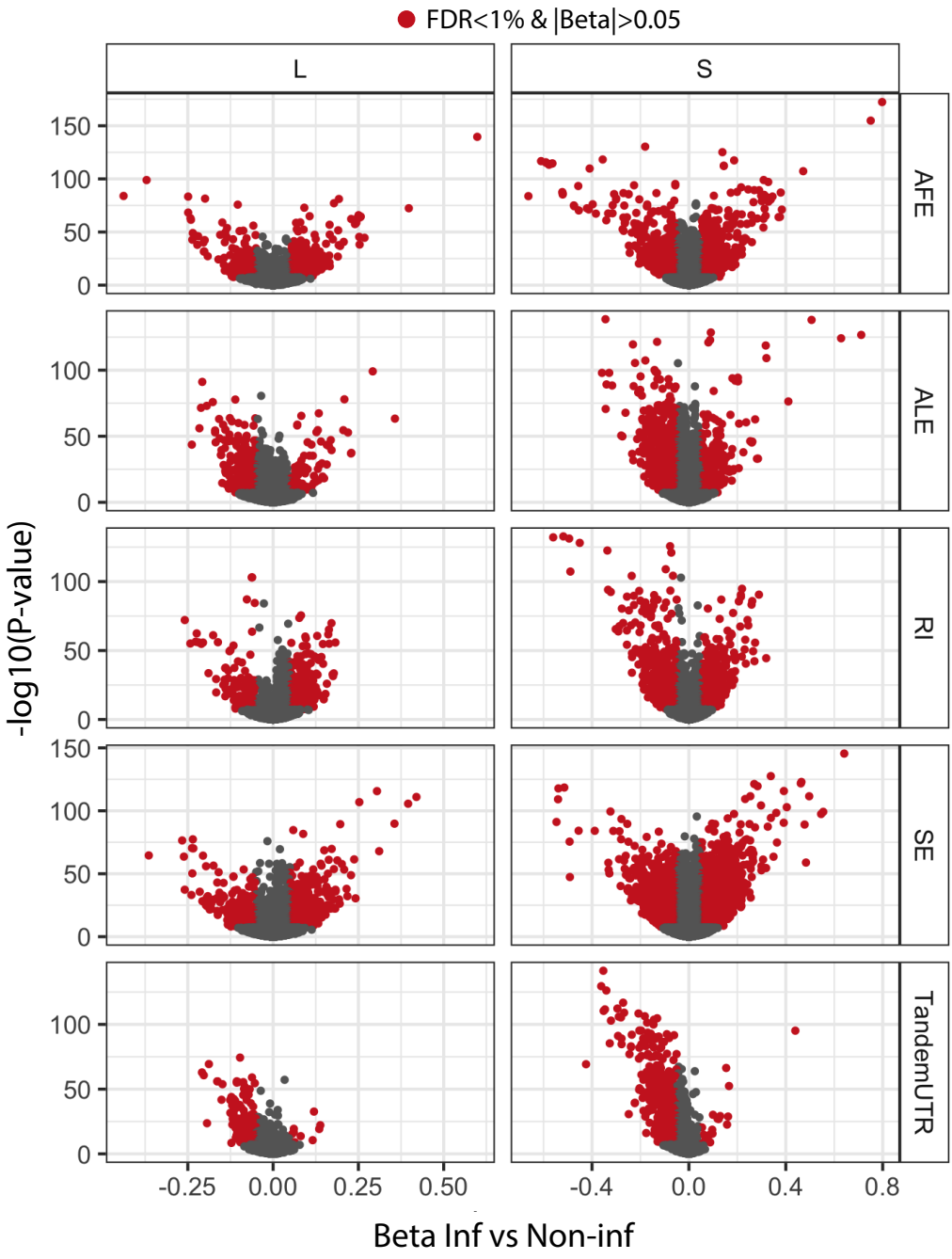
Supplemental Figure 3 PCA des valeurs PSI sans correction

Analyse en composantes principales des valeurs PSI en ne corrigeant pour aucune covariables. Nous avons identifié un effet évident du centre dans lequel les échantillons ont été séquencés. Les conditions ont été séparées pour mettre en évidence l'effet du biais technique.



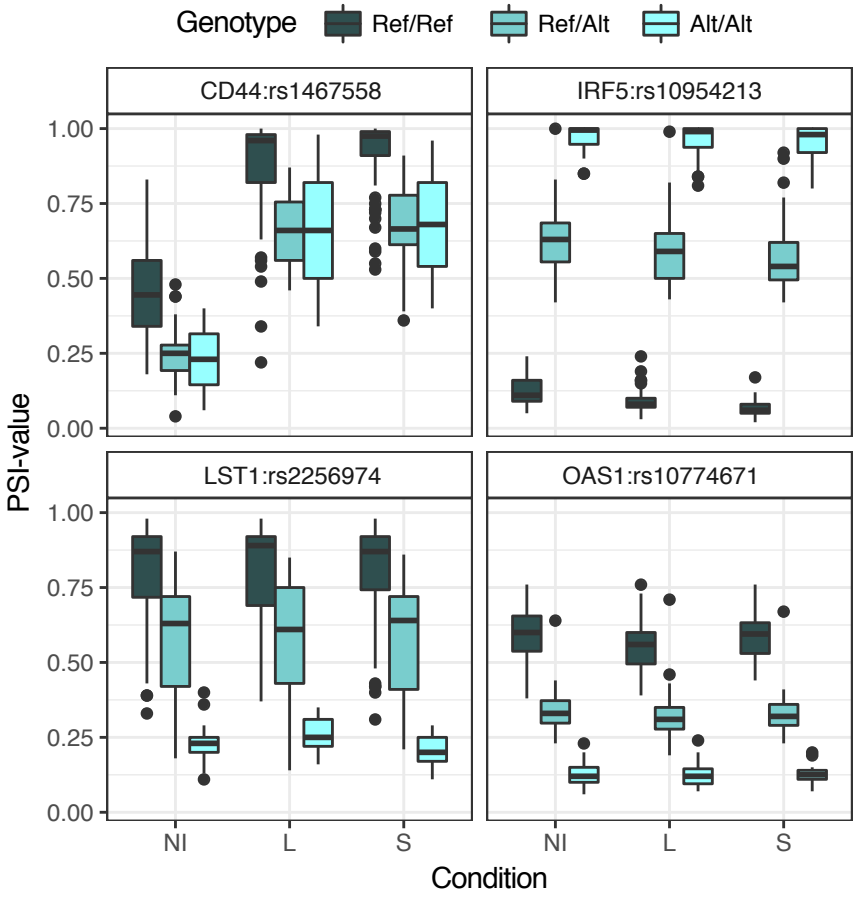
Supplemental Figure 4 PCA des différents types d'évènements après correction

Analyse en composantes principales des valeur PSI en considérant les évènements séparément. La variation liée au centre de séquençage a été retirée des valeurs. L'axe des X représente la première composante principale et l'axe des Y représente la deuxième composante principale. La contribution de chaque axe à la variation des données est indiquée dans l'étiquette de chaque axe.



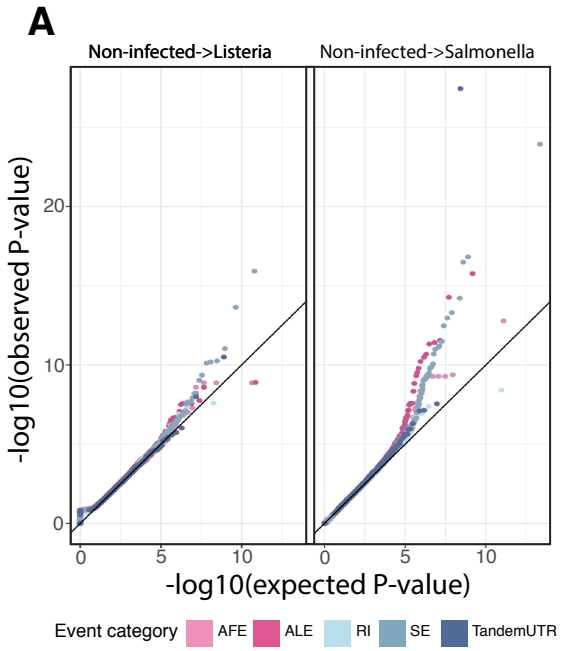
Supplemental Figure 5 Évènements d'épissage différentiel

Graphique en volcan montrant la relation entre le coefficient lié à la condition et la valeur P. Les évènements dont le changement d'épissage suite à l'infection est significatif sont indiqués en rouge. La direction observée dans la section des résultats est également observable dans cette figure où pratiquement tous les TandemUTRs significatifs ont un coefficient négatif.



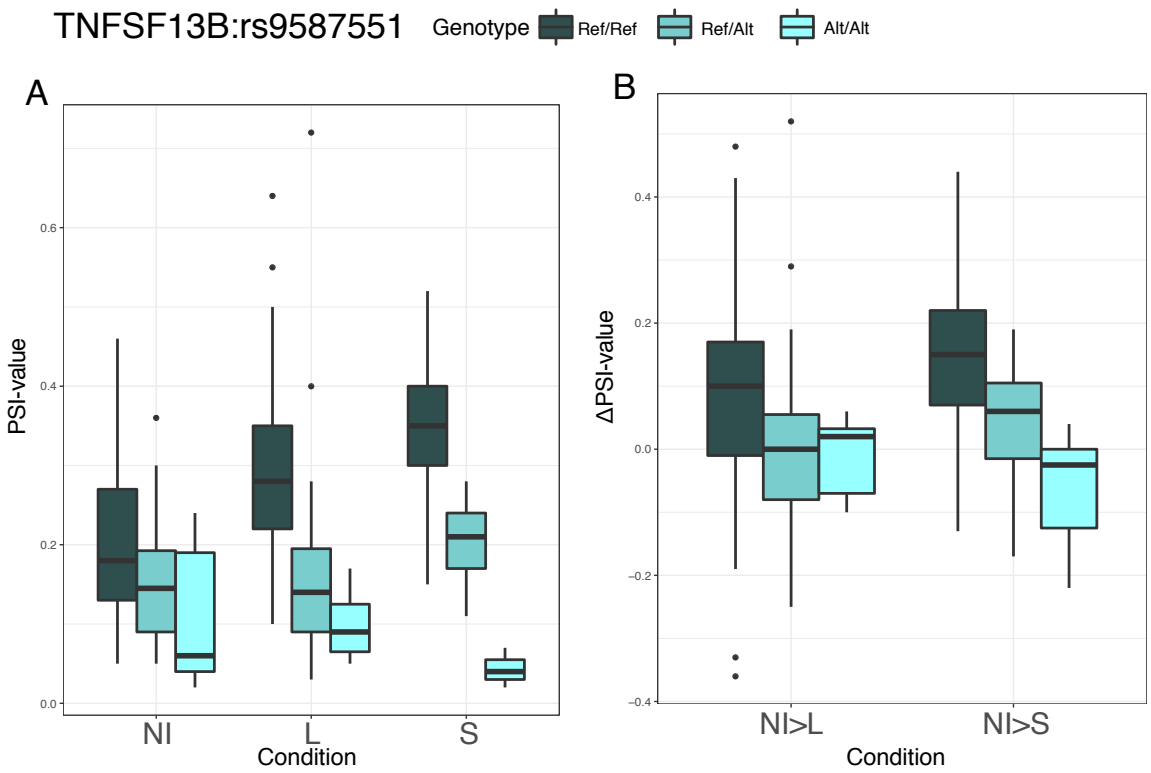
Supplemental Figure 6 Exemples de rpQTL

4 exemples de rpQTL. L'axe des X présente les différentes conditions testées et les groupes de génotypes sont séparés par les couleurs où l'homozygote pour l'allèle de référence est le plus foncé et l'homozygote pour l'allèle alternative est pâle. L'axe des Y représente les valeurs PSI.



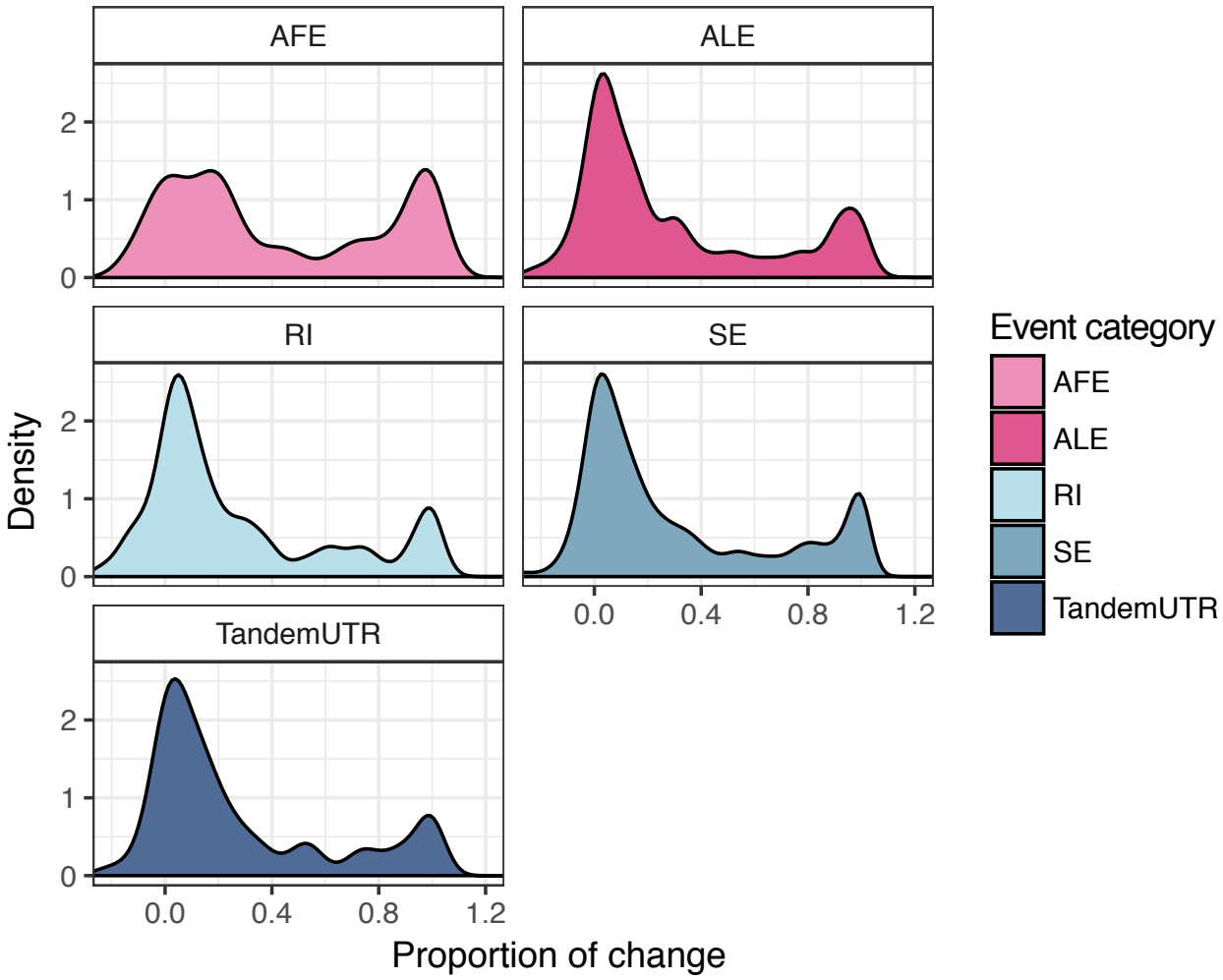
Supplemental Figure 7 ΔPSI QTLs

A) QQ plot des associations en considérant la différence de valeurs PSI comme le phénotype d'intérêt. Les couleurs représentent les 5 différentes classes d'évènements. Les valeurs P attendues sont obtenues selon 10 analyses parallèle en considérant le génotype où les individus sont permutés (voir section Méthodes). **B)** Proportion d'évènements ayant un ΔPSI QTL significatif dans la condition NI>L ou NI>S (FDR<5%).



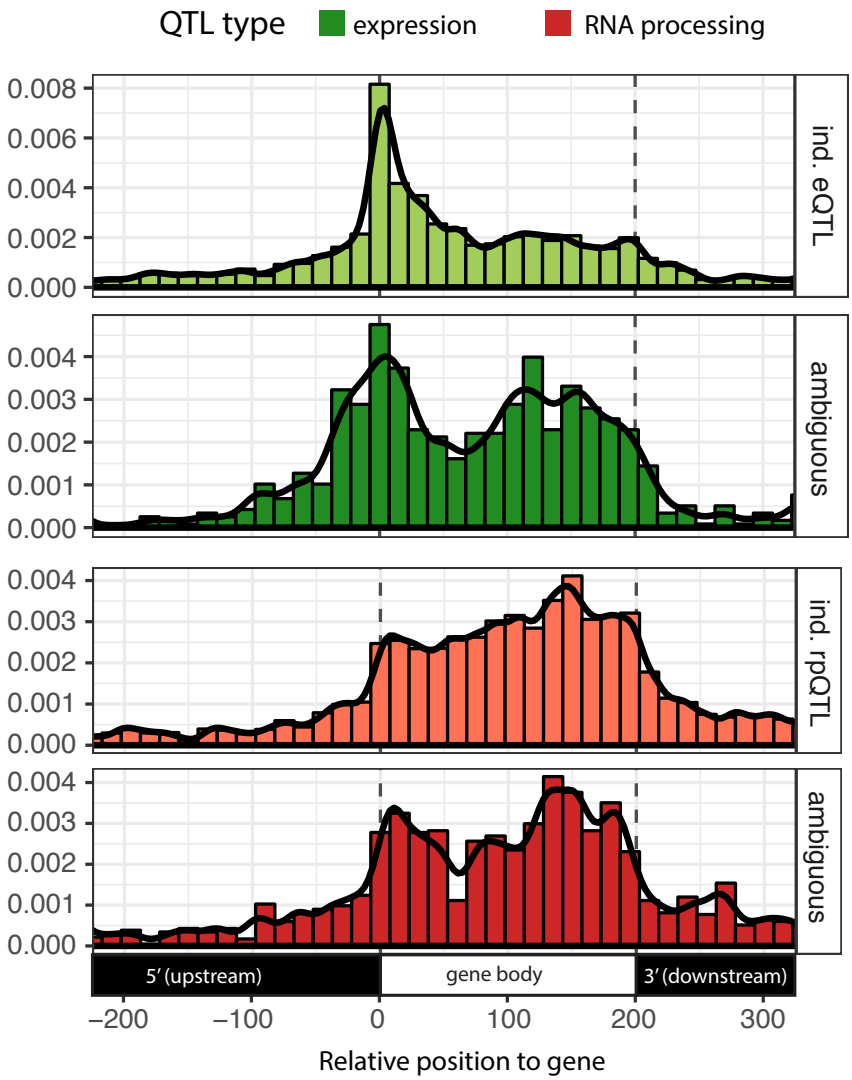
Supplemental Figure 8 Exemple de rpQTL spécifique à l'infection et à la fois ΔPSI QTLs

A) Exemple d'un rpQTL infection-spécifique où l'effet du SNP n'est significatif que dans au moins une condition d'infection. Dans le cas présenté ici, l'association n'est pas significative dans les macrophages sains (FDR=0.52) mais l'est respectivement pour Listeria (FDR=0.0013) et Salmonella (FDR~0). **B)** Exemple d'un ΔPSI QTL. L'association SNP-évènement est la même qu'en a. L'axe des X présente les conditions où NI>L signifie la différence entre les macrophages sains et les macrophages infectés par Listeria et NI>S par Salmonella.



