

Travail de fin d'études

Optimisation d'un algorithme
bioinformatique pour prédire les
origines parentales ancestrales
humaines

Arnaud Cléris

Bachelier en Biotechnique : Bloc 3
HEH - Département des Sciences et technologies
HEPH - Condorcet
Année académique 2023-2024

Travail de fin d'études

Optimisation d'un algorithme
bioinformatique pour prédire les
origines parentales ancestrales
humaines

Arnaud Cléris

Bachelier en Biotechnique : Bloc 3
HEH - Département des Sciences et technologies
HEPH - Condorcet
Année académique 2023-2024

Remerciements

Je tiens à remercier Maxime Lefèvre (ULB), doctorant avec qui j'ai travaillé sur ce projet, et Maxime Tarabichi (ULB), mon maître de stage, pour toutes les explications et l'aide apportées durant la réalisation de ce travail. Je remercie également Mme Léonet (HEH) pour l'encadrement et les conseils.

Table des matières

Table des illustrations.....	7
Table des tableaux.....	8
Informations sur le laboratoire.....	11
Introduction.....	12
Variations génétiques.....	12
Populations.....	13
Résultats obtenus pendant le stage.....	15
Lien avec le projet de stage.....	17
Objectifs.....	18
Matériel et méthode.....	19
Fichier VCF (variant call format).....	19
Données.....	20
1000 genomes (1kGp).....	20
PCAWG.....	21
Méthodes.....	22
PCA.....	22
Rye.....	24
Résultats.....	28
Optimisation des données d'entrée et de RYE.....	28
Évaluation et optimisation de la méthode.....	30
Établissement d'un score pour trier les SNPs en fonction de leur pouvoir discriminant.....	30
Simulation de profils de SNPs pour tester l'algorithme.....	33
Prédictions sur les simulations.....	35
Seuils pour identifier les individus d'origines ancestrales mixtes.....	38
Application aux données PCAWG.....	44
Conclusion.....	46
Perspectives.....	48
Bibliographie.....	49
Lexique.....	51
Table des annexes.....	52
Annexe 1 - Code d'optimisation PCA.....	53
Annexe 2 - Code de création du fichier VCF d'index des SNPs.....	57
Annexe 3 - Code pour séparer le fichier général d'index de SNPs en petit fichier contenant les valeurs de SNPs.....	59
Annexe 4 - Code pour générer les simulations de profils de SNPs "mixed-ancestry" ...	62
Annexe 5 - Code pour obtenir les input (PCA) et les prédictions avec Rye.....	67
Annexe 6 - Script Rye.....	73
Annexe 7 - Benchmarks.....	86
Annexe 8 - Boxplots sur 100.000 SNPs discriminants.....	91
Annexe 9 - Graphiques de densités.....	96

Table des illustrations

Figure 1 : Maxime Tarabichi

Figure 2 : Maxime Lefèvre

Figure 3 : Illustration d'un SNP

Figure 4 : Comparaison des courbes d'exactitudes de prédictions entre le chromosome mitochondrial et le chromosome Y

Figure 5 : Exemple de fichier VCF

Figure 6 : Fichier VCF 1kGp

Figure 7 : Fichier VCF PCAWG

Figure 8 : Exemple d'une analyse en composantes principales (PCA)

Figure 9 : Schéma de fonctionnement de Rye

Figure 10A : Exemple d'input pour la PCA (ensemble de test)

Figure 10B : Exemple d'input pour la PCA (ensemble d'entraînement)

Figure 11 : Exemple d'output de Rye

Figure 12 : Fichier VCF des SNPs du chromosome 22

Figure 13 : Colonne INFO du fichier VFC du chromosome 22

Figure 14 : Colonnes "Score" et "Ethnie" ajoutées au fichier général

Figure 15 : Illustration de la méthode pour simuler des individus de deux origines différentes

Figure 16 : Exemple de prédictions obtenues sur 1.000 SNPs

Figure 17 : Benchmark pour la combinaison d'origines EUR-EAS

Figure 18 : Graphiques "boxplots" illustrant la distribution des probabilités d'appartenance ethnique pour les individus simulés dans la combinaison EUR-EAS

Figure 19A : Graphique de densités des individus d'origine européenne pour la combinaison d'origines EUR-AFR

Figure 19B : Graphique de densités des individus d'origine africaine pour la combinaison d'origines EUR-AFR

Figure 20A : Courbes ROC pour la classification des individus européens dans la combinaison EUR-AFR

Figure 20B : Courbes ROC pour la classification des individus non-européens dans la combinaison EUR-AFR

Table des tableaux

Tableau 1 : Tableau des seuils de classification des individus d'ascendance mixte pour différentes combinaisons ethniques

Tableau 2 : Résultats des prédictions de Rye sur un échantillon de 5 patients PCAWG

Résumé

Ce projet porte sur l'analyse des profils de variations génétiques dans le contexte du cancer. Il est possible et utile dans le cadre de la recherche et de la médecine de déterminer les origines ethniques d'individus en analysant leur ADN et les variations qui y figurent. Il faut pour cela utiliser des algorithmes d'inférence d'ascendance génétique. Cependant obtenir un résultat de qualité dans un court laps de temps reste un défi.

Pour résoudre ce problème, des améliorations ont été apportées à l'utilisation de l'algorithme Rye. Il s'agit d'un outil récent spécialement conçu pour inférer les ascendances génétiques à partir des données de variation génétique. La méthode utilisée par le laboratoire d'accueil pour utiliser Rye a été optimisée pour accroître la rapidité et la précision des analyses. Cette optimisation a permis d'analyser plusieurs individus simultanément, plutôt que de les traiter individuellement. Ensuite, le nombre de variations génétiques prises en compte a été ajusté afin d'atteindre le meilleur équilibre possible entre la vitesse d'exécution et la précision des résultats.

Pour faire suite à ces nouveautés, l'efficacité de la méthode a été testée sur des simulations. L'analyse de ces profils simulés a permis de déterminer les plages de valeurs à utiliser lors de l'évaluation des probabilités d'ascendance. Ces plages permettent de classer un individu comme étant ancestralement mixte en fonction des résultats obtenus.

Le travail effectué permettra de compléter le projet de Maxime Lefèvre, chercheur à l'ULB, visant à quantifier le taux mutationnel par copie de génome parental et fournira ainsi un outil fiable pour déterminer les ascendances génétiques.

En conclusion, le projet propose une amélioration de l'utilisation d'un nouvel algorithme pour déterminer les origines ancestrales à partir de profils génétiques et d'ainsi pouvoir identifier des individus d'origines ancestrales mixtes. Son application contribuera à une meilleure compréhension des processus génétiques sous-jacents au cancer et des variations génétiques entre les populations humaines.

Summary

This project focuses on the analysis of genetic variation profiles in the context of cancer. It is both possible and useful, in research and medicine, to determine the ethnic origins of individuals by analyzing their DNA and the variations within it. This requires the use of genetic ancestry inference algorithms. However, obtaining high-quality results in a short period of time remains a challenge.

To address this issue, improvements were made to the use of the Rye algorithm. Rye is a recent tool specifically designed to infer genetic ancestry from genetic variation data. The method used by the host laboratory to run Rye was optimized to increase the speed and accuracy of the analyses. This optimization allowed for the simultaneous analysis of multiple individuals, rather than processing them one by one. Additionally, the number of genetic variations considered was adjusted to achieve the best possible balance between execution speed and result accuracy.

Following these innovations, the efficiency of the method was tested on simulations. The analysis of these simulated profiles allowed for the determination of value ranges to be used when evaluating ancestry probabilities. These ranges enable the classification of an individual as ancestrally mixed based on the obtained results.

The work carried out will contribute to Maxime Lefèvre's project, a researcher at ULB, which aims to quantify the mutation rate per copy of the parental genome and thus provide a reliable tool for determining genetic ancestry.

In conclusion, the project proposes an improvement in the use of a new algorithm to determine ancestral origins from genetic profiles and to identify individuals of mixed ancestral origins. Its application will contribute to a better understanding of the genetic processes underlying cancer and the genetic variations among human populations.

Informations sur le laboratoire

L'université libre de Bruxelles (ULB) est l'une des plus grandes universités de Belgique. Fondée en 1834, elle dispose actuellement de six campus dont deux se situent à Charleroi. Elle travaille en collaboration avec d'autres universités étrangères comme Berkeley et Oxford, et avec des hôpitaux locaux comme Erasme et Jules Bordet [1]. Mon travail de fin d'études a été réalisé, tout comme le stage précédent, sur le campus Erasme de l'ULB, et plus précisément dans le laboratoire de l'IRIBHM (Institut de Recherche Interdisciplinaire en Biologie Humaine et moléculaire). Fondé dans les années 1960, l'institut regroupe 130 chercheurs, étudiants et techniciens travaillant sur des sujets de recherches diversifiés (comme les neurosciences et le cancer), en utilisant des approches de biologie cellulaire, moléculaire et de bioinformatique. Le service bioinformatique de ce laboratoire regroupe 7 étudiants masters ou (post-)doctorants et est supervisé par un professeur, Maxime Tarabichi, mon maître de stage (figure 1).

Maxime Tarabichi a réalisé un master en sciences appliquées à l'école polytechnique de l'ULB, un master en management à la Solvay business school et un doctorat en biologie computationnelle à l'IRIBHM. Il a ensuite effectué une recherche postdoctorale à l'institut Francis Crick de Londres. Il est, depuis 2020, chercheur principal (Principal Investigator ou P.I. en anglais) à l'IRIBHM. Je travaille sur le projet avec Maxime Lefèvre (figure 2), un des doctorants du service dont le sujet de thèse est l'étude des différences de mutabilité à l'échelle du génome entre les ensembles parentaux de chromosomes. Maxime Lefèvre a terminé un master en sciences biomédicales à l'UMONS ainsi qu'un master en bioinformatique et modélisation à l'ULB.



Figure 1 : Maxime Tarabichi

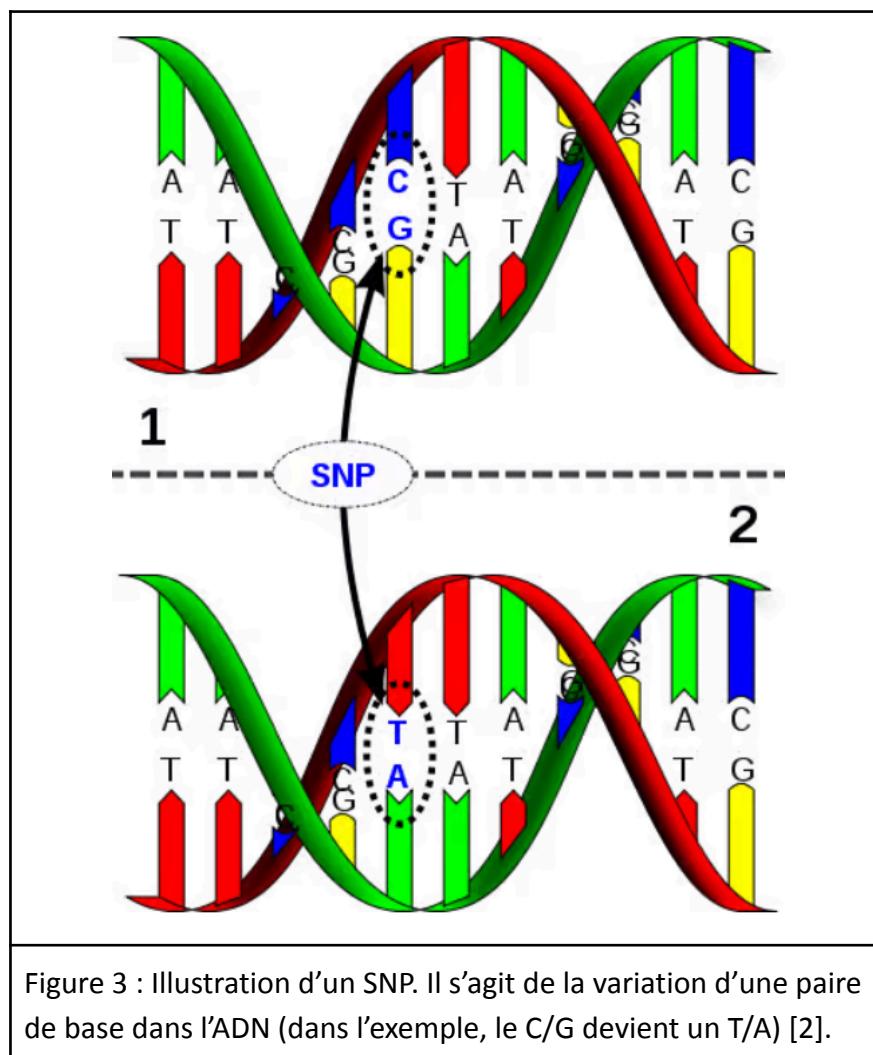


Figure 2 : Maxime Lefèvre

Introduction

Variations génétiques

Les cellules humaines possèdent deux copies du génome, chacune héritée d'un parent. Ces copies génétiques contiennent une multitude de variations telles que les insertions, les délétions, les duplications, les translocations, les inversions ou les SNPs (Single Nucleotide Polymorphisms), qui sont toutes responsables des différences observées entre chaque individu. Les SNPs sont des variations génétiques présentes dans toutes les cellules du corps impliquant la substitution d'un nucléotide par un autre (figure 3). À l'origine, ce sont majoritairement des variations germinales, c'est-à-dire des mutations présentes dans les ovules et les spermatozoïdes qui ont été transmises à la descendance, et qui se sont fixées dans la population lors de l'évolution de l'espèce humaine.



Par définition, pour être considéré comme un SNP, la variation doit apparaître dans minimum 1% de la population [3]. Ces variants de séquence influencent la mutabilité

spécifique aux allèles. De plus, certains d'entre eux sont caractéristiques à certaines populations, présentant une récurrence pour des origines ancestrales spécifiques [4, 5].

Les SNPs représentent en moyenne entre 0,1% et 1% des nucléotides des séquences observées, soit un SNP tous les 100 à 1000 nucléotides. Un SNP isolé n'est pas dangereux en soi et est même anodin la plupart du temps. Cependant, certaines combinaisons de SNPs sont connues pour favoriser l'apparition de maladies, dont des cancers [3]. D'autres combinaisons peuvent quant à elles, influencer le phénotype des individus, contribuant ainsi à la formation de populations et à la différenciation des individus. Il est également intéressant d'étudier ce type de variation dans le contexte de l'évolution humaine car cela permet de retracer les migrations humaines, de reconstruire l'histoire évolutive des populations et de comprendre les facteurs sous-jacents à la diversité génétique observée à travers le monde.

Les SNPs peuvent être mis en évidence en suivant trois étapes, à savoir :

- 1) En utilisant des techniques de séquençage de l'ADN qui permettent de lire la séquence nucléotidique du génome d'un individu. Parmi ces techniques on retrouve par exemple le séquençage Sanger traditionnel, le séquençage par synthèse de nouvelle génération (NGS) tel que proposé par l'entreprise Illumina, ou encore le séquençage de troisième génération à nanopores ou le séquençage de molécule unique en temps réel. Ici, la technologie Illumina a été utilisée;
- 2) Les séquences d'ADN sont ensuite alignées avec un génome de référence pour reconstruire le génome à analyser et détecter les variations.
- 3) La dernière étape, appelée "variant calling", sert à identifier les variations étudiées.

Populations

L'être humain se distingue en différentes populations. Une population peut être décrite comme un groupe d'individus partageant un ou plusieurs ancêtres communs et donc une fréquence allélique élevée pour certains gènes. La fréquence allélique est la fréquence à laquelle les allèles d'un gène (ou par extension une partie du génome) sont retrouvés dans le génome de plusieurs individus. Les membres constituant une population présentent donc des similitudes génétiques (et typiquement phénotypiques) et à l'inverse, les sujets issus de populations différentes vont afficher des caractéristiques génétiques et phénotypiques différentes. Des combinaisons de SNPs sont spécifiques à chaque population, ce qui permet d'identifier les origines des individus assez facilement. Pour représenter les populations en fonction des variations génétiques, il est possible d'utiliser des algorithmes tels que les analyses en composantes principales (ACPs; ou PCAs en Anglais) sur les données de SNPs [6]. D'autres méthodes, comme des plateformes en ligne, existent pour inférer les origines ancestrales et retracer la généalogie à partir d'un simple test ADN. Parmi les plus connus on retrouve MyHeritage [7], 23andMe [8],

AncestryDNA [9], National Geographic's Geno 2.0 [10] mais il en existe beaucoup d'autres encore. Toutes ces entreprises séquentent de l'ADN à partir d'échantillons fournis par des clients, puis comparent les données génétiques avec des bases de données de référence contenant des profils génétiques de populations du monde entier. Ils peuvent alors identifier, grâce à des algorithmes, les similitudes génétiques et estimer les origines ancestrales des individus, offrant ainsi des informations sur leurs ascendances géographiques et leur patrimoine génétique.

En complément de ces méthodes d'analyse des origines ancestrales, des algorithmes d'inférence parentale permettent de déterminer les relations familiales entre les individus à partir de leurs données génétiques. Ces algorithmes comparent les segments d'ADN partagés entre différents individus pour estimer les degrés de parenté, tels que les liens entre parents et enfants, frères et sœurs, ou cousins. Des outils comme "PLINK", "GERMLINE", ou encore "King" sont souvent utilisés pour détecter les segments d'ADN identiques par descendance (ou "Identical by Descent", IBD en anglais). Il est ainsi possible de reconstruire des arbres familiaux et découvrir des liens de parenté entre différents individus.

Les populations, bien que sujettes à des migrations qui modulent les fréquences alléliques des génomes [11], sont généralement regroupées géographiquement, ce qui facilite la transmission des caractéristiques aux générations futures et leur fixation dans le temps. Les rencontres entre individus d'origines différentes contribuent à ce que l'on appelle le brassage génétique ou les recombinaisons génétiques [12]. De nouveaux individus présentant un mélange de génome et potentiellement de caractéristiques propres aux parents voient ainsi le jour. Ce phénomène est à l'origine des individus d'ascendance mixte ("mixed-ancestry"), qui héritent d'un patrimoine génétique reflétant la diversité des populations d'origine [13].

Le processus de mutation est un autre facteur crucial de la diversité génétique. Les mutations, qui sont des modifications de la séquence d'ADN, peuvent se produire spontanément ou être déclenchées par des facteurs environnementaux comme l'exposition à des radiations ou à des agents chimiques. Si ces mutations se produisent dans les cellules germinales (mutations germinales), elles peuvent être transmises à la descendance. Contrairement aux mutations somatiques qui ne sont pas héréditaires, les mutations germinales peuvent entraîner la création de nouveaux allèles qui peuvent être transmis à la descendance, augmentant ainsi la diversité génétique au sein d'une population [14].

En plus des recombinaisons génétiques et des mutations, la dérive génétique et la sélection naturelle jouent également un rôle central dans la diversité génétique d'une population. La dérive génétique est un processus aléatoire par lequel certains allèles

deviennent plus ou moins fréquents au fil du temps (en raison de fluctuations aléatoires dans la reproduction et la survie des individus) [15], tandis que la sélection naturelle favorise les variantes génétiques qui confèrent un avantage adaptatif dans un environnement donné, augmentant ainsi leur prévalence dans la population [16].

La compréhension de la diversité génétique des SNPs a de nombreuses applications pratiques, notamment dans le domaine de la médecine personnalisée. En analysant les profils SNP spécifiques des individus, les chercheurs et médecins peuvent évaluer la prédisposition génétique à certaines maladies, les réactions potentielles à divers traitements, et retracer l'ascendance et l'histoire évolutive d'un individu [17].

Résultats obtenus pendant le stage

Le travail réalisé dans le cadre de ce travail de fin d'études s'inscrit dans la continuité du stage au vu duquel il a pu être démontré l'efficacité de la méthode développée pour inférer l'origine parentale des haplotypes en analysant les profils de SNPs de l'ADN mitochondrial et de l'ADN du chromosome Y des individus d'origines ancestrales mixtes. En utilisant les données du projet "1000 Genomes" (1kGp) [18] et les patients "Pan-Cancer Analysis of Whole Genomes" (PCAWG) [19], la technique a montré une capacité notable à fournir des résultats concordants avec les données ancestrales connues à partir de simulations d'individus d'origines ancestrales mixtes ou non.

L'application de l'algorithme KNN a permis de classifier les individus en fonction de leurs origines ancestrales avec une précision accrue, notamment lorsque le nombre de voisins (k) choisi était faible (figure 4).

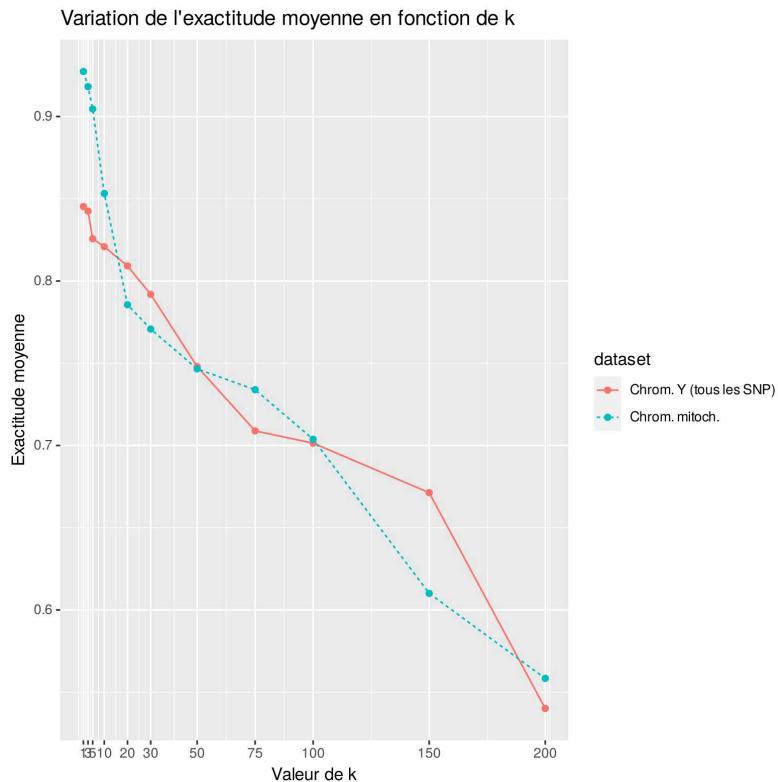


Figure 4 : Comparaison des courbes d'exactitudes de prédictions entre le chromosome mitochondrial et le chromosome Y (données 1kGp). Bootstrapping de 100 itérations pour chaque valeur de k avec des proportions d'ensemble d'entraînement et d'ensemble de test de 90%/10%. Sur l'axe des x : variation de la valeur de k. Sur l'axe des y : évolution de l'exactitude des prédictions (en %)

Après avoir été testée sur des simulations, la méthode a également été appliquée aux patients du projet PCAWG. Les résultats ont montré que la technique était performante, fournissant des résultats cohérents avec les prédictions initiales concernant l'origine parentale des haplotypes. Des ajustements supplémentaires, tels que l'exclusion des Américains des données d'entraînement, ont permis d'améliorer la robustesse et la précision des résultats.

Un autre aspect des résultats a été la démonstration que les performances étaient dégradées quand l'algorithme fonctionnait avec un ensemble d'entraînement constitué uniquement de SNPs des régions exoniques, en raison de la perte de données importantes situées en dehors des exomes.

L'ensemble des résultats obtenus a montré que la méthode développée est non seulement fonctionnelle, mais également robuste et cohérente pour l'inférence de l'origine parentale des haplotypes. Ces conclusions ont ouvert la voie à plusieurs perspectives de recherche, notamment l'exploration d'autres algorithmes pour améliorer

les résultats, l'application de la méthode à une population plus large, et son extension à l'ensemble du génome humain pour des études plus vastes.

Les résultats de ce stage ont donc fourni une base solide pour ce travail de fin d'études, tout en contribuant au projet de thèse de Maxime Lefèvre en offrant une méthode fiable pour quantifier les taux de mutations somatiques par copie parentale du génome.

Lien avec le projet de stage

Ce travail de fin d'études sur l'optimisation d'un algorithme bioinformatique pour prédire les probabilités des origines des individus à partir de leurs profils de SNPs trouve son ancrage dans le travail réalisé lors du stage précédent. En effet, le stage a permis d'explorer les techniques d'analyse génétique pour déterminer les origines ancestrales parentales d'individus en se basant sur les données génétiques représentant chacun des parents (à savoir l'ADN mitochondrial et l'ADN du chromosome Y). Cette exploration a révélé l'importance des profils de SNPs dans cette démarche, ces derniers étant des marqueurs moléculaires permettant de différencier les populations humaines.

Le travail de fin d'études s'appuie donc sur cette base en utilisant un autre algorithme plus récent et en cherchant à l'améliorer pour obtenir une prédition plus précise des origines ancestrales. En étudiant les caractéristiques de cet algorithme, le travail vise à identifier les aspects pouvant être optimisés pour une meilleure performance. Par exemple, l'extension de l'approche du stage, visant à analyser les profils de SNPs de l'ensemble des chromosomes autosomes (allant du chromosome 1 au chromosome 22) des individus, permettrait une vision plus complète des origines de ces derniers, allant au-delà des chromosomes Y et mitochondriaux seuls.

Objectifs

L'objectif de ce projet est d'identifier de manière optimale les individus d'origine ancestrale mixte à partir de leur profil de SNPs, en utilisant l'algorithme d'inférence ancestrale RYE.

Il est possible de scinder ce projet en trois grandes étapes :

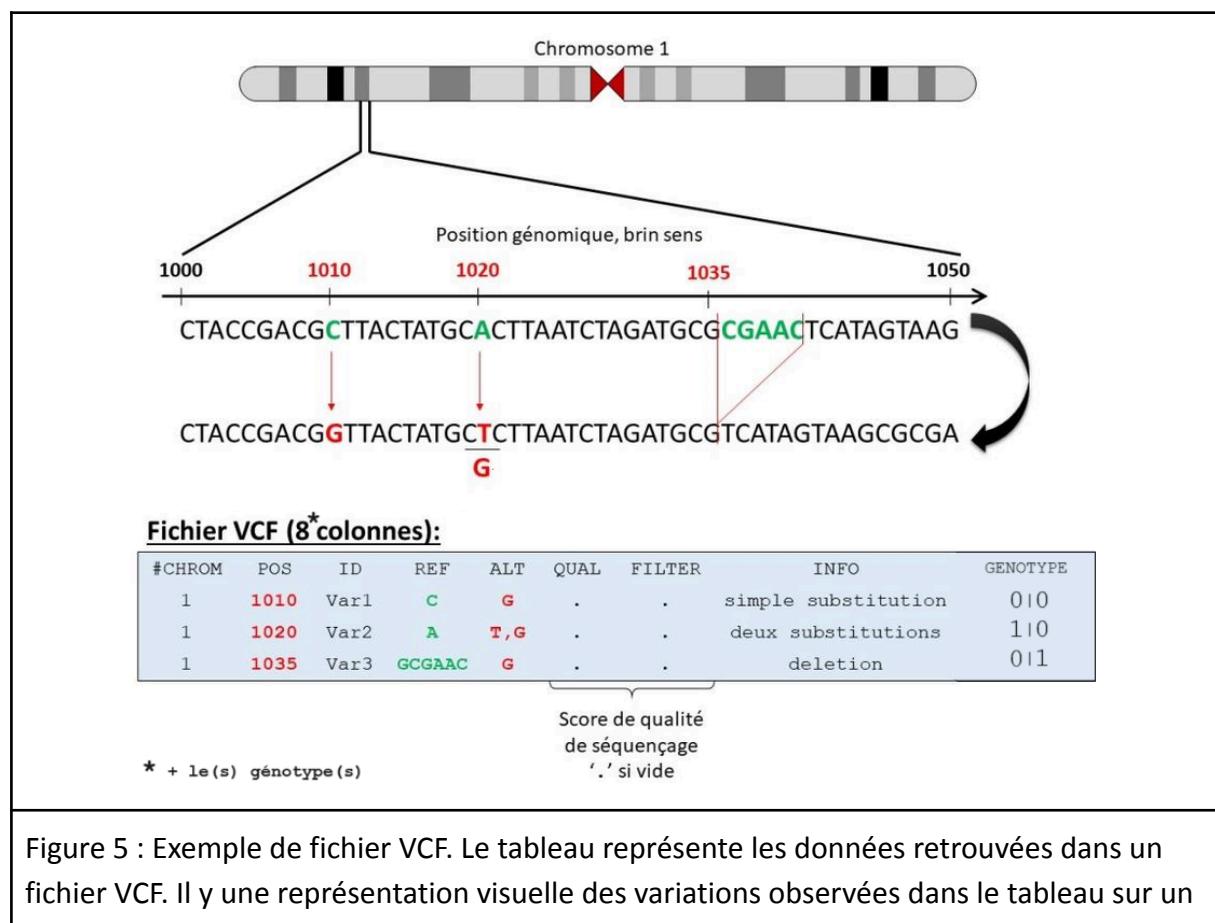
- 1) Optimisation des données d'entrées et de l'algorithme d'inférence ancestrale RYE (explications dans la section “Matériel et méthode”) sur les données PCAWG.
- 2) Évaluation comparative de la méthode (“benchmarking” en anglais) sur les données 1kGp.
- 3) Détermination de seuils à partir desquels on considère un individu étant d'origine ancestrale mixte.

Matériel et méthode

Fichier VCF (variant call format)

Dans le cadre de ce travail, l'utilisation des fichiers VCF (Variant Call Format) est essentielle pour analyser les variations de séquences génétiques d'un ou plusieurs individus (figure 5). Les fichiers VCF sont couramment utilisés en bioinformatique pour stocker des informations détaillées sur les variations génétiques, telles que les SNPs. En utilisant des fichiers VCF, il est possible de visualiser de manière complète et précise des variations génomiques présentes dans des échantillons analysés.

Chaque entrée dans le fichier comprend des données telles que la position du variant sur le génome de référence, l'identifiant du variant, les allèles observés (un allèle de référence et un allèle alterné), les informations sur la qualité du séquençage et les annotations fonctionnelles fournissant des informations sur les effets potentiels du variant sur les gènes et les protéines. Dans la colonne du génotype, pour un génome haploïde, on retrouve une série de 1 ou de 0. Les 0 signifient qu'il n'y a pas d'altération du SNP observé à une position précise (l'allèle de référence est conservé), tandis que les 1 indiquent la présence d'une variation (l'allèle est alterné).



chromosome, avec des exemples sur des séquences d'ADN fictives. La première ligne d'ADN correspond à la référence et la seconde à la séquence partiellement altérée. Image modifiée à partir de la thèse de Raphaël Leman (2019) [20].

Dans le cas d'un génome diploïde, trois génotypes sont possibles : 0|0, 1|0 (ou 0|1), et 1|1.

- 0|0 : Les deux copies du chromosome contiennent l'allèle de référence, il n'y a donc aucune variation par rapport au génome de référence à cette position.
- 1|0 (ou 0|1) : Une des copies du chromosome contient l'allèle alterné, tandis que l'autre conserve l'allèle de référence, indiquant un génotype hétérozygote pour ce SNP.
- 1|1 : Les deux copies du chromosome contiennent l'allèle alterné, indiquant une homozygotie pour l'allèle alterné.

La barre verticale ("|") entre les chiffres, indique que les allèles sont présentés en phase, ce qui signifie que l'origine de chaque allèle (maternelle ou paternelle) est connue et que cette information est conservée dans le fichier VCF. À l'inverse, une barre oblique ("/") indique que les allèles ne sont pas phasés, ce qui signifie que l'origine précise de chaque allèle n'est pas déterminée. Cette distinction entre les formats phasés et non-phasés est importante pour l'interprétation des données génétiques, car elle peut influencer la compréhension de certains mécanismes ou effets, surtout dans les analyses qui cherchent à établir des liens entre les génotypes et les phénotypes.

Dans ce travail, des génomes possédant deux phases ont été manipulés et analysés (contrairement au stage où seuls des génomes n'ayant qu'un seul génotype commun (ADNmt) ou une seule copie (ChrY) avaient été analysés) car cette fois, le travail s'effectue sur l'ensemble des chromosomes autosomiques ; il y aura donc deux chiffres dans la colonne du génotype (séparés par la barre verticale "|").