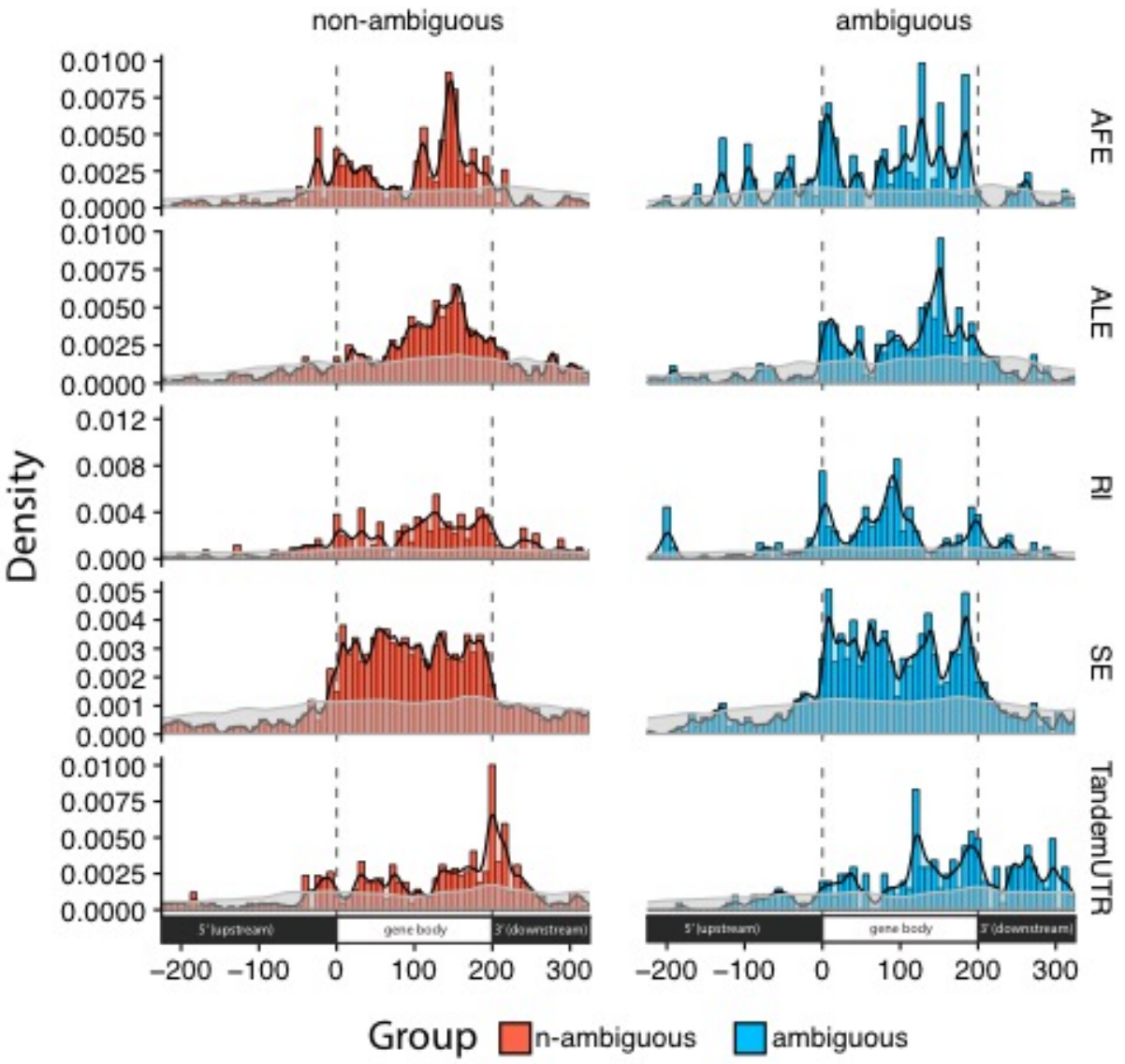
Supplemental Figure 9 Distribution du pourcentage de changement en considérant le meilleur eQTL

L'intégration de l'effet du meilleur eQTL dans l'identification des rpQTL permet de quantifier l'impact de cette intégration. Le calcul de changement se fait en comparant les betas des deux modèles (beta sans considérer l'eQTL et beta' en considérant l'eQTL). La proportion de changement (en Y) est donnée par (beta-beta')/beta. La figure ci-contre montre la distribution de cette valeur parmi les rpQTL significatifs (FDR<5%).



Supplemental Figure 10 Localisation des QTL-SNP

Distribution de la localisation des variants impliqués dans les associations rpQTL et eQTL. Les associations sont séparés en deux catégories : indépendants (L'effet rpQTL et eQTL sont distincts) et ambiguës (L'intégration du eQTL affecte significativement l'effet du rpQTL). Les eQTL ambiguës sont représentés par les gènes comportant au moins un rpQTL ambiguë. Les couleurs permettent de différencier les eQTL et les rpQTL (vert/rouge) et les groupes indépendants et ambiguës (pâle/foncé). L'axe des X représente la position relative du SNP par rapport à son gène associé (voir Méthodes).



Supplemental Figure 11 Localisation des rpQTL-SNP par type d'évènement

Distribution de la localisation des variants impliqués dans les associations rpQTL séparés par type d'évènements. Les associations sont séparés en deux catégories : indépendants (L'effet rpQTL et eQTL sont distincts) et ambiguës (L'intégration du eQTL affecte significativement l'effet du rpQTL. L'axe des X représente la position relative du SNP par rapport à son gène associé (voir Méthodes).

Supplemental Figure 12 Enrichissement GWAS

Enrichissiment des rpQTLs pour des polymorphismes associés à des traits par des études GWAS. Dans ce graphique, les rpQTLs sont divisés en deux groupes; *not ambiguous* : rpQTLs dont l'effet est indépendant de l'effet des eQTL; *ambiguous* : rpQTLs dont l'effet est significativement associé à l'effet de eQTL.

Table I Number of differentially spliced RNA processing events with total proportion

	AFE	ALE	SE	RI	TandemUTR
NI>L	820 (9.7%)	483 (8.3%)	1 615 (9.1%)	499 (14.7%)	201 (10.9%)
NI>S	1 496 (17.5%)	1 005 (17.3%)	4 197 (23.4%)	929 (27.2%)	540 (29.1%)

Table II Number of significant rpQTL events per condition and event type and total tested events

	AFE	ALE	SE	RI	TandemUTR
Non-Infected	80 (7 819)	295 (5 437)	600 (17 183)	137 (3 311)	76 (1 798)
Listeria	95 (7 735)	283 (5 417)	602 (16 956)	103 (3 284)	72 (1 792)
Salmonella	86 (7 827)	217 (5 467)	665 (17 193)	95 (3 311)	74 (1 802)

Table III MEME motif analysis output for rpQTLs

Consensus Motif	Motif width	E-value	RBP associated with motif
CCAHACCC	8	1.7E-06	RNCMPT00026 (HNRNPK)
AMCCCGTC	8	2.8E-06	RNCMPT00020 (FXR2)
ATAAMGC	7	1.7E-05	RNCMPT00034 (KHDRBS3)
TGKSKGGGG	9	2.2E-05	RNCMPT00160 (HNRNPH2)
GTCTTCAK	8	3.8E-05	RNCMPT00157 (PABPN1)
GCCKACC	7	0.00012	RNCMPT00241 (RBM45)
CTTSCAC	7	0.00015	RNCMPT00071 (SNRPA)
AAAMAGCT	8	0.00022	RNCMPT00158 (CPEB4)
CCCCCYAC	8	4E-04	RNCMPT00187 (BRUNOL6)
AGSCTGAA	8	0.00055	RNCMPT00021 (G3BP2)
ACGYGCCC	8	8E-04	RNCMPT00045 (PPRC1)
CAAAAATC	8	0.0013	RNCMPT00025 (HNRNPC)
AGACAWCT	8	0.0013	RNCMPT00084 (YBX2)
CCCGTCT	7	0.0017	RNCMPT00020 (FXR2)
ACTMCTGC	8	0.0041	RNCMPT00154 (RBM5)
GCGCCRCA	8	0.0062	RNCMPT00085 (ZC3H10)
GAGGAAGA	8	0.016	RNCMPT00021 (G3BP2)
AYAAAGG	7	0.027	RNCMPT00169 (KHDRBS1)
AGCCCGCC	8	0.03	RNCMPT00045 (PPRC1)

Recursive GO

READ ME

There are actually two different analysis scripts and a helper module, which are described below.

The gene_ontology.py is a module that is imported by the other scripts. It contains some functions for parsing 'obo' and 'goa' files that you can download from the GO website.

The go_cat_fisher_test.py script tests for enrichment of GO categories using Fisher's exact test. This assumes that you have two sets of genes: a "foreground" set and a "background" set. (You could also consider "foreground" to mean "present" and "background" to mean "absent"). Using Fisher's exact test should be equivalent to using the hypergeometric test that is employed by some of the GO analysis packages.

The go_cat_probeset_ranks.py script assumes that you have a single list of genes that have a numeric score associated with them. It then does a t-test to compare genes within each GO category against genes from all other GO categories. You can also specify that you want to convert the scores for the genes to ranks before performing the test, in which case I think that the test is equivalent to a Mann-Whitney U test. (The "probeset" in the script name is just there because I originally used the script for analyzing microarray data).

Both programs work iteratively and report the GO category that is most significant in each iteration. They then remove all genes that are part of this category and repeat the procedure again. This helps remove redundancy in the reported results that is caused by GO categories that are highly similar to each other.

I believe that the scripts currently use the swissprot (aka uniprot) identifiers parsed from GO annotation (goa) file. It can probably be changed fairly easily in order to use a different kind of identifier.

gene_ontology.py

```
import sys

class GoTerm:
    def __init__(self):
        self.id = None
        self.name = None
        self.namespace = None
        self.definition = None
        self.children = set([])
        self.swiss_ids = set([])
        self.score_ids = set([])
        self.ids_propogated = False

    def __str__(self):
        return self.id + "\t" + self.name

    def add_child(self, child):
        self.children.add(child)

def read_go_terms(obo_file):
    term = None

    term_dict = {}
    f = open(obo_file)

    child_dict = {}

    for l in f:
        line = l.rstrip()
        if term == None:
            if line == "[Term]":
                # start of a new GO term
                term = GoTerm()

        else:
```

```
            if line == "":
                # end of current term
                term_dict[term.id] = term
                term = None
            elif line.startswith("id:"):
                term.id = line[4:]
            elif line.startswith("name:"):
                term.name = line[6:]
            elif line.startswith("namespace:"):
                term.namespace = line[11:]
            elif line.startswith("def:"):
                # get the definition contained within quotes
                term.definition = line[6:line.index('"',6)]
            elif line.startswith("is_a:"):
                cols = line.split(" ")
                parent_id = cols[1]
                if parent_id in child_dict:
                    child_dict[parent_id].add(77)
                else:
                    child_dict[parent_id] = set([term])
    f.close()

    # set children
    for parent_id in child_dict:
        parent_term = term_dict[parent_id]

        for child_term in child_dict[parent_id]:
            parent_term.add_child(child_term)

    return term_dict

def prop_swiss_ids(go_term):
    if go_term.ids_propogated:
        return

    before = len(go_term.swiss_ids)

    for child in go_term.children:
        # propogate ids up from child's children
        prop_swiss_ids(child)
        # add child's ids to this term's ids
        go_term.swiss_ids.update(child.swiss_ids)

    after = len(go_term.swiss_ids)
    go_term.ids_propogated = True

def propogate_swiss_terms(go_term_dict):
    for go_term in go_term_dict.values():
        prop_swiss_ids(go_term)

def read_go_annotation(go_annot_file, go_term_dict):
    """Sets the swissprot identifiers for the GO terms"""
    f = open(go_annot_file)

    go_swiss_mapping = {}

    for l in f:
        if l.startswith("!"):
            # skip comment lines
            continue

        cols = l.rstrip().split("\t")
        swiss_id = cols[1].strip()
        go_id = cols[4].strip()

        go_term = go_term_dict[go_id]

        go_term.swiss_ids.add(swiss_id)

    f.close()
```

```
    propogate_swiss_terms(go_term_dict)
```

go_cat_fisher_test.py

```
import math
import gene_ontology as go
import numpy as np
from scipy import stats
import sys
import os
from optparse import OptionParser


GO_DIR = "/Users/oliviertastet/Downloads/geneontology_McVickerGeorge/databases"

def filter_go_terms(go_terms, namespace="all", min_genes=1):
    """filters out GO terms that do not match criteria
    and only retains score ids (for null distribution)
    that are contained in these GO terms"""
    kept_go_terms = []
    for go_term in go_terms:
        if namespace != "all" and namespace != go_term.namespace:
            # skip term: from different namespace
            continue
        if go_term.n_ids >= min_genes:
            kept_go_terms.append(go_term)
    return kept_go_terms


def read_genes(filename):
    """Reads swiss prot ids from a file"""
    f = open(filename, "r")
    swiss_ids = set([])
    for line in f:
        words = line.strip().split()
        if len(words) != 1:
            raise ValueError("expected line to have single identifier, "
                             "got %d: '%s'" % (len(words), line))
        swiss_ids.add(words[0])
    return swiss_ids



def assign_go_genes(go_terms, fg_swiss_ids, bg_swiss_ids):
    for go_term in go_terms.values():
        go_term.fg_ids = set([])
        go_term.bg_ids = set([])
        for swiss_id in go_term.swiss_ids:
            if swiss_id in fg_swiss_ids:
                go_term.fg_ids.add(swiss_id)
            if swiss_id in bg_swiss_ids:
                go_term.bg_ids.add(swiss_id)
        go_term.n_ids = len(go_term.fg_ids) + len(go_term.bg_ids)




def update_go_terms(go_terms, removed_term):
    for term in go_terms:
        if term.id != removed_term.id:
            term.fg_ids = term.fg_ids - removed_term.fg_ids
            term.bg_ids = term.bg_ids - removed_term.bg_ids
            term.n_ids = len(term.fg_ids) + len(term.bg_ids)

    removed_term.fg_ids = set([])
    removed_term.bg_ids = set([])
```

```
    removed_term.n_ids = 0


def get_go_terms(fg_swiss_ids, bg_swiss_ids,
                 namespace="all", min_genes=1):
    sys.stderr.write("reading GO terms\n")
    # read go terms
    go_terms = go.read_go_terms(GO_DIR + "/gene_ontology.obo")
    # read swissprot ids for GO categories
    sys.stderr.write("reading GO annotation\n")
    go.read_go_annotation(GO_DIR + "/gene_association.goa_ref_human",
                          go_terms)
    assign_go_genes(go_terms, fg_swiss_ids, bg_swiss_ids)
    # drop GO terms which have no associated score identifiers
    return filter_go_terms(go_terms.values(),
                           namespace=namespace,
                           min_genes=min_genes)



def get_genes(fg_gene_filename, bg_gene_filename):
    sys.stderr.write("reading foreground genes from file %s\n" %
                     fg_gene_filename)
    fg_swiss_ids = read_genes(fg_gene_filename)
    sys.stderr.write("reading background genes from file %s\n" %
                     bg_gene_filename)
    bg_swiss_ids = read_genes(bg_gene_filename)
    fg_bg_intersect = fg_swiss_ids.intersection(bg_swiss_ids)
    if len(fg_bg_intersect) != 0:
        sys.stderr.write("WARNING: removing %d genes that are in both "
                         "foreground and background datasets\n" %
                         len(fg_bg_intersect))
        fg_swiss_ids = fg_swiss_ids - fg_bg_intersect
        bg_swiss_ids = bg_swiss_ids - fg_bg_intersect
    sys.stderr.write("there are %d genes in the FG set, and %d genes "
                     "in the background set\n" %
                     (len(fg_swiss_ids), len(bg_swiss_ids)))
    return (fg_swiss_ids, bg_swiss_ids)


def get_remaining_gene_sets(go_terms):
    fg_remain_set = set([])
    bg_remain_set = set([])

    for term in go_terms:
        fg_remain_set.update(term.fg_ids)
        bg_remain_set.update(term.bg_ids)

    return (fg_remain_set, bg_remain_set)



MAX_ITER = 100

NAMESPACES = ["all", "biological_process", "cellular_component",
              "molecular_function"]


def main():
    usage = "usage: %prog [options] <fg_swiss_file> <bg_swiss_file>"
    parser = OptionParser(usage=usage)

    parser.add_option("-n", "--namespace",
                      type="choice", action="store", dest="namespace",
                      default="all",
                      choices=NAMESPACES,
                      help="the GO namespace to use: " + ", ".join(NAMESPACES))
```

```
parser.add_option("-l", "--lower",
                  action="store_true", dest="lower_tail",
                  default=False,
                  help="use the lower tail of the test statistic "
                  "distribution")

parser.add_option("-m", "--mingenes",
                  action="store", type="int", dest="mingenes",
                  default=10,
                  help="only consider GO categories with at least this "
                  "many genes")

parser.add_option("-s", "--signif",
                  action="store", type="float", dest="signif",
                  default=0.05,
                  help="only include GO categories with "
                  "bonferroni corrected p-values <= this value level"
                  "in ranking (use 1.0 to include all categories)")

(options, args) = parser.parse_args()

if options.namespace not in NAMESPACES:
    parser.error("NAMESPACE must be one of: "+", ".join(NAMESPACES))

if len(args) != 2:
    parser.print_help()
    exit(2)

(fg_swiss_ids, bg_swiss_ids) = get_genes(args[0], args[1])

go_terms = get_go_terms(fg_swiss_ids, bg_swiss_ids,
                        namespace=options.namespace,
                        min_genes=options.mingenes)

if len(go_terms) == 0:
    raise Exception("no GO terms for namespace: " + options.namespace)

print "\t".join(["GO.ID", "GO.NAMESPACE", "GO.NAME",  "N.TEST",
                 "CAT.N.FG.GENE", "CAT.N.BG.GENE",
                 "N.FG.GENE", "N.BG.GENE", "P", "ODDS.RATIO", "GENES"])

for i in range(0, MAX_ITER):
    min_p = 1
    min_go_term = None
    n_test = 0

    (remain_fg_ids, remain_bg_ids) = get_remaining_gene_sets(go_terms)
    total_n_fg = len(remain_fg_ids)
    total_n_bg = len(remain_bg_ids)

    for go_term in go_terms:
        if go_term.n_ids >= options.mingenes:
            go_term.n_fg = len(go_term.fg_ids)
            go_term.n_bg = len(go_term.bg_ids)

            table = [[go_term.n_fg, go_term.n_bg],
                     [total_n_fg, total_n_bg]]

            (odds_ratio, p) = stats.fisher_exact(table)


            n_test += 1
            go_term.odds_ratio = odds_ratio
            go_term.p = p
        else:
            go_term.p = None
            go_term.odds_ratio = None

    ranked = sorted(go_terms, key=lambda term: term.p)
```

```
        if len(ranked) < 1:
            break

        min_go_term = ranked[0]

        print "\t".join([min_go_term.id, min_go_term.namespace,
                         '"' + min_go_term.name + '"',
                         str(n_test),
                         str(min_go_term.n_fg),
                         str(min_go_term.n_bg),
                         str(total_n_fg),
                         str(total_n_bg),
                         "%.3g" % min_go_term.p,
                         "%.3g" % min_go_term.odds_ratio,
                         ",".join(min_go_term.fg_ids)])

        # update terms, remaining genes
        update_go_terms(go_terms, min_go_term)

        go_terms = filter_go_terms(go_terms,
                                   namespace=options.namespace,
                                   min_genes=options.mingenes)


if __name__ == "__main__":
    main()
```

Logistic regression enrichments

READ ME

A logistic regression is implemented to estimate enrichment of certain features among the set of significant QTLs. A matrix where each row is a different SNP is passed to the logreg function which will call the other defined function, performing the logistic regression. The first column of the matrix is binary (0/1) where 0 is not involved in a rpQTL and 1 is involved in an rpQTL. Each other column of the matrix is a feature to investigate. It can be general (TF binding site) or more specific (NFKB binding site). This information also needs to be binary. Either the SNP is or is not a feature. The output is a matrix providing P-value and Odds ratios for every investigated feature.

```
error.bar <- function(x, y, upper, lower=upper, length=0.1,vertical=T,...){
  if(length(x) != length(y) | length(y) !=length(lower) | length(lower) != length(upper))
    stop("vectors must be same length")
  if(vertical){
    arrows(x,y+upper, x, y-lower, angle=90, code=3, length=length, ...)
  }else{
    arrows(y+upper,x,y-lower,x, angle=90, code=3, length=length, ...)
  }}
fLogit <- function(BetaLogit,anno,Prob){
    -sum(Prob*(anno %*% BetaLogit) - log(1+exp(anno %*% BetaLogit)))
  }
gLogit <- function(BetaLogit,anno,Prob){
    myPi <- plogis(anno %*% BetaLogit)
    -t(Prob-myPi) %*% anno
  }
simpleLogistic <- function(anno,Prob){
        aux <- cbind(Ict=1,anno)
        BetaFit <-
optim(rep(0,ncol(anno)+1),fLogit,gLogit,anno=as.matrix(aux),Prob=Prob,method="BFGS",control=lis
t(maxit=500),hessian=TRUE);
        save(BetaFit, file="betaFit.rda")
        r <-
as.matrix(replicate(nrow(BetaFit$hessian),sample.int(10,nrow(BetaFit$hessian),rep=TRUE)-
1)/10000)
        BetaFit$hessian = BetaFit$hessian + r
        logitSE <- sqrt(diag(solve(as.matrix(BetaFit$hessian))))
        Zlogit <- BetaFit$par/logitSE
        pval <- pnorm(-abs(Zlogit))*2
        res <- data.frame(Beta=BetaFit$par,SE=logitSE,Z=Zlogit,pval=pval)
        row.names(res) <- colnames(aux)
        res[,"fdr"] = c(0,p.adjust(res[-1,"pval"], "fdr"))
        res = res[order(res$fdr),]}
logreg<-function(mat,group){
        logreg.data = mat
        logreg.data[is.na(logreg.data)]=0
        Prob = as.numeric(as.character(logreg.data[,1]))
        anno = data.frame(logreg.data[,2:(ncol(logreg.data)-1)])
        res <- simpleLogistic(anno, Prob)
        res = res[-1,]
        res[,"fdr"] = p.adjust(res$pval, "fdr")
        res$term=rownames(res)
        res$group=group
        return(res)}
```