Données

1000 genomes (1kGp)

Le « 1000 genomes project » est un projet open source visant à créer un catalogue des variations génétiques humaines courantes, en se basant sur des échantillons ouvertement consentis provenant d'individus d'origines ancestrales diverses, et se déclarant en bonne santé [18]. Au total, ce sont les génomes de plus de 2500 volontaires qui ont été séquencés. Les "ethnicités" sont auto-déclarées par les personnes analysées et se divisent en 26 populations réparties entre 5 groupes continentaux (africains, américains, asiatiques de l'Est, européen et asiatiques du Sud). Les données des individus du projet

sont stockées dans des bases de données publiques incluant le NCBI (National Center for Biotechnology Information) et l'EBI (European Bioinformatics Institute). Les données sont aussi disponibles sur le site web du projet lui-même. Le projet a permis de caractériser plus de 88 millions de variations génétiques dont 84,7 millions de SNPs [21, 22].

Ces données contiennent en fait l'état des SNPs de cette multitude de sujets (figure 6). Elles peuvent être utilisées comme base pour des simulations car nous connaissons les origines ancestrales correspondant à chaque profil de SNPs.

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	HG00096	...	NA21128	NA21129
<int>	<int>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	...	<chr>	<chr>
22	16050115	rs587755077	G	A	.	PASS	.	GT	0 0	...	0 0	0 0
22	16050213	rs587654921	C	T	.	PASS	.	GT	0 0	...	0 0	0 0
22	16050607	rs587720402	G	A	.	PASS	.	GT	0 0	...	0 0	0 0
22	16050840	rs587616822	C	G	.	PASS	.	GT	0 0	...	0 0	0 0
22	16050783	rs587743568	A	G	.	PASS	.	GT	0 0	...	0 0	0 0

Figure 6 : Fichier VCF 1kGp sur l'ADN du chromosome 22. Contient une colonne par génotype d'individu. Chaque ligne représente les SNPs identifiés dans le 1kGp.

Dans cet exemple (figure 6), les “22” dans la colonne “#CHROM” signifient qu'il s'agit de données provenant du chromosome 22. Les colonnes suivantes représentent pour chaque SNP la position, l'id, l'allèle de référence, l'allèle alterné et la qualité de la variation génétique (fiabilité et précision de la variation). La colonne “FILTER” contient des “pass” signifiant “réussi”. Cela indique que le variant a répondu aux critères de qualité et qu'il est donc considéré comme fiable ou validé. L'autre possibilité ici serait de voir des “fa” signifiant que le variant n'est pas fiable. Les “GT” de la colonne “FORMAT” indiquent qu'il s'agit de génotypes. Dans ce fichier, il y a une colonne de génotype par patient et chacune d'entre elles donnent le statut de l'altération par valeurs binaires, séparées par des barres verticales (comme expliqué dans la section du VCF).

PCAWG

Le PCAWG (Pan Cancer Analysis Of Whole Genomes) représente une initiative collaborative de grande envergure qui vise à analyser les caractéristiques génétiques communes à différents types de cancer. Cette collaboration internationale rassemble des chercheurs du monde entier pour étudier près de 2800 génomes entiers de patients atteints de divers types de cancer [19].

Ce projet mené dans le cadre du consortium international du génome du cancer (ICGC), à pour but de comprendre les bases génétiques sous-jacentes aux différents types de cancer, ainsi que d'identifier les mutations génétiques associées à la progression de la maladie et à la réponse aux traitements. En étudiant ces données, les chercheurs espèrent découvrir de nouveaux marqueurs génétiques de la maladie et de potentielles nouvelles pistes pour le développement de traitements.

Les données de génotypes phasés des patients PCAWG (générées par Maxime Tarabichi à partir des données brutes) contiennent, comme pour les données du 1kGp, le génotype des SNPs de patients dont, contrairement aux données de 1kGP, on ne connaît pas les origines ethniques (figure 7).

CHR	SNP_ID	REF	ALT	cda1a403- 16b6-487c- a82a- c377d1d0f89d	448fe471- 3f4e-4dc8- a4e0- 6f147dc93abe	7d2a22eb- 7344-4cba- ad7d- 94c3f9ef3d7c
<int>	<int>	<chr>	<chr>	<chr>	<chr>	<chr>
22	16050115	G	A	0 0	0 0	0 0
22	16050213	C	T	0 0	0 0	0 0
22	16050607	G	A	0 0	0 0	0 0
22	16050783	A	G	0 0	0 0	0 0
22	16050840	C	G	0 0	0 0	0 0
22	16050847	T	C	0 0	0 0	0 0

Figure 7 : Fichier VCF PCAWG sur le chromosome 22. Chaque ligne représente un SNP identifié dans le projet PCAWG. Les colonnes du fichier sont : CHR pour le numéro du chromosome où le variant est situé. SNP_ID : est la position du chromosome utilisé comme ID pour l'identifier. REF est l'allèle de référence (la séquence d'ADN de base). ALT est l'allèle alternatif (la séquence d'ADN modifiée). Les colonnes suivantes représentent les génotypes des échantillons pour chaque SNP, où chaque valeur est un génotype au format "allèle1|allèle2" pour les individus correspondants.

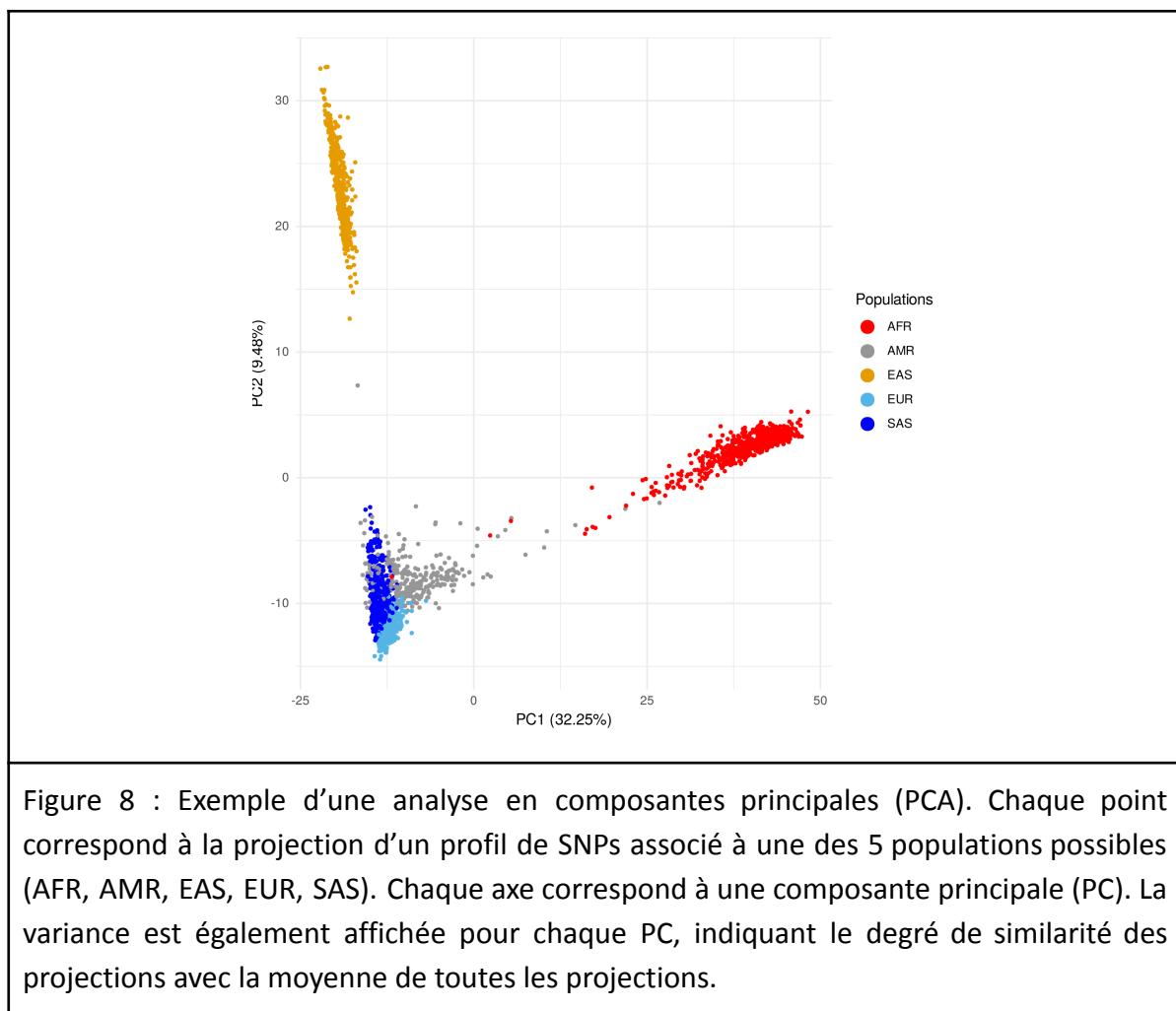
Méthodes

PCA

L'analyse en composantes principales (PCA) est une méthode statistique utilisée pour résumer la structure complexe d'un ensemble de données multidimensionnelles. Elle permet de réduire le nombre de dimensions des données tout en préservant au mieux les informations essentielles contenues dans celles-ci. En utilisant des procédés

mathématiques tels que les calculs vectoriels et la diagonalisation, la PCA identifie les axes de variation principaux présents dans les données, puis projette les observations originales sur ces nouveaux axes. Cela permet de représenter les données de manière plus concise tout en minimisant la perte d'information.

L'utilisation de la PCA sur les données "1000 Genomes" permet de transformer les données de SNPs en un ensemble de composantes principales, qui sont des combinaisons linéaires de ces données. Bien que la PCA soit souvent utilisée pour visualiser les populations et leurs liens génétiques, dans ce cas, les analyses effectuées n'ont pas produit de graphiques de visualisation spécifiques. Cependant, la figure 8 présente un exemple de PCA visuelle pour illustrer comment ces analyses peuvent représenter les regroupements génétiques entre individus. Les résultats des PCA, sous forme de fichiers vecteurs "eigenvec" et valeurs "eigenval", sont utilisés comme inputs pour l'algorithme Rye. Ces fichiers contiennent respectivement les vecteurs propres et les valeurs propres, représentant les axes de variation principaux et leurs significations, qui sont nécessaires pour les analyses ultérieures.



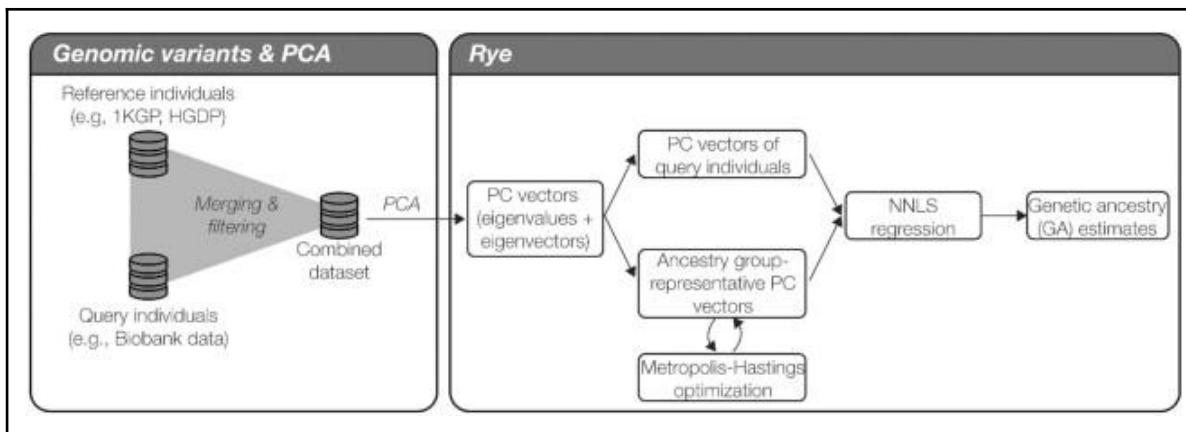
Les PCAs ont été effectuées en R avec la commande :

```
./plink --vcf input_file.vcf.gz --pca --out output_file_pca_1kGp_PCAWG_chrN
```

Cette commande a été appliquée sur les fichiers VCF contenant les profils de SNPs pour calculer les composantes principales. Bien que les graphiques issus de ces données n'aient pas été générés, des fichiers nommés “eigenvec” et “eigenval”, essentiels pour l'analyse avec l'algorithme Rye, ont été créés grâce à cette commande.

Rye

Le script Rye (annexe 6) repose sur une série de fonctions et de procédures qui, ensemble, permettent de réaliser des estimations précises des proportions d'ascendance à partir de données génomiques [23]. Ce processus implique plusieurs étapes (figure 9) et utilise des techniques statistiques avancées pour garantir la fiabilité des résultats.



Avant d'utiliser Rye, un premier script commence par transformer les données fournies en entrée, c'est-à-dire les profils de SNPs des individus à analyser (figure 10), avec une PCA. Les valeurs binaires de SNPs deviennent alors des vecteurs (eigenvectors) et valeurs propres (eigenvalues). Cependant, le fichier eigenvector généré par l'outil Plink ne peut pas être utilisé directement par Rye. Il est nécessaire de le modifier pour y intégrer les informations d'ethnicité des individus. Plus précisément, les individus des données d'entraînement sont associés à leur population d'origine, tandis que les patients dont l'ascendance est inconnue, sont identifiés par un préfixe unique. Cette étape garantit que l'algorithme Rye dispose des bonnes annotations ethniques pour effectuer les prédictions d'ascendance à partir des données de PCA. Une fois ces ajustements effectués, l'algorithme est prêt à analyser les données.

A)											
CHR	SNP_ID	REF	ALT	Patient1	Patient2	Patient3	Patient4	Patient5	Patient6	...	
<int>	<int>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	...
6	134085319	C	T	1 1	0 1	0 1	1 1	0 1	0 1	...	
4	178417106	G	A	1 1	0 1	0 1	0 1	1 1	1 1	...	
4	178417192	G	T	1 1	0 1	0 1	0 1	1 1	1 1	...	
4	55752570	A	G	0 0	0 0	0 0	0 0	1 0	0 0	...	
4	4447064	T	C	1 1	0 1	0 1	1 1	0 1	0 1	...	
4	135274980	T	C	0 1	0 1	1 1	0 1	1 1	0 1	...	

B)											
ID	CHROM	POS	REF	ALT	HG00096	HG00097	HG00099	HG00100	HG00101	...	
<chr>	<int>	<int>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	...
rs1000025	6	134085319	C	T	0 0	0 0	0 0	0 0	0 0	0 0	...
rs10000254	4	178417106	G	A	0 0	1 0	0 0	0 0	0 0	0 0	...
rs10000360	4	178417192	G	T	0 0	1 0	0 0	0 0	0 0	0 0	...
rs10000365	4	55752570	A	G	1 0	0 1	1 1	0 1	0 0	...	
rs10000397	4	4447064	T	C	0 0	0 0	0 0	0 0	0 1	0 0	...
rs10000427	4	135274980	T	C	1 0	0 1	0 0	0 1	0 0	0 0	...

Figure 10 : Exemple d'input (image A : ensemble de test ; image B : ensemble d'entraînement) pour la PCA. Dans l'ensemble de test, la colonne CHR indique sur quel chromosome le SNP se trouve, SNP_ID correspond à la position du SNP sur le chromosome, utilisée ici comme ID car unique dans l'ensemble des données, REF est l'allèle de référence et ALT l'allèle alterné. Les colonnes suivantes sont les états de SNPs des sujets à analyser. Pour l'ensemble d'entraînement, la colonne ID correspond à l'identifiant du SNP, CHROM indique sur quel chromosome le SNP se trouve, POS est à la position du SNP sur le chromosome, REF est l'allèle de référence et ALT l'allèle alterné. Les colonnes suivantes sont les états de SNPs des sujets de référence.

Rye commence par une étape de préparation où les données génomiques, sous forme de vecteurs et valeurs propres, sont lues et normalisées. Cette normalisation permet d'assurer que les différentes caractéristiques des données sont comparables, permettant ainsi une analyse cohérente et précise.

Ensuite, l'algorithme calcule les moyennes des vecteurs caractéristiques des populations de référence. Pour ce faire, il applique une méthode de rétrécissement qui ajuste ces moyennes en fonction d'un paramètre alpha. Ce rétrécissement aide à corriger les biais potentiels et à atténuer les effets des valeurs extrêmes, en favorisant une estimation plus robuste des moyennes de population.

L'étape suivante implique l'utilisation de la méthode des moindres carrés non négatifs (NNLS) pour prédire les proportions d'ascendance. Cette méthode est utilisée pour estimer les proportions en garantissant que les valeurs obtenues sont non négatives et qu'elles s'additionnent correctement. Cela permet de fournir des estimations cohérentes et现实的 des proportions d'ascendance pour chaque individu.

Pour affiner ces estimations, l'algorithme utilise une méthode d'optimisation basée sur l'échantillonnage de Gibbs. Cette technique itérative ajuste progressivement les paramètres alpha et les poids des caractéristiques, cherchant à minimiser l'erreur de prédiction. Au cours de nombreuses itérations, les paramètres sont ajustés pour converger vers des valeurs qui minimisent l'écart entre les valeurs prédites et les valeurs attendues.

Une fois les paramètres optimisés, l'algorithme effectue une série de tentatives d'optimisation globale. Ces tentatives combinent plusieurs "rounds" d'optimisation (ajustables en fonction de la qualité de résultat souhaité), utilisant des threads parallèles pour améliorer l'efficacité du processus. Cette approche permet d'explorer plusieurs configurations de paramètres, augmentant ainsi les chances de trouver l'optimisation la plus précise.

Enfin, après avoir trouvé les paramètres optimaux, l'algorithme calcule les estimations finales des proportions d'ascendance pour chaque individu. Ces estimations sont ensuite sorties sous forme de fichiers, fournissant des résultats agrégés par population (figure 11).

	EAS	SAS	AFR	EUR	AMR
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Patient1	0	0.008658379	0.4624527	0.5288890	0
Patient2	0	0.030059346	0.4987922	0.4711485	0
Patient3	0	0.0000000000	0.5148611	0.4851389	0
Patient4	0	0.0000000000	0.4678313	0.5321687	0
Patient5	0	0.015960010	0.5138737	0.4701663	0
Patient6	0	0.0000000000	0.4725405	0.5274595	0

Figure 11 : Exemple d'output généré par Rye. Chaque colonne correspond à une origine : EAS = Est-asiatique, SAS = Sud-asiatique, AFR = africaine, EUR = européenne, AMR = américaine. Chaque ligne correspond à un individu analysé. Les nombres dans le tableau vont de 0 à 1 et représentent la probabilité que la personne observée soit d'une certaine origine.

En résumé, l'algorithme Rye est conçu pour analyser des données génomiques vastes et complexes, en fournissant des estimations précises des proportions d'ascendance. À travers des étapes successives de normalisation, de régularisation, de prédiction et d'optimisation, Rye garantit une analyse robuste et fiable, capable de s'adapter à des données génétiques variées et potentiellement biaisées.

Résultats

Pour répondre à la problématique de l'identification optimale des individus d'origine ancestrale mixte à partir de leur profil de SNPs, trois étapes clés vont être mises en œuvre. Tout d'abord, la méthode pour obtenir les données d'entrée de la PCA va être optimisée. Ensuite, la robustesse de l'algorithme Rye va être évaluée par un benchmarking sur les données du projet 1kGp, permettant de connaître le nombre optimal de SNPs discriminants à prendre en compte. Enfin, une plage de données sera déterminée pour classer un individu comme "ancestralement mixte". Les résultats obtenus au cours de ces étapes sont détaillés ci-dessous.

Optimisation des données d'entrée et de RYE

Comme expliqué précédemment, Rye nécessite des inputs spécifiques résultant d'une PCA. Ceux-ci sont obtenus grâce à un script utilisé par le laboratoire qui effectue spécifiquement les analyses en composantes principales nécessaires. Cependant, un défaut majeur avait été identifié : le script exécutait une PCA distincte pour chaque patient à analyser, ce qui générait autant d'inputs pour Rye que de patients. Étant donné que le même ensemble d'entraînement est utilisé pour chaque analyse, toutes les PCA effectuées étaient identiques, entraînant une redondance inutile.

L'objectif était donc de modifier le script afin qu'une seule et unique PCA soit réalisée et recyclée pour toutes les analyses afin de gagner considérablement en temps de travail et en efficacité.

Voici les modifications apportées aux scripts (toutes les lignes de code actualisées ainsi que leur ancien équivalent sont dans les codes en annexes 5 et 6) :

1) Fonction "get_pca" du script permettant d'obtenir les inputs et de lancer Rye

L'ancienne version de la fonction traitait un seul individu à la fois, et le fichier VCF était généré individuellement pour chacun d'entre eux. La nouvelle version de la fonction traite plusieurs individus simultanément. L'ordre des colonnes a été modifié et le fichier VCF est maintenant généré pour tous les patients en une seule fois.

2) Fonction "get_rye_predictions" du script permettant d'obtenir les inputs et de lancer Rye

La première version de la fonction générait des prédictions pour chaque individu individuellement et enregistrait les résultats dans des fichiers distincts pour chacun d'entre eux. Dans la nouvelle version, plusieurs individus sont traités simultanément. Les prédictions sont stockées dans un seul fichier texte.

3) Appel de fonction “res_prediction” du script permettant d’obtenir les inputs et de lancer Rye

Dans l’ancienne version, le processus appelant les différentes fonctions était exécuté en parallèle pour chaque patient séparément à l’aide de “mclapply”. Dans la nouvelle version, le processus est exécuté une seule fois pour tous les individus.

4) Modification mineure de Rye

Voici les modifications apportées au script de l’algorithme Rye afin de rendre rendre le traitement simultané de plusieurs patients possible :

```
referenceGroups = pop2group$Group[-length(pop2group$Group)]
names(referenceGroups) = pop2group$Pop[-length(pop2group$Group)]
```

Ces lignes sont devenues :

```
referenceGroups = pop2group$Group
names(referenceGroups) = pop2group$Pop
```

Initialement, le script prenait uniquement en compte le dernier élément de chaque colonne du fichier pop2group_file pour définir les groupes de référence, ce qui pouvait limiter la capacité à traiter toutes les données correctement. La modification permet d’inclure l’ensemble des groupes et des populations de référence pour le traitement.

Avant ces changements, chaque analyse PCA-Rye prenant un peu plus d'une heure, et avec plus de 2200 patients à analyser (ensemble de patients PCAWG), le temps théorique requis était de minimum 2200 heures (soit un peu plus de 91 jours non-stop). Grâce aux modifications apportées, le temps de calcul est désormais d'environ 1h30, quel que soit le nombre de patients analysés (soit le même temps que pour un seul patient).

L'annexe 1 montre l'entièreté du code utilisé pour faire fonctionner Rye à partir des données 1kGp et PCAWG. Les modifications apportées ont permis d'éliminer les PCA redondantes et inutiles. En réalisant une seule PCA initiale et en réutilisant ces résultats pour toutes les analyses subséquentes, le script est désormais beaucoup plus performant. Ce changement réduit non seulement le temps de traitement, mais améliore la cohérence des analyses en utilisant une PCA unique comme base pour tous les patients.

Évaluation et optimisation de la méthode

Maintenant que l'analyse était prête à être réalisée dans un délai optimal, l'étape suivante consistait à optimiser le paramètre du nombre de SNPs considérés.

Plus le nombre de SNPs est grand, plus l'analyse PCA-Rye prend du temps. Afin de déterminer le nombre optimal de SNPs nécessaire pour obtenir des résultats fiables sans analyser un nombre excessif de SNPs, l'efficacité de la méthode a été évaluée en faisant varier le nombre de SNPs considérés. L'objectif est d'identifier le nombre de SNPs à partir duquel l'ajout de SNPs supplémentaires n'améliore plus les performances de la méthode. Le but était de dresser des courbes représentant la précision moyenne des prédictions de l'algorithme en fonction du nombre de SNPs testés. Elles se concentrent sur une combinaison de deux origines (par exemple la combinaison d'origines EUR-AFR). Plusieurs étapes ont été nécessaires pour obtenir cette information.

La première étape a consisté à établir un score pour trier les SNPs en fonction de leur pouvoir discriminant. Ce score a permis de hiérarchiser les SNPs selon leur capacité à distinguer entre différentes origines ancestrales, facilitant ainsi la sélection des SNPs les plus informatifs.

Ensuite, des simulations de profils de SNPs ont été créées pour tester l'algorithme dans des conditions contrôlées. Ces profils simulés ont permis de vérifier la robustesse de l'algorithme Rye face à différents nombres de SNPs, allant du plus discriminant au moins discriminant.

L'étape suivante a impliqué l'obtention des prédictions sur ces profils simulés. En appliquant l'algorithme Rye à ces données simulées, l'influence du nombre de SNPs sur la précision des prédictions d'origine ancestrale a pu être observée. Les performances de l'algorithme ont alors été comparées selon différentes tailles d'ensemble de SNPs. Cette analyse a permis de déterminer le point optimal où l'ajout de SNPs supplémentaires n'améliore plus significativement la précision, fournissant ainsi des directives claires pour l'utilisation de Rye.

Ces étapes, détaillées dans les sections suivantes, ont été cruciales pour optimiser les performances de l'algorithme Rye et garantir des prédictions d'origine ancestrale fiables et précises.

Établissement d'un score pour trier les SNPs en fonction de leur pouvoir discriminant

Afin de rendre les analyses plus rapides, tous les SNPs n'ont pas été pris en compte. En effet, la recherche scientifique a jusqu'à aujourd'hui permis de recenser près de 150

millions de SNPs dans le génome humain [24]. Les données disponibles au laboratoire contiennent environ 80 millions de ces SNPs. Réaliser une analyse sur autant de données n'est bien évidemment pas possible rapidement, même avec du matériel extrêmement performant. Les SNPs utilisés pour prédire les probabilités ethniques n'ont donc pas été choisis au hasard. Un score basé sur la fréquence allélique a été élaboré afin de trier ces variations selon leur degré de discrimination quant à la représentation d'une origine précise.

Le score est calculé selon cette formule :

$$\left(e^{(AF_{max1})} / \sum e^{(AF)} \right) - \left(e^{(AF_{max2})} / \sum e^{(AF)} \right)$$

Dans cette formule, AFmax1 est la plus grande fréquence alléliques, AFmax2 est la deuxième plus grande fréquence alléliques, et $\sum e^{(AF)}$ est la somme des exponentielles de toutes les fréquences alléliques. Pour chaque SNP, les deux plus hautes fréquences alléliques sont extraites des données. Ces fréquences sont ensuite transformées en exponentielles et normalisées en les divisant par la somme des exponentielles de toutes les fréquences alléliques de ce SNP. Après cette normalisation, la différence entre ces deux valeurs est calculée pour produire le score final.

Ce calcul permet de quantifier la différence de dominance entre les deux fréquences alléliques les plus élevées en utilisant une transformation exponentielle pour amplifier ces différences. Le score final reflète ainsi mieux la disparité entre les deux meilleures fréquences alléliques, surtout lorsque l'une est nettement plus élevée que les autres.

Par exemple, voici des fréquences alléliques possibles pour un SNP :

$$EAS_AF = 0 ; AFR_AF = 0.2 ; AMR_AF = 0.1 ; EUR_AF = 0.8 ; SAS_AF = 0$$

Ce SNP va alors être associé à l'origine européenne avec un score de 0.1532541, car il s'agit de l'origine avec la fréquence allélique la plus haute, qui est donc l'origine la plus représentative du SNP.

Voici à présent un second exemple avec des fréquences alléliques plus proches :

$$EAS_AF = 0.2 ; AFR_AF = 0.2 ; AMR_AF = 0.2 ; EUR_AF = 0.3 ; SAS_AF = 0.2$$

Ce SNP va à nouveau être associé à l'origine européenne avec cette fois-ci un score de 0.02060086. Ces deux exemples illustrent l'efficacité de la formule : dans le premier cas, où la fréquence allélique européenne est nettement plus élevée, le score attribué est bien

plus discriminant que dans le second cas, où les fréquences alléliques sont plus équilibrées, ce qui se reflète par un score plus faible.

Les 80 millions de SNPs disponibles sont répartis sur des fichiers VCF différents, un fichier par chromosome (figure 12), tandis que les fréquences alléliques sont stockées dans d'autres fichiers séparés (figure 13).

Afin de trier les SNPs en fonction de leur degré de discrimination, un score est attribué individuellement à chacun d'entre eux. Les SNPs sont ensuite classés par ordre décroissant en fonction de ce score, ce qui permet d'identifier ceux qui sont les plus discriminants pour une origine donnée.

#CHROM	POS	ID	REF	ALT	HG00096	HG00097	HG00099	HG00100	HG00101	...
<int>	<int>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	...
1	22	16050115	rs587755077	G	A	0 0	0 0	0 0	0 0	0 0
2	22	16050213	rs587654921	C	T	0 0	0 0	0 0	0 0	0 0
3	22	16050607	rs587720402	G	A	0 0	0 0	0 0	0 0	0 0
4	22	16050783	rs587743568	A	G	0 0	0 0	0 0	0 0	0 0
5	22	16050840	rs587616822	C	G	0 0	0 0	0 0	0 0	0 0
6	22	16050847	rs587702478	T	C	0 0	0 0	0 0	0 0	0 0

Figure 12 : fichier VCF des SNPs du chromosome 22 (données provenants du 1kGp). La colonne #CHROM indique le chromosome, POS donne la position génomique de la variation, ID est l'id unique des SNPs, REF correspond à l'allèle de référence et ALT correspond à l'allèle alterné. Les colonnes suivantes montrent le génotype des SNPs (deux phases) pour chaque patient (2504 patients au total).

INFO
<chr>
AC=1;AF=0.000199681;AN=5008;NS=2504;DP=8012;EAS_AF=0;AMR_AF=0;AFR_AF=0;EUR_AF=0;SAS_AF=0.001;AA=.; ;VT=SNP
AC=32;AF=0.00638978;AN=5008;NS=2504;DP=11468;EAS_AF=0;AMR_AF=0.0014;AFR_AF=0.0234;EUR_AF=0;SAS_AF=0;AA=.; ;VT=SNP
AC=38;AF=0.00758786;AN=5008;NS=2504;DP=15092;EAS_AF=0;AMR_AF=0.0014;AFR_AF=0.0272;EUR_AF=0.001;SAS_AF=0;AA=.; ;VT=SNP
AC=1;AF=0.000199681;AN=5008;NS=2504;DP=22609;EAS_AF=0;AMR_AF=0.0014;AFR_AF=0;EUR_AF=0;SAS_AF=0;AA=.; ;VT=SNP
AC=1;AF=0.000199681;AN=5008;NS=2504;DP=23591;EAS_AF=0;AMR_AF=0;AFR_AF=0;EUR_AF=0.001;SAS_AF=0;AA=.; ;VT=SNP
AC=2;AF=0.000399361;AN=5008;NS=2504;DP=21258;EAS_AF=0.002;AMR_AF=0;AFR_AF=0;EUR_AF=0;SAS_AF=0;AA=.; ;VT=SNP

Figure 13 : colonne INFO du fichier VFC du chromosome 22. **AC** (allele count) nombre total d'occurrences de l'allèle alterné dans la population étudiée ; **AF** (allele frequency) fréquence de l'allèle alterné dans la population ; **AN** (allele number) nombre total d'allèles

étudiés dans la population ; **NS** (number of samples) nombre de personnes dans l'échantillon ; **DP** (depth) profondeur de séquençage pour cette variante ; **AA** (ancestral allele) allèle ancestral pour ce variant, s'il est connu ; **VT** (variant type) type de variante. Les informations à retenir sont les valeurs de **AF** pour les différentes origines (EAS_AF: fréquence allélique pour population Est-asiatique, EUR_AF: fréquence allélique pour population européenne, AFR_AF: fréquence allélique pour population africaine, AMR_AF: fréquence allélique pour population américaine, SAS_AF: fréquence allélique pour population Sud-asiatique).