FDR Correction

```
library(ggplot2)
library(reshape2)
library(cobs)
#library(grDevices)
#library(gridExtra)
#library(cowplot)


##SUMMARY: Compile and load the C libraries for faster core routines.
system("R CMD SHLIB -Wall fdrSAB.c")
dyn.load("fdrSAB.so")

##SUMMARY:
fdrSAB=function(
##SUMMARY: df or path_to_file containing true p-values
full_data,
##SUMMARY: column of full_data where actual p_values are.
full_column_id,
##SUMMARY: df or path_to_file containing permuted p_values
perm_data,
##SUMMARY: column or columns of perm_data where permuted p_values are
perm_column_ids,
##SUMMARY: list with alternative methods to present in output
alt_methods,
##SUMMARY:Number of knots used for the splines fit (minimum 3, maximum:
min(3,number_of_points_to_fit/100)). If missing the number of knots that will be used is
max(5,min(20,number_of_points_to_fit/1000))
pi_o_knots,
##SUMMARY:Minimum and maximum knopt sizes for the splines fit (relative to the total x-axis
range (p-values) in the fit).
pi_o_knots_range,
##SUMMARY:Number of iterations to be used by the splines fit.
pi_o_iters=10000,
##SUMMARY: Method for estimating p_star: if FALSE: p_star is quickly estimated as the argmin of
the quotient of the local grenander p-value densities estimators; if TRUE, it is identified
from a thorough sweep (computationally exhaustive).
p_star_exhaustive_sweep=FALSE,
##SUMMARY: whether the summary plot is wanted or not.
plot=TRUE,
##SUMMARY: threshold for significance assessments
significance_threshold=0.05,
##SUMMARY: report_threshold in final files and axis limit in fdrs.pdf figure.
report_threshold=1,
##SUMMARY: a folder with such name will be created to store results in working directory.
output_name="outputs_PermFDR",
##SUMMARY: whether progress messages are wanted or not
print_messages=TRUE)
{
    {
        if(print_messages)
        {

cat("\n\n###############################################################################################
#########################","\n")
            cat("################### Sanz-Arregui-Barreiro method for controlling FDRs from a
permuted null ###################","\n")

cat("###############################################################################################
####################","\n")
            cat("\n_____________________________________STEP 1: Check & load input
data_____________________________________\n")
            cat("\nChecking inputs...\n")
        }

        flag_debug_august=1
```

```
 ##1. Declare check_error_vector, perm_fdr_plot & pi_o_fit internal functions.
perm_fdr_plot=function (plots){

     rel_heights <- c(1,1,1,1)
     rel_widths <- 1
     scale <- c(1,1,1,1)
     rows = 4
     cols = 1

     grobs <- lapply(plots, function(x) {ggplot2::ggplotGrob(x)})
     num_widths <- unique(lapply(grobs, function(x) {
         length(x$widths)
     }))
     num_widths[num_widths == 0] <- NULL
     max_widths <- do.call(grid::unit.pmax, lapply(grobs,
     function(x) {
         x$widths
     }))

     for (i in 1:4) {
         grobs[[i]]$widths <- max_widths
     }

     x_deltas <- rel_widths/sum(rel_widths)
     y_deltas <- rel_heights/sum(rel_heights)
     xs <- cumsum(rel_widths)/sum(rel_widths) - x_deltas
     ys <- 1 - cumsum(rel_heights)/sum(rel_heights)
     ggdraw=function (plot = NULL)
     {
         theme_nothing=function(base_size = 12, base_family = "")
         {
             theme_grey(base_size = 12, base_family = "") %+replace%
             theme(rect = element_rect(fill = "transparent", colour = NA,
             color = NA, size = 0, linetype = 0), line = element_blank(),
             text = element_blank(), title = element_blank(),
             panel.background = element_blank(), axis.ticks.length = grid::unit(0,
             "lines"), legend.position = "none", panel.margin = grid::unit(c(0,
             0, 0, 0), "lines"), plot.margin = grid::unit(c(0,
             0, 0, 0), "lines"))
         }

         d <- data.frame(x = 0:1, y = 0:1)
         p <- ggplot(d, aes_string(x = "x", y = "y")) + scale_x_continuous(limits = c(0,
         1), expand = c(0, 0)) + scale_y_continuous(limits = c(0,
         1), expand = c(0, 0)) + theme_nothing() + labs(x = NULL,y = NULL)
         if (!is.null(plot)) {
             if (methods::is(plot, "ggplot")){
                 g<-ggplot2::ggplotGrob(plot)
             }else if (methods::is(plot, "gtable")){
                 g<-plot
             }else{
                 stop('Argument needs to be of class "ggplot" or "gtable"' )
             }
             plot.grob <- grid::grobTree(g)
             p <- p + annotation_custom(plot.grob)
         }
         p
     }
     p <- ggdraw()
     col_count <- 0
     row_count <- 1
     for (i in 1:(rows * cols)) {
         if (i > 4)
         break
         x_delta <- x_deltas[col_count + 1]
         y_delta <- y_deltas[row_count]
         width <- x_delta * scale[i]
         height <- y_delta * scale[i]
         x_off <- (x_delta - width)/2
```

```
                y_off <- (y_delta - height)/2
                x <- xs[col_count + 1] + x_off
                y <- ys[row_count] + y_off
                draw_grob=function (grob, x = 0, y = 0, width = 1, height = 1)
                {
                    annotation_custom(grid::grobTree(grob), xmin = x, xmax = x + width, ymin =
y, ymax = y + height)
                }

                p <- p + draw_grob(grid::grobTree(grobs[[i]]),x,y,width,height)
                col_count <- col_count + 1
                if (col_count >= cols) {
                    col_count <- 0
                    row_count <- row_count + 1
                }
            }
            p
        }

pi_o_fit=function(stats_array,pi_hat,pi_o_knots,pi_o_iters,pi_o_knots_range,trend,full_number,p
erm_number,silent=FALSE,constraint_matrix=NULL){

            if(pi_o_knots==0){
                #pi_o_knots=min(30,as.integer(2+(28/3)*log(length(stats_array)/10)))
                pi_o_knots=min(20,as.integer(2+3*log10(length(stats_array)/10)))
                if(silent==FALSE)cat("Fitting pi_o with ",pi_o_knots," knots")
            }
            if(all(pi_o_knots_range==c(0,0))){
                #pi_o_knots_range=c(1e-12,1)
                pi_o_knots_range=c(0,1)

            }

            p_star=stats_array[length(stats_array)]
            p_star_index=length(stats_array)

            if(trend=="decrease"|trend=="none_left")
            {
                if(missing(constraint_matrix))

#constraint_matrix=as.matrix(data.frame(c(0,2),c(0,p_star),c((full_number/perm_number),0)))
                constraint_matrix=as.matrix(data.frame(c(2),c(p_star),c(0)))

                width_last_one_per_thousand_quantile=p_star-
stats_array[as.integer(length(stats_array)*0.999)]
            }else if(trend=="increase"|trend=="none_right"){

                if(missing(constraint_matrix))
                #constraint_matrix=as.matrix(data.frame(c(0,2),c(1,stats_array[1]),c((1-
full_number)/(1-perm_number),0)))
                constraint_matrix=as.matrix(data.frame(c(2),c(stats_array[1]),c(0)))

width_last_one_per_thousand_quantile=stats_array[as.integer(length(stats_array)*0.001)]-
stats_array[1]
            } else{
                if(missing(constraint_matrix))
                #constraint_matrix=as.matrix(data.frame(c(2),c(p_star),c(0)))

constraint_matrix=as.matrix(data.frame(c(0,2),c(stats_array[1],p_star),c((full_number/perm_numb
er),0)))
                width_last_one_per_thousand_quantile=p_star-
stats_array[as.integer(length(stats_array)*0.999)]
            }


            if(trend %in% c("none_left","none_right")) trend="none"

            df<-data.frame(stats_array,pi_hat)
            write.table(df,paste0(output_name,"/cobs_input.txt"))
```

```
            range=abs(stats_array[1]-stats_array[length(stats_array)])
            max_res=range*max(pi_o_knots_range)
            min_res=range*min(pi_o_knots_range)

            if((trend %in% c("decrease","increase")) &
max_res<width_last_one_per_thousand_quantile)
            {
                max_res=width_last_one_per_thousand_quantile
                if(silent==FALSE)cat(paste0("Fitting true null fraction to a spline with knots
in a range between ",min_res," and ",max_res,"\n"))
                if(silent==FALSE)cat(paste0("(Note: maximum knot size augmented to ",max_res,"
from proposed input to allow the closest knot to p_star to cover the 1 per 1000 of the data at
least)"),"\n")
            }else{
                if(silent==FALSE)cat(paste0("Fitting true null fraction to a spline with knots
in a range between ",min_res," and ",max_res,"\n"))
            }
            nknots=pi_o_knots

            repeat{
                knots=stats_array[1]
                index_last_knot=1
                step=as.integer(p_star_index/nknots)

                repeat
                {
                    past_index=index_last_knot
                    keep_going=(index_last_knot+step)<p_star_index
                    if(keep_going==1){
                        anchura_quantile=stats_array[as.integer(index_last_knot+step)]-
knots[length(knots)]
                        end=stats_array[index_last_knot+step]
                        end_index=index_last_knot+step
                    }else{
                        anchura_quantile=p_star-knots[length(knots)]
                        end=p_star
                        end_index=p_star_index
                    }
                    if(anchura_quantile<min_res){
                        knots=c(knots,min(knots[length(knots)]+min_res,p_star))
                        if(length(which(stats_array>=knots[length(knots)]))>0)
                        {
                            index_last_knot=which(stats_array>=knots[length(knots)])[1]
                        }else{
                            index_last_knot=p_star_index
                            break;
                        }
                    }else{
                        if(anchura_quantile>max_res){
                            knots=c(knots,knots[length(knots)]+max_res)
                            if(length(which(stats_array>=knots[length(knots)]))>0)
                            {
                                index_last_knot=which(stats_array>=knots[length(knots)])[1]
                            }else{
                                index_last_knot=p_star_index
                                break;
                            }
                        }else{
                            knots<-c(knots,end)
                            index_last_knot=end_index
                            if(end==1){break}
                        }
                    }
                    if((p_star-knots[length(knots)])<0.2*(knots[length(knots)]-
knots[length(knots)-1]))
                    {
                        knots[length(knots)]=p_star
                        index_last_knot=length(knots)
                        break
                    }
```

```
                }
                if(length(knots)>=pi_o_knots){
                    break
                }else{nknots=nknots+1}
            }

            if(silent==FALSE){
                f_hat<-
cobs(stats_array,pi_hat,knots=knots,constraint=trend,pointwise=constraint_matrix,maxiter=pi_o_i
ters,print.warn=FALSE,print.mesg=FALSE)
            }else{
                f_hat<-
suppressWarnings(cobs(stats_array,pi_hat,knots=knots,constraint=trend,pointwise=constraint_matr
ix,maxiter=pi_o_iters,print.warn=FALSE,print.mesg=FALSE))
            }
            return(f_hat)
        }
        check_error_vector=function(error_vector,flag_error){
            for(i in 1:length(error_vector))
            {
                if(nchar(error_vector[i])>0)
                {
                    cat(error_vector[i])
                    flag_error=1
                }
            }
        }

        ##SUMMARY 2. Check plot output_name format: possible warning: it's not a single logical
value: will be set to TRUE then.
        {
            warning_output_name=vector(mode = "character", length = 1)
            if(typeof(output_name)!="character"| length(output_name)!=1)
            {
                warning_output_name[1]=paste0("WARNING: Invalid argument output_name: Outputs
will be saved in: ",getwd(),"/outputs_PermFDR\n")
                output_name="outputs_PermFDR"
            }
            if(output_name==""){
                command="mkdir -p outputs_PermFDR"
                output_name="outputs_PermFDR"
                warning_output_name[1]=paste0("WARNING: Invalid argument output_name: Outputs
will be saved in: ",getwd(),"/outputs_PermFDR\n")
                system(command)
            }else{
                command=paste0("mkdir -p ",output_name)
                system(command)
            }
            if(nchar(warning_output_name)>0) cat(warning_output_name)
        }
        ##SUMMARY: 3. Check full_data & full_column_id inputs formats:
        ##For full_data: possible errors:
        ##1. Missing argument.
        ##2. Bad-format argument (neither df nor file).
        ##3. If file, and the file is absent or impossible to load: invalid file.
        ##4: if file: bad parsing.
        ##5: Not numeric, or values outside interval [0,1] in tests column (invalid p-values).
        ##For full_column_id: possible errors.
        ##1. Missing argument.
        ##2. Bad-format argument (neither column name or ordinal).
        ##3. Out-of-range argument(column name or ordinal that doesn't match with present
columns ids in full_data
        {
            error_full_data=vector(mode = "character", length = 1)
            error_full_column_id=vector(mode = "character", length = 1)
            if(missing(full_data)){
                error_full_data[1]="ERROR: Absent argument full_data.\n"
            }else if((typeof(full_data)!="character"|(typeof(full_data)=="character" &
length(full_data)>1)) & is.data.frame(full_data)==FALSE){
```

```
                error_full_data[1]="ERROR: Invalid argument full_data: must be a data frame or
a path to a file.\n"
            }else if(typeof(full_data)=="character"){
                full_data_is_file=1
                #Check wether a rownames column is present: NOTE that a colnames row is needed
                head=try(read.table(full_data,nrows=2,header=TRUE),silent=TRUE)
                head_2=try(read.table(full_data,nrows=2,header=FALSE),silent=TRUE)
                if(is(head,"try-error"))
                {error_full_data[1]=paste0("ERROR: Invalid argument full_data: file:
",full_data," is missing or bad formatted.\n")
                }else{
                    if(is(head_2,"try-error")) {
                        rownames_flag_full=1
                    }else{
                        rownames_flag_full=0
                    }
                    columns_full_data=ncol(head)+rownames_flag_full
                }

                #Check if full_column_id argument matches a column in full_data file
                if(nchar(error_full_data[1])==0)
                {
                    if(missing(full_column_id)){
                        error_full_column_id[1]="ERROR: Absent argument full_column_id.\n"
                    }else if((typeof(full_column_id)!="character" &
typeof(full_column_id)!="double") | length(full_column_id)>1){
                        error_full_column_id[1]="ERROR: Invalid argument full_column_id: must
be a valid column identificator (column name or integer ordinal index)\n"
                    }else if(typeof(full_column_id)=="character"){
                        buffer=which(colnames(head) %in% full_column_id)
                        if(length(buffer)!=1) error_full_column_id[1]=paste0("ERROR: Invalid
argument full_column_id:\n","Column name introduced: ",full_column_id," was not found in
full_data file: ",full_data,".\n")
                    }else if(full_column_id%%1!=0){
                        error_full_column_id[1]="ERROR: Invalid argument full_column_id: must
be a valid column identificator (column name or integer ordinal index)\n"
                    }else if(full_column_id<1 | full_column_id>ncol(head))
                    {
                        error_full_column_id[1]=paste0("ERROR:  Invalid argument
full_column_id:\nColumn ordinal: ",full_column_id," is out of range: full_data file contains
",ncol(head)," columns.\n")
                    }else{
                        buffer=full_column_id
                    }
                    full_column_id=buffer+rownames_flag_full
                }

                #If everything is yet, now check file parsing and get number of rows:
                if(nchar(error_full_data[1])==0 & nchar(error_full_column_id[1])==0)
                {
                    fields=ncol(head)
                    if(rownames_flag_full==1) fields=fields+1
                    if(print_messages){cat("Checking full_data file...\n")}
                    ##This will do the following:
                    ##Keep track of max_p and min_p
                    ##Print one line per bad-parsed line.
                    ## IF any value outside interval [0,1] is found standing for a p-value:
flag_badp=1
                    ## Print a final 4 lines with NR (number of rows in file (number of tests
plus one for the colnames)) min_p, max_p, flag_badp

                    command=paste0("awk 'BEGIN{RS=\"\\r\";print 1;}{if(NR>1){print NR; exit
0;}}' ",full_data,"> parsing_full1_",output_name,".txt")
                    system(command)
                    full_parsing_name<-paste0("parsing_full1_",output_name,".txt")
                    parsing=read.table(full_parsing_name,header=FALSE)
                    if(nrow(parsing)==2){
                        command=paste0("awk 'BEGIN
{max=",fields,";flag_bad_p=0;RS=\"\\r\";min_stat=1;max_stat=0} {if(NF!=max) {print
NR;}if(NR>1){if(($",full_column_id,"<0)||($",full_column_id,"!~ '/^[0-9.eE+'-
```

```
']*$/')||($",full_column_id,">1))  {flag_bad_p=1; exit 0;}} if(NR>1)
{if(($",full_column_id,"<min_stat)&&($",full_column_id,">0))
min_stat=$",full_column_id,";if($",full_column_id,">max_stat) max_stat=$",full_column_id,";}}
END{print NR; print min_stat; print max_stat; print flag_bad_p}' ",full_data," >
parsing_full_",output_name,".txt\n")
                        system(command)
                    }else{
                        command=paste0("awk 'BEGIN{RS=\"\\n\";print 1;}{if(NR>1){print NR; exit
0;}}' ",full_data,"> parsing_full2_",output_name,".txt")
                        system(command)
                        full_parsing_name<-paste0("parsing_full2_",output_name,".txt")
                        parsing=read.table(full_parsing_name,header=FALSE)
                        if(nrow(parsing)==2){
                           command=paste0("awk 'BEGIN
{max=",fields,";flag_bad_p=0;RS=\"\\n\";min_stat=1;max_stat=0} {if(NF!=max) {print
NR;}if(NR>1){if(($",full_column_id,"<0)||($",full_column_id,"!~ '/^[0-9.eE+'-
']*$/')||($",full_column_id,">1))  {flag_bad_p=1; exit 0;}} if(NR>1)
{if(($",full_column_id,"<min_stat)&&($",full_column_id,">0))
min_stat=$",full_column_id,";if($",full_column_id,">max_stat) max_stat=$",full_column_id,";}}
END{print NR; print min_stat; print max_stat; print flag_bad_p}' ",full_data," >
parsing_full_",output_name,".txt\n")
                           system(command)
                        }else{
                            error_full_data[1]="ERROR: Invalid argument full_data\n"
                        }
                    }
                    full_parsing_name<-paste0("parsing_full_",output_name,".txt")
                    parsing=read.table(full_parsing_name,header=FALSE)
                    ##Lines that the file will have by default: the final 4 lines plus another
one (indicating that the first row contains one field less than the default) if a rownames
column is present.
                    parsing_lines=4+rownames_flag_full
                    if(nrow(parsing)>parsing_lines)
                    {error_full_data[1]="ERROR: Invalid argument full_data: bad parsing.\n"
                    }else{
                        full_tests=as.double(parsing[nrow(parsing)-3,1])-1
                        processed_tests=full_tests

                        min_stat=parsing[nrow(parsing)-2,1]
                        max_stat=parsing[nrow(parsing)-1,1]
                        if(parsing[nrow(parsing),1]==1)