Le score a dès lors été attribué à chaque SNP. Un seul fichier contenant les scores des SNPs des chromosomes 1 à 22 a été généré (figure 14).

ID	#CHROM	POS	REF	ALT	Score	Ethnie
<chr>	<int>	<int>	<chr>	<chr>	<dbl>	<chr>
rs2814778	1	159174683	T	C	0.2296365	AFR_AF
rs1871534	8	145639681	G	C	0.2156987	AFR_AF
rs58827274	4	3666494	C	G	0.2128039	AFR_AF
rs6869589	5	178626609	G	C	0.2125906	AFR_AF
rs10152250	15	29427400	T	C	0.2107404	AFR_AF
rs3768641	2	72368190	C	G	0.2097378	AFR_AF

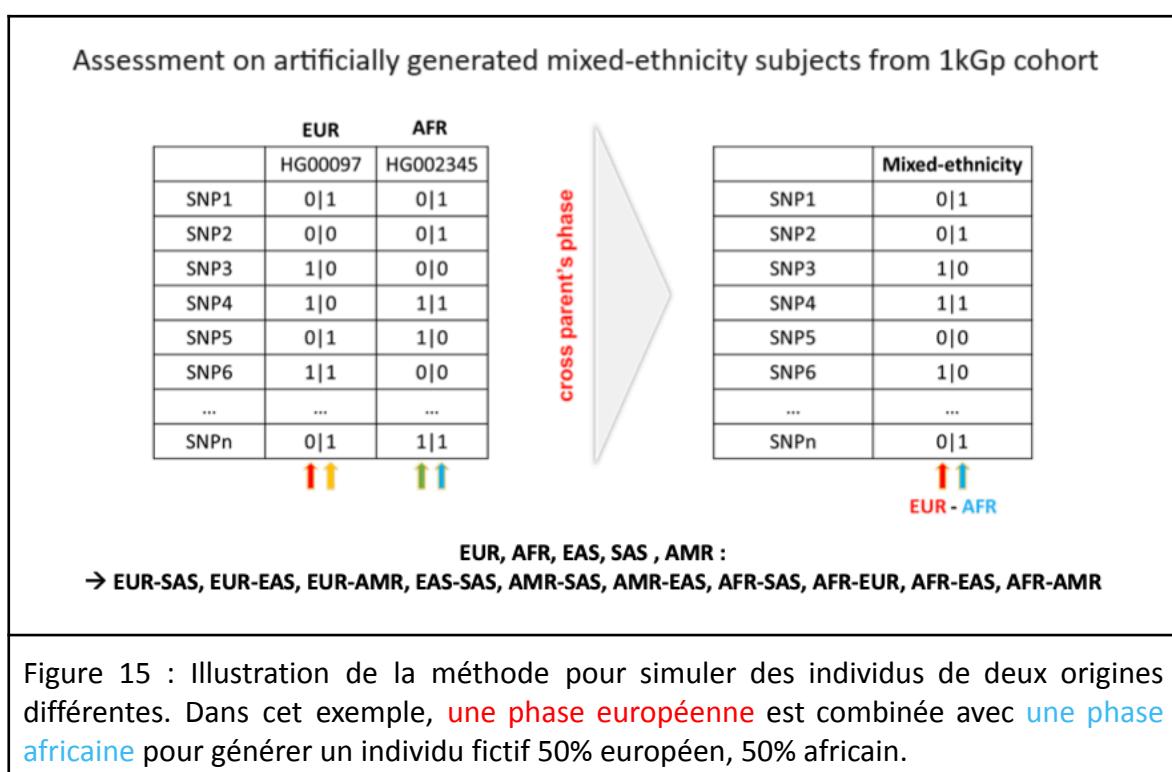
Figure 14 : colonnes “Score” et “Ethnie” ajoutées au fichier général. “Score” est le résultat de la formule expliquée précédemment pour le SNP observé, et “Ethnie” est l'origine considérée comme discriminante pour celui-ci. La colonne ID correspond à l'id unique des SNPs, #CHROM indique le chromosome, POS donne la position génomique de la variation, REF correspond à l'allèle de référence et ALT correspond à l'allèle alterné.

Simulation de profils de SNPs pour tester l'algorithme

Le but est de tester la robustesse de l'algorithme RYE en utilisant différents nombres de SNPs discriminants afin d'évaluer à partir de quelle valeur nous ne gagnons plus d'information. Ces SNPs sont en fait utilisés comme données pour la PCA qui servira d'input pour Rye. Pour chaque nombre de SNPs testés, un nombre égal de SNPs représentant chaque ethnie a été sélectionné (par exemple, dans un fichier contenant 10.000 SNPs discriminants, il y a 2.000 SNPs représentant l'origine africaine, 2.000

représentant l'origine américaine, etc.). Pour chaque origine, les SNPs les plus discriminants ont été sélectionnés sur base de leur score (0,25 étant le plus discriminant).

Pour évaluer l'efficacité de l'algorithme Rye dans l'identification des individus d'ascendance mixte, des profils de SNPs fictifs ont été générés à partir des données 1kGp (figure 15). Ces profils simulés représentent des individus hypothétiques d'origines ancestrales mixtes. Pour cela, des SNPs provenant d'individus de différentes origines ethniques connues dans les données 1kGp ont été combinés. Comme les données sont phasées, une phase de l'ADN d'un individu d'une certaine ethnie a été croisée avec une phase d'un individu d'une autre ethnie. Ce processus a été répété pour toutes les combinaisons possibles, créant ainsi des profils qui simulent des individus d'ascendances mixtes, utilisés ensuite pour tester l'algorithme Rye.



En utilisant ce principe sur le fichier de données 1kGp, 5 personnes d'une origine et 5 d'une autre ont été retirées. Les phases de chaque origine ont été séparées, donnant 10 phases pour chaque ethnie. Ensuite, chacune des phases d'une origine a été combinée avec chacune des phases de l'autre origine pour créer 100 individus d'origines mixtes. Pour chaque ensemble de test créé, un ensemble d'entraînement correspondant à la combinaison utilisée a été généré simultanément, constitué du reste du fichier initial 1kGP, sans les 10 individus utilisés pour le test set. Chaque combinaison ethnique est obtenue à partir des cinq origines disponibles (AFR, AMR, EUR, EAS et SAS). Cette approche a permis de préparer les fichiers nécessaires au test de l'algorithme (code annexe 4), pour simuler différents scénarios, et déterminer le nombre optimal de SNPs à utiliser via le benchmark.

Prédictions sur les simulations

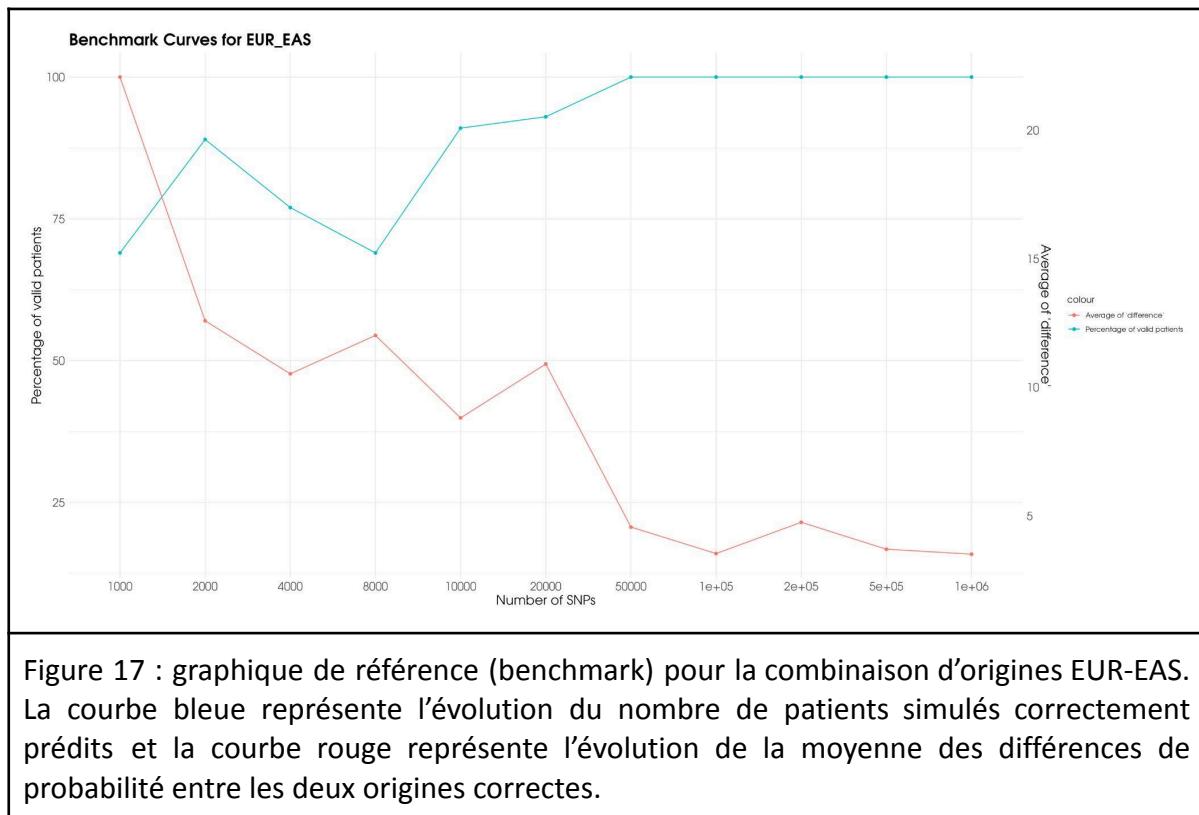
La dernière étape consistait à obtenir les prédictions de l'algorithme Rye (exemple pour 1.000 SNPs discriminants figure 16) sur les profils simulés afin de mesurer l'efficacité de Rye à prédire les origines ancestrales de patients d'origines ancestrales mixtes. Pour ce faire, les profils simulés ont été utilisés comme données d'entrée en ensemble de test, et le reste des individus 1kGp non-utilisés dans la génération des profils a été utilisé comme ensemble d'entraînement. Une PCA a alors été effectuée sur l'ensemble de test. Les résultats de cette PCA ont été modifiés et adaptés pour être utilisés comme données d'entrées dans Rye. Tous les fichiers résultants des prédictions sur les simulations ont ensuite été analysés pour évaluer les performances de l'algorithme (code annexe 5).

	EAS	SAS	AFR	EUR	AMR
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Patient1	0.013870742	0.00000000000	0.5173469	0.4327415	0.03604091
Patient2	0.00000000000	0.00000000000	0.5125527	0.4874473	0.000000000
Patient3	0.009772467	0.00000000000	0.5810411	0.3883569	0.02082956
Patient4	0.001993749	0.00000000000	0.5191656	0.4788407	0.000000000
Patient5	0.013675977	0.0001511258	0.5443337	0.4156642	0.02617501
Patient6	0.030768851	0.00000000000	0.5625790	0.3914748	0.01517738
Patient7	0.00000000000	0.00000000000	0.4553396	0.4684469	0.07621351
Patient8	0.035611966	0.00000000000	0.5648131	0.3995750	0.000000000
Patient9	0.021326619	0.00000000000	0.5510078	0.4276655	0.000000000
Patient10	0.011498114	0.00000000000	0.5598786	0.4286233	0.000000000

Figure 16 : exemple de prédictions obtenues sur 1.000 SNPs après avoir utilisé Rye sur les profils créés lors de l'étape précédente (AFR-EUR). Chaque colonne correspond à une origine, et les valeurs qui s'y trouvent correspondent aux probabilités que le patient observé soit de l'origine concernée.

Cet exemple montre une partie des résultats de prédictions sur la simulation de patients d'origines ancestrales africaines et européennes. Les résultats semblent corrects dans l'ensemble, la plus grande partie des probabilités correspond soit à l'origine européenne soit à l'origine africaine. Puisque deux origines ont été mélangées, chacune de ces deux dernières présente une probabilité proche de 50% (0,5 dans le tableau figure 16).

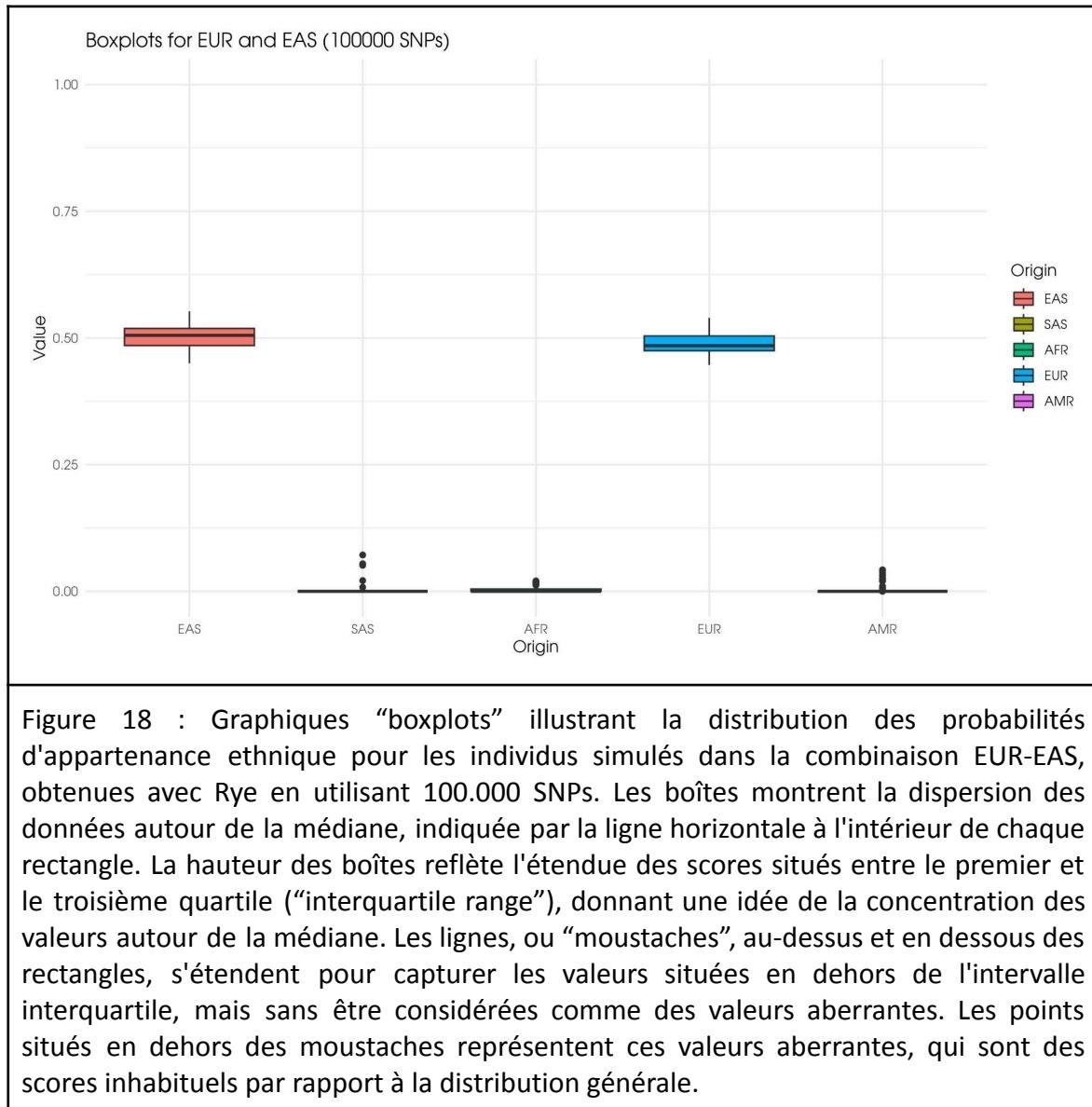
Une technique de validation des individus testés a été élaborée pour savoir combien de personnes avaient été prédites correctement par l'algorithme. Tout d'abord, les données ont été lues et traitées afin de vérifier que les plus hautes probabilités correspondaient bien aux deux origines constituant le profil fictif. Ensuite, ces deux probabilités ont été manipulées de deux manières. Pour être considérés comme valides, leur addition devait au moins valoir 0,8 et leur soustraction en valeur absolue ne devait dépasser 0,3. Un individu simulé est dès lors considéré comme correctement prédict lorsqu'il remplit ces critères: les origines sont correctement prédites, avec un haut taux de probabilité, et une différence réduite entre celles-ci. Ces critères permettent de s'assurer qu'il n'y ait pas une troisième origine avec un haut taux de probabilité et que la personne analysée tend à être 50% d'une origine et 50% d'une autre. Pour chaque nombre de SNPs testé, le nombre de sujets prédits correctement (considérés comme valides), ainsi que la moyenne des différences entre les origines correctes, ont été calculées et sont représentées dans l'annexe 7. La figure 17 représente les résultats de la combinaison EUR-EAS.



Ces résultats montrent que le nombre d'individus correctement prédits augmente progressivement avec le nombre de SNPs, tandis que la différence entre les résultats des prédictions correctes diminue. Un plateau apparaît autour de 100.000 SNPs discriminants, suggérant que ce nombre est optimal pour les analyses ultérieures. La faible différence entre les prédictions pour ce nombre de SNPs confirme également ce choix. 100.000 SNPs est donc le nombre de SNPs optimal retenu. De plus, ces graphiques montrent que les

combinaisons impliquant l'origine américaine sont bien moins bonnes et constantes, rejoignant ce qui avait été prouvé dans le stage. En effet, la diversité génétique des peuples américains étant très riche à cause de leur histoire (guerres, migrations, commerce, ...), les prédictions sont impactées négativement, rendant l'obtention de résultats corrects sur ceux-ci compliquée.

Afin de compléter l'analyse des prédictions et visualiser celles-ci, des graphiques de “boîtes à moustaches” (ou “boxplots” en anglais) ont été réalisés pour visualiser les scores obtenus par Rye avec 100.000 SNPs sur les individus simulés (figure 18 - annexe 8). Ces boxplots montrent la distribution des différentes probabilités prédictives par RYE pour les différentes combinaisons d'origines ancestrales.



La figure 18 montre les prédictions de la combinaison EUR-EAS. Ce graphique montre des distributions autour de 0,5 pour EUR et EAS et proches de 0 pour les autres origines ancestrales, ce qui illustre la précision du modèle pour cette combinaison. Ce genre de représentations permet de visualiser clairement la capacité de Rye à attribuer des scores cohérents aux individus simulés, en montrant comment les scores se regroupent autour des valeurs attendues pour les origines mixtes.

Rye prédit donc correctement les origines des individus simulés, avec des prédictions proches des 50%-50% (sauf pour les américains).

Seuils pour identifier les individus d'origines ancestrales mixtes

Maintenant que le nombre de SNPs optimal pour lequel RYE permet de prédire avec précision les origines ancestrales a été trouvé, la dernière étape consistait à déterminer une plage de valeurs au sein de laquelle un patient sera considéré comme ayant une ascendance mixte. Pour cela, il fallait trouver deux seuils (“thresholds” en anglais), un inférieur et un supérieur.

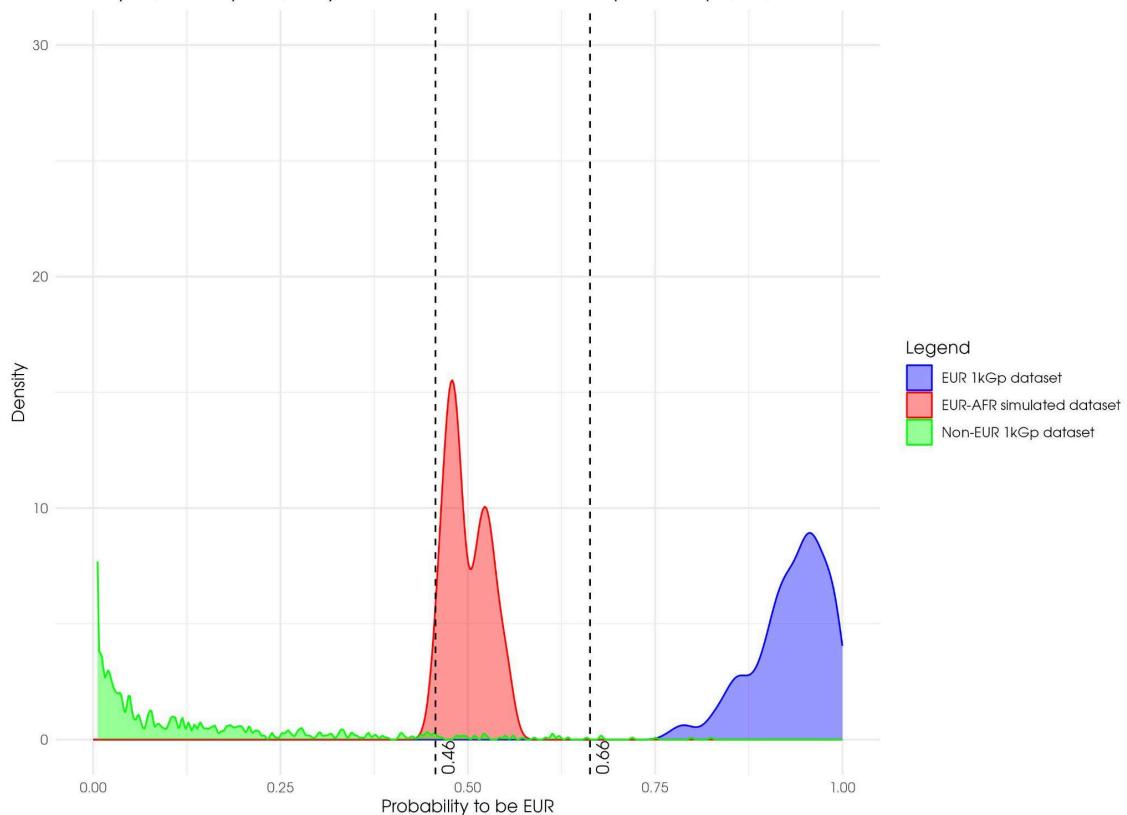
Pour obtenir ces seuils, plusieurs résultats obtenus avec Rye ont été représentés graphiquement. Pour chaque combinaison d'origines ancestrales, deux graphiques ont été créés, un pour chaque origine (A et B). Par exemple pour l'origine A, le graphique montre :

- 1) La densité de la probabilité d'ascendance A pour les individus 1kGP (ensemble d'entraînement) dont l'origine est différente de A.
- 2) La densité de la probabilité d'ascendance A pour les individus 1kGP (ensemble d'entraînement) purement d'origine A.
- 3) La densité de la probabilité d'ascendance A pour les individus simulés (ensemble de test) d'origine mixte A-B.

Le même principe est appliqué pour l'origine B. Les graphiques obtenus sont donc des graphiques de densité représentant ces trois ensembles (figure 19) (autres graphiques en annexe 9).

A)

Density curves of probability to be EUR on EUR-AFR and pure 1kGp datasets



B)

Density curves of probability to be AFR on EUR-AFR and pure 1kGp datasets

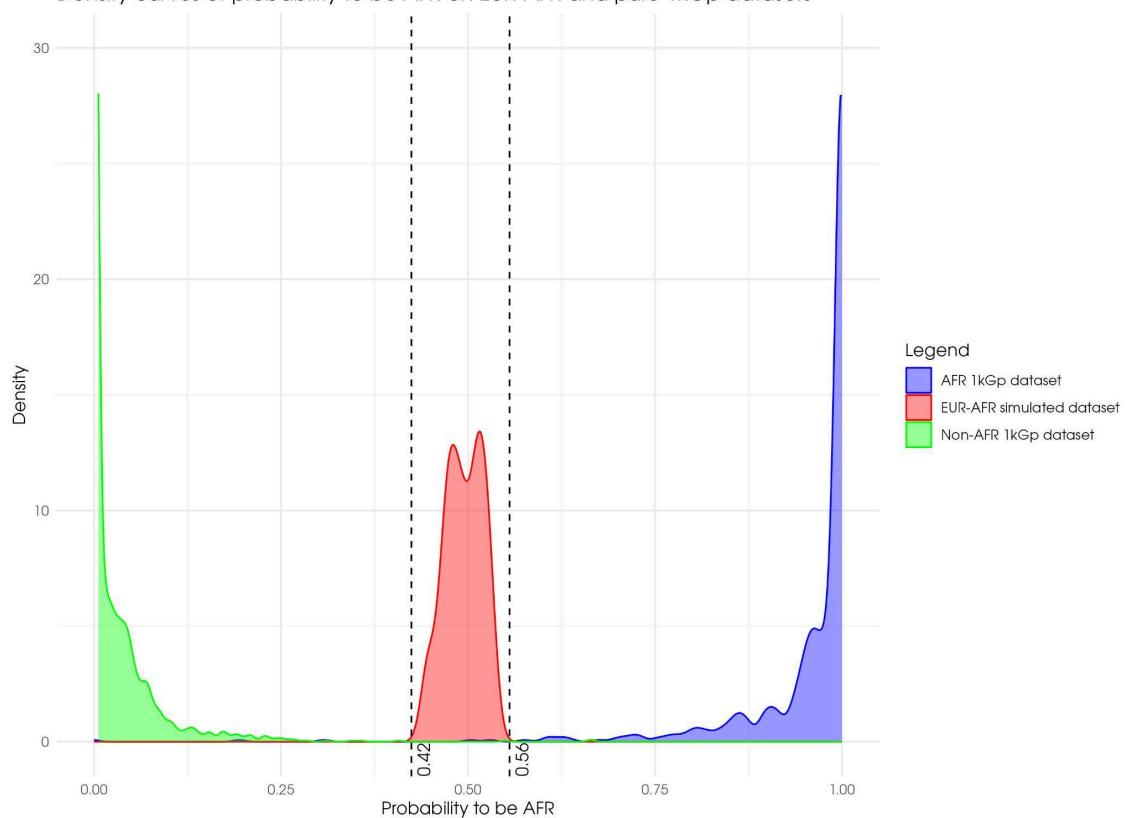


Figure 19 : Graphiques de densités des individus d'origine européenne (image A) et africaine (image B) sur différents ensembles, pour la combinaison d'origines EUR-AFR. Ces graphiques sont des graphiques en bâtonnets dont les sommets ont été reliés pour visualiser la densité sous forme de courbe. Les aires sous les courbes ont été coloriées pour plus de visibilité. La courbe rouge correspond à la densité d'individus mi-européens mi-africains observée dans l'ensemble d'individus simulés, la courbe bleue représente la densité d'individus dits "purs" pour l'origine observée dans l'ensemble 1kGp, et la courbe verte représente la densité d'individus dits "purs" pour les origines non-observées dans l'ensemble 1kGp. Les lignes pointillées indiquent les seuils définis pour classer les individus selon leur probabilité d'appartenance à une origine ethnique ou à un mélange de deux origines. Attention, ces graphiques ont été limités à une valeur de 30 sur l'axe y pour améliorer la visibilité. Les courbes verte et bleue ont pu être rognées.

Les probabilités affichées sur l'axe horizontal de ces graphiques correspondent aux probabilités d'appartenance à une origine ethnique donnée, calculées pour chaque individu. Une valeur proche de 0 indique une faible probabilité d'appartenance à l'origine observée (et donc une probabilité élevée d'appartenance à une autre origine), tandis qu'une valeur proche de 1 indique une forte probabilité d'appartenance à cette origine. Une valeur autour de 0,5 représente une situation intermédiaire, où l'individu présente une probabilité équilibrée d'appartenance à l'origine observée, suggérant un mélange d'origines ethniques. Ici, les ensembles étudiés se situent dans les zones de probabilités attendues : les individus n'appartenant pas à l'origine observée sont proches de 0, les individus appartenant à l'origine observée sont proches de 1, et les individus simulés avec des origines mixtes se situent autour de 0,5. Cela est cohérent puisque, pour simuler les profils, une phase d'une origine et une phase d'une autre origine ont été associées, générant ainsi un profil 50/50. La première conclusion de ces résultats est que Rye calcule correctement les probabilités d'appartenance ethnique des individus simulés. Les autres graphiques sont globalement bons avec des distributions bien distinctes, seuls ceux impliquant des américains présentent des courbes incohérentes.

Les seuils utilisés pour classifier les individus comme "ancestralement mixtes" sont déterminés en fonction des probabilités d'appartenance ethnique. Par exemple, un individu dont la probabilité d'appartenance à l'origine EUR est située entre 0,46 et 0,66, et la probabilité d'appartenance à l'origine AFR est située entre 0,42 et 0,56 peut être considéré comme "mixte" dans le contexte de la combinaison EUR-AFR. Ces seuils sont établis à partir de courbes ROC (pour "Receiver Operating Characteristic" en anglais). L'objectif est de déterminer deux seuils distincts à partir des distributions de densité des individus en fonction de leur origine ancestrale. Le premier seuil est destiné à différencier la densité des individus d'origines ancestrales mixtes simulées de celle des individus appartenant à l'ethnie correspondante (figure 20A). Le second seuil vise à séparer la

densité des individus d'origines ancestrales mixtes simulées de celle des individus qui ne correspondent pas à l'ethnie en question (figure 20B).

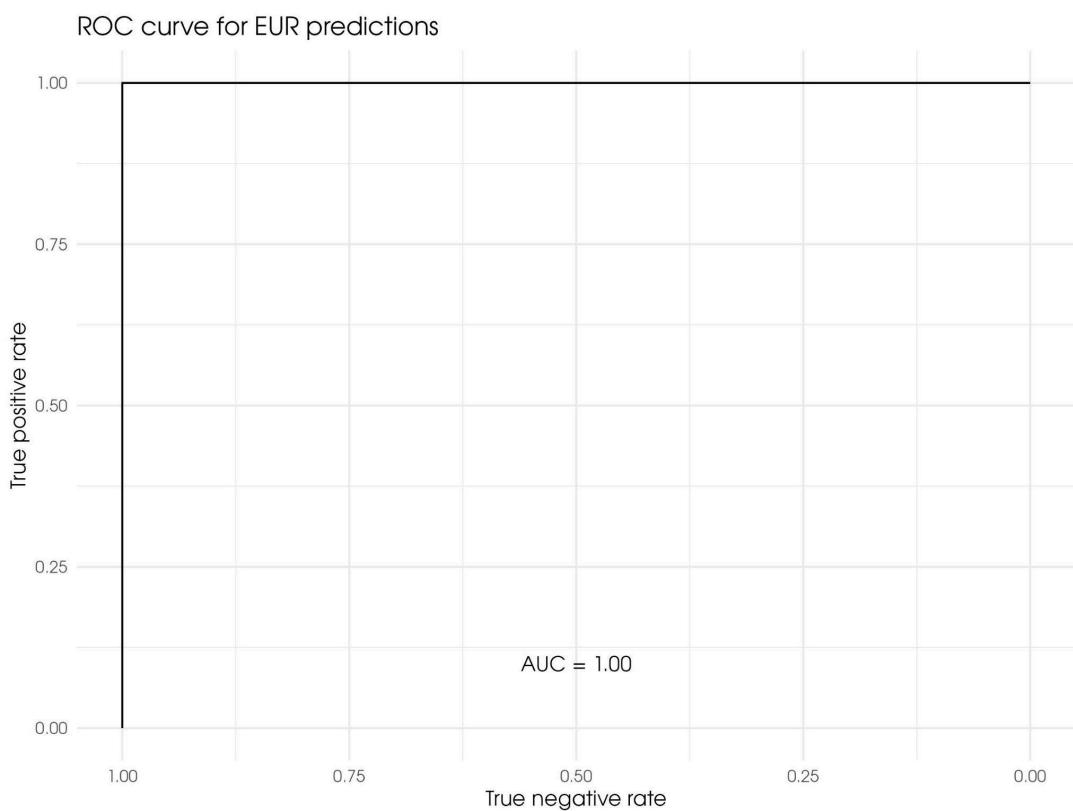
Les courbes ROC ont donc été calculées dans chaque scénario. Ces courbes permettent d'identifier les points (ou seuils) maximisant la sensibilité "Se" (c'est-à-dire, dans l'exemple de la figure 19A, la capacité du modèle à identifier correctement les individus européens), et la spécificité "Sp" (la capacité à exclure correctement les individus non-européens), dans les différentes distributions de densités. Cette pratique vise à maximiser l'indice de Youden (J) [25], obtenu avec la formule suivante :

$$J = Se + Sp - 1$$

Les points maximisant cet indice sont situés dans les coins supérieurs gauches de la courbe ROC et représentent le seuil optimal où le compromis entre les faux positifs et les faux négatifs est le meilleur. C'est donc à ce point que la détermination d'un seuil permettant de distinguer les individus d'origine ancestralement mixtes des autres est la plus parfaite. L'aire sous la courbe (AUC ou "Area Under the Curve" en anglais) donne une indication sur la qualité du seuil obtenu. Plus l'AUC tend vers 1, plus le seuil est bon.