                        {error_full_data[1]=paste0("ERROR: Invalid p-values (non-numeric and/or
out of range) at full_data file:",full_data,".\n")}
                    }
                    rm(parsing)
                }
            }else if (is.data.frame(full_data)==TRUE){

                full_data_is_file=0

                #Check if full_column_id argument matches a column in full_data dataframe
                if(missing(full_column_id)){
                    error_full_column_id[1]="ERROR: Absent argument full_column_id.\n"
                }else if((typeof(full_column_id)!="character" &
typeof(full_column_id)!="double") | length(full_column_id)>1){
                    error_full_column_id[1]="ERROR: Invalid argument full_column_id: must be a
valid column identificator (column name or integer ordinal index)\n"
                }else if(typeof(full_column_id)=="character")
                {
                    buffer=which(colnames(full_data) %in% full_column_id)
                    if(length(buffer)!=1)
                    error_full_column_id[1]=paste0("ERROR: Invalid argument
full_column_id:\nColumn name introduced: ",full_column_id," was not found in full_data
dataframe.\n")
                }else if(full_column_id%%1!=0){
                    error_full_column_id[1]="ERROR: Invalid argument full_column_id: must be a
valid column identificator (column name or integer ordinal index)\n"
                }else if(full_column_id<1 | full_column_id>ncol(full_data))
                {
```

```
                    error_full_column_id[1]=paste0("ERROR:  Invalid argument
full_column_id:\nColumn ordinal: ",full_column_id," is out of range: full_data file contains
",ncol(full_data)," columns.\n")
                }else{
                    buffer=full_column_id
                }

                if(nchar(error_full_column_id[1])==0)
                {
                    full_column_id=buffer
                    failed_full_entries=length(which(is.na(full_data[,full_column_id]) |
(full_data[,full_column_id]<0) | (full_data[,full_column_id]>1)))
                    if(failed_full_entries>0) {error_full_data[1]="ERROR: Invalid p-values at
full_data dataframe (non-numeric and/or out of range).\n"}
                }

                if(nchar(error_full_column_id[1])==0 & nchar(error_full_data[1])==0)
                {
                    full_tests=nrow(full_data)
                    processed_tests=full_tests
                }
            }
            check_error_vector(error_full_data,flag_error)
            check_error_vector(error_full_column_id,flag_error)
        }
        ##SUMMARY: 4 Check perm_data & perm_column_ids inputs formats:
        ##For perm_data: possible errors:
        ##1. Missing argument.
        ##2. Bad-format argument (neither df nor file).
        ##3. If file, and the file is absent or impossible to load: invalid file.
        ##4: if file: bad parsing.
        ##5: Not numeric, or values outside interval [0,1] in tests column (invalid p-values).
        ##For perm_column_ids: possible errors.
        ##1. Missing argument.
        ##2. Bad-format argument (neither vector of column names or ordinals or atring "all").
        ##3. The argument contains out-of-range entries (column name or ordinal that doesn't
match with present columns ids in perm_data
        {
            error_perm_data=vector(mode = "character", length = 1)
            error_perm_column_id=vector(mode = "character", length = 1)
            if(missing(perm_data)){
                error_perm_data[1]="ERROR: Absent argument perm_data.\n"
            }else if((typeof(perm_data)!="character"|(typeof(perm_data)=="character" &
length(perm_data)>1)) & is.data.frame(perm_data)==FALSE){
                error_perm_data[1]="ERROR: Invalid argument perm_data: must be a data frame or
a path to a file.\n"
            }else if(typeof(perm_data)=="character"){
                perm_data_is_file=1
                #Check wether a rownames column is present: NOTE that a colnames row is needed
                head=try(read.table(perm_data,nrows=2,header=TRUE),silent=TRUE)
                head_2=try(read.table(perm_data,nrows=2,header=FALSE),silent=TRUE)
                if(is(head,"try-error"))
                {error_perm_data[1]=paste0("ERROR: Invalid argument perm_data: file:
",perm_data,"missing file or bad formatted.\n")
                }else{
                    if(is(head_2,"try-error")){
                        rownames_flag_perm=1
                    }else{
                        rownames_flag_perm=0
                    }
                    columns_perm_data=ncol(head)+rownames_flag_perm
                }

                #Check if all entries in perm_column_id argument match columns in perm_data
file
                if(nchar(error_perm_data[1])==0)
                {
                    if(missing(perm_column_ids)){
                        error_perm_column_ids[1]="ERROR: Absent argument perm_column_ids.\n"
```

```
                    }else if(typeof(perm_column_ids)!="character" &
(typeof(perm_column_ids)!="double")){
                        error_perm_column_ids[1]="ERROR: Invalid argument perm_column_ids: must
be a vector of valid column identificators (column names, integer ordinal indexes or
\"all\")\n"
                    }else if(typeof(perm_column_ids)=="character")
                    {
                        if(perm_column_ids=="all")
                        {buffer=c(1:(columns_perm_data-rownames_flag_perm))
                        }else{
                            buffer=which(colnames(head) %in% perm_column_ids)
                            not_found=which(!(perm_column_ids %in% colnames(head)))
                            if(length(not_found)>0)
                            error_perm_column_id[1]="ERROR: Invalid entries at perm_column_ids:
names of columns introduced not found in perm_data.\n"
                        }
                    }else if(sum(perm_column_ids%%1)!=0){
                        error_perm_column_id[1]="ERROR: Invalid argument perm_column_ids: must
be a vector of valid column identificators (column names, integer ordinal indexes or
\"all\")\n"
                    }else{
                        valid_set=which((perm_column_ids>0)&(perm_column_ids<(ncol(head)+1)))
                        buffer=perm_column_ids[valid_set]
                        if(length(valid_set)<length(perm_column_ids))
                        error_perm_column_id[1]="ERROR: Invalid entries at perm_column_ids: out
of range column ordinals introduced.\n"
                    }
                    perm_column_ids=buffer+rownames_flag_perm
                }

                #Check file parsing and get number of rows:
                if(nchar(error_perm_data[1])==0 & nchar(error_perm_column_id[1])==0)
                {
                    fields=ncol(head)
                    if(rownames_flag_perm==1) fields=fields+1
                    if(print_messages){cat("Checking perm_data file...\n")}

                    command=paste0("awk 'BEGIN{RS=\"\\r\";print 1;}{if(NR>1){print NR; exit
0;}}' ",perm_data,"> parsing_perm1_",output_name,".txt")
                    system(command)
                    perm_parsing_name<-paste0("parsing_perm1_",output_name,".txt")
                    parsing=read.table(perm_parsing_name,header=FALSE)
                    if(nrow(parsing)==2){
                        command=paste0("awk 'BEGIN {max=",fields,";flag_bad_p=0;RS=\"\\r\";}
{if(NF!=max) {print NR;}if(NR>1){if(($",perm_column_ids[1],"<0)||($",perm_column_ids[1],"!~
'/^[0-9.eE+'-']*$/')||($",perm_column_ids[1],">1)")
                        if(length(perm_column_ids)>1){
                            for(i in 2:length(perm_column_ids))

command=paste0(command,"||($",perm_column_ids[i],"<0)||($",perm_column_ids[i],"!~ '/^[0-9.e+'-
']*$/')||($",perm_column_ids[i],">1) ")
                        }
                        command=paste0(command,"){flag_bad_p=1; exit 0;}}} END{print NR; print
flag_bad_p}' ",perm_data," > parsing_perm_",output_name,".txt\n")
                        system(command)
                    }else{
                        command=paste0("awk 'BEGIN{RS=\"\\n\";print 1;}{if(NR>1){print NR; exit
0;}}' ",perm_data,"> parsing_perm2_",output_name,".txt")
                        system(command)
                        perm_parsing_name<-paste0("parsing_perm2_",output_name,".txt")
                        parsing=read.table(perm_parsing_name,header=FALSE)
                        if(nrow(parsing)==2){
                            command=paste0("awk 'BEGIN
{max=",fields,";flag_bad_p=0;RS=\"\\n\";} {if(NF!=max) {print
NR;}if(NR>1){if(($",perm_column_ids[1],"<0)||($",perm_column_ids[1],"!~ '/^[0-9.eE+'-
']*$/')||($",perm_column_ids[1],">1)")
                            if(length(perm_column_ids)>1){
                                for(i in 2:length(perm_column_ids))
```

```
command=paste0(command,"||($",perm_column_ids[i],"<0)||($",perm_column_ids[i],"!~ '/^[0-9.e+'-
']*$/')||($",perm_column_ids[i],">1) ")
                            }
                            command=paste0(command,"){flag_bad_p=1; exit 0;}}} END{print NR;
print flag_bad_p}' ",perm_data," > parsing_perm_",output_name,".txt\n")
                            system(command)
                        }else{
                            error_perm_data[1]="ERROR: Invalid argument perm_data\n"
                        }
                    }

                    perm_parsing_name<-paste0("parsing_perm_",output_name,".txt")
                    parsing=read.table(perm_parsing_name,header=FALSE)
                    parsing_lines=2+rownames_flag_perm
                    if(nrow(parsing)>parsing_lines)
                    {
                        error_perm_data[1]="ERROR: Invalid argument perm_data: bad parsing.\n"
                        return(0)
                    }else
                    {
                        perm_tests=(parsing[nrow(parsing)-1,1]-1)*length(buffer)
                        processed_perm_tests=perm_tests

                        if(parsing[nrow(parsing),1]==1)
                        {error_perm_data[1]="ERROR: Invalid p-values at perm_data file (non-
numeric and/or out of range).\n"}

                        rm(parsing)
                    }
                }
            } else if (is.data.frame(perm_data)==TRUE){
                perm_data_is_file=0
                #Check if all entries in perm_column_id argument match columns in perm_data
dataframe
                if(missing(perm_column_ids)){
                    error_perm_column_ids[1]="ERROR: Absent argument perm_column_ids.\n"
                }else if(typeof(perm_column_ids)!="character" &
(typeof(perm_column_ids)!="double")){
                    error_perm_column_ids[1]="ERROR: Invalid argument perm_column_ids: must be
a vector of valid column identificators (column names, integer ordinal indexes or \"all\")\n"
                }else if(typeof(perm_column_ids)=="character")
                {
                    if(perm_column_ids=="all")
                    {
                        buffer=c(1:ncol(perm_data))
                    }else{
                        buffer=which(colnames(perm_data) %in% perm_column_ids)
                        not_found=which(!(perm_column_ids %in% colnames(perm_data)))
                        if(length(buffer)==0 | length(not_found)>0)
                        error_perm_column_id[1]="ERROR: Invalid entries at perm_column_ids:
names of columns introduced not found in perm_data.\n"
                    }

                }else if(sum(perm_column_ids%%1)!=0){
                    error_perm_column_id[1]="ERROR: Invalid argument perm_column_ids: must be a
vector of valid column identificators (column names, integer ordinal indexes or \"all\")\n"
                }else
                {
                    valid_set=which((perm_column_ids>0)&(perm_column_ids<(ncol(perm_data)+1)))
                    buffer=perm_column_ids[valid_set]
                    if(length(valid_set)<length(perm_column_ids))
                    error_perm_column_id[1]="ERROR: Invalid entries at perm_column_ids:out of
range column ordinals introduced.\n"
                }
                perm_column_ids=buffer

                failed_perm_entries=length(which(is.na(perm_data[,perm_column_ids]) |
(perm_data[,perm_column_ids]<0) | (perm_data[,perm_column_ids]>1)))
```

```
                if(failed_perm_entries>0) {error_perm_data[1]="ERROR: Invalid p-values at full
data file (non-numeric and/or out of range).\n"}

                if(nchar(error_perm_column_id[1])==0 & nchar(error_perm_data[1])==0)
                {
                    #when a df is provided all data is sampled:
                    #load only columns of interest in perm_data into a vector
                    perm_data=unlist(perm_data[,perm_column_ids])
                    perm_tests=length(perm_data)
                    processed_perm_tests=perm_tests
                    perm_data=perm_data[order(as.double(perm_data))]
                    perm_data<-as.double(as.character(perm_data))
                }
            }
            check_error_vector(error_perm_data,flag_error)
            check_error_vector(error_perm_column_id,flag_error)
        }

        ##SUMMARY: 5 Check alt_methods input format: possible warning: bad format, none or some
of the entries do not match one of the possible values: alt_methods set to the subset of valid
entries, if any.
        {
            warning_alt_methods=vector(mode = "character", length = 1)
            valid_alt_methods=c("Fdr_ST","Fndr_ST","Fdr_BH_unif","Fdr_BH_perm","Fdr_MV","BF")
            if(missing(alt_methods)){alt_methods="none"}
            set_alt_methods=which(alt_methods %in% valid_alt_methods)
            if(typeof(alt_methods)!="character")
            {
                alt_methods="none"
                warning_alt_methods[1]="WARNING: Invalid argument alt_methods: if any, it must
be a vector containing some of the alternative methods ids:
'Fdr_ST','Fndr_ST','Fdr_BH_unif','Fdr_BH_perm','Fdr_MV','BF'; no alternative fdrs estimator
will be computed.\n"
            }else if(length(alt_methods)>length(set_alt_methods)){
                if(length(set_alt_methods)==0){
                    alt_methods="none"
                    warning_alt_methods[1]="WARNING: Invalid entries at alt_methods:
alt_methods set to 'none'\n"
                }else{
                    alt_methods=alt_methods[set_alt_methods]
                    warning_alt_methods[1]="WARNING: Invalid entries at alt_methods;
alt_methods set to:"
                    for(i in 1:length(alt_methods))
                    warning_alt_methods[1]=paste0(warning_alt_methods[1],"'",alt_methods[i],"'
")
                    warning_alt_methods[1]=paste0(warning_alt_methods[1],".\n")
                }
            }
            if(nchar(warning_alt_methods)>0) cat(warning_alt_methods)
        }
        ##SUMMARY: 6 Check pi_o_knot and pi_o_knots_range formats: possible warnings:
        ## pi_o_knots it's not a single integer value: will be set to 0 then (to be chose
authomatically)
        ##And or pi_o_knots_range is not a vector of two numbers between 0 and 1: set to zero
then (to be chose authomatically)

        warning_pi_o_knots=vector(mode = "character", length = 1)

        if(missing(pi_o_knots)){pi_o_knots=0}else{
            if(!is.numeric(pi_o_knots)|length(pi_o_knots)!=1|pi_o_knots%%1!=0)
            {
                warning_pi_o_knots[1]=paste0("WARNING: invalid argument pi_o_knots: must be a
single integer value. The number of knots will be calculated automatically\n")
                pi_o_knots=0
            }else if(pi_o_knots<2|pi_o_knots>processed_tests/10)
            {
                warning_pi_o_knots[1]=paste0("WARNING: invalid argument pi_o_knots (out of
range): The number of knots will be calculated automatically\n")
                pi_o_knots=0
            }
```

```
        }


        if(!missing(pi_o_knots_range)){
            if(!is.numeric(pi_o_knots_range)| length(pi_o_knots_range)!=2 |
length(which(pi_o_knots_range<0|pi_o_knots_range>1))>0){
                if(length(warning_pi_o_knots)>1)
                {
                    warning_pi_o_knots[1]="WARNING: Invalid pi_o_knot and pi_o_knots_range
arguments: the number and sizes of knots for pi_o fit will be chosen automatically.\n"
                }else{
                    warning_pi_o_knots[1]="WARNING: Invalid pi_o_knots_range argument: the
allowed knot range for pi_o fit will be chosen automatically.\n"}
                pi_o_knots_range=c(0,0)
            }
        }else{pi_o_knots_range=c(0,0)}
        if(nchar(warning_pi_o_knots)>0) cat(warning_pi_o_knots)

        warning_pi_o_knots=vector(mode = "character", length = 1)
        flag=0
        if(missing(pi_o_knots)){pi_o_knots=0}else{
            if(!is.numeric(pi_o_knots)|length(pi_o_knots)!=1|pi_o_knots%%1!=0)
            {
                warning_pi_o_knots[1]=paste0("WARNING: invalid argument pi_o_knots: must be a
single integer value. The number of knots will be calculated automatically\n")
                pi_o_knots=0
            }else if(pi_o_knots<3|pi_o_knots>processed_tests/100)
            {
                warning_pi_o_knots[1]=paste0("WARNING: invalid argument pi_o_knots (out of
range): The number of knots will be calculated automatically\n")
                pi_o_knots=0
            }
            if(nchar(warning_pi_o_knots)>0) cat(warning_pi_o_knots)
        }

        ##SUMMARY: 7 Check pi_o_iters input format: possible warning: it's not a single numeric
value, or it is not integer. In that case, put to 10000.
        {
            warning_pi_o_iters=vector(mode = "character", length = 1)
            if(!is.numeric(pi_o_iters)|length(pi_o_iters)!=1|pi_o_iters%%1!=0)
            {
                warning_pi_o_iters[1]=paste0("WARNING: invalid argument pi_o_iters: must be a
single integer value. Set to 10000\n")
                pi_o_iters=10000
            }
            if(nchar(warning_pi_o_iters)>0) cat(warning_pi_o_iters)
        }
        ##SUMMARY: 8 Check p_star_exhaustive_sweep input format: possible warning: it's not a
single logical value: will be set to TRUE then.
        {
            warning_p_star_exhaustive_sweep=vector(mode = "character", length = 1)
            if(typeof(p_star_exhaustive_sweep)!="logical"| length(p_star_exhaustive_sweep)!=1)
            {
                warning_p_star_exhaustive_sweep[1]=paste0("WARNING: invalid argument
p_star_exhaustive_sweep: must be a logical variable, set to FALSE.\n")
                p_star_exhaustive_sweep=FALSE
            }
            if(nchar(warning_p_star_exhaustive_sweep)>0) cat(warning_p_star_exhaustive_sweep)
        }
        ##SUMMARY: 9 Check plot input format: possible warning: it's not a single logical
value: will be set to TRUE then.
        {
            warning_plot=vector(mode = "character", length = 1)
            if(typeof(plot)!="logical"| length(plot)!=1)
            {
                warning_plot[1]=paste0("WARNING: invalid argument plot: must be a logical
variable, set to TRUE.\n")
                plot=TRUE
            }
            if(nchar(warning_plot)>0) cat(warning_plot)
```

```
        }
        ##SUMMARY: 10 Check significance_threshold input format: possible warning: it's not a
single numeric value value or it is not within (0,1]. Set to 0.05 then
        {
            warning_significance_threshold=vector(mode = "character", length = 1)
            if(!is.numeric(significance_threshold)|length(significance_threshold)!=1)
            {
                warning_significance_threshold[1]=paste0("ERROR: invalid argument
significance_threshold: must be a single numeric value in (0,1]. Set to 0.05\n")
                significance_threshold=0.05
            }else if(significance_threshold<=0 |significance_threshold>1){
                warning_significance_threshold[1]=paste0("ERROR: invalid argument
significance_threshold: out of range: it must be a numeric value in (0,1]. Set to 0.05\n")
                significance_threshold=0.05
            }
            if(nchar(warning_significance_threshold)>0) cat(warning_significance_threshold)
        }
        ##SUMMARY: 11 Check report_threshold input format: possible warning: it's not a single