En d'autres termes, les seuils sont choisis pour maximiser la capacité de Rye à classer correctement les individus mixtes tout en minimisant les erreurs de classification tels que les faux positifs et faux négatifs. Ces seuils permettent ainsi de distinguer de manière précise les individus "mixtes" des "purs" en se basant sur leur probabilité de mélange ethnique.

A)



B)

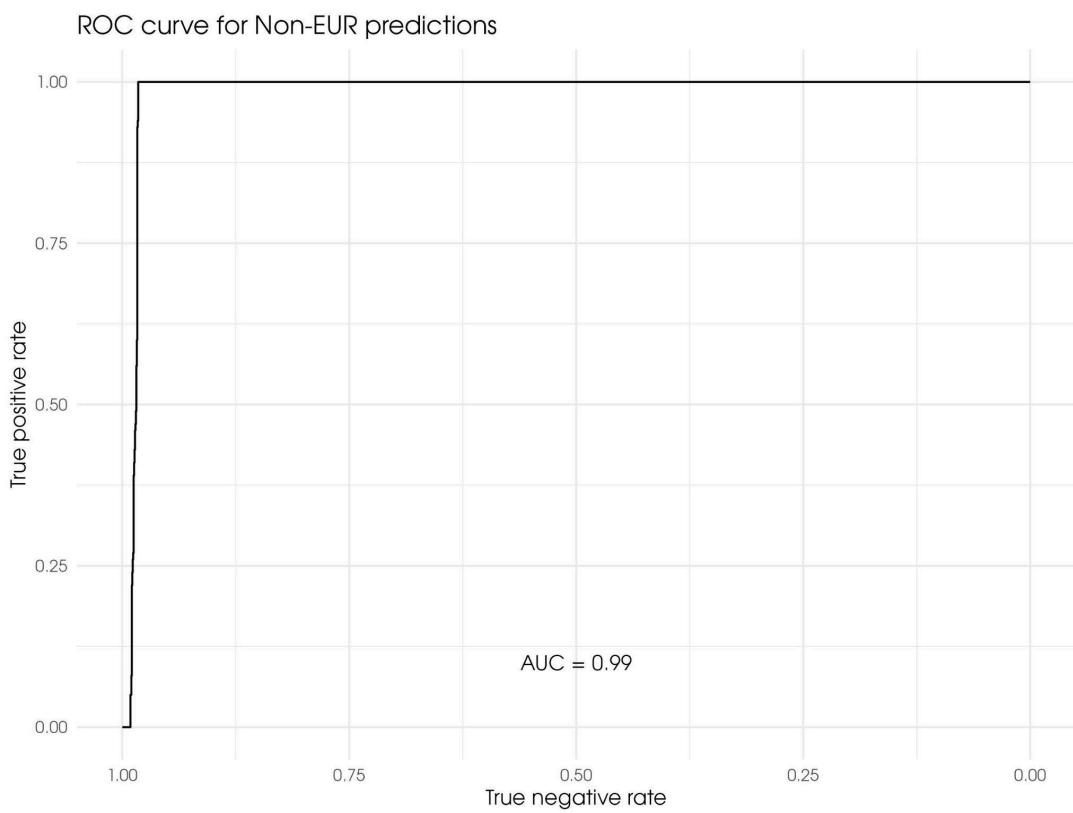


Figure 20 : Courbes ROC pour la classification des individus européens (image A) et non-européens (image B) dans la combinaison EUR-AFR. La courbe ROC illustre la capacité du modèle Rye à distinguer entre les deux groupes d'origine.

La figure 20A permet d'identifier le seuil supérieur de la plage de valeurs dans laquelle une personne sera considérée comme mixte. Sur celle-ci, la courbe ROC des individus européens montre un angle droit parfait dans le coin supérieur gauche, indiquant une AUC de 1. Cela signifie que la détermination du seuil est parfaite pour la classification des individus mixtes européens dans la combinaison EUR-AFR, sans erreurs de faux positifs ou de faux négatifs, car les deux distributions ne se chevauchent pas du tout. La figure 20B quant à elle, permet d'identifier le seuil inférieur de la plage de valeurs. Sur celle-ci, la courbe ROC des individus non-européens a une AUC de 0,99, avec une légère déviation de la courbe vers la droite lors de la montée vers le coin supérieur gauche. Cette légère déviation indique que la détermination du seuil est presque parfaite pour la classification des individus non-européens, avec une très faible proportion d'erreurs, car les ensembles se chevauchent un petit peu.

Tous les intervalles ainsi formés permettent de classifier les individus comme "purs" ou "mixtes". Par exemple, le seuil supérieur, déterminé à partir de la courbe ROC des individus européens (figure 20A), correspond à la valeur maximale de probabilité pour laquelle un individu est considéré comme mixte européen. Inversement, le seuil inférieur, déterminé à partir de la courbe ROC des individus non-européens (image B), correspond à la valeur minimale pour laquelle un individu est considéré comme mixte européen. En dessous du seuil inférieur l'individu est considéré comme non-européen et au-dessus du seuil supérieur, il est considéré comme purement européen. Les autres plages de valeurs ont été calculées et référencées dans un tableau (tableau 1) :

	AFR	AMR	EAS	EUR	SAS
EUR-AFR	0.42 - 0.56	NA	NA	0.46 - 0.66	NA
EUR-AMR	NA	0.13 - 0.56	NA	0.16 - 0.88	NA
EUR-EAS	NA	NA	0.32 - 0.65	0.45 - 0.67	NA
EUR-SAS	NA	NA	NA	0.35 - 0.60	0.24 - 0.61
AFR-AMR	0.12 - 0.55	0.00 - 0.26	NA	NA	NA
AFR-EAS	0.35 - 0.55	NA	0.43 - 0.72	NA	NA
AFR-SAS	0.34 - 0.54	NA	NA	NA	0.22 - 0.60
AMR-EAS	NA	0.18 - 0.73	0.06 - 0.71	NA	NA

AMR-SAS	NA	0.20 - 0.59	NA	NA	0.23 - 0.69
EAS-SAS	NA	NA	0.35 - 0.71	NA	0.28 - 0.64

Tableau 1 : Ensemble des intervalles de probabilité (“seuils” ou “thresholds” en anglais) utilisés pour classer les individus comme mixtes pour différentes combinaisons d’origines ethniques. Chaque case contient un intervalle de valeurs, correspondant à la plage dans laquelle un individu est considéré comme ayant une ascendance mixte pour l’origine observée (dans la colonne), entre les deux groupes ethniques spécifiés par la ligne. Les origines présentes sont les origines africaines (AFR), américaines (AMR), Est-asiatiques (EAS), européennes (EUR) et Sud-asiatiques (SAS). Par exemple, la première cellule (croisement de la ligne EUR-AFR et de la colonne AFR) montre des valeurs entre 0,42 et 0,56. Cela signifie que les individus AFR observés dans la combinaison EUR-AFR sont considérés comme mixte AFR si la probabilité d’être AFR retournée par l’algorithme est entre 0,42 et 0,56. Les “NA” signifie que l’origine déterminée par la colonne n’est pas observable dans la combinaison d’origines déterminée par la ligne.

Les plages de valeurs impliquant l’origine américaine (AMR) sont plus étendues que les autres. Cela montre à nouveau le caractère aléatoire et instable de cette origine.

Application aux données PCAWG

Pour terminer, la méthode a été appliquée sur les données de la cohorte de patients PCAWG. Les plages de valeurs déterminées au point précédent ont été utilisées sur les résultats obtenus pour déterminer combien d’entre eux étaient considérés comme ancestralement mixtes (tableau 2).

	EAS	SAS	AFR	EUR	AMR
Patient 1	0	0.007931462	0	0.99206854	0
Patient 2	0.997598672	0	0.002401328	0	0
Patient 3	0.024690643	0.016248244	0.861013509	0.07547788	0.02256973
Patient 4	0.039908446	0.416023385	0.01172920	0.5306930	0.001645949
Patient 5	0.003079028	0.250931286	0.11124479	0.5916964	0.043048466

Tableau 2 : Résultats des prédictions de Rye sur un échantillon de 5 patients PCAWG. Les valeurs **vertes** sont celles qui représentent une probabilité ethnique élevée chez les individus observés. Ces patients sont alors considérés comme purs, car une origine est prépondérante et que la valeur correspondante ne se situe pas dans les plages de valeurs du tableau 1. Les valeurs **rouges** représentent les deux plus hautes probabilités chez les individus n’ayant pas qu’une seule origine prédominante. Si ces deux origines sont comprises dans les plages de valeurs du tableau 1, alors ces patients sont considérés comme ancestralement mixtes.

Sur les 2246 patients PCAWG analysés, un groupe de 28 patients a été identifié comme ayant une ascendance mixte. Ces individus présentent des valeurs de probabilité pour deux origines distinctes qui se situent entre le seuil supérieur et le seuil inférieur déterminés par les ROC curves. Ce résultat montre que la plupart des patients référencés dans le projet PCAWG sont des individus purs pour une origine précise et donc non mixtes.

Conclusion

Dans le cadre de sa thèse, Maxime Lefèvre a pour projet de mesurer les différences de taux de mutations somatiques entre les copies parentales du génome. Pour ce faire, il est nécessaire d'identifier les individus d'origines ancestrales mixtes dont chaque copie du génome représente une origine ancestrale. Cela nécessite d'analyser les variations génétiques dans l'intégralité des chromosomes autosomiques et d'attribuer une origine ethnique à chaque profil de variation avant de se concentrer sur l'attribution d'une origine parentale.

Pour inférer les origines ancestrales des individus, il est nécessaire d'utiliser des algorithmes spécifiques. Cependant, ces algorithmes, comme Rye qui utilisé dans ce travail, peuvent être longs et complexes à exécuter. Afin d'obtenir des prédictions précises, fiables et rapides, plusieurs améliorations ont été développées. Cela inclut l'optimisation du script qui génère les données de PCA, servant d'entrée pour Rye, ainsi que des tests visant à identifier le meilleur nombre de SNPs discriminants à utiliser pour obtenir des résultats optimaux. Une analyse a ensuite été menée pour déterminer les plages de valeurs propres à chaque combinaison d'origines ancestrales, permettant de classifier un individu comme ancestralement mixte.

L'optimisation du script pour générer les données d'entrée de Rye (données de PCA) a d'abord permis d'accélérer les prédictions sur un grand nombre d'individus simultanément. Plutôt que de réaliser une PCA pour chaque personne individuellement, une seule PCA est effectuée pour l'ensemble des sujets à analyser, entraînant un gain de temps considérable dans le traitement de grands nombre d'individus. Ensuite, le nombre optimal de SNPs discriminants a été déterminé en évaluant la précision de RYE à identifier des individus d'origines ancestrales mixtes simulés à partir de différents nombres de SNPs. Ces résultats, ont révélé un plateau dans la courbe du nombre d'individus correctement prédits aux alentours de 100.000 SNPs. Ces tests ont également montré une faible différence entre les probabilités que ces individus appartiennent à l'une des deux origines correctes autour de cette valeur, confirmant ainsi qu'il s'agit du point optimal où les deux origines étaient le mieux détectées. Enfin, l'analyse finale a permis de définir des plages de valeurs spécifiques pour chaque combinaison d'origine (EUR-AFR, EUR-AMR, ...), permettant ainsi de déterminer si un individu est considéré comme "pure" avec des traces d'autres origines, ou comme ancestralement mixte. Ces plages seront très utiles pour les analyses futures menées au laboratoire.

Ce travail va permettre d'aider Maxime Lefèvre dans sa thèse car la méthode développée est une amélioration de celle qu'il a élaborée au début de son travail. Sa technique probabiliste permettait de déterminer les différentes origines des individus sans obtenir les valeurs de probabilités qui y étaient associées. La nouvelle méthode permet désormais une identification plus précise avec le détail des probabilités.

Perspectives

Ce travail de fin d'études donne lieu à diverses perspectives pour les recherches futures. Tout d'abord, l'identification des individus à ascendance mixte dans la cohorte PCAWG montre l'importance d'une analyse plus approfondie de l'impact génétique de ces populations, notamment dans le contexte d'études sur le cancer. Une meilleure compréhension des caractéristiques biologiques spécifiques à ces individus pourrait révéler des éléments importants sur la prédisposition au cancer et la réponse aux traitements, en fonction de la diversité génétique.

Ensuite, bien que l'application du modèle Rye aux différentes combinaisons d'origines ethniques ait démontré sa robustesse, cette approche pourrait être optimisée pour déterminer plus précisément la complexité génétique des individus. À l'avenir, l'intégration de données provenant de populations plus diversifiées permettrait de renforcer la précision des classifications et de mieux comprendre les interactions entre les différentes origines.

Enfin, il serait pertinent d'appliquer cette méthode sur d'autres cohortes de patients atteints de cancer. L'application de Rye à d'autres ensembles de données variés permettrait de tester la généralisation des résultats obtenus avec l'ensemble PCAWG. Cette extension offrirait une validation plus robuste de la méthode.

Bibliographie

1. Vaissaud, S. (s. d.). *L'ULB, un esprit libre depuis 1834.* ULB.
<https://www.ulb.be/fr/l-universite>
2. Rédaction, L. (s. d.). SNP : qu'est-ce que c'est ? Futura.
<https://www.futura-sciences.com/sante/definitions/genetique-snp-6348/>
3. Garvan Institute of Medical Research. (s. d.). <https://www.garvan.org.au/>
4. National Institutes of Health (US); Biological Sciences Curriculum Study. NIH Curriculum Supplement Series [Internet]. Bethesda (MD): National Institutes of Health (US); 2007. Understanding Human Genetic Variation. Available from:
<https://www.ncbi.nlm.nih.gov/books/NBK20363/>
5. Korzeniewski, S., Hofman, P., & Brest, P. (2013). Des polymorphismes silencieux plutôt bruyants. MéDecine/Sciences/MS. MéDecine Sciences, 29(2), 124-126.
<https://doi.org/10.1051/medsci/2013292003>
6. Gaspar, H.A., Breen, G. Probabilistic ancestry maps: a method to assess and visualize population substructures in genetics. BMC Bioinformatics 20, 116 (2019).
<https://doi.org/10.1186/s12859-019-2680-1>
7. Arbre généalogique, généalogie, antécédents familiaux et tests ADN gratuits. (s. d.). MyHeritage. <https://www.myheritage.fr/>
8. 23andMe. (s. d.). DNA genetic testing for health, ancestry and more - 23andMe.
<https://www.23andme.com/>
9. AncestryDNA® | DNA Tests for Ethnicity & Genealogy DNA Test. (s. d.).
<https://www.ancestry.com/dna/>
10. The Genographic Project® Geno 2.0 Next Generation Helix Product Privacy Policy. (s. d.). Pages. <https://www.nationalgeographic.com/pages/article/genographic>
11. AncestrySupport. (s. d.).
https://support.ancestry.com/s/article/Unexpected-Ethnicity-Results?language=en_US
12. Stoppa-Lyonnet, D. & Lyonnet, S. (2017). Chapitre VII - Génétique, évolution, population. Dans : Dominique Stoppa-Lyonnet éd., *Les 100 mots de la génétique* (pp. 91-104). Paris cedex 14: Presses Universitaires de France.
13. J C Long, The genetic structure of admixed populations., *Genetics*, Volume 127, Issue 2, 1 February 1991, Pages 417–428, <https://doi.org/10.1093/genetics/127.2.417>
14. Thibault Leroy. Estimations et déterminants de la diversité génétique. Génétique des populations [q-bio.PE]. Université d'Angers, 2022.
15. Pascal Angst, Camille Ameline, Christoph R Haag, Frida Ben-Ami, Dieter Ebert, Peter D Fields, La dérive génétique façonne l'évolution d'une métapopulation hautement dynamique, *Biologie moléculaire et évolution*, Volume 39, Numéro 12, Décembre 2022, msac264, <https://doi.org/10.1093/molbev/msac264>
16. Gregory, T.R. Understanding Natural Selection: Essential Concepts and Common Misconceptions. *Evo Edu Outreach* 2, 156–175 (2009).
<https://doi.org/10.1007/s12052-009-0128-1>

17. Inchingolo, F.; Martelli, F.S.; Gargiulo Isacco, C.; Borsani, E.; Cantore, S.; Corcioli, F.; Boddi, A.; Nguyễn, K.C.D.; De Vito, D.; Aityan, S.K.; et al. *Chronic Periodontitis and Immunity, Towards the Implementation of a Personalized Medicine: A Translational Research on Gene Single Nucleotide Polymorphisms (SNPs) Linked to Chronic Oral Dysbiosis in 96 Caucasian Patients*. *Biomedicines* **2020**, *8*, 115.
<https://doi.org/10.3390/biomedicines8050115>
18. 1000 Genomes / A Deep Catalog of Human Genetic Variation. (s. d.).
<https://www.internationalgenome.org/>
19. The ICGC/TCGA Pan-Cancer Analysis of Whole Genomes Consortium. Pan-cancer analysis of whole genomes. *Nature* **578**, 82–93 (2020).
<https://doi.org/10.1038/s41586-020-1969-6>
20. Raphaël, L. (2019). Développement d'outils biostatistiques et bioinformatiques de prédiction et d'analyse des défauts de... *ResearchGate*.
https://www.researchgate.net/publication/338829076_Développement_d'outils_biostatistiques_et_bioinformatiques_de_prediction_et_d'analyse_des_defauts_de_l'épissage_application_aux_genes_de_predisposition_aux_cancers_du_sein_et_de_l'ovaire
21. Devuyst O. (2015). The 1000 Genomes Project: Welcome to a New World. *Peritoneal dialysis international : journal of the International Society for Peritoneal Dialysis*, *35*(7), 676–677. <https://doi.org/10.3747/pdi.2015.00261>
22. 1000 Genomes Project Consortium, Auton, A., Brooks, L. D., Durbin, R. M., Garrison, E. P., Kang, H. M., Korbel, J. O., Marchini, J. L., McCarthy, S., McVean, G. A., & Abecasis, G. R. (2015). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74.
<https://doi.org/10.1038/nature15393>
23. Conley AB, Rishishwar L, Ahmad M, Sharma S, Norris ET, Jordan IK, Mariño-Ramírez L. Rye: genetic ancestry inference at biobank scale. *Nucleic Acids Res.* **2023 May**; *51*(8):e44. doi: 10.1093/nar/gkad149. PMID: 36928108; PMCID: PMC10164567.
24. Jordan, B. (2017). Variants fréquents et rares, caractères multigéniques et héritabilité perdue. *Médecine/Sciences/MS. Médecine Sciences*, *33*(6–7), 674–676.
<https://doi.org/10.1051/medsci/20173306028>
25. Delacour, H., Servonnet, A., Perrot, A., Vigezzi, J. F., Ramirez, J. M., Laboratoire de biochimie, toxicologie cliniques, Hôpital d'Instruction des Armées du Val-de-Grâce, Paris, & Laboratoire de biochimie, toxicologie cliniques, Hôpital d'Instruction des Armées Robert Picqué, Villenave d'Ornon. (2004). La courbe ROC (receiver operating characteristic) : principes et principales applications en biologie clinique. In *La Courbe ROC (Receiver Operating Characteristic) : Principes Et Principales Applications En Biologie Clinique* (Vols. 2–2, pp. 145–154) [Journal-article].
<https://www.reseau-naissance.fr/data/mediashare/t3/pr3xql3uzbdbgvdkpemed657ohvpki-org.pdf>

Lexique

Allèles : variantes, versions d'un même gène. Par extension, les deux copies génomiques parentales, qui varient l'une de l'autre.

Autosome : chromosome non-sexuel.

Brassage génétique : processus par lequel les gènes provenant de deux individus différents sont mélangés pour former de nouvelles combinaisons génétiques chez leur descendance.

Ethnicité : identité ethnique d'un individu. Fait référence à un groupe partageant des caractéristiques culturelles ou génétiques communes.

Exome : partie du génome constituée par les exons, c'est-à-dire les parties des gènes qui permettent la synthèse de protéines.

Génotype : patrimoine héréditaire d'un individu, représente l'ensemble des allèles d'un organisme.

Haplotype : groupe d'allèles de différents gènes situés sur un même chromosome et habituellement transmis ensemble.

Mutation somatique : mutation touchant les cellules non germinales (non reproductrices).

NGS (Next Generation Sequencing) : technologie de séquençage d'ADN qui permet de séquencer rapidement et en parallèle de grandes quantités d'ADN à moindre coût.

Origine ancestrale : origine géographique des ancêtres ayant transmis une copie du génome.

Phénotype : ensemble des attributs visibles d'un individu.

Séquençage Sanger : méthode de séquençage de l'ADN basée sur la synthèse de chaînes d'ADN marquées par fluorescence.

SNP (Single Nucleotide Polymorphism) : variations mineures du génome au sein d'une population (un seul nucléotide change). La plupart du temps, ces variations sont bénignes et provoquent simplement des différences morphologiques entre les individus. Peuvent aussi être à l'origine de certaines maladies.

Variant calling : processus d'identification des variations génétiques.

Table des annexes

Annexe 1 : Code d'optimisation PCA

Annexe 2 : Code de création du fichier VCF unique contenant tous les SNPs disponibles

Annexe 3 : Code pour séparer le fichier général d'index de SNPs en petit fichier contenant les valeurs de SNPs

Annexe 4 : Code pour générer les faux profils de SNPs “mixed-ancestry”

Annexe 5 : Code pour obtenir les prédictions sur les faux profils

Annexe 6 : Script Rye

Annexe 7 : Benchmarks

Annexe 8 : Boxplots sur 100.000 SNPs discriminants

Annexe 9 : Graphiques de densités

Annexe 1 - Code d'optimisation PCA

```
# -----
# Libraries
# -----  
  
library(data.table)
library(VariantAnnotation)
library(dplyr)
library(parallel)  
  
# -----
# Constants
# -----  
  
chr = "all"
threads = 10  
  
# -----
# Pathways
# -----  
  
input_pop =
"/mnt/iribhm/homes/mlef0011/rawdata/1000G_data/Population/population.ts
v"
input_pop2subpop =
"/mnt/iribhm/homes/acle0017/output/mutation_rate_project/pop2subpop_PCA
WG.txt"
rye_exec = "/mnt/iribhm/homes/acle0017/script/Dissertation/rye.R"
input_1kgP <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/F
inal_Rye_train_set_updated_2.vcf.gz")
input_PCAWG <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/F
inal_Rye_test_set_updated_2.vcf.gz")
output_vcf_1kGp_PCAWG_patient <- "/mnt/iribhm/homes/acle0017/tmp/"
output_PCA <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/other_outputs/PC
A/Final_Rye_run")
output_rye <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/other_outputs/Ry
e_prediction/mixed-ancestry/Final_Rye_run")
```

```

# -----
# Functions
# -----


writevcf.beagle <- function(vcf, filepath, vcfversion="4.2",
genomereference="GRCh37") {
  cat(paste0('##fileformat=VCFv', vcfversion,
'\n##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">\n##reference=',
            genomereference,
            '\n'), file=filepath)
  suppressWarnings(write.table(vcf, file=filepath, sep="\t",
col.names=TRUE, row.names=FALSE, quote=FALSE, append=TRUE))
}

get_PCA <- function(patients, chr, data_PCAWG, data_1kGp) {
  data_patient_PCAWG = data_PCAWG[, c("#CHROM", "SNP_ID", "REF", "ALT",
patients)]
  colnames(data_patient_PCAWG)[2] = "POS"

  merged_data = merge(data_1kGp, data_patient_PCAWG)

  vcf_1kGp_PCAWG = cbind(merged_data[, c(1:5)],
                           QUAL = rep(".", nrow(merged_data)),
                           FILTER = rep("PASS", nrow(merged_data)),
                           INFO = rep(".", nrow(merged_data)),
                           FORMAT = rep("GT", nrow(merged_data)),
                           merged_data[, c(6:ncol(merged_data))])

  col_names <- colnames(vcf_1kGp_PCAWG)
  new_order <- c(col_names[1:2], col_names[5], col_names[3:4],
col_names[-c(1:5)])
  vcf_1kGp_PCAWG <- vcf_1kGp_PCAWG[, new_order]

  output_vcf_path <- paste0(output_vcf_1kGp_PCAWG_patient,
"vcf_1kGP_PCAWG_.vcf")
  writevcf.beagle(vcf_1kGp_PCAWG, output_vcf_path)
  system(paste("gzip -f", output_vcf_path))

  system(paste0("/mnt/iribhm/homes/mlef0011/software./plink --vcf ",
output_vcf_path, ".gz --pca --out ", output_PCA, "_chr",
as.character(chr), "_pca_1kGp_PCAWG"))
  return(paste0(output_PCA, "_chr", as.character(chr),
"_pca_1kGp_PCAWG"))
}

```

```

get_input_rye <- function(path_pca, input_pop) {
  sub_pop = fread(input_pop, sep = '\t', header = TRUE)
  sub_pop = data.frame(sub_pop = sub_pop$Population, V2 =
  sub_pop`Individual ID`)
  fullPCA = fread(paste0(path_pca, ".eigenvec"), header = FALSE)
  fullPCA = fullPCA[, 2:ncol(fullPCA)]
  data_1kGp = merge(sub_pop, fullPCA, by.x = "V2", by.y = "V2", all =
FALSE)
  data_1kGp <- data_1kGp[, c(2, 1, seq(from = 3, to =
  ncol(data_1kGp)))]
  data_PCAWG <- anti_join(fullPCA, sub_pop, by = "V2")
  data_PCAWG = cbind(paste0("NA", 1:nrow(data_PCAWG)), data_PCAWG)
  colnames(data_PCAWG) = colnames(data_1kGp)
  fullPCA_1kgP_PCAWG <- rbind(data_1kGp, data_PCAWG)
  fullPCA_1kgP_PCAWG[, 3:ncol(fullPCA_1kgP_PCAWG)] <-
  lapply(fullPCA_1kgP_PCAWG[, 3:ncol(fullPCA_1kgP_PCAWG)], as.numeric)
  write.table(fullPCA_1kgP_PCAWG, paste0(path_pca, ".eigenvec"),
  col.names = FALSE, row.names = FALSE, quote = FALSE, sep = '\t')
}

get_rye_prediction <- function(patients, chr, rye_exec, path_pca,
threads, output_rye) {
  system(paste0(rye_exec, " --eigenvec=", path_pca, ".eigenvec
--eigenval=", path_pca, ".eigenval --pop2group=", input_pop2subpop, "
--threads=", threads, " --out=", output_rye,
"rye_prediction_PCAWG_chr", as.character(chr)))

  prediction = read.table(paste0(output_rye,
"rye_prediction_PCAWG_chr", as.character(chr), "-20.5.Q"))
  prediction = prediction[patients, ]
  write.table(prediction, paste0(output_rye, "_rye_prediction.txt"),
  col.names = TRUE, row.names = TRUE, quote = FALSE, sep = '\t')
  return(prediction)
}

process <- function(patients, chr, data_PCAWG, data_1kGp, input_pop,
rye_exec, threads, output_rye) {
  pca = get_PCA(patients, chr, data_PCAWG, data_1kGp)
  get_input_rye(pca, input_pop)
  prediction = get_rye_prediction(patients, chr, rye_exec, pca,
threads, output_rye)
  return(prediction)
}

```

```
# -----
# Main code
# -----  
  
    # Read datasets  
    train_set <- fread(input_1kgP, header = TRUE, sep = '\t', data.table  
= FALSE)  
    colnames(train_set)[2] <- "#CHROM"  
    test_set <- fread(input_PCAWG, header = TRUE, sep = '\t', data.table  
= FALSE)  
    colnames(test_set)[1] <- "#CHROM"  
  
    patients <- colnames(test_set)[5:ncol(test_set)]  
  
    # Execute the process  
    res <- process(patients = patients, chr = chr, data_PCAWG = test_set,  
data_1kGp = train_set, input_pop = input_pop, rye_exec = rye_exec,  
threads = threads, output_rye = output_rye)
```

Annexe 2 - Code de création du fichier VCF d'index des SNPs

```
library(data.table)
library(VariantAnnotation)
library(dplyr)
library(parallel)
library(stringr)

# Function to calculate the result for each column of allele frequencies
calcul_resultat <- function(col) {
  exp_col <- exp(col)
  somme_exp <- sum(exp_col)
  deux_plus_hautes <- head(sort(col, decreasing = TRUE), 2)
  indices_plus_hauts <- head(order(col, decreasing = TRUE), 2)
  ethnies <- c("EAS_AF", "AFR_AF", "AMR_AF", "EUR_AF", "SAS_AF")
  ethnies_plus_haute <- ethnies[indices_plus_hauts]
  part_max1 <- exp(deux_plus_hautes[1]) / somme_exp
  part_max2 <- exp(deux_plus_hautes[2]) / somme_exp
  resultat_final <- part_max1 - part_max2
  return(list(Resultat = resultat_final, Ethnie =
ethnies_plus_haute[1]))
}

# Function to process each chromosome
process_chromosome <- function(chr) {
  vcf_1kGp <- paste0("/mnt/iribhm/homes/acle0017/rawdata/1kGp/chr",
as.character(chr), ".1kGp.phase3.v5a.vcf.gz")

  vcf_data <- fread(vcf_1kGp, skip = "##", header = TRUE, sep = "\t")
  vcf_data <- vcf_data[, c("ID", "#CHROM", "POS", "REF", "ALT",
"INFO")]

  champs_AF <- c("EAS_AF", "AFR_AF", "AMR_AF", "EUR_AF", "SAS_AF")

  # Extract allele frequencies from the INFO column for each population
  valeurs_AF <- list()
  for (champ in champs_AF) {
    valeurs_AF[[champ]] <- as.numeric(str_extract(vcf_data$INFO,
paste0("(?<=", champ, "=)[0-9.]+")))
  }
}
```

```

valeurs_AF_df <- do.call(rbind, valeurs_AF)

resultats <- apply(valeurs_AF_df, 2, calcul_resultat)

nouvelle_dataframe <- do.call(rbind, lapply(resultats, unlist))
merged_df <- cbind(vcf_data, nouvelle_dataframe)

names(merged_df)[ncol(merged_df) - 1] <- "Score"
names(merged_df)[ncol(merged_df)] <- "Ethnie"

cat(paste0("Finished processing chromosome ", as.character(chr),
"\n"))

return(merged_df)
}

# Main script

chromosomes <- 1:22
threads <- 11

# Process each chromosome in parallel using multiple cores
merged_df_list <- mclapply(chromosomes, process_chromosome, mc.cores =
threads)

# Combine all individual chromosome data frames into one
genome_wide_df <- do.call(rbind, merged_df_list)

# Print the dimensions and the first few rows of the final data frame
print(dim(genome_wide_df))
head(genome_wide_df)

# Write the combined data frame to a file
write.table(genome_wide_df, file =
"/mnt/iribhm/homes/acle0017/script/Dissertation/NEW_1kGp_1_22.txt", sep
= "\t", row.names = FALSE)

```

Annexe 3 - Code pour séparer le fichier général d'index de SNPs en petit fichier contenant les valeurs de SNPs

```
# Load necessary libraries
library(dplyr)
library(data.table)
library(class)
library(optparse)

# Define and parse command-line arguments
option_list <- list(
  make_option(c("-n", "--nbr"), type="integer", default=NULL,
              help="Number of most discriminant SNPs to keep in the
main SNPs file", metavar="number")
)
opt_parser <- OptionParser(option_list=option_list)
opt <- parse_args(opt_parser)

# Ensure the mandatory argument is provided
if (is.null(opt$nbr)) {
  stop("The argument -n/--nbr is mandatory. Please choose a number of
SNPs to keep to start the analysis.")
}

nbr_to_divide <- opt$nbr

# -----
# Functions
# -----


# Function to select the most discriminant SNPs based on the 'Score'
# column
select_best_SNPs <- function(vcf_to_cut, nbr_total_SNPs) {
  indices_tries <- order(vcf_to_cut$Score, decreasing = TRUE)
  vcf_to_cut <- vcf_to_cut[indices_tries, ]

  # Group by 'Ethnie' and select the top SNPs within each group
  df_CUT <- vcf_to_cut %>%
```

```

group_by(Ethnie) %>%
  slice_head(n = nbr_total_SNPs %% 5)

# Ungroup and arrange the rows
df_CUT <- df_CUT %>%
  ungroup() %>%
  arrange(row_number())

return(df_CUT)
}

# -----
# Main code
# -----
cat("Reading the main SNPs file")
PCAWG_general <-
read.table("/mnt/iribhm/homes/acle0017/script/Dissertation/all_1kGp.txt",
  header=TRUE, sep="\t")
cat("\nMain SNPs file read")

cat("\n\nSelecting best SNPs")
selected_SNPs <- select_best_SNPs(PCAWG_general, nbr_to_divide)
cat(paste("\nBest", nbr_to_divide, "SNPs selected"))

# Rename the 'CHROM' column and identify the maximum chromosome number
names(selected_SNPs)[2] <- "CHROM"
unique_chrom_values <- unique(selected_SNPs$CHROM)
max_unique_chrom_value <- max(unique_chrom_values)
cat(paste("\nBest SNPs found up to chromosome",
as.character(max_unique_chrom_value)))

# Remove unnecessary columns and save the selected SNPs to a file
selected_SNPs <- subset(selected_SNPs, select = -INFO)
divided_file <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/SNPs_selection_mixed_ancestry_",
  as.character(nbr_to_divide %% 1000), ".txt")
write.table(selected_SNPs, file = divided_file, sep = "\t", row.names =
FALSE, col.names = TRUE, quote = FALSE)

# Start the recovery and merging process for the best SNP values
cat(paste("\n\nRecovery of the", nbr_to_divide, "best SNP
values\nProgress :"))

execution_time <- system.time({
  df_general <- read.table(divided_file, header=TRUE, sep="\t")
  final_df <- df_general
})

```

```

chromosome_dfs <- list()

# Loop through each chromosome and process the SNP data
for (chr in 1:max_unique_chrom_value) {
  file_path <-
  paste0("/mnt/iribhm/homes/mlef0011/rawdata/1000G_data/SNP/1kGp_SNPs_chr",
  ", as.character(chr), ".txt")
  snp_data <- fread(file_path, header = TRUE, sep = "\t",
  data.table = FALSE)

  # Filter the SNP data based on the selected SNP IDs
  filtered_snp_data <- snp_data %>%
    filter(ID %in% df_general$ID) %>%
    select(-'#CHROM', -POS, -REF, -ALT)

  chromosome_dfs[[chr]] <- filtered_snp_data
  cat(paste("\nChromosome", chr, "processed"))
}

# Combine all chromosome dataframes into one large dataframe
combined_snp_data <- bind_rows(chromosome_dfs)

# Merge the combined data with the original dataframe containing
selected SNPs
final_df <- merge(final_df, combined_snp_data, all.x = TRUE)

# Write the final merged dataframe to a file
write.table(final_df, file =
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/final_set_mixed_",
ancestry_, as.character(nbr_to_divide %% 1000), ".k.txt"), sep = "\t",
row.names = FALSE, col.names = TRUE, quote = FALSE)
}

# Output the total execution time for the merging process
cat(paste("\nTotal execution time for merging data:",
execution_time["elapsed"], "seconds\n"))

```

Annexe 4 - Code pour générer les simulations de profils de SNPs “mixed-ancestry”

```
# -----
# Libraries
# -----



library(data.table)
library(dplyr)
library(tidyr)
library(optparse)

# -----
# Argument Parsing
# -----



# Define command-line options
option_list <- list(
  make_option(c("-n", "--nbr"), type="integer", default=NULL,
             help="Number of most discriminant SNPs to keep in the
SNPs file (choose a number divisible by 1000). Possible values: 1, 2,
4, 8, 10, 20, 40, 80, 200, 400, 500",
             metavar="number")
)
opt_parser <- OptionParser(option_list = option_list)
opt <- parse_args(opt_parser)

# Ensure the number of SNPs is provided
if (is.null(opt$nbr)) {
  stop("The argument -n/--nbr is mandatory. Please choose a valid
number of SNPs to keep (must be divisible by 1000)!")
}

nbr_of_SNPs_kept <- opt$nbr

# -----
# Functions
# -----



# Function to write VCF data in Beagle format
```

```

writevcf.beagle <- function(vcf, filepath, vcfversion = "4.2",
genomereference = "GRCh37") {
  cat(paste0('##fileformat=VCFv', vcfversion,
'\n##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">\n##reference=',
genomereference, '\n'),
  file = filepath)
  suppressWarnings(write.table(vcf, file = filepath, sep = "\t",
col.names = TRUE, row.names = FALSE, quote = FALSE, append = TRUE))
}

# Function to select n columns of patients at random and remove them
# from the original dataframe
select_and_remove_random_patients <- function(df, n) {
  patient_columns <- colnames(df)[6:ncol(df)]
  selected_columns <- sample(patient_columns, n)
  selected_df <- df[, c(colnames(df)[1:5], selected_columns), with =
FALSE]
  remaining_df <- df[, !(colnames(df) %in% selected_columns), with =
FALSE]
  return(list(selected_df = selected_df, remaining_df = remaining_df))
}

# Function to separate phases in patient data and create new columns
split_snp_columns <- function(df) {
  patient_columns <- colnames(df)[6:ncol(df)]

  for (col in patient_columns) {
    df[[paste0(col, "_after")]] <- sapply(strsplit(df[[col]], "\\|"),
"[[", 2)
    df[[col]] <- sapply(strsplit(df[[col]], "\\|"), "[[", 1)
  }

  return(df)
}

# Function to combine SNP columns by matching rows of two dataframes
combine_snp_columns_by_order <- function(df1, df2) {
  stopifnot(identical(df1[, 1:5], df2[, 1:5]))
  combined_df <- df1[, 1:5]
  num_patient_columns1 <- ncol(df1) - 5
  num_patient_columns2 <- ncol(df2) - 5

  # Create all combinations of SNP columns from df1 and df2
  column_combinations <- expand.grid(1:num_patient_columns1,
1:num_patient_columns2)
}

```

```

for (i in 1:nrow(column_combinations)) {
  col1_index <- column_combinations[i, 1] + 5
  col2_index <- column_combinations[i, 2] + 5
  combined_values <- paste(df1[[col1_index]], df2[[col2_index]], sep = "|")
  combined_values <- gsub(" ", "", combined_values) # Remove any spaces
  combined_df[[paste0("Patient", i)]] <- combined_values
}

return(combined_df)
}

# Function to process ethnic groups, select and transform specific columns, and write results to files
process_ethnicities <- function(onevcf_df, sample_pop, origin1, origin2, n, n_SNPs) {
  # Update ethnicity labels in the sample population
  sample_pop$Ethnicity[sample_pop$Ethnicity == origin1] <- "ORIGIN1"
  sample_pop$Ethnicity[sample_pop$Ethnicity == origin2] <- "ORIGIN2"

  # Select columns for ORIGIN1
  ORIGIN1_to_keep <- sample_pop[sample_pop$Ethnicity == "ORIGIN1", "PatientID"]
  ORIGIN1_columns <- c("#CHROM", "POS", "ID", "REF", "ALT",
  as.character(ORIGIN1_to_keep))
  ORIGIN1 <- onevcf_df[, ..ORIGIN1_columns]

  # Select columns for ORIGIN2
  ORIGIN2_to_keep <- sample_pop[sample_pop$Ethnicity == "ORIGIN2", "PatientID"]
  ORIGIN2_columns <- c("#CHROM", "POS", "ID", "REF", "ALT",
  as.character(ORIGIN2_to_keep))
  ORIGIN2 <- onevcf_df[, ..ORIGIN2_columns]

  # Randomly select and split columns for both origins
  result_ORIGIN1 <- select_and_remove_random_patients(ORIGIN1, n)
  random_ORIGIN1 <- result_ORIGIN1$selected_df
  remaining_ORIGIN1 <- result_ORIGIN1$remaining_df

  result_ORIGIN2 <- select_and_remove_random_patients(ORIGIN2, n)
  random_ORIGIN2 <- result_ORIGIN2$selected_df
  remaining_ORIGIN2 <- result_ORIGIN2$remaining_df

  # Split SNP columns for both origins
}

```

```

random_ORIGIN1 <- split_snp_columns(random_ORIGIN1)
random_ORIGIN2 <- split_snp_columns(random_ORIGIN2)

# Combine SNP columns from both origins
mixed_ancestry <- combine_snp_columns_by_order(random_ORIGIN1,
random_ORIGIN2)
mixed_ancestry <- mixed_ancestry[, -3] # Remove unnecessary columns
names(mixed_ancestry)[1:4] <- c("CHR", "SNP_ID", "REF", "ALT")

remaining_ORIGIN2 <- remaining_ORIGIN2[, -(1:5)] # Remove unwanted
columns

# Create train set excluding test set patients
all_patient_ids <- colnames(onevcf_df)[6:ncol(onevcf_df)]
test_set_ids <- colnames(random_ORIGIN1)[6:ncol(random_ORIGIN1)]
test_set_ids <- c(test_set_ids,
colnames(random_ORIGIN2)[6:ncol(random_ORIGIN2)])

train_set_ids <- setdiff(all_patient_ids, test_set_ids)
train_set_columns <- c("ID", "#CHROM", "POS", "REF", "ALT",
train_set_ids)
train_set <- onevcf_df[, ..train_set_columns]
train_set <- train_set[, !(colnames(train_set) %in% c("QUAL",
"FILTER", "INFO", "FORMAT")), with = FALSE]
colnames(train_set)[colnames(train_set) == "#CHROM"] <- "CHROM"

# Write the generated VCF files
writevcf.beagle(train_set,
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/" ,
n_SNPs, "SNPs/combined_train_set_", origin1, "_", origin2, ".vcf"),
nbr_of_SNPs_kept)
writevcf.beagle(mixed_ancestry,
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/" ,
n_SNPs, "SNPs/combined_test_set_", origin1, "_", origin2, ".vcf"),
nbr_of_SNPs_kept)

# Compress the generated VCF files
vcf_train <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/" ,
n_SNPs, "SNPs/combined_train_set_", origin1, "_", origin2, ".vcf")
vcf_test <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/" ,
n_SNPs, "SNPs/combined_test_set_", origin1, "_", origin2, ".vcf")

system(paste("gzip", vcf_train))
system(paste("gzip", vcf_test))

```

```

    return(list(mixed_ancestry = mixed_ancestry, train_set = train_set))
}

# -----
# Main Code
# -----


# Read SNP file based on the number of SNPs provided
file_path <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/final_set_mixed_
ancestry/final_set_mixed_ancestry_", nbr_of_SNPs_keped, "k.vcf.gz")
selected_SNPs <- fread(file_path, header = TRUE, sep = "\t")

# Check for any NA values in the SNP data
na_count <- sum(apply(selected_SNPs, 1, function(x) any(is.na(x))))
print(paste("Number of rows with at least one NA:", na_count))

# Load sample population data
sample_pop <-
read.table("/mnt/iribhm/homes/mlef0011/rawdata/1000G_data/Population/sa
mple_pop.txt", header = FALSE, col.names = c("PatientID", "Ethnicity"))

# Define ethnicity groups
ethnies <- c("EUR", "AFR", "AMR", "EAS", "SAS")

# Generate all combinations of two ethnicities
combinations <- combn(ethnies, 2, simplify = FALSE)

# Format the SNP count for file naming
nbr_of_SNPs_keped = format((nbr_of_SNPs_keped * 1000), scientific =
FALSE)

# Process each ethnicity combination
for (combo in combinations) {
  origin1 <- combo[1]
  origin2 <- combo[2]
  result <- process_ethnicities(selected_SNPs, sample_pop, origin1,
origin2, 5, nbr_of_SNPs_keped)
  cat("Finished processing combination:", origin1, "-", origin2, "\n")
}

```

Annexe 5 - Code pour obtenir les input (PCA) et les prédictions avec Rye

```
# -----
# Libraries
# -----



library(data.table)
library(VariantAnnotation)
library(dplyr)
library(parallel)

# -----
# Constants
# -----



chr <- "all"
threads <- 10

# -----
# Pathways
# -----



input_pop <-
"/mnt/iribhm/homes/mlef0011/rawdata/1000G_data/Population/population.ts
v"
input_pop2subpop <-
"/mnt/iribhm/homes/acle0017/output/mutation_rate_project/pop2subpop_PCA
WG.txt"
rye_exec <- "/mnt/iribhm/homes/acle0017/script/Dissertation/rye.R"

# -----
# Functions
# -----



# Function to write VCF data in Beagle format
writevcf.beagle <- function(vcf, filepath, vcfversion = "4.2",
genomereference = "GRCh37") {
  cat(paste0('##fileformat=VCFv', vcfversion,
```

```

'\n##FORMAT<ID=GT,Number=1>Type=String,Description="Genotype">\n##reference=',
            genomereference, '\n'), file = filepath)
suppressWarnings(write.table(vcf, file = filepath, sep = "\t",
col.names = TRUE, row.names = FALSE, quote = FALSE, append = TRUE))
}

# Function to perform PCA for the patient and 1000G datasets
get_PCA <- function(patients, chr, data_PCAWG, data_1kGp) {
  data_patient_PCAWG <- data_PCAWG[, c("#CHROM", "SNP_ID", "REF",
"ALT", patients)]
  colnames(data_patient_PCAWG)[2] <- "POS"

  merged_data <- merge(data_1kGp, data_patient_PCAWG)

  # Prepare VCF format for 1kGp and PCAWG combined data
  vcf_1kGp_PCAWG <- cbind(merged_data[, c(1:5)],
                           QUAL = rep(".", nrow(merged_data)),
                           FILTER = rep("PASS", nrow(merged_data)),
                           INFO = rep(".", nrow(merged_data)),
                           FORMAT = rep("GT", nrow(merged_data)),
                           merged_data[, c(6:ncol(merged_data))])

  # Reorder columns for VCF format
  col_names <- colnames(vcf_1kGp_PCAWG)
  new_order <- c(col_names[1:2], col_names[5], col_names[3:4],
col_names[-c(1:5)])
  vcf_1kGp_PCAWG <- vcf_1kGp_PCAWG[, new_order]

  # Write VCF to file and compress it
  output_vcf_path <- paste0(output_vcf_1kGp_PCAWG_patient,
"vcf_1kGP_PCAWG_.vcf")
  writevcf.beagle(vcf_1kGp_PCAWG, output_vcf_path)
  system(paste("gzip -f", output_vcf_path))

  # Run PCA using PLINK
  system(paste0("/mnt/iribhm/homes/mlef0011/software./plink --vcf ",
output_vcf_path, ".gz --pca --out ", output_PCA, "_chr",
as.character(chr), "_pca_1kGp_PCAWG"))
  return(paste0(output_PCA, "_chr", as.character(chr),
"_pca_1kGp_PCAWG"))
}

# Function before changes
#

```

```

#
# get_PCA <- function(patient, chr, data_PCAWG, data_1kGp){
#   data_patient_PCAWG =
data_PCAWG[,c("#CHROM", "POS", "REF", "ALT", patient)]
#   merged_data = merge(data_1kGp, data_patient_PCAWG)
#   vcf_1kGp_PCAWG = cbind(merged_data[,c(1:5)],
#                           QUAL = rep(".", nrow(merged_data)),
#                           FILTER = rep("PASS", nrow(merged_data)),
#                           INFO = rep(".", nrow(merged_data)),
#                           FORMAT = rep("GT", nrow(merged_data)),
#                           merged_data[,c(6:ncol(merged_data))])
#
#   col_names <- colnames(vcf_1kGp_PCAWG)
#   new_order <- c(col_names[1], col_names[2], col_names[5],
#   col_names[-c(1,2,5)])
#   vcf_1kGp_PCAWG <- vcf_1kGp_PCAWG[, new_order]
#
#   writevcf.beagle(vcf_1kGp_PCAWG,
paste0(output_vcf_1kGp_PCAWG_patient,
#                                     "vcf_1kGP_PCAWG_", patient,
".vcf"))
#
#   system(paste("gzip -f", paste0(output_vcf_1kGp_PCAWG_patient,
#                                     "vcf_1kGP_PCAWG_", patient,
".vcf")))
#
#   system(paste0("/srv/home/mlef0011/software./plink --vcf ",
#               output_vcf_1kGp_PCAWG_patient,
#               "vcf_1kGP_PCAWG_", patient, ".vcf.gz --pca --out ",
#               output_PCA, patient, "_chr", as.character(chr),
"_pca_1kGp_PCAWG"))
#
#   return(paste0(output_PCA, patient, "_chr", as.character(chr),
"_pca_1kGp_PCAWG"))
# }

# Function to prepare input for Rye using PCA results and population
data
get_input_rye <- function(path_pca, input_pop) {
  sub_pop <- fread(input_pop, sep = '\t', header = TRUE)
  sub_pop <- data.frame(sub_pop = sub_pop$Population, V2 =
sub_pop$`Individual ID`)

  # Load PCA results and merge with population data
  fullPCA <- fread(paste0(path_pca, ".eigenvec"), header = FALSE)

```

```

fullPCA <- fullPCA[, 2:ncol(fullPCA)]


# Merge population and PCA data for 1000G
data_1kGp <- merge(sub_pop, fullPCA, by.x = "V2", by.y = "V2", all =
FALSE)
data_1kGp <- data_1kGp[, c(2, 1, seq(from = 3, to =
ncol(data_1kGp)))]


# Extract PCA data for PCAWG patients
data_PCAWG <- anti_join(fullPCA, sub_pop, by = "V2")
data_PCAWG <- cbind(paste0("NA", 1:nrow(data_PCAWG)), data_PCAWG)
colnames(data_PCAWG) <- colnames(data_1kGp)


# Combine 1000G and PCAWG data
fullPCA_1kgP_PCAWG <- rbind(data_1kGp, data_PCAWG)
fullPCA_1kgP_PCAWG[, 3:ncol(fullPCA_1kgP_PCAWG)] <-
lapply(fullPCA_1kgP_PCAWG[, 3:ncol(fullPCA_1kgP_PCAWG)], as.numeric)


# Save the combined PCA data
write.table(fullPCA_1kgP_PCAWG, paste0(path_pca, ".eigenvec"),
col.names = FALSE, row.names = FALSE, quote = FALSE, sep = '\t')
}

# Function before changes
#
#
# get_rye_prediction <- function(patient, chr, rye_exec, path_pca,
threads, output_rye) {
#   system(paste0(rye_exec,
#                 " --eigenvec=", path_pca, ".eigenvec",
#                 " --eigenval=", path_pca, ".eigenval",
#                 " --pop2group=", input_pop2subpop,
#                 " --threads=", threads,
#                 " --out=", output_rye, "rye_prediction_PCAWG_",
#                 patient, "_chr", as.character(chr)))
#
#   prediction = read.table(paste0(output_rye,
#                                   "rye_prediction_PCAWG_", patient,
# "_chr",
#                                   as.character(chr), "-20.5.Q"))
#
#   prediction = prediction[patient, ]
#
#   write.table(prediction, paste0(output_rye,
#                                   "rye_prediction_", patient, "_chr",
#                                   as.character(chr), "-20.5.Q"))
}

```

```

#                                     as.character(chr), ".txt"),
#           col.names = TRUE,
#           row.names = TRUE,
#           quote = FALSE,
#           sep = '\t')
#
#   return(prediction)
# }

# Function to run Rye prediction using the PCA data and population
mapping
get_rye_prediction <- function(patients, chr, rye_exec, path_pca,
threads, output_rye) {
  # Run Rye prediction
  system(paste0(rye_exec, " --eigenvec=", path_pca, ".eigenvec
--eigenval=", path_pca, ".eigenval -pop2group=", input_pop2subpop, "
--threads=", threads, " --out=", output_rye,
"rye_prediction_PCAWG_chr", as.character(chr)))

  # Read the prediction results
  prediction <- read.table(paste0(output_rye,
"rye_prediction_PCAWG_chr", as.character(chr), "-20.5.Q"))
  prediction <- prediction[patients, ]

  # Save the prediction results
  write.table(prediction, paste0(output_rye, "_rye_prediction.txt"),
col.names = TRUE, row.names = TRUE, quote = FALSE, sep = '\t')
  return(prediction)
}

# Function to execute the entire process for ethnicity inference
process <- function(patients, chr, data_PCAWG, data_1kGp, input_pop,
rye_exec, threads, output_rye) {
  pca <- get_PCA(patients, chr, data_PCAWG, data_1kGp) # Perform PCA
  get_input_rye(pca, input_pop) # Prepare input for Rye
  prediction <- get_rye_prediction(patients, chr, rye_exec, pca,
threads, output_rye) # Get Rye prediction
  return(prediction)
}

# -----
# Main code
# -----

```

```

# Update input and output paths based on the combination
input_1kgP <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/final_train_set_LAST_RYE.vcf.gz")
input_PCAWG <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/mixed_ancestry/final_test_set_LAST_RYE.vcf.gz")
output_vcf_1kGp_PCAWG_patient <- "/mnt/iribhm/homes/acle0017/tmp/"
output_PCA <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/other_outputs/PCA/NEW_Final_Rye_run")
output_rye <-
paste0("/mnt/iribhm/homes/acle0017/script/Dissertation/other_outputs/Rye_prediction/mixed-ancestry/NEW_Final_Rye_run")

# Read training and test datasets
train_set <- fread(input_1kgP, header = TRUE, sep = '\t', data.table =
FALSE)
colnames(train_set)[2] <- "#CHROM"
test_set <- fread(input_PCAWG, header = TRUE, sep = '\t', data.table =
FALSE)
colnames(test_set)[1] <- "#CHROM"

patients <- colnames(test_set)[5:ncol(test_set)] # Get patient
identifiers from test dataset

# Execute the full prediction process
res <- process(patients = patients, chr = chr, data_PCAWG = test_set,
data_1kGp = train_set, input_pop = input_pop, rye_exec = rye_exec,
threads = threads, output_rye = output_rye)

# Execution of the prediction process before changes
#
#
# res_prediction = mclapply(patients,
#                               function(patient) process(patient, chr,
# data_PCAWG, data_1kGp, input_pop, rye_exec, threads, output_rye),
# #                                     mc.cores = threads)
#
#
# df_final <- do.call(rbind, res_prediction)
#
#
# write.table(df_final,
#             paste0(output_prediction,
# "rye_prediction_PCAWG_mixed_ancestry_ANNm1.txt"),
#             col.names = TRUE,
#             row.names = TRUE,

```

```
#           quote = FALSE,  
#           sep = '\t')
```

Annexe 6 - Script Rye

```
#!/usr/bin/env Rscript
script_author = "Andrew Conley, Lavanya Rishishwar"
script_copyright = "Copyright 2021, Andrew Conley, Lavanya Rishishwar"
script_credits = c("Andrew Conely", "Lavanya Rishishwar", "Maria
Ahmad", "Shivam Sharma", "Emily Norris")
script_license = "GPL"
script_version = "0.1"
script_maintainer = "Andrew Conley, Lavanya Rishishwar"
script_email = "aconley@ihrc.com; lrishishwar@ihrc.com"
script_status = "Development"
script_title = "rye.R"

#####
#####
### Load libraries
requiredPackages = c('nnls','Hmisc','parallel', 'optparse', 'crayon')
for(p in requiredPackages){
  if(!suppressMessages(require(p,character.only = TRUE, quietly = T))){
    stop(paste0("Library ", p, " is required, I can't seem to find
it."))
  }
}

options(width = 220)
options(scipen = 999)
options(digits = 4)

#####
#####
### Function definition
printDims = function(X, msg){cat(paste(msg, ':', paste(dim(X), collapse
= 'x'), "\n"))}

printError = function(msg){cat(red(paste(msg, collapse = " "), "\n"))}
printWarn = function(msg){cat(yellow(paste(msg, collapse = " "),
"\n"))}
logmsg = function(msg){cat(green(paste(format(Sys.time(), "[ %b %d %Y -
%X ]"), msg, collapse = " "), "\n"))}
progressmsg = function(msg){cat(magenta(paste(msg, collapse = " "),
"\n"))}
pretty_time = function(time){
```

```

out_string = ""
if(time > 60*60*24){
  days = round(time/(60*60*24))
  time = time %% (60*60*24)
  out_string = paste0(out_string, days, " days, ")
}
if(time > 60*60){
  hours = round(time/(60*60))
  time = time %% (60*60)
  out_string = paste0(out_string, hours, " hours, ")
}
if(time > 60){
  mins = round(time/60)
  time = time %% 60
  out_string = paste0(out_string, mins, " mins, ")
}
out_string = paste0(out_string, round(time, 2), " seconds")
return(out_string)
}

rye.scale = function(X = NULL) {
  return(apply(X, 2, function(i){i = i - min(i); i / max(i)}))
}

rye.populationMeans = function(X = NULL, fam = NULL, alpha = NULL,
weight = NULL, fn = median, referenceGroups = NULL) {

  ## Find the mean of each reference population
  if (!is.null(referenceGroups)) {
    means = aggregate(X, by = list(referenceGroups[fam[ ,
'population']]), fn)
  } else {
    means = aggregate(X, by = list(fam[ , 'population']), fn)
  }

  ## Reformat
  rownames(means) = means[ , 1]
  means = means[ , 2:ncol(means)]

  ## Apply shrinkage by given method and alpha
  means = apply(means, 2, function(i) i + (((1/2 - i)**2) * (((i >
1/2) * -1) + (i <= 1/2) * alpha)))

  ## Weight each feature
  means = t(t(means) * weight)
}

```

```

    return(means)
}

rye.predict = function(X = NULL, means = NULL, weight = NULL,
referenceGroups = NULL) {

  estimates = t(apply(t(t(X) * weight), 1, function(i){c = coef(nnls(A=
as.matrix(t(means)), b = i)); c / sum(c)}))
  colnames(estimates) = rownames(means)

  if (!is.null(referenceGroups)) {
    estimates = do.call(cbind, lapply(unique(referenceGroups),
function(i) apply(estimates[ , names(referenceGroups)[referenceGroups
== i], drop = FALSE], 1, sum)))
    colnames(estimates) = unique(referenceGroups)
  }
  return(estimates)
}

rye.squaredError = function(expected = NULL, predicted = NULL) {
  return((expected - predicted) ** 2)
}

rye.absoluteError = function(expected = NULL, predicted = NULL) {
  return(abs(expected - predicted))
}

rye.gibbs = function(X = NULL, fam = NULL, referenceGroups = NULL,
alpha = NULL, optimizeAlpha = TRUE,
weight = NULL, optimizeWeight = TRUE,
iterations = 100, sd = 0.0001) {

  pops = names(alpha)

  ## Assume the correct ref assignment is 100% their population
  expected = matrix(0, nrow = nrow(X), ncol = length(pops), dimnames =
list(rownames(fam), pops))
  expected[fam[ , c('id', 'population')]] <- 1

  ## Make each pop its own group if groups aren't given
  if (is.null(referenceGroups)) {
    referenceGroups = pops
    names(referenceGroups) = pops
  }

  expected = matrix(0, nrow = nrow(X), ncol =

```

```

length(unique(referenceGroups)), dimnames = list(rownames(fam),
unique(referenceGroups)))
  expected[cbind(fam[ , 'id'], referenceGroups[fam[ , 'population']])] =
= 1

fam = cbind(fam, referenceGroups[fam[ , 'population']])
colnames(fam)[ncol(fam)] = 'group'

## Get the starting error
means = rye.populationMeans(X = X, fam = fam, alpha = alpha, weight =
weight, referenceGroups = referenceGroups)[pops, ]
predicted = rye.predict(X = X, means = means, weight = weight,
referenceGroups = referenceGroups)
oldError = rye.absoluteError(expected = expected, predicted =
predicted)
oldError = cbind(apply(oldError, 1, mean))
oldError = aggregate(oldError, by = list(fam[ , 'group']), mean)
oldError = oldError[ , -1]
oldError = mean(oldError)

## Return values
minError = oldError
minParams = list(minError, alpha, weight, means, predicted)

## Momentum between iterations
alphaMomentum = rep(0, length(alpha))
weightMomentum = rep(0, length(weight))
momentum = 1/10

for (iteration in seq(iterations)) {

  ## Pick new alpha and weight for this iteration
  newAlpha = alpha
  if (optimizeAlpha) {
    toUpdate = sample(seq(length(newAlpha)))[1]
    newAlpha[toUpdate] = newAlpha[toUpdate] + rnorm(n = 1, sd =
(abs(newAlpha[toUpdate]) + 0.001) * sd) + alphaMomentum[toUpdate]
    newAlpha[newAlpha < 0] = 0
  }

  newWeight = weight
  if (optimizeWeight) {
    toUpdate = sample(seq(length(newWeight)))[1]
    newWeight[toUpdate] = newWeight[toUpdate] + rnorm(n = 1, sd =
(newWeight[toUpdate] + 0.001) * sd) + weightMomentum[toUpdate]
    newWeight[newWeight < 0] = 0
  }
}

```

```

}

## Find the new errors
means = rye.populationMeans(X = X, fam = fam, alpha = newAlpha,
weight = newWeight, referenceGroups = referenceGroups)[pops, ]
predicted = rye.predict(X = X, means = means, weight = newWeight,
referenceGroups = referenceGroups)
newError = rye.absoluteError(expected = expected, predicted =
predicted)
newError = cbind(apply(newError, 1, mean))
newError = aggregate(newError, by = list(fam[ , 'group']), mean)
newError = newError[ , -1]
newError = mean(newError)

## Find the jump odds
odds = pnorm(newError, mean = oldError, sd = oldError / 1000)
odds = c(1 - odds, odds)

## If this is the best error we've seen, then keep it
if (newError < minError) {
  minError = newError
  minParams = list(minError, alpha, weight, means, predicted)
}

## See if we jump
if (runif(n = 1, min = 0, max = 1) < odds[1]) {
  oldError = newError
  alphaMomentum = (alphaMomentum / 2) + ((newAlpha - alpha) *
momentum)
  weightMomentum = (weightMomentum / 2) + ((newWeight - weight) *
momentum)
  alpha = newAlpha
  weight = newWeight
}

}

return(minParams)
}

rye.optimize = function(X = NULL, fam = NULL,
referencePops = NULL, referenceGroups = NULL,
alpha = NULL, optimizeAlpha = TRUE,
weight = NULL, optimizeWeight = TRUE, attempts

```

```

= 4,
iterations = 100, rounds = 25, threads = 1,
startSD = 0.005, endSD = 0.001,
populationError = FALSE) {

## Pull out the reference PCs
referenceFAM = fam[fam[, 'population'] %in% referencePops, ]
referenceX = X[rownames(referenceFAM), ]

## Start with the shrinking at 0.05 for all pops by default
if (is.null(alpha)) {
  alpha = rep(0.001, length(referencePops))
}
names(alpha) = referencePops

## Weights
if (is.null(weight)) {
  weight = 1 / seq(ncol(X))
}

allErrors = c()

for (round in seq(rounds)) {

  sd = startSD - (startSD - endSD) * log(round)/log(rounds)
  if (threads > 1) {
    params = mclapply(seq(attempts), function(i) rye.gibbs(X =
referenceX, fam = referenceFAM, referenceGroups = referenceGroups,
iterations
= iterations,
alpha =
alpha, weight = weight, sd = sd,
optimizeAlpha = optimizeAlpha, optimizeWeight = optimizeWeight),
mc.cores = threads)
  } else {
    params = lapply(seq(attempts), function(i) rye.gibbs(X =
referenceX, fam = referenceFAM, referenceGroups = referenceGroups,
iterations =
iterations,
alpha =
alpha, weight = weight, sd = sd,
optimizeAlpha
= optimizeAlpha, optimizeWeight = optimizeWeight))
  }
}