numeric value value or it is not within (0,1]. Set to 1 then.
        {
            warning_report_threshold=vector(mode = "character", length = 1)
            if(!is.numeric(report_threshold)|length(report_threshold)!=1)
            {
                warning_report_threshold[1]=paste0("ERROR: invalid argument report_threshold:
must be a single numeric value in (0,1]. Set to 1\n")
                report_threshold=1
            }else if(report_threshold<=0 |report_threshold>1){
                warning_report_threshold[1]=paste0("ERROR: invalid argument report_threshold:
out of range: it must be a numeric value in (0,1]. Set to 1\n")
                report_threshold=1
            }
            if(nchar(warning_report_threshold)>0) cat(warning_report_threshold)
        }
        ##SUMMARY: 12 Check print_messages input format: possible warning: it's not a single
logical value: will be set to TRUE then.
        {
            warning_print_messages=vector(mode = "character", length = 1)
            if(typeof(print_messages)!="logical"| length(print_messages)!=1)
            {
                warning_output_name[1]=paste0("WARNING: invalid argument print_messages: must
be a logical variable, set to TRUE.\n")
                print_messages=TRUE
            }
            if(nchar(warning_print_messages)>0) cat(warning_print_messages)
        }

        if(exists("flag_error")) return(0)
        reference_size=min(processed_tests,100000)

        #Other warnings:
        warning_full_size=vector(mode = "character", length = 1)
        warning_perm_size=vector(mode = "character", length = 1)

        ##There are less than 1000 tests (pi_o estimation loses roboustness, makes little sense
to run the algorithm on files)
        if(typeof(full_data)=="character")
        {
            if(processed_tests<1000)
            {
                if(plot==FALSE)
                {
                    plot=TRUE
                    warning_full_size[1]=paste0("WARNING: few tests, you might want to load
full_data from data.frame for non-interpolated fdr calculations.\nWARNING: pi_o stimations are
more robust for larger number of tests. Check distributions plots. (plot set to TRUE).\n")
                }else{
                    warning_full_size[1]=paste0("WARNING: few tests, you might want to load
full_data from data.frame for non-interpolated fdr calculations.\nWARNING: pi_o stimations are
more robust for larger number of tests. Check distributions plots.\n")
                }
```

```
            }else if(processed_tests<100000){
                {
                    warning_full_size[1]=paste0("WARNING: moderate number of tests: you might
want to load full_data from data.frame for non-interpolated fdr calculations.\n")
                }
            }
        }else if(processed_tests<1000)
        {
            if(plot==FALSE)
            {
                plot=TRUE
                warning_full_size[1]=paste0("WARNING: pi_o stimations are more robust for
larger number of tests. Check distributions plots (plot set to TRUE).\n")
            }else{
                warning_full_size[1]=paste0("WARNING: few tests, pi_o stimations are more
robust for larger number of tests. Check distributions plots.\n")
            }
        }
        ##There are less than 10 permutations of the true data (might not be enough:
fdr_Bootstrap recommended)
        if(processed_perm_tests<10*processed_tests)
        {
            warning_perm_size[1]=paste0("WARNING: too few permutations: typically result in
poor fdr resolution. You might want to run fdr_Bootstrap.\n")
        }
        if(nchar(warning_full_size)>0) cat(warning_full_size)
        if(nchar(warning_perm_size)>0) cat(warning_perm_size)

    }
    if(print_messages)
    {
        cat("\n_____________________________________STEP 2: Obtain full and permuted statistics
distibutions_____________________\n")
    }

    if(full_data_is_file)
    {
        grid_unif=vector(mode = "double", length = reference_size)
        grid_log=vector(mode = "double", length = reference_size)
        F_unif=vector(mode = "double", length = reference_size)
        f_unif=vector(mode = "double", length = reference_size)
        F_log=vector(mode = "double", length = reference_size)
        f_log=vector(mode = "double", length = reference_size)

        if(print_messages){cat("\nLoading actual tests distributions...\n")}
        grids_full=.C("get_grids_full",
        file=full_data,
        columns=as.integer(columns_full_data),
        full_tests=as.double(full_tests),
        rows=as.double(processed_tests),
        stat_column=as.integer(full_column_id),
        grid_size=as.integer(reference_size),
        grid_unif=grid_unif,
        grid_log=grid_log,
        F_log=F_log,
        f_log=f_log,
        F_unif=F_unif,
        f_unif=f_unif,
        min_stat=min_stat,
        max_stat=max_stat,
        zero_count=as.integer(0),
        print_messages=print_messages,
        rownames_flag=as.integer(rownames_flag_full))

        grid_unif<-grids_full$grid_unif
        grid_log<-grids_full$grid_log
        f_log<-grids_full$f_log
        F_log<-grids_full$F_log
        f_unif<-grids_full$f_unif
        F_unif<-grids_full$F_unif
```

```
    zero_count_full<-grids_full$zero_count
    rm(grids_full)

    if(flag_debug_august==TRUE){
        df=data.frame(grid_unif,f_unif,F_unif,grid_log,f_log,F_log)
        write.table(df,paste0(output_name,"/file_full_dist.txt"))
    }

}else{

    full_data[,full_column_id]=as.double(as.matrix(full_data[,full_column_id]))
    if(length(full_data)>1){full_data=full_data[order(full_data[,full_column_id]),]}
    else{full_data=as.matrix(full_data[order(full_data[,full_column_id]),])}
    zero_count_full=length(which(full_data[,full_column_id]==0))
    data=full_data[1:processed_tests,full_column_id]
    grid=unique(data)
    F=vector(mode = "double", length = length(grid))
    f=vector(mode = "double", length = length(grid))
    min_stat=grid[1]
    max_stat=grid[length(grid)]

    dist=.C("get_distribution_df",
    data=as.double(data),
    data_size=as.integer(full_tests),
    processed_data_size=as.integer(processed_tests),
    grid=as.double(grid),
    grid_size=as.integer(length(grid)),
    f=f,
    F=F)

    f<-dist$f
    F<-dist$F
    rm(dist)

    if(flag_debug_august==TRUE){
        df=data.frame(grid,f,F)
        write.table(df,paste0(output_name,"/df_full_dist.txt"))

    }
}

if(perm_data_is_file)
{
    if(full_data_is_file)
    {
        F_o_unif=vector(mode = "double", length = reference_size)
        f_o_unif=vector(mode = "double", length = reference_size)
        F_o_log=vector(mode = "double", length = reference_size)
        f_o_log=vector(mode = "double", length = reference_size)

        if(print_messages){cat("\nLoading permutation tests distributions...\n")}

        grids_perm=.C("get_grids_perm",
        file=perm_data,
        columns=as.integer(columns_perm_data),
        perm_tests=as.double(perm_tests),
        rows=as.double(processed_perm_tests),
        stat_columns=length(perm_column_ids),
        stat_columns_array=as.integer(perm_column_ids),
        grid_size=as.integer(reference_size),
        grid_unif=grid_unif,
        grid_log=grid_log,
        F_o_log=F_o_log,
        f_o_log=f_o_log,
        F_o_unif=F_o_unif,
        f_o_unif=f_o_unif,
        zero_count=as.integer(0),
        print_messages=print_messages,
        rownames_flag=as.integer(rownames_flag_perm))
```

```
        f_o_log<-grids_perm$f_o_log
        F_o_log<-grids_perm$F_o_log
        f_o_unif<-grids_perm$f_o_unif
        F_o_unif<-grids_perm$F_o_unif
        zero_count_perm<-grids_perm$zero_count
        rm(grids_perm)

        if(flag_debug_august==TRUE){
            df=data.frame(grid_unif,f_o_unif,F_o_unif,grid_log,f_o_log,F_o_log)
            write.table(df,paste0(output_name,"/file_file_perm_dist.txt"))
        }
    }else{
        F_o=vector(mode = "double", length = length(grid))
        f_o=vector(mode = "double", length = length(grid))

        grids_perm=.C("get_grids_perm_from_data",
        file=perm_data,
        columns=as.integer(columns_perm_data),
        perm_tests=as.double(perm_tests),
        rows=as.double(processed_perm_tests),
        stat_columns=length(perm_column_ids),
        stat_columns_array=as.integer(perm_column_ids),
        grid_size=as.integer(length(grid)),
        grid=grid,
        F_o=F_o,
        f_o=f_o,
        zero_count=as.integer(0),
        print_messages=print_messages,
        rownames_flag=as.integer(rownames_flag_perm))

        f_o<-grids_perm$f_o
        F_o<-grids_perm$F_o
        zero_count_perm<-grids_perm$zero_count
        rm(grids_perm)

        if(flag_debug_august==TRUE){
            df=data.frame(grid,f_o,F_o)
            write.table(df,paste0(output_name,"/df_file_perm_dist.txt"))
        }
    }
}else{
    perm_data=perm_data[order(perm_data)]
    zero_count_perm=length(which(perm_data==0))

    if(full_data_is_file)
    {
        F_o_log=vector(mode = "double", length = reference_size)
        f_o_log=vector(mode = "double", length = reference_size)
        F_o_unif=vector(mode = "double", length = reference_size)
        f_o_unif=vector(mode = "double", length = reference_size)

        dist_perm=.C("get_grids_perm_df",
        perm_tests_array=as.double(as.matrix(perm_data)),
        perm_tests=as.integer(perm_tests),
        processed_perm_tests=as.integer(processed_perm_tests),
        grid_size=as.integer(reference_size),
        grid_log=grid_log,
        grid_unif=grid_unif,
        F_o_log=F_o_log,
        f_o_log=f_o_log,
        F_o_unif=F_o_unif,
        f_o_unif=f_o_unif,
        zero_count=as.integer(0))

        F_o_log=as.double(dist_perm$F_o_log)
        F_o_unif=as.double(dist_perm$F_o_unif)
        f_o_log=as.double(dist_perm$f_o_log)
        f_o_unif=as.double(dist_perm$f_o_unif)
        zero_count_perm<-dist_perm$zero_count
        rm(dist_perm)
```

```
            if(flag_debug_august==TRUE){
                df=data.frame(grid_unif,f_o_unif,F_o_unif,grid_log,f_o_log,F_o_log)
                write.table(df,paste0(output_name,"/file_df_perm_dist.txt"))
            }

        }else{
            F_o=vector(mode = "double", length = length(grid))
            f_o=vector(mode = "double", length = length(grid))

            dist_perm=.C("get_distribution_df",
            data=as.double(perm_data),
            data_size=as.integer(perm_tests),
            processed_data_size=as.integer(processed_perm_tests),
            grid=as.double(grid),
            grid_size=as.integer(length(grid)),
            f=f_o,
            F=F_o)

            f_o<-dist_perm$f
            F_o<-dist_perm$F
            rm(dist_perm)

            if(flag_debug_august==TRUE){
                df=data.frame(grid,f_o,F_o)
                write.table(df,paste0(output_name,"/df_df_perm_dist.txt"))
            }
        }
    }

    if(print_messages)
    {
        cat("\n_____________________________________STEP 3: Obtain fraction of true negatives
pi_o_____________________________________\n\n")
    }

    if(full_data_is_file)
    {
        F_g=vector(mode = "double", length = reference_size)
        F_o_g=vector(mode = "double", length = reference_size)
        f_g=vector(mode = "double", length = reference_size)
        f_o_g=vector(mode = "double", length = reference_size)
        fitting_goodness=vector(mode = "double", length = 2)
        flag_grenander_choice=vector(mode = "integer", length = 2)

        grid=grid_unif
        F=F_unif
        F_o=F_o_unif
        f=f_unif
        f_o=f_o_unif

        if(flag_debug_august){
            df=data.frame(grid,F,F_o,f,f_o)
            write.table(df,"grenander_input.txt")
        }

        if(print_messages){   cat("Estimating scaling point p*...\n")}

        grenander=.C("estimate_p_star",
        grid=grid,
        grid_size=as.integer(reference_size),
        F=F,
        F_o=F_o,
        f=f,
        f_o=f_o,
        F_g=F_g,
        F_o_g=F_o_g,
        f_g=f_g,
        f_o_g=f_o_g,
        index_p_star=as.integer(1),
```

```
    index_p_prime=as.integer(c(-1,-1)),
    fitting_goodness=fitting_goodness,
    flag_grenander_choice=flag_grenander_choice)


    F_g=grenander$F_g
    F_o_g=grenander$F_o_g
    f_g=grenander$f_g
    f_o_g=grenander$f_o_g
    index_p_star_unif=grenander$index_p_star
    index_p_prime=grenander$index_p_prime
    fitting_goodness=grenander$fitting_goodness
    flag_grenander_choice=grenander$flag_grenander_choice
    rm(grenander)

    p_star=grid_unif[index_p_star_unif]
    log_max_ratio=log10(max_stat/min_stat)
    delta_log=log_max_ratio/(reference_size-1)
    index_p_star_log=(as.integer)(log10(p_star/min_stat)/delta_log+1)

    #if(flag_debug_august)
    #{
    #      df=data.frame(grid,F_g,f_g,F_o_g,f_o_g)
    #      write.table(df,paste0(output_name,"/grenander.txt"))
    #}
}else{
    F_g=vector(mode = "double", length = length(grid))
    F_o_g=vector(mode = "double", length = length(grid))
    f_g=vector(mode = "double", length = length(grid))
    f_o_g=vector(mode = "double", length = length(grid))
    fitting_goodness=vector(mode = "double", length = 2)
    flag_grenander_choice=vector(mode = "integer", length = 2)
    if(print_messages){cat("Estimating scaling point p*...\n")}
    grenander=.C("estimate_p_star",
    grid=grid,
    grid_size=as.integer(length(grid)),
    F=F,
    F_o=F_o,
    f=f,
    f_o=f_o,
    F_g=F_g,
    F_o_g=F_o_g,
    f_g=f_g,
    f_o_g=f_o_g,
    index_p_star=as.integer(1),
    index_p_prime=as.integer(c(-1,-1)),
    fitting_goodness=fitting_goodness,
    flag_grenander_choice=flag_grenander_choice)

    F_g=grenander$F_g
    F_o_g=grenander$F_o_g
    f_g=grenander$f_g
    f_o_g=grenander$f_o_g
    index_p_star=grenander$index_p_star
    index_p_prime=grenander$index_p_prime
    fitting_goodness=grenander$fitting_goodness
    flag_grenander_choice=grenander$flag_grenander_choice
    rm(grenander)

    p_star=grid[index_p_star]
}

if(full_data_is_file){
    grid<-grid_unif
    F<-F_unif
    F_o<-F_o_unif
    index_p_star=index_p_star_unif
}

#cat("debug\n");
```

```
    if(flag_debug_august==TRUE)
    {
        df=data.frame(grid,F_g,f_g,F_o_g,f_o_g)
        write.table(df,paste0(output_name,"/grenander.txt"))
    }

    #df=data.frame(c(1,2),fitting_goodness)
    write.table(fitting_goodness,paste0(output_name,"/fitting_goodness.txt"))


if(((fitting_goodness[1]<0.9)||(fitting_goodness[2]<0.9))&&(p_star_exhaustive_sweep==FALSE)){
        cat("WARNING: \n")
        if(fitting_goodness[1]<0.9){cat("   True data distribution fitting goodness is
",fitting_goodness[1],"\n")}
        if(fitting_goodness[2]<0.9){cat("   Perm. data distribution fitting goodness is
",fitting_goodness[2],"\n")}

        cat("You might want to use p_star_exhaustive_sweep=TRUE\n")
    }


    if(p_star_exhaustive_sweep==FALSE)
    {
        if(index_p_star>length(F)/2)
        {
            pi_hat=vector(mode = "double", length = index_p_star)
            pi_hat_fitted=vector(mode = "double", length = index_p_star)
            #Introducir F_g?
            pi_hat=(F[index_p_star]-F[1:index_p_star])/(F_o[index_p_star]-F_o[1:index_p_star])
            end_plateau=which(F_o==F_o[index_p_star])
            end_plateau=end_plateau[which(end_plateau<=index_p_star)]
            pi_hat[end_plateau]=pi_hat[end_plateau[1]-1]
            finite_set=which(is.finite(as.numeric(as.character(pi_hat))))

            if(flag_debug_august==TRUE){
                df=data.frame(x=grid[finite_set],y=pi_hat[finite_set])
                write.table(df,paste0(output_name,"/pi_o_fit_input.txt"))
            }

f_hat=pi_o_fit(grid[finite_set],pi_hat[finite_set],pi_o_knots,pi_o_iters,pi_o_knots_range,trend
="decrease",full_number=F[index_p_star],perm_number=F_o[index_p_star])
            pi_hat_fitted<-f_hat$fitted
            if(flag_debug_august==TRUE){
                df=data.frame(x=grid[finite_set],y=pi_hat[finite_set],yfit=pi_hat_fitted)
                write.table(df,paste0(output_name,"/sweep_debug.txt"))
                p=ggplot(df)+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=yfit))
                pdf(paste0("NOSweep_left_fit.pdf"))
                print(p)
                dev.off()
            }
            pi_o_scaling=pi_hat_fitted[length(pi_hat_fitted)]
            pi_o_scaling=min(pi_o_scaling,1)
            pi_o_scaling=max(pi_o_scaling,0)
            rm(f_hat)
        }else{
            pi_hat=vector(mode = "double", length = (length(F)-index_p_star+1))
            pi_hat_fitted=vector(mode = "double", length = (length(F)-index_p_star+1))

            pi_hat=(F[index_p_star]-F[index_p_star:length(F)])/(F_o[index_p_star]-
F_o[index_p_star:length(F_o)])
            begin_plateau=which(F_o==F_o[index_p_star])
            begin_plateau=begin_plateau[which(begin_plateau>=index_p_star)]
            pi_hat[begin_plateau]=pi_hat[begin_plateau[length(begin_plateau)]+1]
            finite_set=which(is.finite(as.numeric(as.character(pi_hat))))
            finite_set_grid=finite_set+index_p_star-1

f_hat=pi_o_fit(grid[finite_set_grid],pi_hat[finite_set],pi_o_knots,pi_o_iters,pi_o_knots_range,
trend="increase",full_number=F[index_p_star],perm_number=F_o[index_p_star])

            pi_hat_fitted<-f_hat$fitted
```

```
            if(flag_debug_august==TRUE){
                df=data.frame(x=grid[finite_set_grid],y=pi_hat[finite_set],yfit=pi_hat_fitted)
                #write.table(df,paste0(output_name,"/sweep_debug.txt"))
                p=ggplot(df)+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=yfit))
                pdf(paste0("NOSweep_right_fit.pdf"))
                print(p)
                dev.off()
            }
            pi_o_scaling=pi_hat_fitted[1]
            pi_o_scaling=min(pi_o_scaling,1)
            pi_o_scaling=max(pi_o_scaling,0)
            rm(f_hat)
        }
    }else{
        reduction_factor=(length(F))/100

        if(full_data_is_file){
            p_star_indexes_array=seq(1,(length(F)),as.integer(reduction_factor))
            p_star_indexes_array=c(1,p_star_indexes_array,(length(F)))
            p_star_indexes_array=unique(p_star_indexes_array)
        }else{
            v_aux=vector(mode = "double", length = as.integer(length(grid)/reduction_factor+1))
            delta=as.double((max_stat-min_stat)/(length(grid))*reduction_factor)
            grid_aux=seq(min_stat,max_stat,delta)
            index_aux=1
            #print(head(grid_aux))
            for(i in 2:length(v_aux))
            {
                while(grid_aux[i]>grid[index_aux]) {index_aux=index_aux+1}

                v_aux[i]=index_aux

            }
            v_aux=c(v_aux[2:length(v_aux)],length(grid))
            p_star_indexes_array=unique(v_aux)
        }

        #print(head(p_star_indexes_array))

        pi_o_scalingloop=p_star_indexes_array
        pi_o_loop=p_star_indexes_array

        min=2
        for(i in 1:length(p_star_indexes_array))
        {
            if(print_messages)
            {
                cat("Iteration ",i," over ",length(p_star_indexes_array)," of pi_o_fit loop\n")
            }
            if(p_star_indexes_array[i]>length(F)/2)
            {
                pi_hat_aux=vector(mode = "double", length = p_star_indexes_array[i])
                pi_hat_fitted_aux=vector(mode = "double", length = p_star_indexes_array[i])

                pi_hat_aux=(F[p_star_indexes_array[i]]-
F[1:p_star_indexes_array[i]])/(F_o[p_star_indexes_array[i]]-F_o[1:p_star_indexes_array[i]])
                end_plateau=which(F_o==F_o[p_star_indexes_array[i]])

                end_plateau=end_plateau[which(end_plateau<=p_star_indexes_array[i])]
                pi_hat_aux[end_plateau]=pi_hat_aux[end_plateau[1]-1]
                finite_set=which(is.finite(as.numeric(as.character(pi_hat_aux))))


f_hat=pi_o_fit(grid[finite_set],pi_hat_aux[finite_set],pi_o_knots,pi_o_iters,pi_o_knots_range,t
rend="none_left",full_number=F_g[p_star_indexes_array[i]],perm_number=F_o_g[p_star_indexes_arra
y[i]],silent=TRUE)

                pi_hat_fitted_aux<-f_hat$fitted

                if(flag_debug_august==TRUE){
```

```
df=data.frame(x=grid[finite_set],y=pi_hat_aux[finite_set],yfit=pi_hat_fitted_aux)