```

```

errors = unlist(lapply(params, function(i) i[[1]]))

bestError = which.min(errors)
meanError = mean(errors)
progressmsg(paste0('Round ', round, '/', rounds, ' Mean error: ',
sprintf("%.6f", meanError),
', Best error: ', sprintf('.6f', errors[bestError])))

bestParams = params[[bestError]]
alpha = bestParams[[2]]
weight = bestParams[[3]]

allErrors = c(allErrors, errors[bestError])

## See if our error hasn't decreased substantially in 5 rounds
if (round > 5) {
  errorChange = allErrors[(round - 5):round]
  errorChange = max(errorChange) - min(errorChange)
  if (errorChange <= 0.000025) {
    break
  }
}

return(bestParams)
}

rye = function(eigenvec_file = NULL, eigenval_file = NULL,
              pop2group_file = NULL, output_file = NULL,
              threads = 4, pcs = 20, optim_rounds = 200,
              optim_iter = 100, attempts=4){
  ## Perform core operation
  #TODO: Change file reading method to data.table
  logmsg("Reading in Eigenvector file")
  fullPCA = read.table(eigenvec_file, header = T, row.names = NULL)
  fullPCA = read.table(eigenvec_file, header = F, row.names = NULL)
  rownames(fullPCA) = fullPCA[, 2]
  logmsg("Reading in Eigenvalue file")
  fullEigenVal = read.table(eigenval_file, header = FALSE, row.names =
NULL)[,1]
  logmsg("Reading in pop2group file")
  pop2group = read.table(pop2group_file, header = T, stringsAsFactors =
F)

```

```

#These two lines must be changed to make the new method works :
#
#referenceGroups = pop2group$Group[-length(pop2group$Group)]
#names(referenceGroups) = pop2group$Pop[-length(pop2group$Group)]
referenceGroups = pop2group$Group
names(referenceGroups) = pop2group$Pop

## Regenerate the FAM from the PCA input
logmsg("Creating individual mapping")
fam = as.matrix(fullPCA[ , c(1, 2)])
colnames(fam) = c('population', 'id')
rownames(fam) = fam[ , 'id']
allPops = unique(fam[ , 'population'])

## Cast PCA to a matrix & scale the PCs
logmsg("Scaling PCs")
fullPCA = fullPCA[ , 3:ncol(fullPCA)]
fullPCA = as.matrix(fullPCA)
fullPCA = rye.scale(fullPCA)

## Weight the PCs by their eigenvalues
logmsg("Weighting PCs")
weight = fullEigenVal / max(fullEigenVal)

## Using each region as a population, e.g., combine British and
French to WesternEuropean
logmsg("Aggregating individuals to population groups")
regionFAM = fam
regionFAM[fam[,1] %in% names(referenceGroups),1] =
referenceGroups[fam[fam[,1] %in% names(referenceGroups), 1]]
referenceGroups = unique(referenceGroups)
names(referenceGroups) = referenceGroups
referencePops = referenceGroups

## Optimize estimates using NNLS
logmsg("Optimizing estimates using NNLS")
scaledWeight = weight[seq(pcs)]
unifAlpha = rep(0.001, length(referencePops))
names(unifAlpha) = referencePops
optParams = rye.optimize(X = fullPCA[,seq(pcs)], fam = regionFAM,
                        referencePops = referencePops,
referenceGroups = referenceGroups,
                        startSD = 0.01, endSD = 0.005,
                        threads = threads, iterations = optim_iter,
                        rounds = optim_rounds, attempts=attempts,
                        weight = scaledWeight, alpha = unifAlpha,

```

```

    optimizeWeight = TRUE, optimizeAlpha = TRUE)
optWeight = optParams[[3]]
optMeans = optParams[[4]]

## Calculate ancestry estimates
logmsg("Calculate per-individual ancestry estimates")
optEstimates = rye.predict(X = fullPCA[, seq(pcs)], means = optMeans,
weight = optWeight)
optEstimates = t(apply(optEstimates, 1, function(i) i /sum(i)))

## Find the mean of each population
# logmsg("Calculate per-population mean ancestry estimates")
# optEstimateMeans = do.call(cbind, lapply(allPops, function(i)
cbind(apply(t(optEstimates[fam[,1] == i, ,drop = FALSE]), 1, mean)))
# colnames(optEstimateMeans) = allPops
optEstimatesAgg = NULL
for(group in referenceGroups){
  optEstimatesAgg = cbind(optEstimatesAgg, apply(optEstimates[ ,
group, drop = FALSE], 1, sum))
}
colnames(optEstimatesAgg) = as.character(referenceGroups)

## Create output files
logmsg("Create output files")
write.table(x = optEstimatesAgg,
            file = paste0(output_file, '-', pcs, '.', 
length(referenceGroups), '.Q'),
            col.names = TRUE, row.names = TRUE, quote = FALSE, sep =
'\t')
write.table(x = optEstimates,
            file = paste0(output_file, '-', pcs, '.', 
ncol(optEstimates), '.Q'),
            col.names = TRUE, row.names = TRUE, quote = FALSE, sep =
'\t')
write.table(x = fam[rownames(optEstimatesAgg), ],
            file = paste0(output_file, '-', pcs, '.fam'), col.names =
TRUE,
            row.names = TRUE, quote = FALSE, sep = '\t')

}

validate_arguments <- function(opt){
## Verify the arguments
#TODO: Move argument validation to its own function
argumentsGood = TRUE

```

```

if (is.null(opt$eigenvec)) {
  argumentsGood = FALSE
  printError('Eigenvector file not given (--eigenvec)')
} else if (!file.exists(opt$eigenvec)) {
  argumentsGood = FALSE
  printError(paste('Eigenvector file (--eigenvec=', opt$eigenvec, ')'
not found'))
}
if (is.null(opt$eigenval)) {
  argumentsGood = FALSE
  printError('Eigenvalue file not given (--eigenval)')
} else if (!file.exists(opt$eigenval)) {
  argumentsGood = FALSE
  printError(paste('Eigenvalue file (--eigenval=', opt$eigenval, ')'
not found'))
}
if (is.null(opt$pop2group)) {
  argumentsGood = FALSE
  printError('Population-to-group mapping file not given
(--pop2group)')
} else if (!file.exists(opt$pop2group)) {
  argumentsGood = FALSE
  printError(paste('Population-to-group mapping file (--pop2group=',
opt$pop2group, ') not found'))
}
if (is.null(opt$output)) {
  argumentsGood = FALSE
  printError('Output prefix not given (--output)')
}

#TODO: Implement check for file dimensions
#TODO: Ensure number of threads don't exceed machine capacity

if (!argumentsGood){
  printError('Incomplete/incorrect arguments were observed, cannot
continue.')
  printError(c("Run", script_title, "-h for usage information"))
  # stop("Exiting...") # I don't like the error message
  q(save = "no", status = 1)
}
#####

optionList = list(

```

```

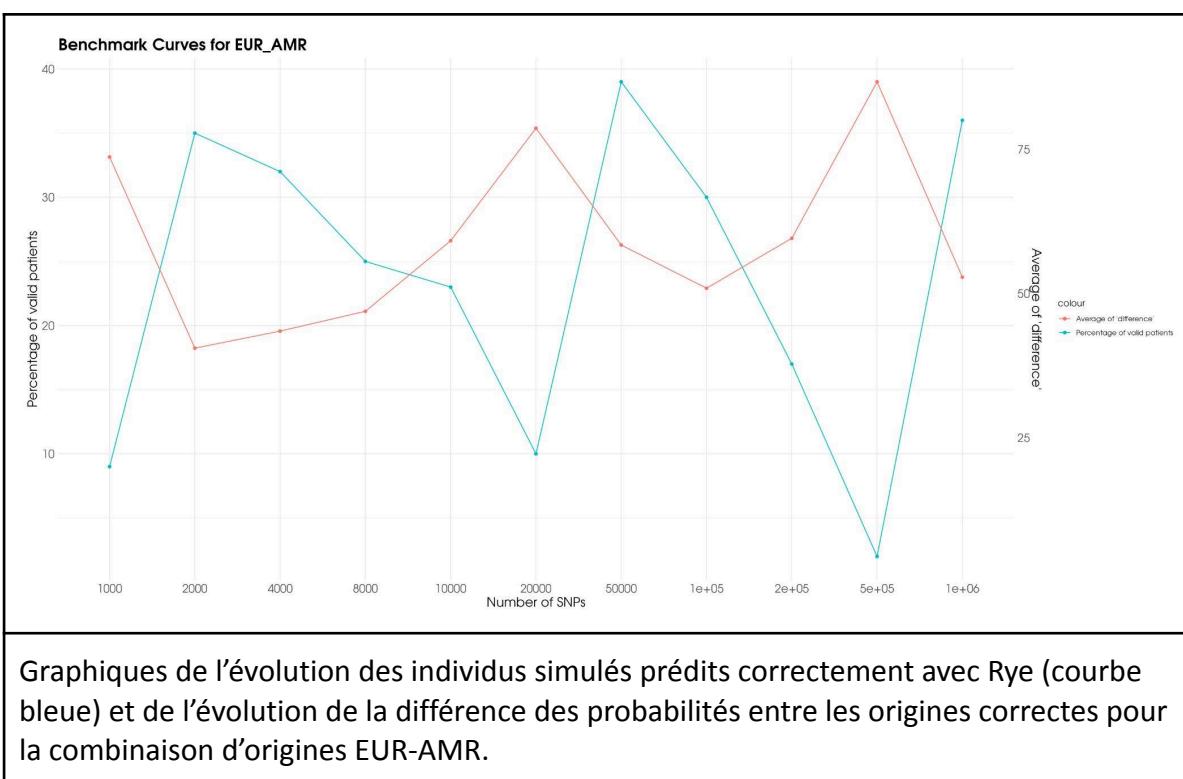
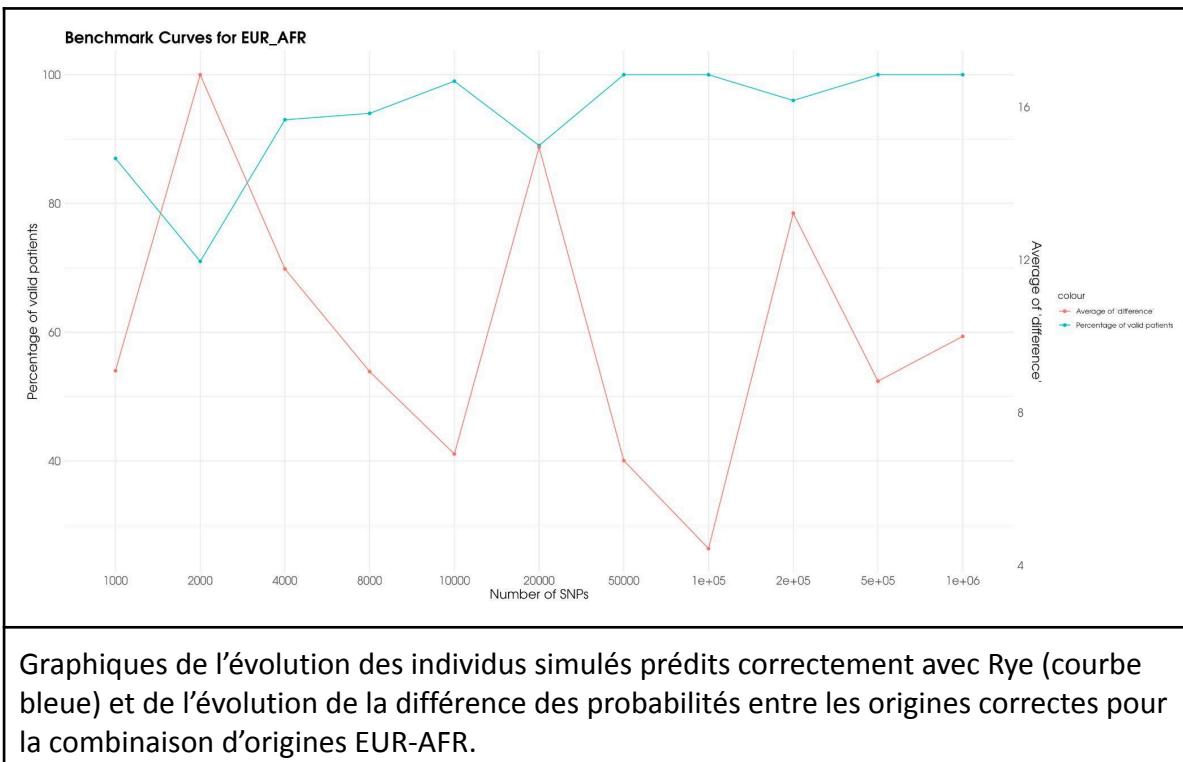
make_option('--eigenval', type = 'character', default = NULL,
           help = 'Eigenvalue file [REQUIRED]', metavar =
'<EVAL_FILE>'),
make_option('--eigenvec', type = 'character', default = NULL,
           help = 'Eigenvector file [REQUIRED]', metavar =
'<EVEC_FILE>'),
make_option('--pop2group', type = 'character', default = NULL,
           help = 'Population-to-group mapping file [REQUIRED]',
metavar = '<P2G_FILE>'),
make_option('--output', type = 'character', default = "output",
           help = 'Output prefix (Default = output)', metavar =
'<OUTPUT_PREFIX>'),
make_option('--threads', type = 'numeric', default = 4,
           help = 'Number of threads to use (Default = 4)', metavar =
'<THREADS>'),
make_option('--pcs', type = 'numeric', default = 20,
           help = 'Number of PCs to use (Default = 20)', metavar =
'<#PCs>'),
make_option('--rounds', type = 'numeric', default = 200,
           help = 'Number of rounds to use for optimization (higher
number = more accurate but slower; Default=200)',
           metavar = '<optim-rounds>'),
make_option('--iter', type = 'numeric', default = 100,
           help = 'Number of iterations to use for optimization
(higher number = more accurate but slower; Default=100)',
           metavar = '<optim-iters>'),
make_option('--attempts', type = 'numeric', default = 4,
           help = 'Number of attempts to find the optimum values
(Default = 4)', metavar = '<ATTEMPTS>')
)

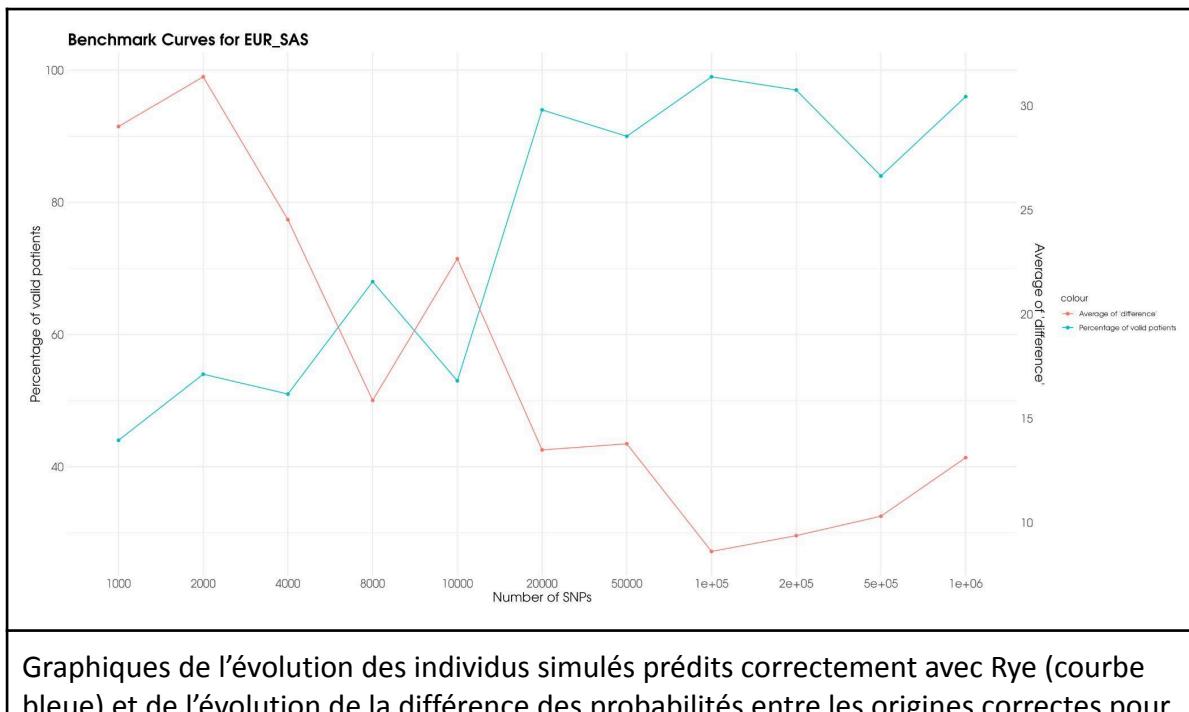
optParser = OptionParser(option_list = optionList)
opt = parse_args(optParser)
# Debug only
# opt = parse_args(optParser, args =
c("--eigenvec=extractedChrAllPrunedNoSan.25.eigenvec.gz",
#
"--eigenval=extractedChrAllPrunedNoSan.25.eigenval",
#                                     "--pop2group=pop2group.txt"))
# print(opt)
start_time <- Sys.time()
logmsg("Parsing user supplied arguments...")
validate_arguments(opt)
logmsg("Arguments passed validation")
logmsg(paste0("Running core rye with ", opt$threads, " threads"))
rye(eigenvec_file = opt$eigenvec,

```

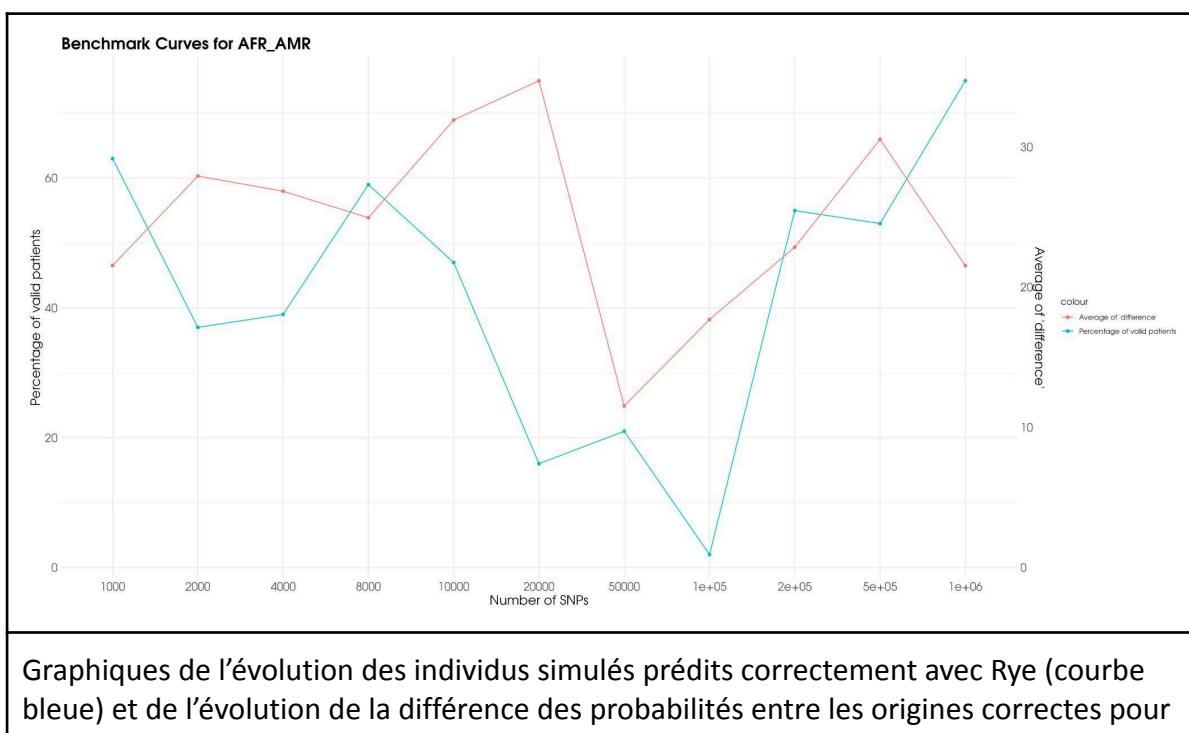
```
eigenval_file = opt$eigenval,
pop2group_file = opt$pop2group,
output_file = opt$output,
threads = opt$threads,
attempts = opt$attempts,
pcs = opt$pcs,
optim_rounds = opt$rounds,
optim_iter = opt$iter)
logmsg("Process completed")
end_time <- difftime(Sys.time(), start_time, units = "secs")[[1]]
# print(end_time)
logmsg(paste0("The process took ", pretty_time(end_time)))
```

Annexe 7 - Benchmarks

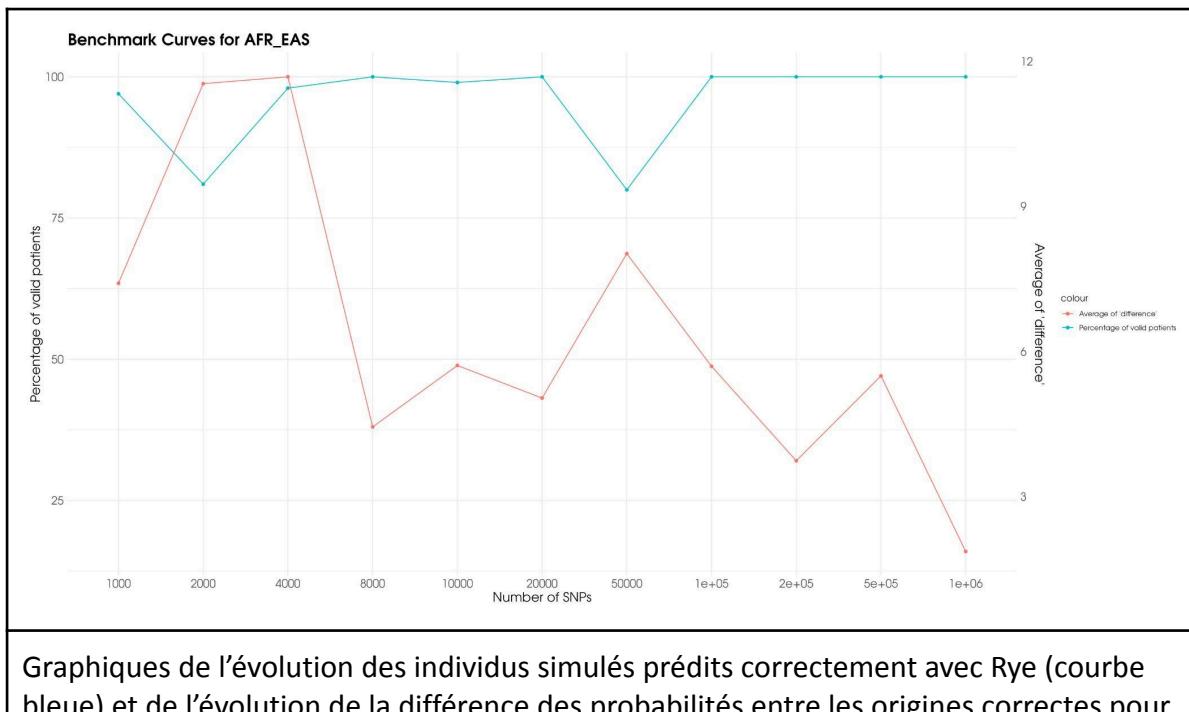




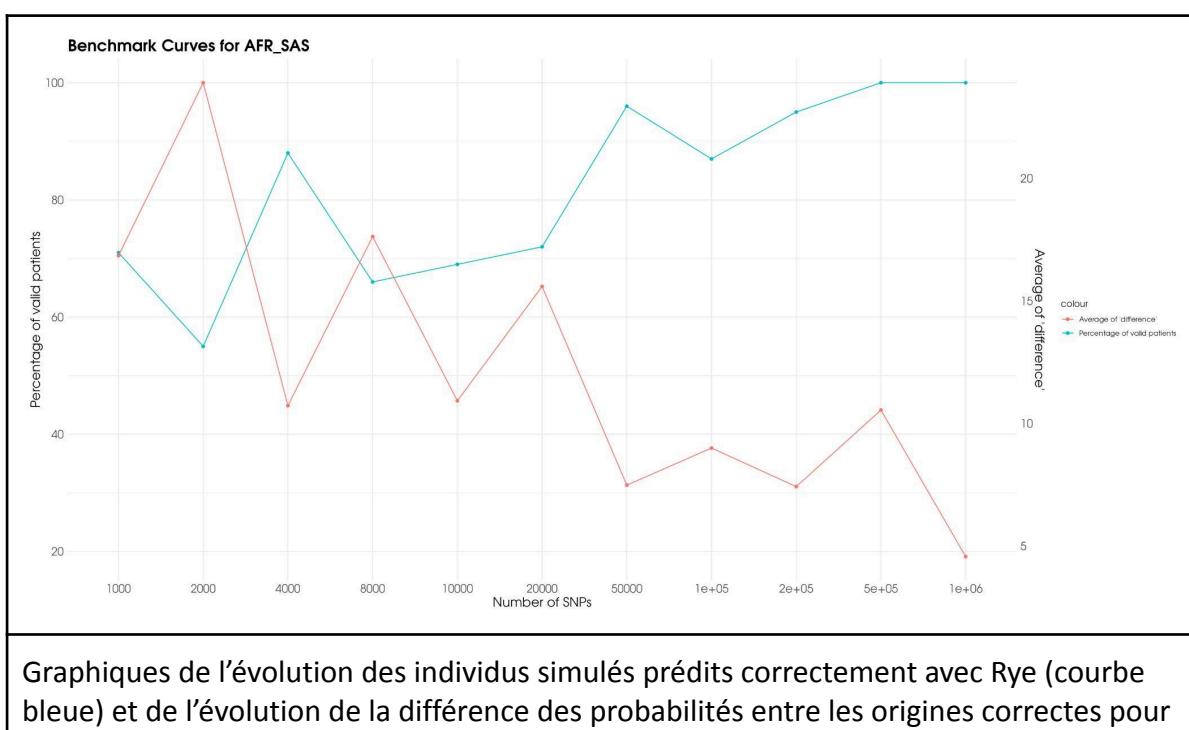
Graphiques de l'évolution des individus simulés prédits correctement avec Rye (courbe bleue) et de l'évolution de la différence des probabilités entre les origines correctes pour la combinaison d'origines EUR-EAS.



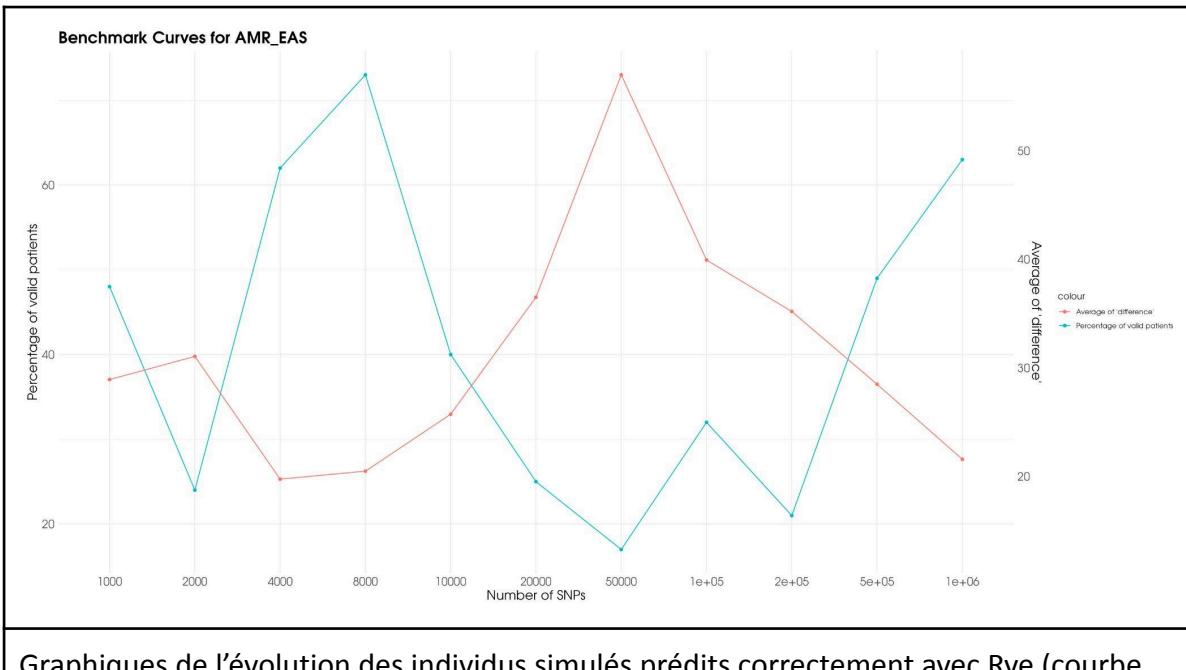
Graphiques de l'évolution des individus simulés prédits correctement avec Rye (courbe bleue) et de l'évolution de la différence des probabilités entre les origines correctes pour la combinaison d'origines AFR-AMR.



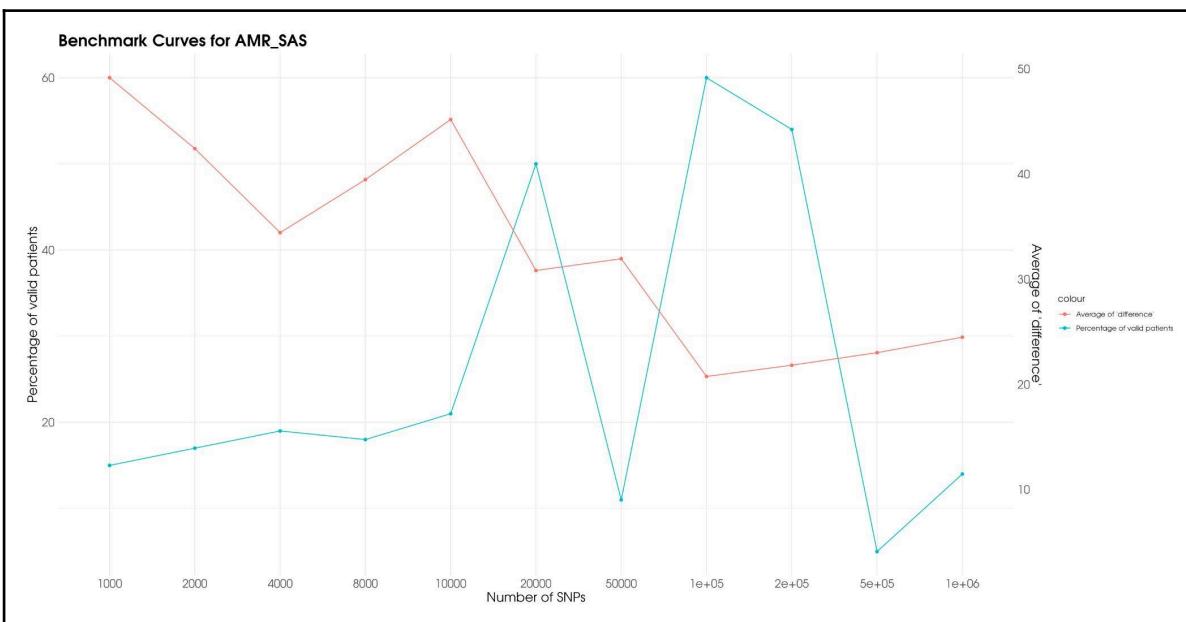
Graphiques de l'évolution des individus simulés prédits correctement avec Rye (courbe bleue) et de l'évolution de la différence des probabilités entre les origines correctes pour la combinaison d'origines AFR-EAS.



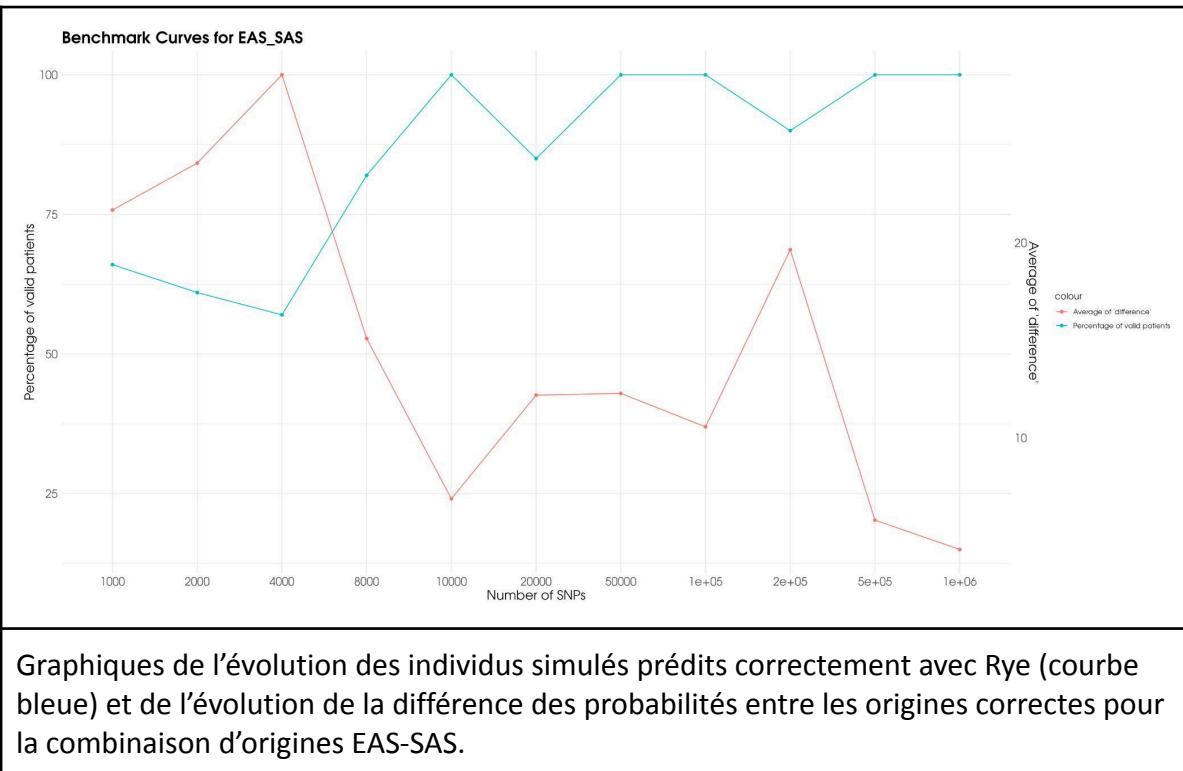
Graphiques de l'évolution des individus simulés prédits correctement avec Rye (courbe bleue) et de l'évolution de la différence des probabilités entre les origines correctes pour la combinaison d'origines AFR-SAS.



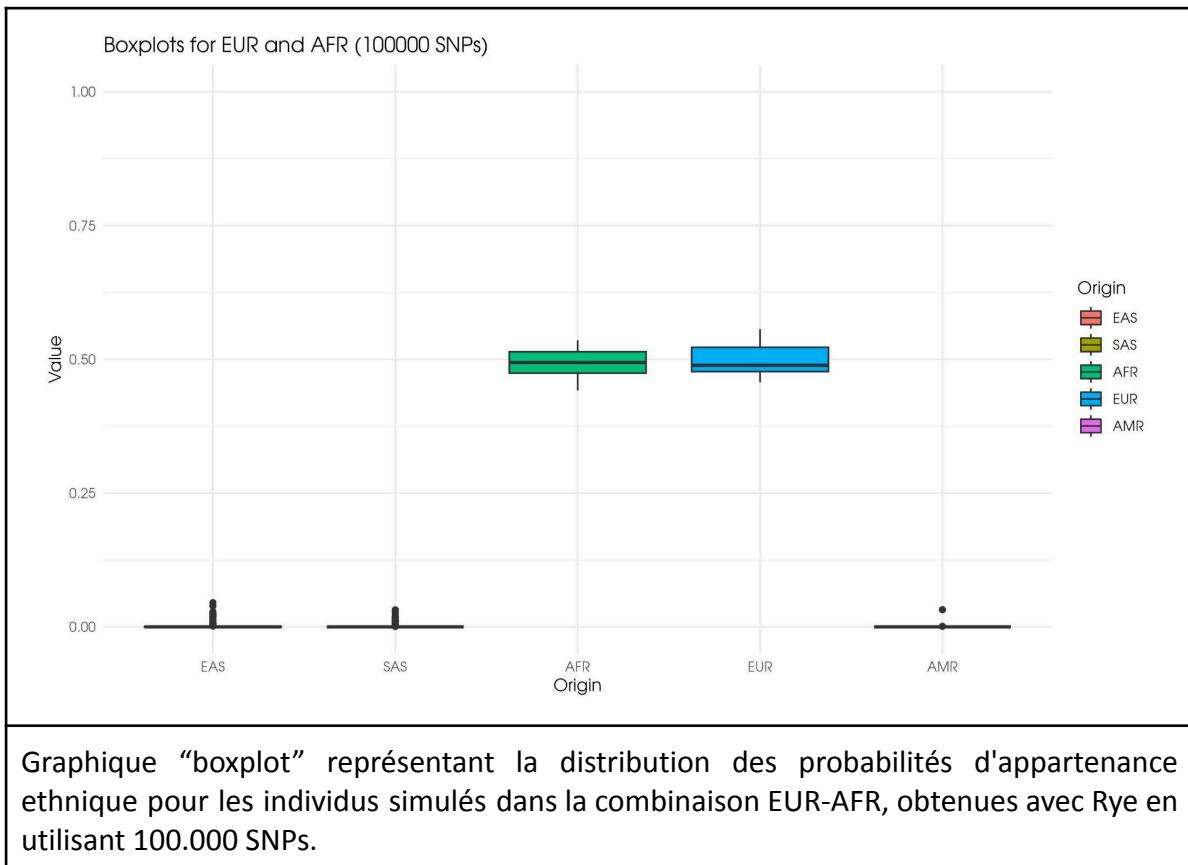
Graphiques de l'évolution des individus simulés prédits correctement avec Rye (courbe bleue) et de l'évolution de la différence des probabilités entre les origines correctes pour la combinaison d'origines AMR-EAS.



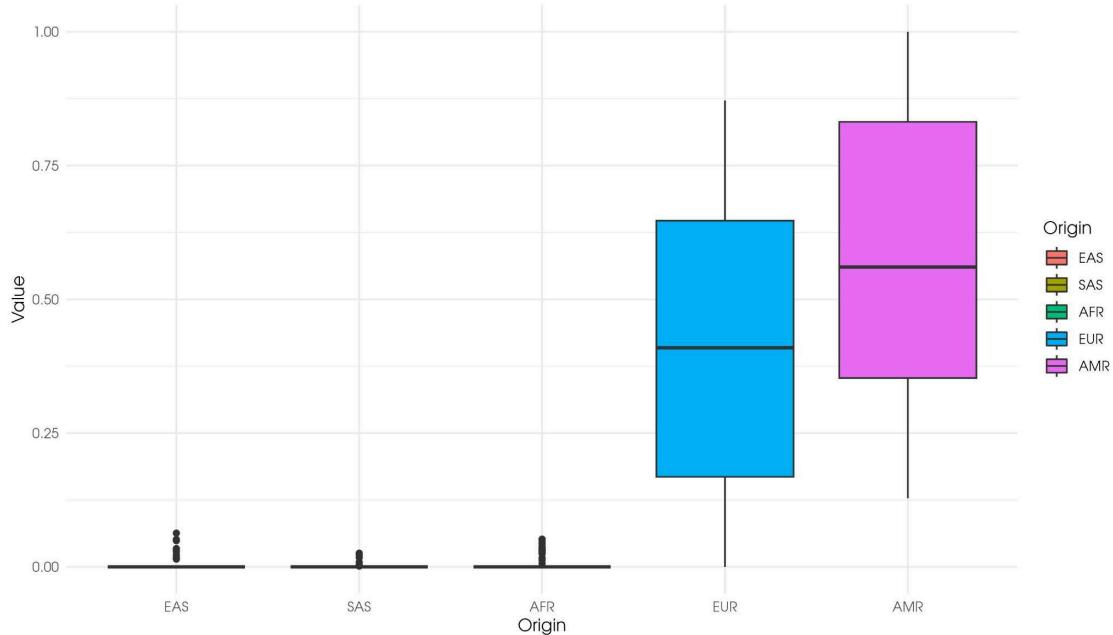
Graphiques de l'évolution des individus simulés prédits correctement avec Rye (courbe bleue) et de l'évolution de la différence des probabilités entre les origines correctes pour la combinaison d'origines AMR-SAS.



Annexe 8 - Boxplots sur 100.000 SNPs discriminants

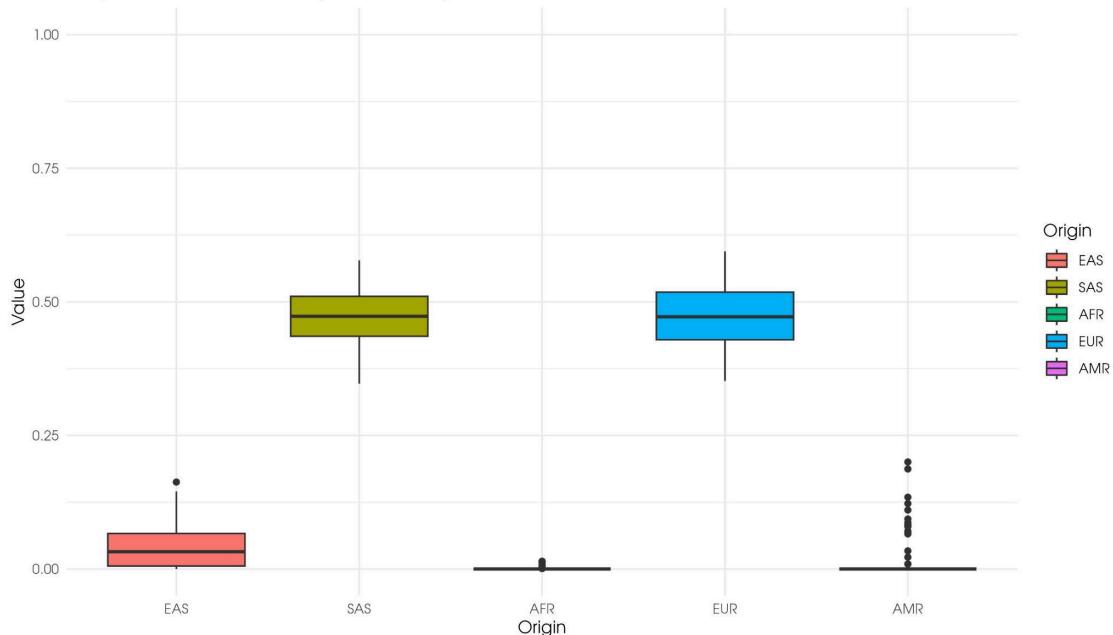


Boxplots for EUR and AMR (100000 SNPs)



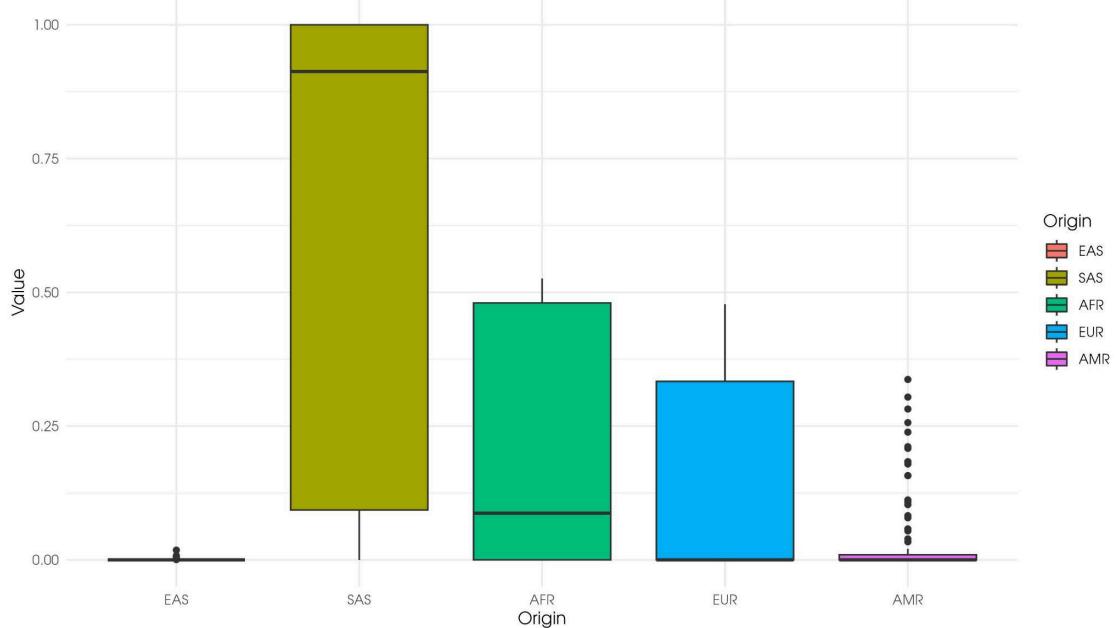
Graphique “boxplot” représentant la distribution des probabilités d'appartenance ethnique pour les individus simulés dans la combinaison EUR-AMR, obtenues avec Rye en utilisant 100.000 SNPs.

Boxplots for EUR and SAS (100000 SNPs)



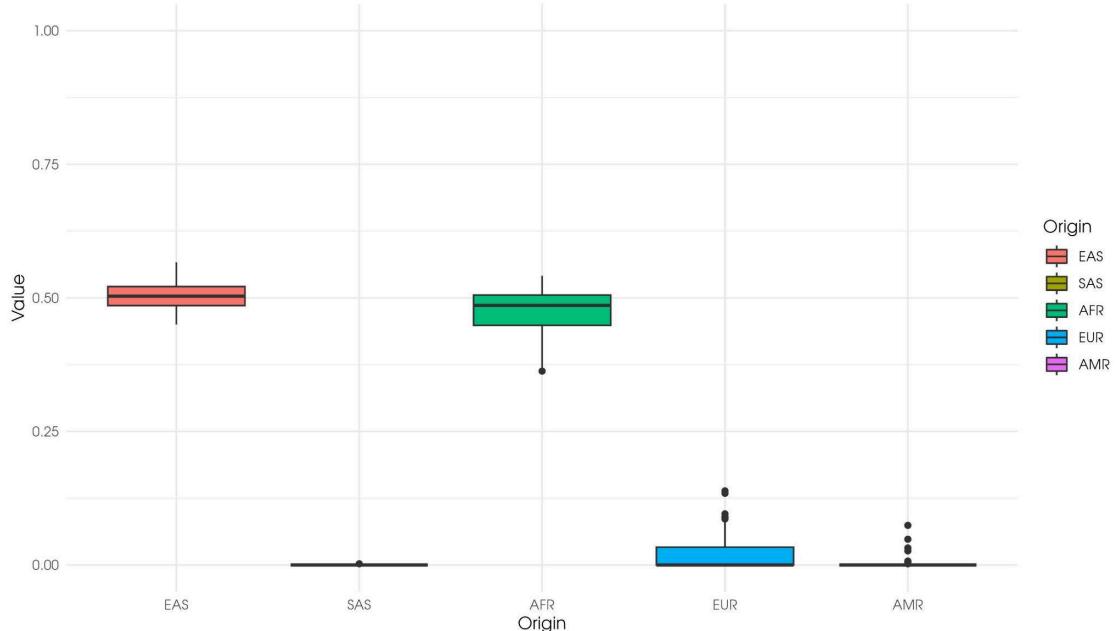
Graphique “boxplot” représentant la distribution des probabilités d'appartenance ethnique pour les individus simulés dans la combinaison EUR-SAS, obtenues avec Rye en utilisant 100.000 SNPs.

Boxplots for AFR and AMR (100000 SNPs)

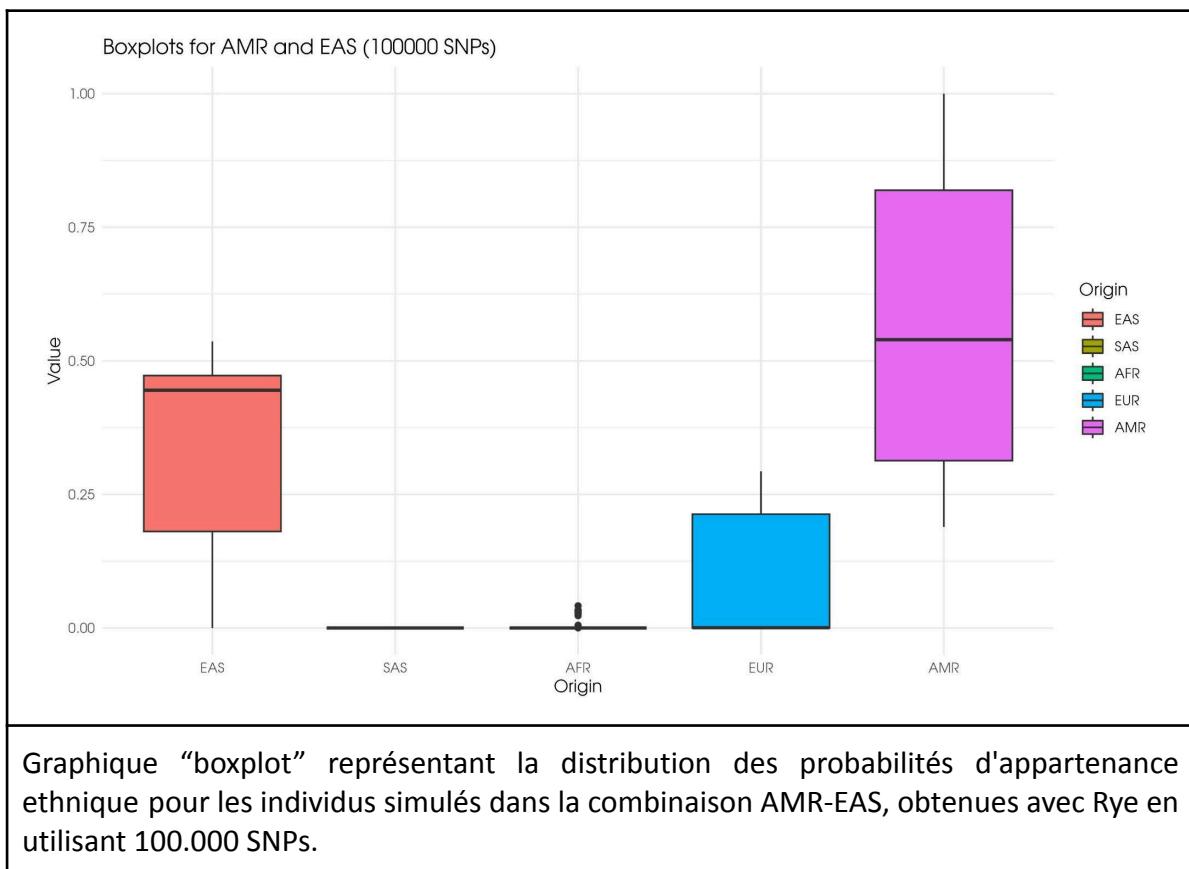
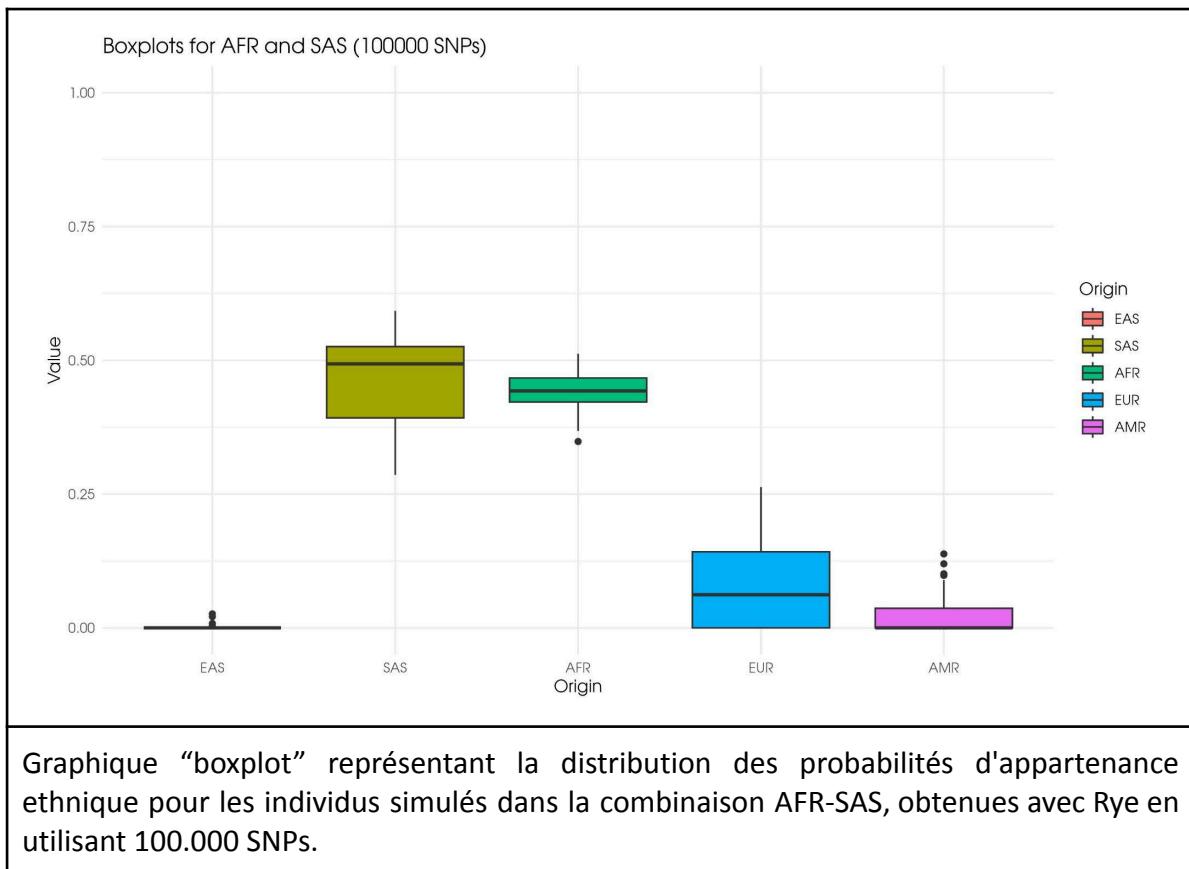


Graphique “boxplot” représentant la distribution des probabilités d'appartenance ethnique pour les individus simulés dans la combinaison AFR-AMR, obtenues avec Rye en utilisant 100.000 SNPs.

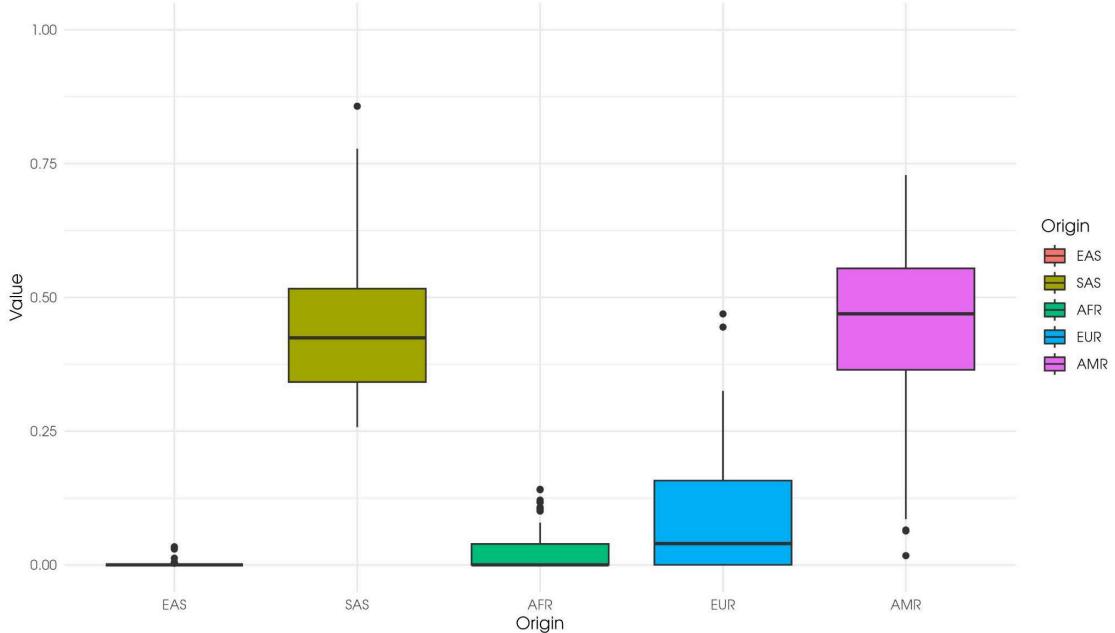
Boxplots for AFR and EAS (100000 SNPs)



Graphique “boxplot” représentant la distribution des probabilités d'appartenance ethnique pour les individus simulés dans la combinaison AFR-EAS, obtenues avec Rye en utilisant 100.000 SNPs.

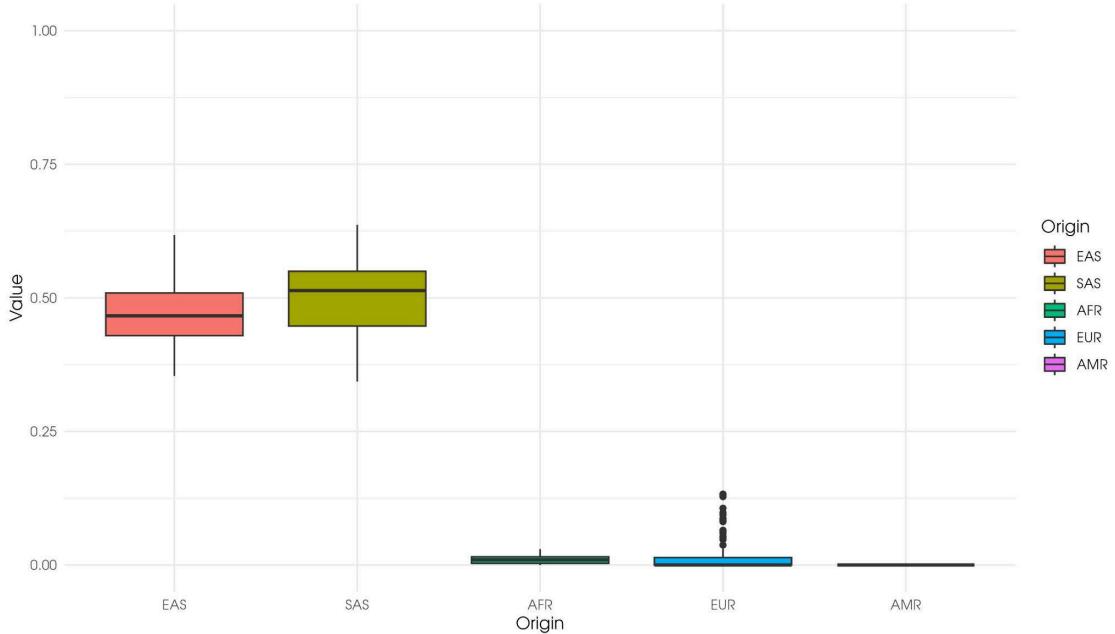


Boxplots for AMR and SAS (100000 SNPs)



Graphique “boxplot” représentant la distribution des probabilités d'appartenance ethnique pour les individus simulés dans la combinaison AMR-SAS, obtenues avec Rye en utilisant 100.000 SNPs.

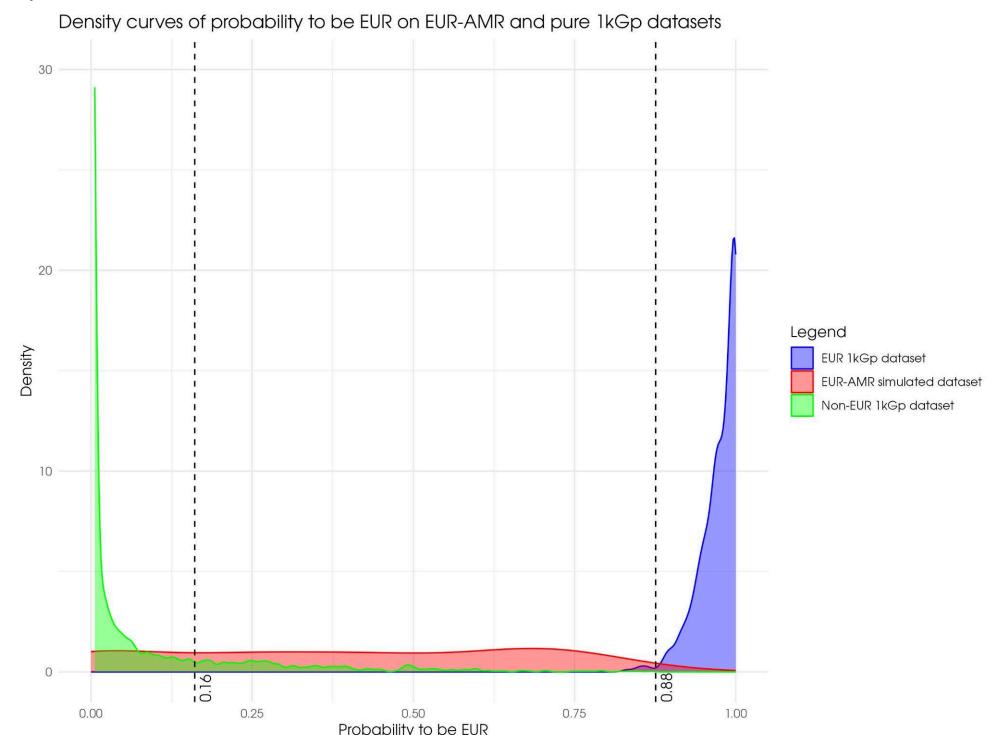
Boxplots for EAS and SAS (100000 SNPs)



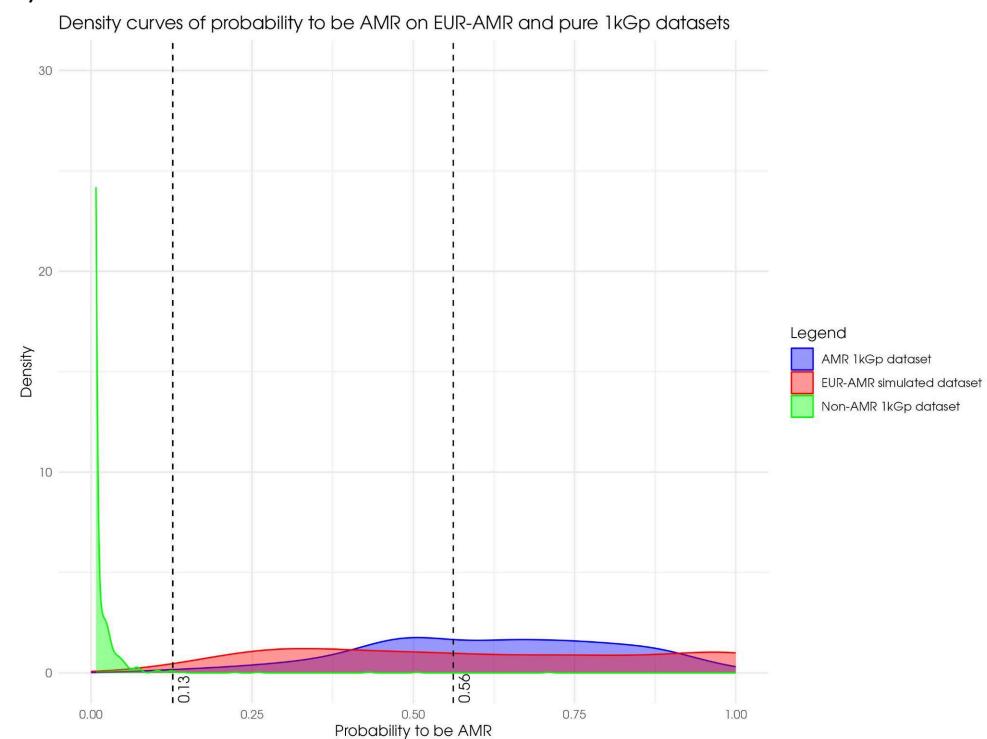
Graphique “boxplot” représentant la distribution des probabilités d'appartenance ethnique pour les individus simulés dans la combinaison EAS-SAS, obtenues avec Rye en utilisant 100.000 SNPs.