                    #write.table(df,paste0(output_name,"/sweep_debug.txt"))
                    p=ggplot(df)+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=yfit))
                    pdf(paste0("Sweep_left_fit_",i,".pdf"))
                    print(p)
                    dev.off()
                }
                pi_o_scalingloop[i]=pi_hat_fitted_aux[length(pi_hat_fitted_aux)]
                pi_o_scalingloop[i]=min(pi_o_scalingloop[i],1)
                pi_o_scalingloop[i]=max(pi_o_scalingloop[i],0)

                #Introducir F_g?
                pi_o_loop[i]=1+pi_o_scalingloop[i]*F_o[p_star_indexes_array[i]]-
F[p_star_indexes_array[i]]
                pi_o_loop[i]=min(pi_o_loop[i],1)
                pi_o_loop[i]=max(pi_o_loop[i],0)

                rm(f_hat)
            }else{
                pi_hat_aux=vector(mode = "double", length = (length(grid)-
p_star_indexes_array[i]+1))
                pi_hat_fitted_aux=vector(mode = "double", length = (length(grid)-
p_star_indexes_array[i]+1))

                pi_hat_aux=(F[p_star_indexes_array[i]]-
F[p_star_indexes_array[i]:length(F)])/(F_o[p_star_indexes_array[i]]-
F_o[p_star_indexes_array[i]:length(F_o)])
                begin_plateau=which(F_o==F_o[p_star_indexes_array[i]])

                begin_plateau=begin_plateau[which(begin_plateau>=p_star_indexes_array[i])]
                pi_hat_aux[begin_plateau]=pi_hat_aux[begin_plateau[length(begin_plateau)]+1]
                finite_set=which(is.finite(as.numeric(as.character(pi_hat_aux))))
                finite_set_grid=finite_set+p_star_indexes_array[i]-1

                #cat("Fit\n")

f_hat=pi_o_fit(grid[finite_set_grid],pi_hat_aux[finite_set],pi_o_knots,pi_o_iters,pi_o_knots_ra
nge,trend="none_right",full_number=F_g[p_star_indexes_array[i]],perm_number=F_o_g[p_star_indexe
s_array[i]],silent=TRUE)
                #cat("End fit\n")

                pi_hat_fitted_aux<-f_hat$fitted
                pi_o_scalingloop[i]=pi_hat_fitted_aux[1]
                pi_o_scalingloop[i]=min(pi_o_scalingloop[i],1)
                pi_o_scalingloop[i]=max(pi_o_scalingloop[i],0)

                if(flag_debug_august==TRUE){

df=data.frame(x=grid[finite_set_grid],y=pi_hat_aux[finite_set],yfit=pi_hat_fitted_aux)
                    #write.table(df,paste0(output_name,"/sweep_debug.txt"))
                    p=ggplot(df)+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=yfit))
                    pdf(paste0("Sweep_right_fit_",i,".pdf"))
                    print(p)
                    dev.off()
                }

                pi_o_loop[i]=1+pi_o_scalingloop[i]*F_o[p_star_indexes_array[i]]-
F[p_star_indexes_array[i]]
                pi_o_loop[i]=min(pi_o_loop[i],1)
                pi_o_loop[i]=max(pi_o_loop[i],0)

                rm(f_hat)
            }
            print(pi_o_scalingloop[i])
            if(pi_o_scalingloop[i]<min)
            {
                if(p_star_indexes_array[i]<=length(F)/2)
                {
                    finite_set_grid_min<-finite_set_grid
```

```
                }
                finite_set_min<-finite_set
                pi_hat_min<-pi_hat_aux
                pi_hat_fitted_min<-pi_hat_fitted_aux
                min=pi_o_scalingloop[i]
            }
        }
        if(print_messages)
        {
            cat("\n")
        }
        if(p_star_indexes_array[which.min(pi_o_scalingloop)]<=length(F)/2)
        {
            finite_set_grid<-finite_set_grid_min
            rm(finite_set_grid_min)
        }

        finite_set<-finite_set_min
        rm(finite_set_min)
        pi_hat<-pi_hat_min[finite_set]
        rm(pi_hat_min)
        pi_hat_fitted<-pi_hat_fitted_min
        rm(pi_hat_fitted_min)

        pi_o_scaling=min(pi_o_scalingloop)


        if(full_data_is_file){
            index_p_star_unif=p_star_indexes_array[which.min(pi_o_scalingloop)]
            index_p_star=index_p_star_unif

            p_star=grid_unif[index_p_star_unif]
            log_max_ratio=log10(max_stat/min_stat)
            delta_log=log_max_ratio/(reference_size-1)
            index_p_star_log=(as.integer)(log10(p_star/min_stat)/delta_log)
        }else{
            index_p_star=p_star_indexes_array[which.min(pi_o_scalingloop)]
        }

        p_star=grid[index_p_star]
        ##DECLARE pi_o_plot df to be plotted later on

pi_o_plot=data.frame(statistic=grid[p_star_indexes_array],variable="pi_o_scalingloop",value=pi_
o_scalingloop)
    }

    if(print_messages)
    {
        cat("\n_____________________________________STEP 4: Obtain false discovery rates: SAB
method_____________________________\n")
    }

    if(full_data_is_file)
    {
        grid=grid_log
        F=F_log
        F_o=F_o_log
        index_p_star=index_p_star_log
    }

    pi_o=1+pi_o_scaling*F_o[index_p_star]-F[index_p_star]
    pi_o=min(pi_o,1)
    pi_o=max(pi_o,0)
    if(pi_o!=0){
        F_null=F_o*pi_o_scaling/pi_o

if(full_data_is_file){F_null[index_p_star:reference_size]=(F_o[index_p_star]*pi_o_scaling+F[ind
ex_p_star:reference_size]-F[index_p_star])/pi_o
        }else{
```

```
F_null[index_p_star:length(grid)]=(F_o[index_p_star]*pi_o_scaling+F[index_p_star:length(grid)]-
F[index_p_star])/pi_o
        }
    }else{
        F_null=rep(0,reference_size)
    }
    Fdr=pi_o*F_null/F
    Fndr=(1-F-pi_o*(1-F_null))/(1-F)
    Fndr[length(Fndr)]=Fndr[length(Fndr)-1]
    if(zero_count_full!=0)
    {
        zero_Fdr=pi_o*zero_count_perm/zero_count_full
    }else{
        zero_Fdr=pi_o
    }

    alt_methods_aux=as.integer(valid_alt_methods %in% alt_methods)

    #### Comprobar que F_null<F###
    null_check=which(F<F_null)
    if(length(null_check)>0){
        cat("WARNING: No se cumple F>F_null en todo el rango\n")
        df=data.frame(grid,F,F_null)
        write.table(df,"F_vs_Fnull.txt")
    }
    ###################

    Fdr_ST=0
    Fndr_ST=0
    Fdr_BH_unif=0
    Fdr_BH_perm=0
    Fdr_MV=0
    BF=0

    if(length(which(alt_methods_aux==1))>0 & print_messages)
    {
        cat("\n_____________________________________STEP 5: Obtain false discovery rates: Alternative
methods_____________________\n\n")
    }

    if(full_data_is_file)
    {
                F=F_unif
                F_o=F_o_unif
                grid=grid_unif
    }

    discard_methods=0
    threshold=0.15
    threshold_2=0.8
    threshold_3=0.5

    if((flag_grenander_choice[1]==2)||(flag_grenander_choice[1]==4)){
                cat("For the full: left ",F[index_p_prime[1]])
                cat(" right",(1-F[index_p_prime[1]]),"\n")
                if((F[index_p_prime[1]]>threshold)&&((1-F[index_p_prime[1]])>threshold)){
                        cv_left=2*abs(f_g[1]-
f_g[index_p_prime[1]])/(f_g[1]+f_g[index_p_prime[1]])
                        cv_right=2*abs(f_g[length(f_g)]-
f_g[index_p_prime[1]])/(f_g[length(f_g)]+f_g[index_p_prime[1]])
                        cat("cvs: ",cv_left," ",cv_right,"\n")
                        if((cv_left>threshold_3)&&(cv_right>threshold_3)){discard_methods=1}

                }
    }
    cat("Discard methods full: ",discard_methods,"\n");

    if(discard_methods==0){
                if((flag_grenander_choice[2]==2)||(flag_grenander_choice[2]==4)){
```

```
                        cat("For the perm: left ",F_o[index_p_prime[2]])
                        cat(" right",(1-F_o[index_p_prime[2]]),"\n")
                        if((F_o[index_p_prime[2]]>threshold)&&((1-
F_o[index_p_prime[2]])>threshold)){
                                cv_left=2*abs(f_o_g[1]-
f_o_g[index_p_prime[2]])/(f_o_g[1]+f_o_g[index_p_prime[2]])
                                cv_right=2*abs(f_o_g[length(f_o_g)]-
f_o_g[index_p_prime[2]])/(f_o_g[length(f_o_g)]+f_o_g[index_p_prime[2]])
                                cat("cvs: ",cv_left," ",cv_right,"\n")

        if((cv_left>threshold_3)&&(cv_right>threshold_3)){discard_methods=1}
                        }
                }
    }
    cat("Discard methods perm: ",discard_methods,"\n");

    #Up-down situation
    if(discard_methods==0){
                print(flag_grenander_choice[1])

        if((flag_grenander_choice[1]==3)||((flag_grenander_choice[1]==4)&&(F[index_p_prime[1]]>
threshold_2))||(((flag_grenander_choice[1]==2)&&((1-F[index_p_prime[1]])>threshold_2)))){
                        if(flag_grenander_choice[1]==3){cv_inc=2*abs(f_g[length(f_g)]-
f_g[1])/(f_g[length(f_g)]+f_g[1])}
                        if(flag_grenander_choice[1]==4){cv_inc=2*abs(f_g[index_p_prime[1]]-
f_g[1])/(f_g[index_p_prime[1]]+f_g[1])}
                        if(flag_grenander_choice[1]==2){cv_inc=2*abs(f_g[length(f_g)]-
f_g[index_p_prime[1]])/(f_g[length(f_g)]+f_g[index_p_prime[1]])}
                        print(cv_inc)
                        if(cv_inc>threshold_3){
                                print(flag_grenander_choice[2])
                                if(flag_grenander_choice[2]==1){discard_methods=1}
                                else{

        if((flag_grenander_choice[2]==3)||((flag_grenander_choice[2]==4)&&(F_o[index_p_prime[2]
]>threshold_2))||(((flag_grenander_choice[2]==2)&&((1-F_o[index_p_prime[2]])>threshold_2)))){

        if(flag_grenander_choice[2]==3){cv_inc=2*abs(f_o_g[length(f_o_g)]-
f_o_g[1])/(f_o_g[length(f_o_g)]+f_o_g[1])}

        if(flag_grenander_choice[2]==4){cv_inc=2*abs(f_o_g[index_p_prime[2]]-
f_o_g[1])/(f_o_g[index_p_prime[2]]+f_o_g[1])}

        if(flag_grenander_choice[2]==2){cv_inc=2*abs(f_o_g[length(f_o_g)]-
f_o_g[index_p_prime[2]])/(f_o_g[length(f_o_g)]+f_o_g[index_p_prime[2]])}

                                        print(cv_inc)
                                        if(cv_inc<threshold_3){discard_methods=1}

                                        }
                                        else{discard_methods=1}
                                }


        #if((flag_grenander_choice[2]==1)||((flag_grenander_choice[2]==2)&&(F_o[index_p_prime[2
]]>threshold_2))||(((flag_grenander_choice[2]==4)&&((1-F_o[index_p_prime[2]])>threshold_2)))){
                                #
        if(flag_grenander_choice[2]==1){cv_dec=2*abs(f_o_g[length(f_o_g)]-
f_o_g[1])/(f_o_g[length(f_o_g)]+f_o_g[1])}
                                #
        if(flag_grenander_choice[2]==2){cv_dec=2*abs(f_o_g[index_p_prime[1]]-
f_o_g[1])/(f_o_g[index_p_prime[1]]+f_o_g[1])}
                                #
        if(flag_grenander_choice[2]==4){cv_dec=2*abs(f_o_g[length(f_o_g)]-
f_o_g[index_p_prime[1]])/(f_o_g[length(f_o_g)]+f_o_g[index_p_prime[1]])}

                                #       if(cv_dec>threshold_3){discard_methods=1}
                                #}
                        }
                }
```

```
    }
    cat("Discard methods up-down: ",discard_methods,"\n");
    print(flag_grenander_choice)

    if(full_data_is_file)
    {
               F=F_log
               F_o=F_o_log
               grid=grid_log
    }

    if(length(which(alt_methods_aux==1))>0)
    {
        if("Fdr_ST" %in% alt_methods)
        {
            if(print_messages){cat("Calculating Storey-Tibshinari FDRs...\n")}

            if(discard_methods==1){
                                cat("Storey-Tibshirani not suitable for these
distributions...\n");
                                alt_methods_aux[1]=0;
                                alt_methods=alt_methods[which(alt_methods!="Fdr_ST")]
                        }else{

                                if(full_data_is_file)
                                {
                                        grid=grid_unif
                                        F=F_unif
                                        F_o=F_o_unif
                                }

                                ##if(flag_debug_august==TRUE){
                                ##      x=read.table("perm_dist.txt",header=TRUE)
                                ##      F_o_new=x$F_o
                                ##      df=data.frame(grid,F_o,F_o_new)
                                ##      write.table(df,paste0(output_name,"/perm_dist.txt"))
                                ##      F_o=F_o_new
                                ##}

                                pi_hat_ST=(1-F)/(1-F_o)
                                end_plateau_ST=which(F_o==1)
                                pi_hat_ST[end_plateau_ST]=pi_hat_ST[end_plateau_ST[1]-1]

        finite_set_ST=which(is.finite(as.numeric(as.character(pi_hat_ST))))


        f_hat_ST=pi_o_fit(grid[finite_set_ST],pi_hat_ST[finite_set_ST],pi_o_knots,pi_o_iters,pi
_o_knots_range,trend="decrease",full_number=1,perm_number=1,silent=TRUE)

                                pi_hat_fitted_ST<-f_hat_ST$fitted
                                pi_o_ST=pi_hat_fitted_ST[length(pi_hat_fitted_ST)]
                                pi_o_ST=min(pi_o_ST,1)
                                pi_o_ST=max(pi_o_ST,0)
                                rm(f_hat_ST)

                                if(flag_debug_august==TRUE){

        df=data.frame(x=grid[finite_set_ST],y=pi_hat_ST[finite_set_ST],yfit=pi_hat_fitted_ST)
                                        #write.table(df,paste0(output_name,"/ST_debug.txt"))

        p=ggplot(df)+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=yfit))
                                        pdf("ST_fit.pdf")
                                        print(p)
                                        dev.off()
                                }

                                if(full_data_is_file)
                                {
                                        grid=grid_log
                                        F=F_log
```

```
                                        F_o=F_o_log
                                }

                                Fdr_ST=pi_o_ST*grid/F
                        }
        }
        if("Fndr_ST" %in% alt_methods)
        {
            if(print_messages){cat("Calculating Storey-Tibshinari FNDRs...\n")}

            if(discard_methods==1){
               cat("Storey-Tibshirani not suitable for these distributions...\n");
                                alt_methods_aux[2]=0;
                                alt_methods=alt_methods[which(alt_methods!="Fndr_ST")]
                        }else{

                                if(!("Fdr_ST" %in% alt_methods))
                                {
                                        if(full_data_is_file)
                                        {
                                                grid=grid_unif
                                                F=F_unif
                                                F_o=F_o_unif
                                        }

                                        pi_hat_ST=(1-F)/(1-F_o)
                                        end_plateau_ST=which(F_o==1)
                                        pi_hat_ST[end_plateau_ST]=pi_hat[end_plateau_ST[1]-1]

        finite_set_ST=which(is.finite(as.numeric(as.character(pi_hat_ST))))

        f_hat_ST=pi_o_fit(grid[finite_set_ST],pi_hat_ST[finite_set_ST],pi_o_knots,pi_o_iters,pi
_o_knots_range,trend="decrease",full_number=1,perm_number=1,silent=TRUE)

                                        pi_hat_fitted_ST<-f_hat_ST$fitted
                                        if(flag_debug_august==TRUE){

        df=data.frame(x=grid[finite_set_ST],y=pi_hat_ST[finite_set_ST],yfit=pi_hat_fitted_ST)

        p=ggplot(df)+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=yfit))
                                                pdf("ST_when_entering_by_fndr_fit.pdf")
                                                print(p)
                                                dev.off()
                                        }
                                        pi_o_ST=pi_hat_fitted_ST[length(pi_hat_fitted_ST)]
                                        pi_o_ST=min(pi_o_ST,1)
                                        pi_o_ST=max(pi_o_ST,0)
                                        rm(f_hat_ST)
                                }

                                if(full_data_is_file)
                                {
                                        grid=grid_log
                                        F=F_log
                                        F_o=F_o_log
                                }

                                Fndr_ST=(1-F-pi_o_ST*(1-grid))/(1-F)
                                Fndr_ST[length(Fndr_ST)]=Fndr_ST[length(Fndr_ST)-1]
                        }
                }
        if("Fdr_BH_unif" %in% alt_methods)
        {
            if(print_messages){cat("Calculating Benjamini-Hochberg FDRs (uniform null)...\n")}
            Fdr_BH_unif=grid/F
        }
        if("Fdr_BH_perm" %in% alt_methods)
        {
            if(print_messages){cat("Calculating Benjamini-Hochberg FDRs (permutations-based
null)...\n")}
```

```
            Fdr_BH_perm=F_o/F
        }
        if("Fdr_MV" %in% alt_methods)
        {
            if(print_messages){cat("Calculating Millstein-Volfson FDRs...\n")}

            if(discard_methods==1){
                                cat("Storey-Tibshirani not suitable for these
distributions...\n");
                                alt_methods_aux[5]=0;
                                alt_methods=alt_methods[which(alt_methods!="Fdr_MV")]
                        }else{

                                pi_hat_MV=(1-F)/(1-F_o)
                                end_plateau_MV=which(F_o==1)
                                pi_hat_MV[end_plateau_MV]=pi_hat_MV[end_plateau_MV[1]-1]

                                if((flag_debug_august==TRUE)&&(full_data_is_file)){
                                        df=data.frame(grid_log,grid_unif,pi_hat_MV)
                                        write.table(df,paste0(output_name,"/MV_fit_input.txt"))
                                }
                                if((flag_debug_august==TRUE)&&(full_data_is_file==0)){
                                        df=data.frame(grid,pi_hat_MV)
                                        write.table(df,paste0(output_name,"/MV_fit_input.txt"))
                                }

                                grid_aux_2=grid[which(grid<1e-50)]
                                if(length(grid_aux_2)>0){
                                        peak=F[length(grid_aux_2)]
                                }else{peak=0}
                                if(peak>0.10) #Si hay un 10% de los datos por debajo de 1e-50
                                {

        pi_o_knots=min(30,as.integer(2+(28/3)*log(length(grid)/10)))
                                        if(grid[1]==0){
                                                grid_aux=grid[2:length(grid)]
                                                pi_hat_MV_aux=pi_hat_MV[2:length(grid)]

        constraint_matrix=as.matrix(data.frame(c(0),c(log10(grid_aux)[1]),c(pi_hat_MV_aux[1])))

        f_hat_MV_aux=pi_o_fit(log10(grid_aux),pi_hat_MV_aux,pi_o_knots,pi_o_iters,pi_o_knots_ra
nge,trend="none",full_number=1,perm_number=1,silent=FALSE,constraint_matrix=constraint_matrix)
                                                pi_hat_fitted_MV_aux<-f_hat_MV_aux$fitted

        pi_hat_fitted_MV=c(pi_hat_MV[1],pi_hat_fitted_MV_aux)
                                                pi_hat_long<-pi_hat_fitted_MV
                                        }else{

        constraint_matrix=as.matrix(data.frame(c(0),c(log10(grid)[1]),c(pi_hat_MV[1])))

        f_hat_MV=pi_o_fit(log10(grid),pi_hat_MV,pi_o_knots,pi_o_iters,pi_o_knots_range,trend="n
one",full_number=1,perm_number=1,silent=FALSE,constraint_matrix=constraint_matrix)
                                                pi_hat_fitted_MV<-f_hat_MV$fitted
                                                pi_hat_long<-pi_hat_fitted_MV
                                        }
                                }else{

        f_hat_MV=pi_o_fit(grid,pi_hat_MV,pi_o_knots,pi_o_iters,pi_o_knots_range,trend="none",fu
ll_number=1,perm_number=1,silent=FALSE)
                                        pi_hat_fitted_MV<-f_hat_MV$fitted
                                        pi_hat_long<-pi_hat_fitted_MV
                                }

                                if(flag_debug_august==TRUE){
                                        df=data.frame(x=grid,y=pi_hat_MV,yfit=pi_hat_long)
                                        write.table(df,paste0(output_name,"/MV_fit_output.txt"))

        p=ggplot(df)+geom_point(aes(x=x,y=y))+geom_line(aes(x=x,y=yfit))

        pp=ggplot(df)+geom_point(aes(x=log10(x),y=y))+geom_line(aes(x=log10(x),y=yfit))
```

```
                                    pdf("MV_fit.pdf")
                                    print(p)
                                    dev.off()
                                    pdf("MV_fit_log.pdf")
                                    print(pp)
                                    dev.off()
                                }

                                Fdr_MV=pi_hat_long*F_o/F
                            }
        }
        if("BF" %in% alt_methods)
        {
            if(print_messages){cat("Calculating Bonferroni adjusted p-values...\n")}

            if(full_data_is_file){
                BF=grid*full_tests
            }else{
                BF=grid*length(grid)
            }
            BF[which(BF>1)]=1
        }
    }

    if(full_data_is_file){
        significant_tests=vector(mode = "double", length = (2+sum(alt_methods_aux)))
        if(print_messages){cat("Interpolating FDRs from grid p-values...\n")}
        result=.C("interpolate_fdrs",
        file=full_data,
        columns=as.integer(columns_full_data),
        flag_rownames=as.integer(rownames_flag_full),
        rows=as.double(full_tests),
        stat_column=as.integer(full_column_id),
        sampled_stats=as.integer(reference_size),
        sampled_stats_array=grid_log,
        Fdr=Fdr,
        Fndr=Fndr,
        Fdr_ST=Fdr_ST,
        Fndr_ST=Fndr_ST,
        Fdr_BH_unif=Fdr_BH_unif,
        Fdr_BH_perm=Fdr_BH_perm,
        Fdr_MV=Fdr_MV,
        BF=BF,
        name=output_name,
        alt_methods=as.integer(alt_methods_aux),
        zero_Fdr=as.double(zero_Fdr),
        report_threshold=as.double(report_threshold),
        print_messages=print_messages,
        significance_threshold=significance_threshold,
        significant_tests=as.double(significant_tests))
        significant_tests_vector=as.double(result$significant_tests)

        Fdr<-as.double(result$Fdr)
        Fndr<-as.double(result$Fndr)
        Fdr_ST<-as.double(result$Fdr_ST)
        Fndr_ST<-as.double(result$Fndr_ST)
        Fdr_BH_unif<-as.double(result$Fdr_BH_unif)
        Fdr_BH_perm<-as.double(result$Fdr_BH_perm)
        Fdr_MV<-as.double(result$Fdr_MV)
        BF<-as.double(result$BF)

        significant_tests=data.frame(significant_tests_vector[1],significant_tests_vector[2])
        cont=1
        for(i in 1:length(alt_methods_aux))
        {
            if(alt_methods_aux[i]==1)
            {
                significant_tests=cbind(significant_tests,significant_tests_vector[cont+2])
                cont=cont+1
```

```
            }
        }

if(sum(alt_methods_aux)>0){colnames(significant_tests)=c("Fdr_SAB","Fndr_SAB",alt_methods)}
        else{colnames(significant_tests)=c("Fdr_SAB","Fndr_SAB")}
    }else{
        result=.C("monotone_fdrs",Fdr=Fdr,Fndr=Fndr,
        Fdr_ST=Fdr_ST,
        Fndr_ST=Fndr_ST,
        Fdr_BH_unif=Fdr_BH_unif,
        Fdr_BH_perm=Fdr_BH_perm,
        Fdr_MV=Fdr_MV,
        alt_methods=as.integer(alt_methods_aux),
        sampled_stats=as.integer(length(grid)))

        Fdr<-as.double(result$Fdr)
        Fndr<-as.double(result$Fndr)
        Fdr_ST<-as.double(result$Fdr_ST)
        Fndr_ST<-as.double(result$Fndr_ST)
        Fdr_BH_unif<-as.double(result$Fdr_BH_unif)
        Fdr_BH_perm<-as.double(result$Fdr_BH_perm)
        Fdr_MV<-as.double(result$Fdr_MV)

        significant_tests=data.frame(0,0)
        if(sum(alt_methods_aux)>0){
            for(i in 1:sum(alt_methods_aux))
            {
                significant_tests=cbind(significant_tests,0)
            }
        }

if(sum(alt_methods_aux)>0){colnames(significant_tests)=c("Fdr_SAB","Fndr_SAB",alt_methods)}
        else{colnames(significant_tests)=c("Fdr_SAB","Fndr_SAB")}

        significant_tests$Fdr_SAB=length(which(Fdr<=significance_threshold))
        significant_tests$Fndr_SAB=length(which(Fndr<=significance_threshold))
        if("Fdr_ST" %in% alt_methods){
significant_tests$Fdr_ST=length(which(Fdr_ST<=significance_threshold))  }
        if("Fndr_ST" %in% alt_methods){
significant_tests$Fndr_ST=length(which(Fndr_ST<=significance_threshold))  }
        if("Fdr_BH_unif" %in% alt_methods){
significant_tests$Fdr_BH_unif=length(which(Fdr_BH_unif<=significance_threshold))  }
        if("Fdr_BH_perm" %in% alt_methods){
significant_tests$Fdr_BH_perm=length(which(Fdr_BH_perm<=significance_threshold))  }
        if("Fdr_MV" %in% alt_methods){
significant_tests$Fdr_MV=length(which(Fdr_MV<=significance_threshold))  }
        if("BF" %in% alt_methods){
significant_tests$BF=length(which(BF<=significance_threshold))  }
    }

    if(print_messages)
    {
        if(length(which(alt_methods_aux==1))>0)
        {
            cat("\n_____________________________________STEP 6: Print
outputs_____________________________________________________________\n")
        }else{
            cat("\n_____________________________________STEP 5: Print
outputs_____________________________________________________________\n")
        }
    }

output=list(full_tests,perm_tests,pi_o,p_star,significant_tests,fitting_goodness,call=deparse(m
atch.call()))
    names(output)=c("full_tests","perm_tests","pi_o","p_star","hits","fitting_goodness","call")

    if("Fdr_ST" %in% alt_methods | "Fndr_ST" %in% alt_methods)
    {
        names=c(names(output),"pi_o_ST")
        output[[length(output)+1]]=pi_o_ST
```

```
        names(output)=names
    }


    if(full_data_is_file==0)
    {
        fdrs=data.frame(full_data,Fdr_SAB=-1,Fndr_SAB=-1)
        first_entries=which(!duplicated(fdrs[,full_column_id]))
        duplicated_entries=which(duplicated(fdrs[,full_column_id]))

        fdrs$Fdr_SAB[first_entries]=Fdr
        fdrs$Fndr_SAB[first_entries]=Fndr

        for(i in 1:length(duplicated_entries))
        {
            fdrs$Fdr_SAB[duplicated_entries[i]]=fdrs$Fdr_SAB[duplicated_entries[i]-1]
            fdrs$Fndr_SAB[duplicated_entries[i]]=fdrs$Fndr_SAB[duplicated_entries[i]-1]
        }

        if(alt_methods_aux[1]==1)
        {
            fdrs=data.frame(fdrs,Fdr_ST=0)
            fdrs$Fdr_ST[first_entries]=Fdr_ST

            for(i in 1:length(duplicated_entries))
            {
                fdrs$Fdr_ST[duplicated_entries[i]]=fdrs$Fdr_ST[duplicated_entries[i]-1]
            }
        }

        if(alt_methods_aux[2]==1)
        {
            fdrs=data.frame(fdrs,Fndr_ST=0)
            fdrs$Fndr_ST[first_entries]=Fndr_ST
            for(i in 1:length(duplicated_entries))
            {
                fdrs$Fndr_ST[duplicated_entries[i]]=fdrs$Fndr_ST[duplicated_entries[i]-1]
            }
        }

        if(alt_methods_aux[3]==1)
        {
                        fdrs=data.frame(fdrs,Fdr_BH_unif=0)
            fdrs$Fdr_BH_unif[first_entries]=Fdr_BH_unif
            for(i in 1:length(duplicated_entries))
            {
                fdrs$Fdr_BH_unif[duplicated_entries[i]]=fdrs$Fdr_BH_unif[duplicated_entries[i]-
1]
            }
        }

        if(alt_methods_aux[4]==1)
        {
                        fdrs=data.frame(fdrs,Fdr_BH_perm=0)
            fdrs$Fdr_BH_perm[first_entries]=Fdr_BH_perm
            for(i in 1:length(duplicated_entries))
            {
                fdrs$Fdr_BH_perm[duplicated_entries[i]]=fdrs$Fdr_BH_perm[duplicated_entries[i]-
1]
            }
        }

        if(alt_methods_aux[5]==1)
        {
            fdrs=data.frame(fdrs,Fdr_MV=0)
            fdrs$Fdr_MV[first_entries]=Fdr_MV
            for(i in 1:length(duplicated_entries))
            {
                fdrs$Fdr_MV[duplicated_entries[i]]=fdrs$Fdr_MV[duplicated_entries[i]-1]
            }
```

```
        }

        if(alt_methods_aux[6]==1)
        {
            fdrs=data.frame(fdrs,BF=0)
            fdrs$BF[first_entries]=BF
            for(i in 1:length(duplicated_entries))
            {
                fdrs$BF[duplicated_entries[i]]=fdrs$BF[duplicated_entries[i]-1]
            }
        }

        names=c(names(output),"fdrs")
        output[[length(output)+1]]=fdrs
        names(output)=names
    }

   if(plot==TRUE)
    {
        ###Build data.frames to plot: 1) local distributions 2) Grenander (local)
distributions. 3) ECDFs 4)fdrs

        if(full_data_is_file)
        {
            grid=grid_log
            F=F_log
        }


        ##1. Load fdrs table

fdrs=data.frame(statistic=grid,quantile=(F/F[length(F)])*processed_tests,Fdr=Fdr,Fndr=Fndr)
        labels_fdr=c(expression(Fdr[{SAB}]),expression(Fndr[{SAB}]))

        if(alt_methods_aux[1]==1)
        {
            fdrs=data.frame(fdrs,Fdr_ST=Fdr_ST)
            labels_fdr=c(labels_fdr,expression(Fdr[{ST}]))
        }

        if(alt_methods_aux[2]==1)
        {
            fdrs=data.frame(fdrs,Fndr_ST=Fndr_ST)
            labels_fdr=c(labels_fdr,expression(Fndr[{ST}]))
        }

        if(alt_methods_aux[3]==1)
        {
            fdrs=data.frame(fdrs,Fdr_BH_unif=Fdr_BH_unif)
            labels_fdr=c(labels_fdr,expression(Fdr[{BH(unif)}]))
        }

        if(alt_methods_aux[4]==1)
        {
            fdrs=data.frame(fdrs,Fdr_BH_perm=Fdr_BH_perm)
            labels_fdr=c(labels_fdr,expression(Fdr[{BH(perm)}]))
        }

        if(alt_methods_aux[5]==1)
        {
            fdrs=data.frame(fdrs,Fdr_MV=Fdr_MV)
            labels_fdr=c(labels_fdr,expression(Fdr[{MV}]))
        }

        if(alt_methods_aux[6]==1)
        {
            fdrs=data.frame(fdrs,BF=BF)
            labels_fdr=c(labels_fdr,expression(BF))
        }
```

```
        ## Restrict it to the segment of interest
        if(report_threshold<max_stat)
        {
            end=fdrs$statistic[which(fdrs$Fdr>=report_threshold)][1]
            if("Fdr_ST" %in% alt_methods)
end=max(end,fdrs$statistic[which(fdrs$Fdr_ST>=report_threshold)][1])
            if("Fdr_BH_perm" %in% alt_methods)
end=max(end,fdrs$statistic[which(fdrs$Fdr_BH_perm>=report_threshold)][1])
            if("Fdr_BH_unif" %in% alt_methods)
end=max(end,fdrs$statistic[which(fdrs$Fdr_BH_unif>=report_threshold)][1])
            if("Fdr_MV" %in% alt_methods)
end=max(end,fdrs$statistic[which(fdrs$Fdr_MV>=report_threshold)][1])
            if(is.na(end)) end=max_stat
        }else{
            end=max_stat
        }

        ##Load tab: dataframe with distributions, both local and cumulative (some are still
provisional)
        if(full_data_is_file==0)
        {
            F_null_g=F_o_g*pi_o_scaling

F_null_g[index_p_star:length(grid)]=(F_o_g[index_p_star]*pi_o_scaling+F_g[index_p_star:length(g
rid)]-F_g[index_p_star])
            pi_o_g=F_null_g[length(F_null_g)]
            F_null_g=F_null_g/F_null_g[length(F_null_g)]
            tab<-
data.frame(statistic=grid,f_o=f_o,f=f,f_o_g=f_o_g,f_g=f_g,F_o=F_o,F=F,F_o_g=F_o_g,F_g=F_g,pi_o_
F_null=pi_o_g*F_null_g,pi_o_f_null=pi_o_g*F_null_g)
        }else{
            F_null_unif=F_o_g*pi_o_scaling

F_null_unif[index_p_star_unif:reference_size]=(F_o_g[index_p_star_unif]*pi_o_scaling+F_g[index_
p_star_unif:reference_size]-F_g[index_p_star_unif])
            pi_o_g=F_null_unif[length(F_null_unif)]
            F_null_unif=F_null_unif/F_null_unif[length(F_null_unif)]
            tab<-
data.frame(statistic=grid_unif,f_o=f_o_unif,f=f_unif,f_o_g=f_o_g,f_g=f_g,F_o=F_o_unif,F=F_unif,
F_o_g=F_o_g,F_g=F_g,pi_o_F_null=pi_o_g*F_null_unif,pi_o_f_null=pi_o_g*F_null_unif)
        }

        cut=which(tab$statistic>=end*0.999999)[1]
        tab=tab[1:cut,]


        ###Reduce points/bins to plot to a constant number (to avoid generating pdfs with as
many points as tests, which might be a lot) and get correct versions of the distributions.
Report threshold will be accepted only if allows plotting 20 points.

        bins_array=c(500,450,400,350,300,250,200,150,100,50,20)
        enough_bins=0;
        for(bins_number in c(500,450,400,350,300,250,200,150,100,50,20))
        {
            if(nrow(tab)>bins_number)
            {
                enough_bins=1
                break
            }else{
                bins_array=bins_array[(which(bins_array==bins_number)+1):length(bins_array)]
            }
        }
        if(enough_bins==0)
        {
            end=max_stat
            bins_array=c(500,450,400,350,300,250,200,150,100,50,20)
            if(full_data_is_file){
                tab<-
data.frame(statistic=grid_unif,f_o=f_o_unif,f=f_unif,f_o_g=f_o_g,f_g=f_g,F_o=F_o_unif,F=F_unif,
F_o_g=F_o_g,F_g=F_g,pi_o_F_null=pi_o_g*F_null_unif,pi_o_f_null=pi_o_g*F_null_unif)
```

```
            }else{
                tab<-
data.frame(statistic=grid,f_o=f_o,f=f,f_o_g=f_o_g,f_g=f_g,F_o=F_o,F=F,F_o_g=F_o_g,F_g=F_g,pi_o_
F_null=pi_o_g*F_null_g,pi_o_f_null=pi_o_g*F_null_g)
            }
            cut=which(tab$statistic>=end*0.999999)[1]
            tab=tab[1:cut,]
            cat("WARNING: Not enough resolution for this report_threshold. Set to 1 in the
plot\n")
        }

        for(bins_number in bins_array)
        {
            if(full_data_is_file)
            {
                reduction_factor=as.integer(nrow(tab)/bins_number)
                v_aux=seq(1,nrow(tab),reduction_factor)
                v_aux=c(v_aux[2:length(v_aux)],nrow(tab))
                v_aux=unique(v_aux)
            }else{
                reduction_factor=as.integer(nrow(tab)/bins_number)
                v_aux=vector(mode = "double", length = bins_number)
                delta=as.double((end-min_stat)/bins_number)
                grid_aux=seq(min_stat,end,delta)

                index_aux=1
                for(i in 2:length(v_aux))
                {
                    while(grid_aux[i]>grid[index_aux])
                    index_aux=index_aux+1
                    v_aux[i]=index_aux
                }
                v_aux=c(v_aux[2:length(v_aux)],nrow(tab))
                v_aux=unique(v_aux)
            }

            v=v_aux
            rm(v_aux)

            tab_reduced=tab[v,]
            tab_reduced$f_o[1]=tab_reduced$F_o[1]/(tab_reduced$statistic[1]-min_stat)
            tab_reduced$f_o_g[1]=tab_reduced$F_o_g[1]/(tab_reduced$statistic[1]-min_stat)
            tab_reduced$f[1]=tab_reduced$F[1]/(tab_reduced$statistic[1]-min_stat)
            tab_reduced$f_g[1]=tab_reduced$F_g[1]/(tab_reduced$statistic[1]-min_stat)
            tab_reduced$pi_o_f_null[1]=tab_reduced$pi_o_F_null[1]/(tab_reduced$statistic[1]-
min_stat)

            for(i in 2:nrow(tab_reduced))
            {
                tab_reduced$f_o[i]=(tab_reduced$F_o[i]-tab_reduced$F_o[i-
1])/(tab_reduced$statistic[i]-tab_reduced$statistic[i-1])
                tab_reduced$f_o_g[i]=(tab_reduced$F_o_g[i]-tab_reduced$F_o_g[i-
1])/(tab_reduced$statistic[i]-tab_reduced$statistic[i-1])
                tab_reduced$f[i]=(tab_reduced$F[i]-tab_reduced$F[i-
1])/(tab_reduced$statistic[i]-tab_reduced$statistic[i-1])
                tab_reduced$f_g[i]=(tab_reduced$F_g[i]-tab_reduced$F_g[i-
1])/(tab_reduced$statistic[i]-tab_reduced$statistic[i-1])
                tab_reduced$pi_o_f_null[i]=(tab_reduced$pi_o_F_null[i]-
tab_reduced$pi_o_F_null[i-1])/(tab_reduced$statistic[i]-tab_reduced$statistic[i-1])
            }

maxim=max(tab_reduced$f_o,tab_reduced$f,tab_reduced$f_o_g,tab_reduced$f_g,tab_reduced$pi_o_f_nu
ll)
            if(maxim<10)
            {break}
        }
        tab_reduced=rbind(tab_reduced[1,],tab_reduced)
        tab_reduced[1,1]=min_stat

        tab_1=data.frame(tab_reduced[,c(1,2)],Data="Permuted")
```

```
        colnames(tab_1)=c("statistic","series","Data")

        tab_2=data.frame(tab_reduced[,c(1,3)],Data="Full")
        colnames(tab_2)=c("statistic","series","Data")

        tab_hists=rbind(tab_1,tab_2)

        tab_3=data.frame(tab_reduced[,c(1,4)],Estimated_distribution="f_perm")
        colnames(tab_3)=c("statistic","series","Estimated_distribution")

        tab_4=data.frame(tab_reduced[,c(1,5)],Estimated_distribution="f_full")
        colnames(tab_4)=c("statistic","series","Estimated_distribution")

        tab_5=data.frame(tab_reduced[,c(1,11)],Estimated_distribution="pi_o_f_null")
        colnames(tab_5)=c("statistic","series","Estimated_distribution")

        tab_lines=rbind(tab_3,tab_4,tab_5)

        maxim_entry_hists=max(tab_hists$series)
        maxim_entry_lines=max(tab_lines$series)


        ### Generate X axis tickmarks: Intercalate among x axis tickmarks one at p*

        if(10^(-2)<p_star)
        {
            p_star_write=format(round(p_star, 2))
            if(p_star==1) {p_star_write=1}
        }else{
            p_star_write=format(p_star,scientific=TRUE)
            if(p_star==0) {p_star_write=0}
        }

        if(10^(-2)<pi_o_scaling)
        {
            rho_o_write=round(pi_o_scaling,2)
            if(pi_o_scaling==1) {rho_o_write=1}

        }else{
            rho_o_write=format(pi_o_scaling,digits=2,scientific=TRUE)
            if(pi_o_scaling==0) {rho_o_write=0}
        }

        if(10^(-2)<pi_o)
        {
            pi_o_write=round(pi_o,2)
            if(pi_o==1) {pi_o_write=1}
        }else{
            pi_o_write=format(pi_o,digits=2,scientific=TRUE)
            if(pi_o==0) {pi_o_write=0}
        }


        tickmarks_seq=seq(min_stat,end,(end-min_stat)/5)

        index_p_star_ticks=which.min(abs(tickmarks_seq-p_star))
        tickmarks_length=length(tickmarks_seq)

        if(tickmarks_seq[index_p_star_ticks]>p_star)
        {
            if(min(abs(tickmarks_seq-p_star))>0.09*(end-min_stat))
            {
                tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-
1],p_star,tickmarks_seq[index_p_star_ticks:length(tickmarks_seq)])
                tickmarks_length=tickmarks_length+1
            }else{
                if(index_p_star_ticks<length(tickmarks_seq))
                {
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-
1],p_star,tickmarks_seq[index_p_star_ticks+1:length(tickmarks_seq)])
```

```
                }else{
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-1],p_star)
                }
            }
        }else{
            if(min(abs(tickmarks_seq-p_star))>0.09*(end-min_stat))
            {
                if(index_p_star_ticks<length(tickmarks_seq))
                {

tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks],p_star,tickmarks_seq[(index_p_star_ticks+1)
:length(tickmarks_seq)])
                    tickmarks_length=tickmarks_length+1
                    index_p_star_ticks=index_p_star_ticks+1
                }else{
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks],p_star)
                    tickmarks_length=tickmarks_length+1
                    index_p_star_ticks=index_p_star_ticks+1
                }
            }else{
                if(index_p_star_ticks<length(tickmarks_seq))
                {
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-
1],p_star,tickmarks_seq[index_p_star_ticks+1:length(tickmarks_seq)])
                }else{
                    tickmarks_seq=c(tickmarks_seq[1:(index_p_star_ticks-1)],p_star)
                }
            }
        }

        if(end>0.1)
        {
            labels_tickmarks=round(tickmarks_seq,2)
        }else{
            labels_tickmarks=format(tickmarks_seq,digits=3,scientific=TRUE)
        }

        labels_tickmarks[index_p_star_ticks]=paste0(p_star_write)
        #labels_tickmarks[index_p_star_ticks]=paste0("p*=\n",p_star_write)

        if(all(flag_grenander_choice==c(1,4))|all(flag_grenander_choice==c(4,1)))
        {
            x_annotation=(end-min_stat)*0.7
        }else
if(all(flag_grenander_choice==c(3,4))|all(flag_grenander_choice==c(4,3))|all(flag_grenander_cho
ice==c(4,4))){
            x_annotation=(end-min_stat)*0.3
        }else{
            x_annotation=(end-min_stat)*0.5}

        p1<-ggplot() +
        geom_area(data=tab_hists,
aes(x=statistic,y=series,fill=Data),alpha=0.5,position="identity", na.rm=TRUE)+

geom_line(data=tab_lines,aes(x=statistic,y=series,colour=Estimated_distribution),size=0.7,posit
ion="identity", na.rm=TRUE)+
        scale_fill_manual(values=c("firebrick2","dodgerblue1"),labels=c("Full"="Actual
tests","Permuted"="Permut. tests"))+

scale_colour_manual(values=c("firebrick","dodgerblue1","grey50"),labels=c("f_full"=expression(p
aste("Actual tests ",f)),"f_perm"=expression(paste("Permut. tests
",f[{o}])),"pi_o_f_null"=expression(paste("Estimated null ",pi[{o}],"·",f[{null}])))))+
        ylab("Local densities")+
        xlab("P-value")+
        guides(colour=guide_legend(title="Local densities",label.hjust = 0, label.vjust = 0,
override.aes = list(size=1.5)))+
        guides(fill=FALSE)+
        #guides(fill=guide_legend(title="Local densities",override.aes = list(size=3)))+
        theme(axis.text.y   = element_text(size=10),
        axis.text.x   = element_text(size=10),
```

```
        axis.title.x  = element_text(size=10),
        axis.title.y  = element_text(size=10),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill=NA, size=1),
        legend.title=element_text(size=12),
        legend.text=element_text(size=12))+
        scale_x_continuous(breaks=tickmarks_seq,labels=labels_tickmarks,limits =
c(min_stat*0.999,end))+
        geom_vline(xintercept=p_star, colour="grey", linetype = "longdash", na.rm=TRUE)+
        annotate("text", x = x_annotation, y=maxim*0.9, label =paste("f", "==
paste(pi[{o}],f[{null}],+(1-pi[{o}]),f[{alt}])"),parse=TRUE)

        ####################

        if(full_data_is_file){
            if(index_p_star_unif>length(F)/2)
            {
                x=data.frame(statistic=grid_unif,F=F_unif,F_o=F_o_unif)

y=data.frame(statistic=grid_unif[finite_set],pi_hat=pi_hat[finite_set],pi_fit=pi_hat_fitted)
            }
            else
            {
                x=data.frame(statistic=grid_unif,F=F_unif,F_o=F_o_unif)

y=data.frame(statistic=grid_unif[finite_set_grid],pi_hat=pi_hat[finite_set],pi_fit=pi_hat_fitte
d)
            }
            x=x[which(x$statistic<end),]
        }else{
            if(index_p_star>length(F)/2)
            {
                x=data.frame(statistic=grid,F=F,F_o=F_o)

y=data.frame(statistic=grid[finite_set],pi_hat=pi_hat[finite_set],pi_fit=pi_hat_fitted)
            }else{
                x=data.frame(statistic=grid,F=F,F_o=F_o)

y=data.frame(statistic=grid[finite_set_grid],pi_hat=pi_hat[finite_set],pi_fit=pi_hat_fitted)
            }
            x=x[which(x$statistic<end),]
        }
        #ECDFs
        x=melt(x,id="statistic")
        #pi_o_fit_data
        y=melt(y,id="statistic")

        #pi_o_fit_data:just the dots
        z=y[which(y$variable=="pi_hat"),]
        fact=100
        if(nrow(z)>fact)
        {
            set_step=as.integer(nrow(z)/fact)
            z=z[seq(1,nrow(z),set_step),]
        }

        #pi_o_fit_data:just the fit
        zz=y[which(y$variable=="pi_fit"),]

        #x=rbind(x,zz)
        x$variable=factor(x$variable)

        maxim_cumulative=max(x$value[which(x$statistic<=end)])

        tickmarks_seq_y_distributions=seq(0,maxim_cumulative*1.01,maxim_cumulative/5)
        labels_tickmarks_seq_y_distributions=round(tickmarks_seq_y_distributions,2)
```

```
        p2<-ggplot() +
        geom_line(data=x, aes(x=statistic,y=value,colour=variable),size=0.7, na.rm=TRUE)+

scale_colour_manual(values=c("dodgerblue1","firebrick","mediumpurple4"),labels=c("F"=expression
(F),"F_o"=expression(F[{o}]),"pi_fit"=expression(paste(tilde(pi)[{o}],"  fit"))))+
        ylab("Cumulative distributions")+
        xlab("P-value")+
        guides(colour=guide_legend(title="CDFs",label.hjust = 0, label.vjust = 0,override.aes =
list(size=1.5),order = 1),
        fill=guide_legend(title="",override.aes = list(size=2),order = 2))+
        theme(axis.text.y   = element_text(size=10),
        axis.text.x   = element_text(size=10),
        axis.title.y  = element_text(size=10),
        axis.title.x  = element_text(size=10),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill=NA, size=1),
        legend.title=element_text(size=12),
        legend.text=element_text(size=12))+
        scale_x_continuous(breaks=tickmarks_seq,labels=labels_tickmarks,limits =
c(min_stat*0.999,end))+

scale_y_continuous(breaks=tickmarks_seq_y_distributions,labels=labels_tickmarks_seq_y_distribut
ions,limits = c(-0.0001,maxim_cumulative*1.01))+
        geom_vline(xintercept=p_star, colour="grey", linetype = "longdash", na.rm=TRUE)

        ##############################################################################

        rm(tickmarks_seq)
        #end_pi_o_x_axis=(max(p_star,end,as.numeric(p_star_exhaustive_sweep==FALSE)*max_stat))
        end_pi_o_x_axis=max_stat
        tickmarks_seq=seq(min_stat,end_pi_o_x_axis,(end_pi_o_x_axis-min_stat)/5)
        index_p_star_ticks=which.min(abs(tickmarks_seq-p_star))
        tickmarks_length=length(tickmarks_seq)

        if(tickmarks_seq[index_p_star_ticks]>p_star)
        {
            if(min(abs(tickmarks_seq-p_star))>0.1*(end_pi_o_x_axis-min_stat))
            {
                tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-
1],p_star,tickmarks_seq[index_p_star_ticks:length(tickmarks_seq)])
                tickmarks_length=tickmarks_length+1
            }else{
                if(index_p_star_ticks<length(tickmarks_seq))
                {
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-
1],p_star,tickmarks_seq[index_p_star_ticks+1:length(tickmarks_seq)])
                }else{
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-1],p_star)
                }
            }
        }else{
            if(min(abs(tickmarks_seq-p_star))>0.1*(end_pi_o_x_axis-min_stat))
            {
                if(index_p_star_ticks<length(tickmarks_seq))
                {

tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks],p_star,tickmarks_seq[(index_p_star_ticks+1)
:length(tickmarks_seq)])
                    tickmarks_length=tickmarks_length+1
                    index_p_star_ticks=index_p_star_ticks+1
                }else{
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks],p_star)
                    tickmarks_length=tickmarks_length+1
                    index_p_star_ticks=index_p_star_ticks+1
                }
            }else{
                if(index_p_star_ticks<length(tickmarks_seq))
```

```
                {
                    tickmarks_seq=c(tickmarks_seq[1:index_p_star_ticks-
1],p_star,tickmarks_seq[index_p_star_ticks+1:length(tickmarks_seq)])
                }else{
                    tickmarks_seq=c(tickmarks_seq[1:(index_p_star_ticks-1)],p_star)
                }
            }
        }

        if(end_pi_o_x_axis>0.1)
        {
            labels_tickmarks=round(tickmarks_seq,2)
        }else{
            labels_tickmarks=format(tickmarks_seq,digits=3,scientific=TRUE)
        }
        labels_tickmarks[index_p_star_ticks]=paste0(p_star_write)

        tickmarks_seq_y=c(0,0.25,0.50,0.75,1.00)
        index_pi_o_ticks=which.min(abs(tickmarks_seq_y-pi_o_scaling))

        if(tickmarks_seq_y[index_pi_o_ticks]>pi_o_scaling)
        {
            if(min(abs(tickmarks_seq_y-pi_o_scaling))>0.1)
            {
                tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-
1],pi_o_scaling,tickmarks_seq_y[index_pi_o_ticks:length(tickmarks_seq_y)])
            }else{
                if(index_pi_o_ticks<length(tickmarks_seq_y)){
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-
1],pi_o_scaling,tickmarks_seq_y[index_pi_o_ticks+1:length(tickmarks_seq_y)])
                }else{
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-1],pi_o_scaling)
                }
            }
        }else{
            if(min(abs(tickmarks_seq_y-pi_o_scaling))>0.1)
            {
                if(index_pi_o_ticks<length(tickmarks_seq_y)){

tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks],pi_o_scaling,tickmarks_seq_y[index_pi_o_t
icks+1:length(tickmarks_seq_y)])
                }else{
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks],pi_o_scaling)
                }

            }else{
                if(index_pi_o_ticks<length(tickmarks_seq_y)){
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-
1],pi_o_scaling,tickmarks_seq_y[index_pi_o_ticks+1:length(tickmarks_seq_y)])
                }else{
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-1],pi_o_scaling)
                }
            }
        }
        labels_tickmarks_seq_y=round(tickmarks_seq_y,2)
        index_pi_o_ticks=which.min(abs(tickmarks_seq_y-pi_o_scaling))

eval(parse(text=paste0("labels_tickmarks_seq_y[index_pi_o_ticks]=expression(rho[{o}]==",rho_o_w
rite,")")))

        if(pi_o_scaling<0.5){
            x_legend=0.55
            y_legend=0.95
        }else if(p_star>0.5)
        {
            x_legend=0.4
            y_legend=0.4
        }else{
            x_legend=0.55
            y_legend=0.4
```

```
        }

        if(p_star_exhaustive_sweep==TRUE){
            z=rbind(z,pi_o_plot)
            lbl_o=paste0('paste("Scaling p-value
",p^symbol("\052")==arg.min(rho[{o}](p)),"=',p_star_write,'")')
            #Permuted distribution scaling factor ",rho[{o}]==min(rho[{o}](p)))'

            lbl=paste0('paste("Permuted distribution scaling factor
",rho[{o}]==min(rho[{o}](p)),"=',rho_o_write,'")')
            lbl_2=paste0('paste("Fraction of true null tests
",pi[{o}]==rho[{o}]*F[{o}](p^symbol("\052"))+(1-F(p^symbol("\052"))),"=',pi_o_write,'")')

            #lbl_o='paste("Scaling p-value ",p^symbol("\052")==arg.min(rho[{o}](p)))'
            #Permuted distribution scaling factor ",rho[{o}]==min(rho[{o}](p)))'

            #lbl='paste("Permuted distribution scaling factor ",rho[{o}]==min(rho[{o}](p)))'
            #lbl_2=paste0('paste("Fraction of true null tests
",pi[{o}]==rho[{o}]*F[{o}](p^symbol("\052"))+(1-F(p^symbol("\052"))),"=',pi_o,'")')


            p3<-ggplot() +
            geom_line(data=zz, aes(x=statistic,y=value),size=0.7,colour="dodgerblue1",
na.rm=TRUE)+
            geom_point(data=z, aes(x=statistic,y=value,fill=variable),shape=21,size=0.8,
na.rm=TRUE)+
            scale_colour_continuous(guide = FALSE)+

scale_fill_manual(values=c("dodgerblue1","firebrick"),labels=c("pi_hat"=expression(paste(tilde(
rho)[{o}](p,p^symbol("\052"))==frac(F(p^symbol("\052"))-F(p),F[{o}](p^symbol("\052"))-
F[{o}](p)))),"pi_o_scalingloop"=expression(paste(rho[{o}],"(p)=",lim(tilde(rho)[{o}](p*symbol("
\242"),p), p*symbol("\242") %->% p)))))+
            ylab(expression(paste("Scaling factors  ",tilde(rho)[{o}]," , ",rho[{o}])))+
            xlab("P-value")+
            guides(fill=guide_legend(title="",override.aes = list(size=2),label.hjust = 0,
order = 2))+
            theme(axis.text.y   = element_text(size=10),
            axis.text.x   = element_text(size=10),
            axis.title.y  = element_text(size=10),
            axis.title.x  = element_text(size=10),
            #legend.position = c(x_legend, y_legend),
            panel.background = element_blank(),
            panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            axis.line = element_line(colour = "black"),
            panel.border = element_rect(colour = "black", fill=NA, size=1),
            legend.title=element_text(size=12),
            legend.key.size = unit(3, "lines"),
            legend.text=element_text(size=12))+
            scale_x_continuous(breaks=tickmarks_seq,labels=labels_tickmarks,limits =
c(min_stat*0.999,end_pi_o_x_axis))+
            scale_y_continuous(breaks=tickmarks_seq_y,labels=labels_tickmarks_seq_y,limits =
c(-0.0001,1.0001))+
            geom_vline(xintercept=p_star, colour="grey", linetype = "longdash", na.rm=TRUE)+
            geom_hline(yintercept=pi_o_scaling, colour="grey", linetype = "longdash",
na.rm=TRUE)+
            annotate("text", x = x_legend, y=y_legend, label =lbl_o,size=3,parse=TRUE)+
            annotate("text", x = x_legend, y=y_legend-0.17, label =lbl,size=3,parse=TRUE)+
            annotate("text", x = x_legend, y=y_legend-0.3, label =lbl_2,size=3,parse=TRUE)
        }else{

            lbl_o=paste0('paste("Scaling p value
",p^symbol("\052")==arg.min(f(p)/f[{o}](p)),"=',p_star_write,'")')
            lbl=paste0('paste("Permuted distribution scaling factor
",rho[{o}](p^symbol("\052"))==lim(tilde(rho)[{o}](p,p^symbol("\052")), p %->%
p^symbol("\052")),"=',rho_o_write,'")')
            lbl_2=paste0('paste("Fraction of true null tests
",pi[{o}](p^symbol("\052"))==rho[{o}](p^symbol("\052"))*F[{o}](p^symbol("\052"))+(1-
F(p^symbol("\052"))),"=',pi_o_write,'")')
```

```
            p3<-ggplot() +
            geom_line(data=zz, aes(x=statistic,y=value),size=0.7,colour="dodgerblue1",
na.rm=TRUE)+
            geom_point(data=z,
aes(x=statistic,y=value,fill=variable),shape=21,color="grey95",size=0.8, na.rm=TRUE)+
            scale_colour_continuous(guide = FALSE)+

scale_fill_manual(values=c("dodgerblue1"),labels=c("pi_hat"=expression(paste(tilde(rho)[{o}](p,
p^symbol("\052"))==frac(F(p^symbol("\052"))-F(p),F[{o}](p^symbol("\052"))-F[{o}](p))))))+
            ylab(expression(paste("Scaling factors  ",tilde(rho)[{o}]," , ",rho[{o}])))+
            xlab("P-value")+
            guides(fill=guide_legend(title="",override.aes = list(size=2),label.hjust = 0,
order = 2))+
            theme(axis.text.y   = element_text(size=10),
            axis.text.x   = element_text(size=10),
            axis.title.y  = element_text(size=10),
            axis.title.x  = element_text(size=10),
            #legend.position = c(x_legend, y_legend),
            panel.background = element_blank(),
            panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            axis.line = element_line(colour = "black"),
            panel.border = element_rect(colour = "black", fill=NA, size=1),
            legend.title=element_text(size=12),
            legend.key.size = unit(3, "lines"),
            legend.text=element_text(size=12))+
            scale_x_continuous(breaks=tickmarks_seq,labels=labels_tickmarks,limits =
c(min_stat*0.999,end_pi_o_x_axis))+
            scale_y_continuous(breaks=tickmarks_seq_y,labels=labels_tickmarks_seq_y,limits =
c(-0.0001,1.0001))+
            geom_vline(xintercept=p_star, colour="grey", linetype = "longdash", na.rm=TRUE)+
            geom_hline(yintercept=pi_o_scaling, colour="grey", linetype = "longdash",
na.rm=TRUE)+
            annotate("text", x = x_legend, y=y_legend, label =lbl_o,size=3,parse=TRUE)+
            annotate("text", x = x_legend, y=y_legend-0.17, label =lbl,size=3,parse=TRUE)+
            annotate("text", x = x_legend, y=y_legend-0.3, label =lbl_2,size=3,parse=TRUE)



        }
        ############################
        ############################

        cut=which(fdrs$statistic>=end*0.999999)[1]
        fdrs=fdrs[1:cut,c(1,(ncol(fdrs):2))]
        tab=tab[1:cut,]

        labels_fdr=labels_fdr[length(labels_fdr):1]

        colors_number=ncol(fdrs)-2

palette_set_1=c("dodgerblue1","firebrick","mediumpurple4","#4DAF4A","#FF7F00","#FFFF33","#A6562
8","#F781BF")
        palette=palette_set_1[colors_number:1]

        fdrs=fdrs[which(fdrs$statistic<end),]
        fdrs=fdrs[,c(2:ncol(fdrs))]
        fdrs=melt(fdrs,id="quantile")

        maxim_fdrs=max(fdrs$value[which(fdrs$variable %in%
c("Fdr_MV","Fdr_BH_perm","Fdr_BH_unif","Fdr_ST","Fdr"))])


        threshold_tick=significance_threshold
        tickmarks_seq_y=c(0,0.25,0.50,0.75,1.00)
        index_pi_o_ticks=which.min(abs(tickmarks_seq_y-threshold_tick))

        if(tickmarks_seq_y[index_pi_o_ticks]>threshold_tick)
        {
            if(min(abs(tickmarks_seq_y-threshold_tick))>0.1)
```

```
            {
                tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-
1],threshold_tick,tickmarks_seq_y[index_pi_o_ticks:length(tickmarks_seq_y)])
            }else{
                if(index_pi_o_ticks<length(tickmarks_seq_y)){
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-
1],threshold_tick,tickmarks_seq_y[index_pi_o_ticks+1:length(tickmarks_seq_y)])
                }else{
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-1],threshold_tick)
                }
            }
        }else{
            if(min(abs(tickmarks_seq_y-threshold_tick))>0.1)
            {
                if(index_pi_o_ticks<length(tickmarks_seq_y)){

tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks],threshold_tick,tickmarks_seq_y[index_pi_o
_ticks+1:length(tickmarks_seq_y)])
                }else{
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks],threshold_tick)
                }

            }else{
                if(index_pi_o_ticks<length(tickmarks_seq_y)){
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-
1],threshold_tick,tickmarks_seq_y[index_pi_o_ticks+1:length(tickmarks_seq_y)])
                }else{
                    tickmarks_seq_y=c(tickmarks_seq_y[1:index_pi_o_ticks-1],threshold_tick)
                }
            }
        }

        if(maxim_fdrs>0.1)
        {
            labels_tickmarks_seq_y=round(tickmarks_seq_y,2)
        }else{
            labels_tickmarks_seq_y=format(tickmarks_seq_y,digits=3,scientific=TRUE)
        }



        #tickmarks_seq_y_fdrs=seq(0,maxim_fdrs*1.01,maxim_fdrs/5)
        #labels_tickmarks_seq_y_fdrs=round(tickmarks_seq_y_fdrs,2)
        p4<-ggplot() +
        geom_line(data=fdrs, aes(x=quantile,y=value,colour=variable),size=0.7, na.rm=TRUE)+
        scale_colour_manual(values=palette,labels=labels_fdr)+
        ylab("Fdr & Fndr")+
        guides(colour=guide_legend(title="Methods",label.hjust = 0, label.vjust =
0,override.aes = list(size=1.5),order = 1))+
        ylab("Fdr & Fndr")+
        xlab("Number of tests")+
        theme(axis.text.y   = element_text(size=10),
        axis.text.x   = element_text(size=10),
        axis.title.y  = element_text(size=10),
        axis.title.x  = element_text(size=10),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.border = element_rect(colour = "black", fill=NA, size=1),
        legend.title=element_text(size=12),
        legend.text=element_text(size=12))+
        scale_y_continuous(breaks=labels_tickmarks_seq_y,labels=labels_tickmarks_seq_y,limits =
c(-0.0001,maxim_fdrs*1.01))+
        geom_vline(xintercept=p_star, colour="grey", linetype = "longdash", na.rm=TRUE)+
        geom_hline(yintercept=significance_threshold, colour="grey", linetype = "longdash",
na.rm=TRUE)
        #annotate("text", x = (max(fdrs$quantile))*0.9,
y=significance_threshold+0.05*maxim_fdrs, label
=paste0(as.integer(significance_threshold*100,0),"% FDR"))
```

```
        pdf(paste0(output_name,"/fdrs.pdf"),height=9,width=8,onefile=FALSE)
        print(perm_fdr_plot(list(p1,p2,p3,p4)))
        graphics.off()
    }
   ######################################## QUITAR
#################################################################
    if(flag_debug_august==TRUE)
    {
        if(full_data_is_file==1)
        {

distributions_and_fdrs_at_grid=data.frame(p_value=grid_log,F_perm=F_o_log,F=F_log,f_perm=f_o_lo
g,f=f_log,Fdr_SAB=Fdr,Fndr_SAB=Fndr)

pi_hat_fit=data.frame(p=grid_unif[finite_set],pi_hat=pi_hat[finite_set],pi_fit=pi_hat_fitted)

write.table(distributions_and_fdrs_at_grid,paste0(output_name,"/distributions_and_fdrs_at_grid.
txt"))
            write.table(pi_hat_fit,paste0(output_name,"/pi_hat_fit.txt"))
        }else{

distributions=data.frame(p=grid,F_perm=F_o,F_null=F_null,F=F,f_perm=f_o,f=f,F_perm_g=F_o_g,F_g=
F_g,f_perm_g=F_o_g,f_g=f_g)

pi_hat_fit=data.frame(p=grid[finite_set],pi_hat=pi_hat[finite_set],pi_fit=pi_hat_fitted)

distributions_all=data.frame(p=full_data[,full_column_id],F_perm=0,F_null=0,F=0,f_perm=0,f=0,F_
perm_g=0,F_g=0,f_perm_g=0,f_g=0)
            finite_ps_indexes=which(full_data[,full_column_id] %in% grid[finite_set])
            finite_ps_indexes=finite_ps_indexes[order(finite_ps_indexes)]


pi_hat_fit_all=data.frame(p=full_data[finite_ps_indexes,full_column_id],pi_hat=0,pi_fit=0)
            for(i in 1:nrow(pi_hat_fit_all))
            {
                index=which(pi_hat_fit[,1]==pi_hat_fit_all[i,1])
                pi_hat_fit_all[index,]=pi_hat_fit[index,]
            }

            write.table(distributions_all,paste0(output_name,"/distributions.txt"))
            write.table(pi_hat_fit_all,paste0(output_name,"/pi_hat_fit.txt"))
        }
    }
    ######################################## CIERRO QUITAR
#########################################################

    if(print_messages)
    {

cat("\n\n#############################################################################################
#######################","\n")
        cat("####################                                    Done!
##################","\n")

cat("#############################################################################################
####################","\n")
    }

    return(output)
}
```