Annexe 9 - Graphiques de densités

A)

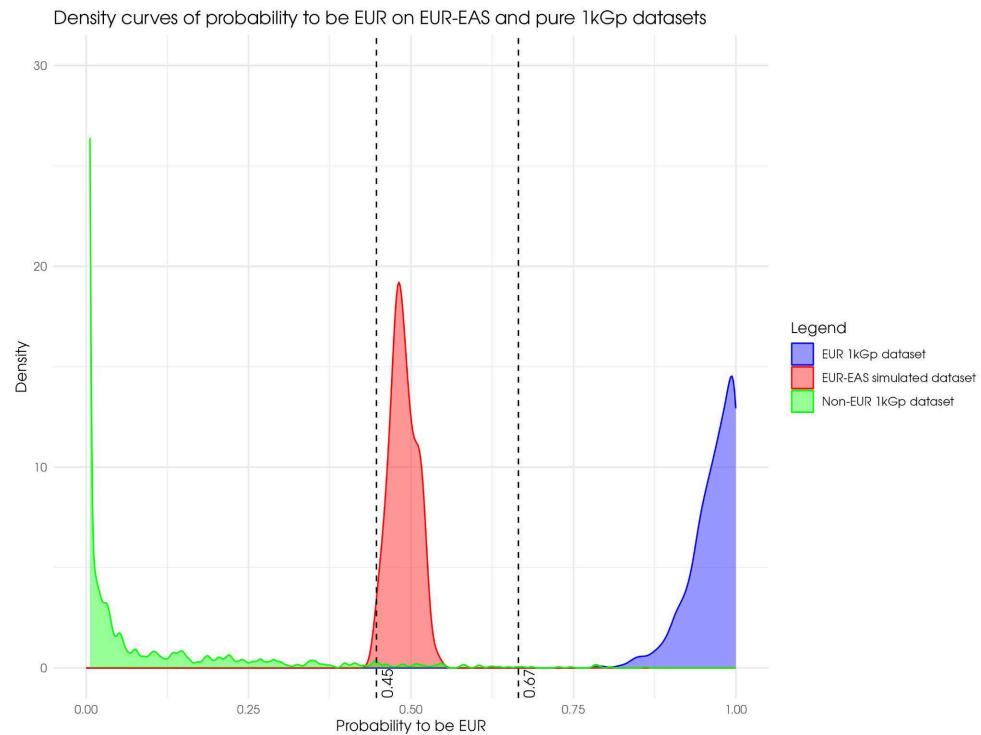


B)

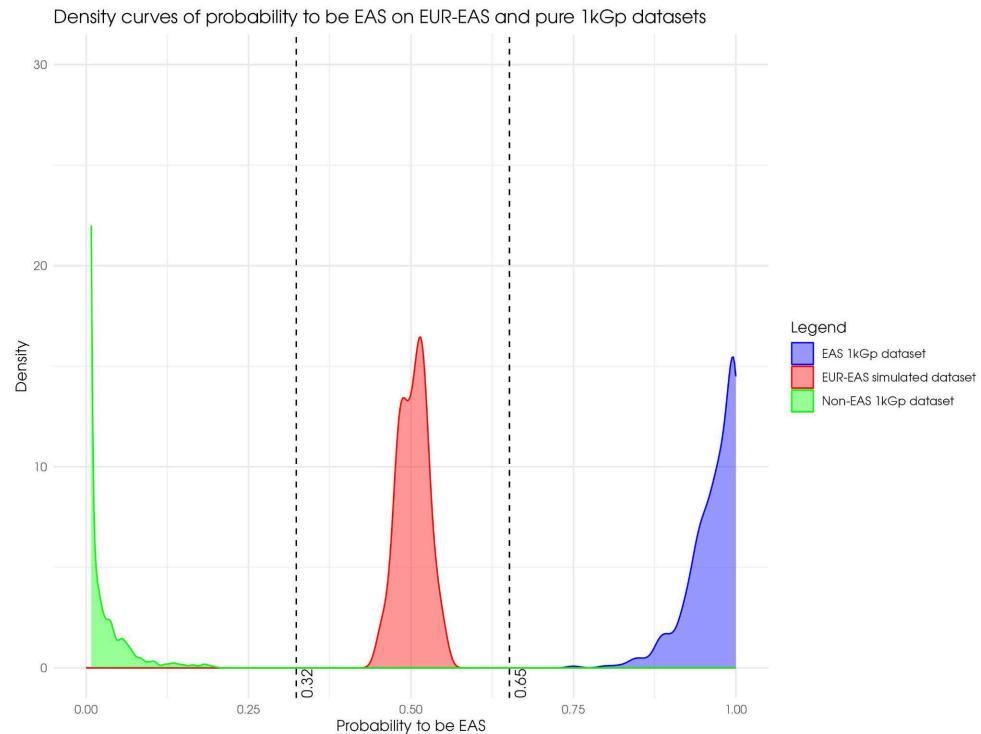


Graphiques de densités des individus d'origine européenne (image A) et américaine (image B) sur différents ensembles, pour la combinaison d'origines EUR-AMR.

A)

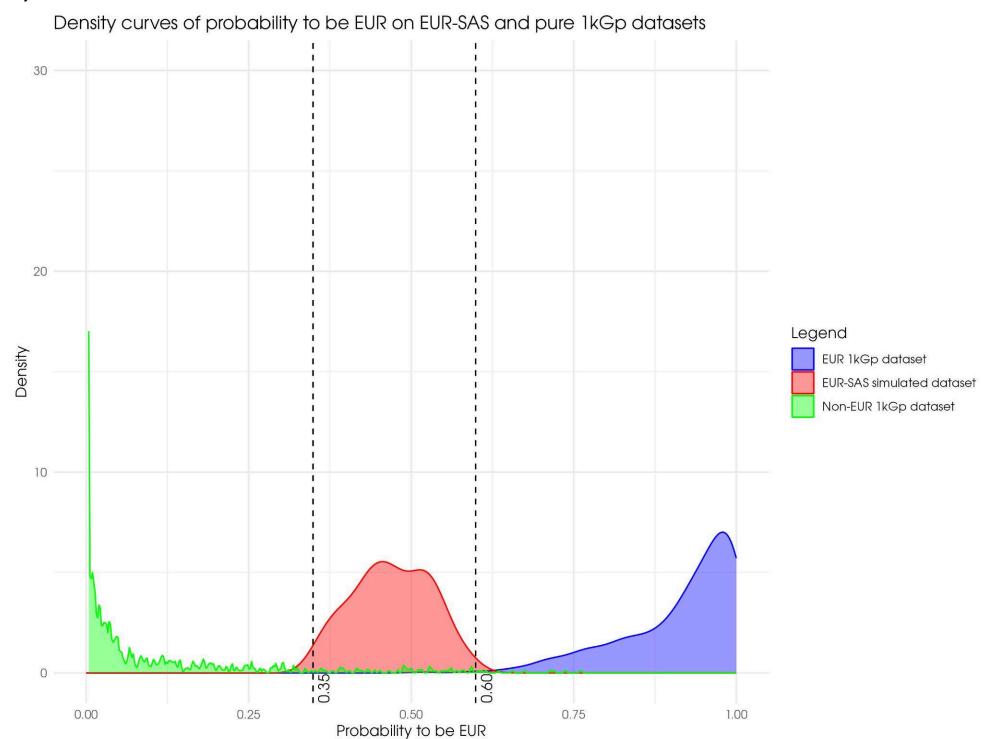


B)

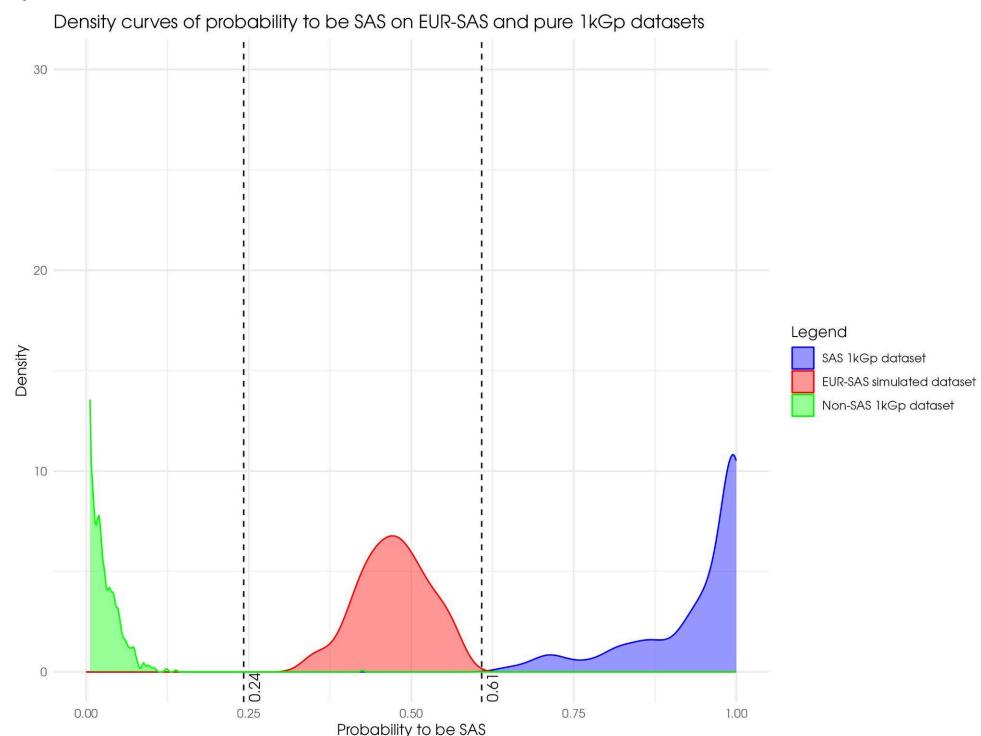


Graphiques de densités des individus d'origine européenne (image A) et Est-asiatique (image B) sur différents ensembles, pour la combinaison d'origines EUR-EAS.

A)

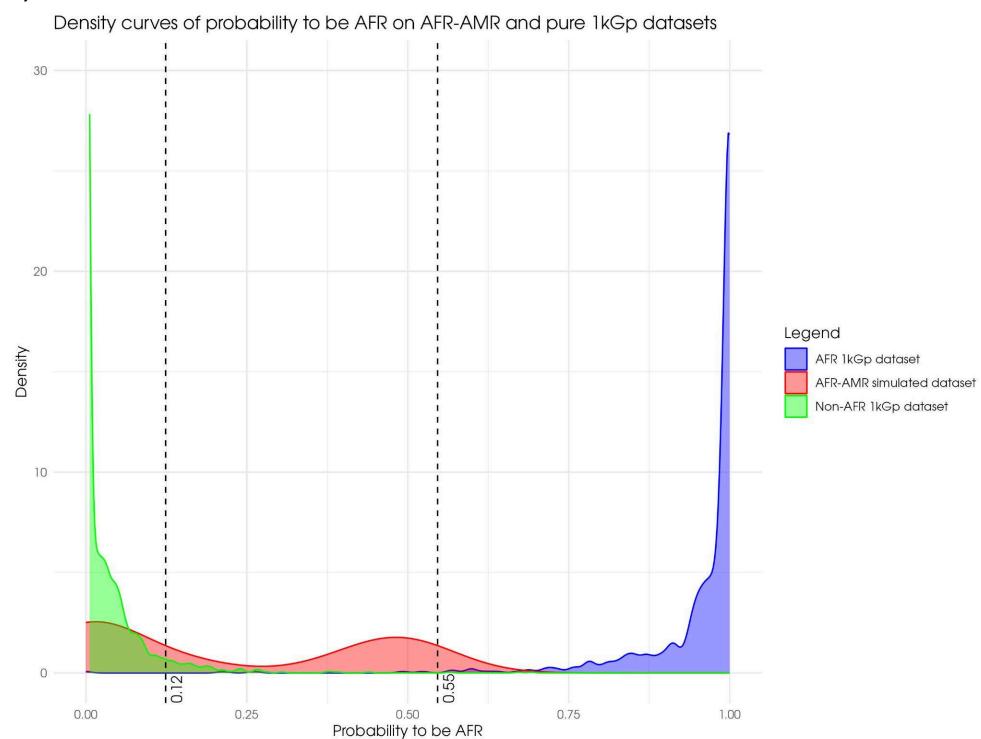


B)

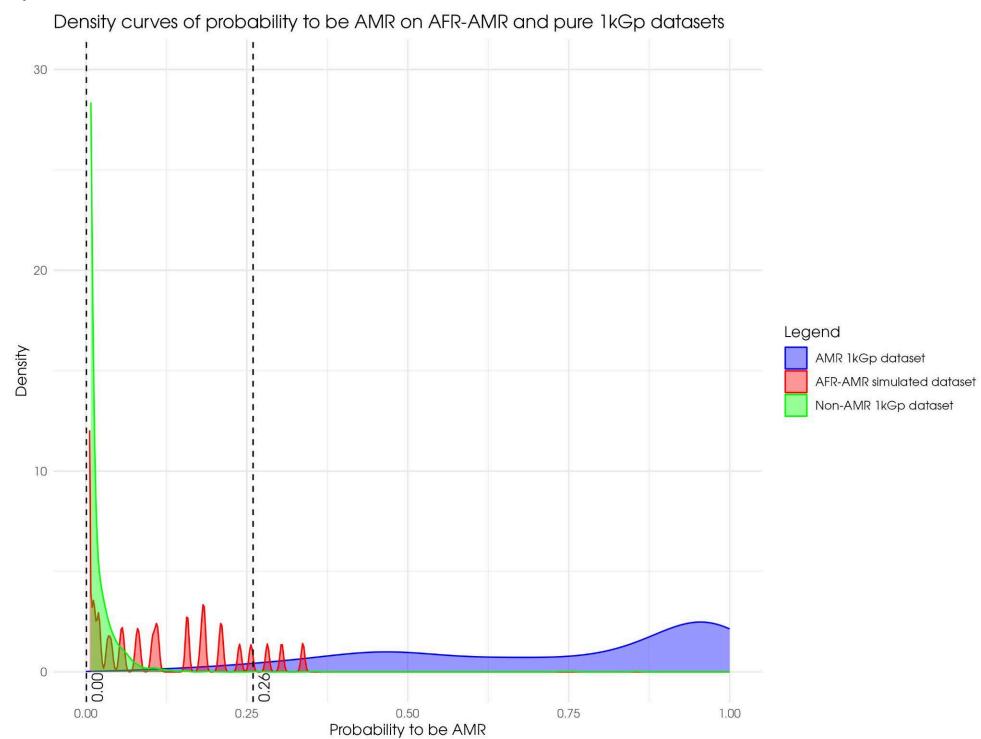


Graphiques de densités des individus d'origine européenne (image A) et Sud-asiatique (image B) sur différents ensembles, pour la combinaison d'origines EUR-SAS.

A)

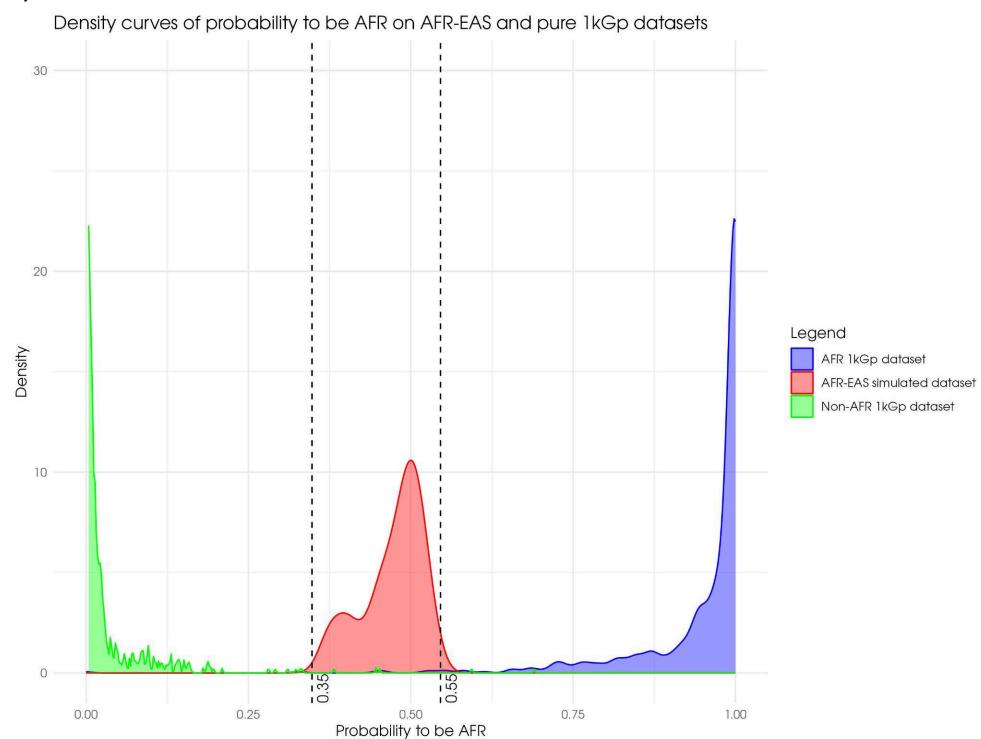


B)

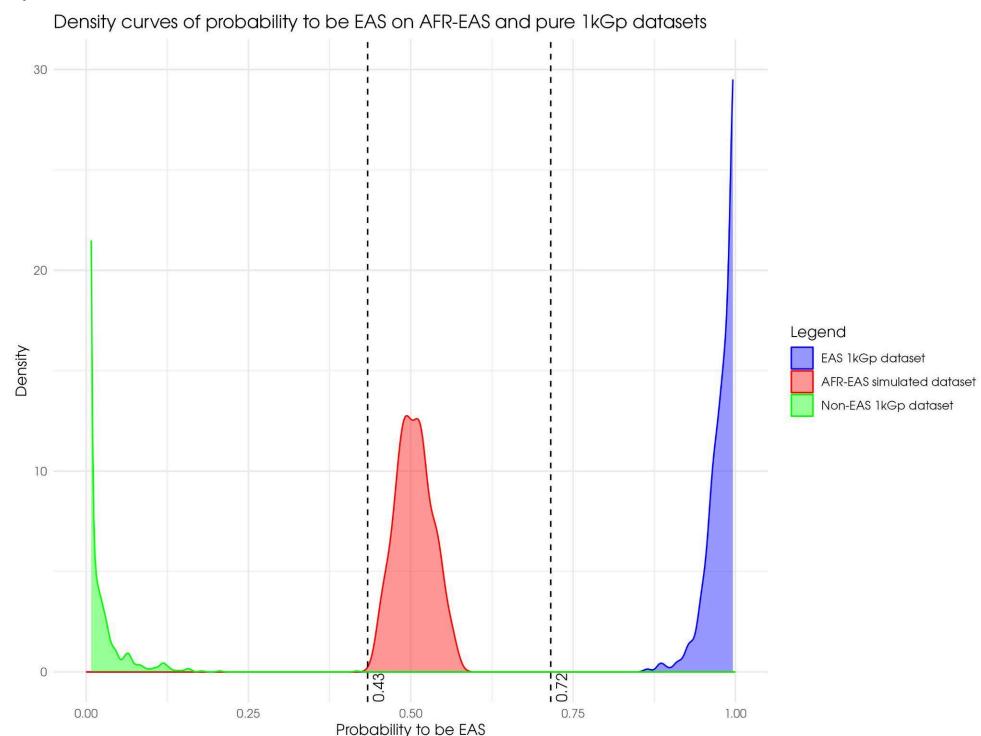


Graphiques de densités des individus d'origine africaine (image A) et américaine (image B) sur différents ensembles, pour la combinaison d'origines AFR-AMR.

A)

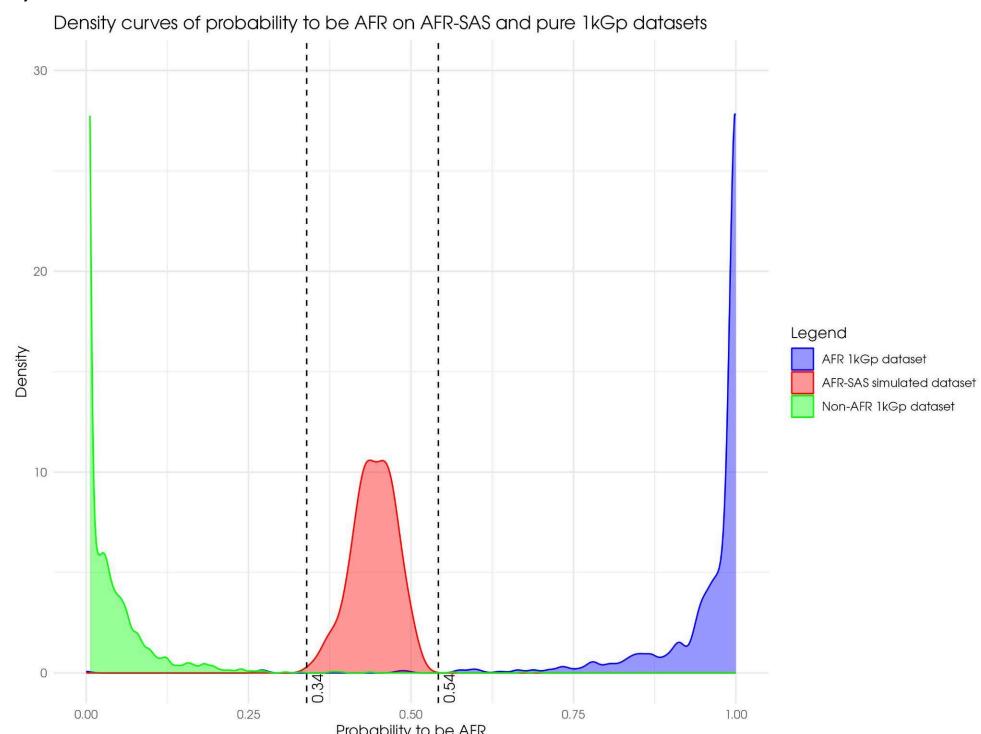


B)

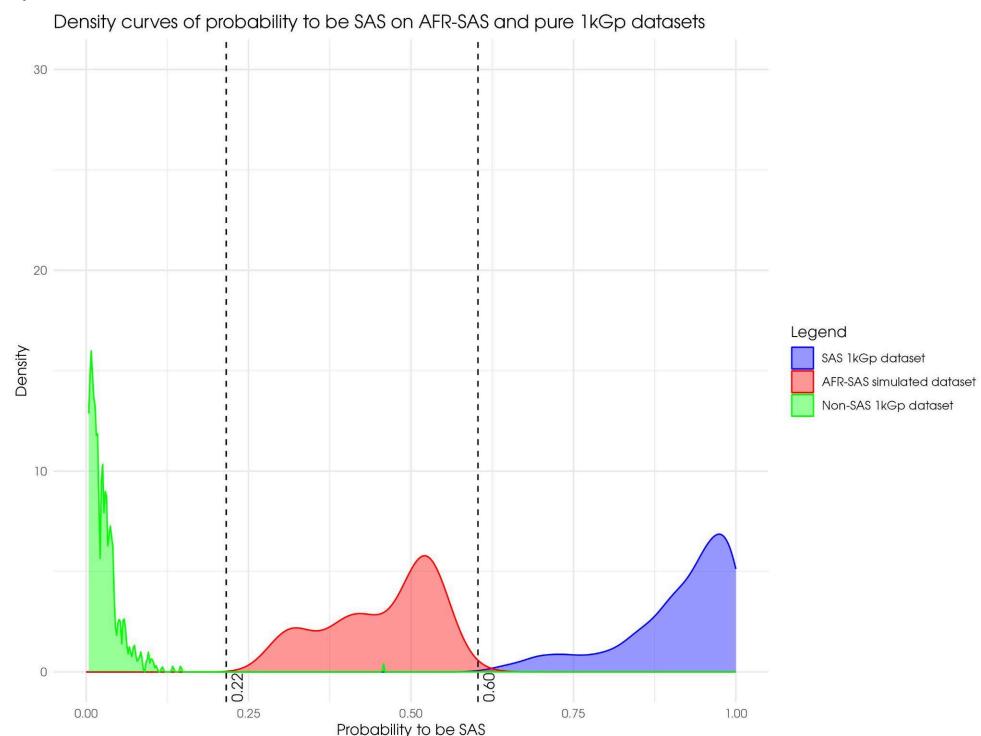


Graphiques de densités des individus d'origine africaine (image A) et Est-asiatique (image B) sur différents ensembles, pour la combinaison d'origines AFR-EAS.

A)



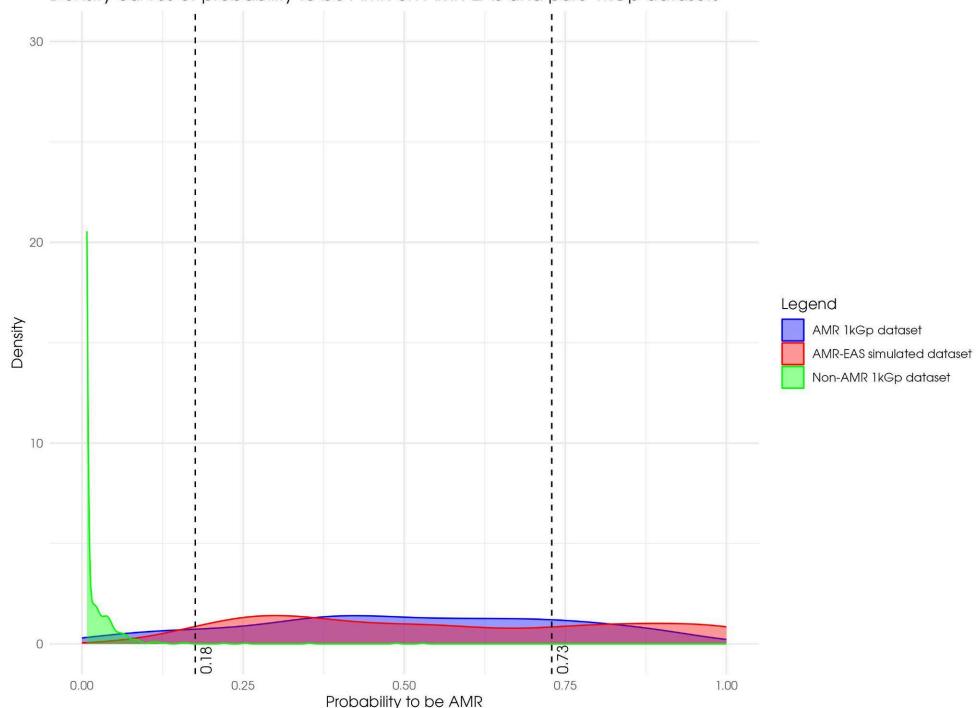
B)



Graphiques de densités des individus d'origine africaine (image A) et Sud-asiatique (image B) sur différents ensembles, pour la combinaison d'origines AFR-SAS.

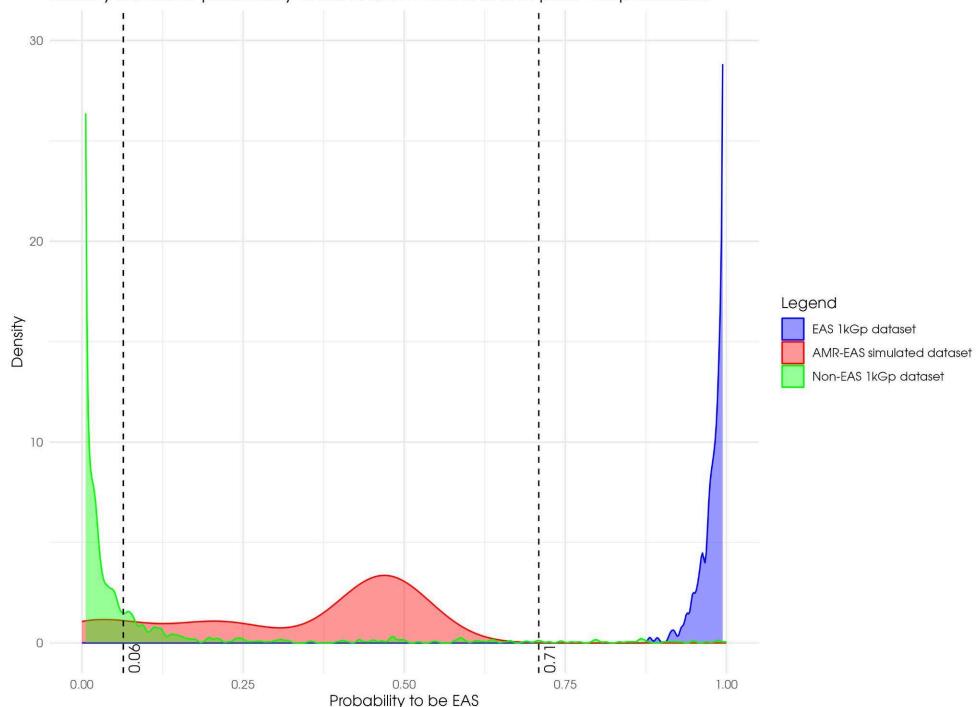
A)

Density curves of probability to be AMR on AMR-EAS and pure 1kGp datasets



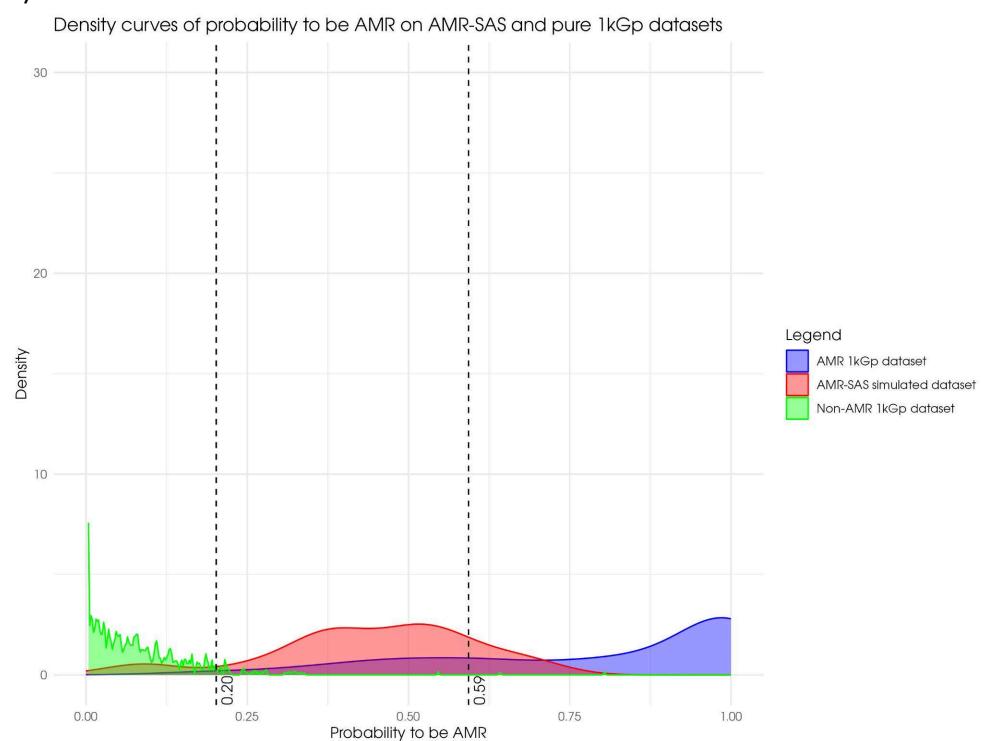
B)

Density curves of probability to be EAS on AMR-EAS and pure 1kGp datasets

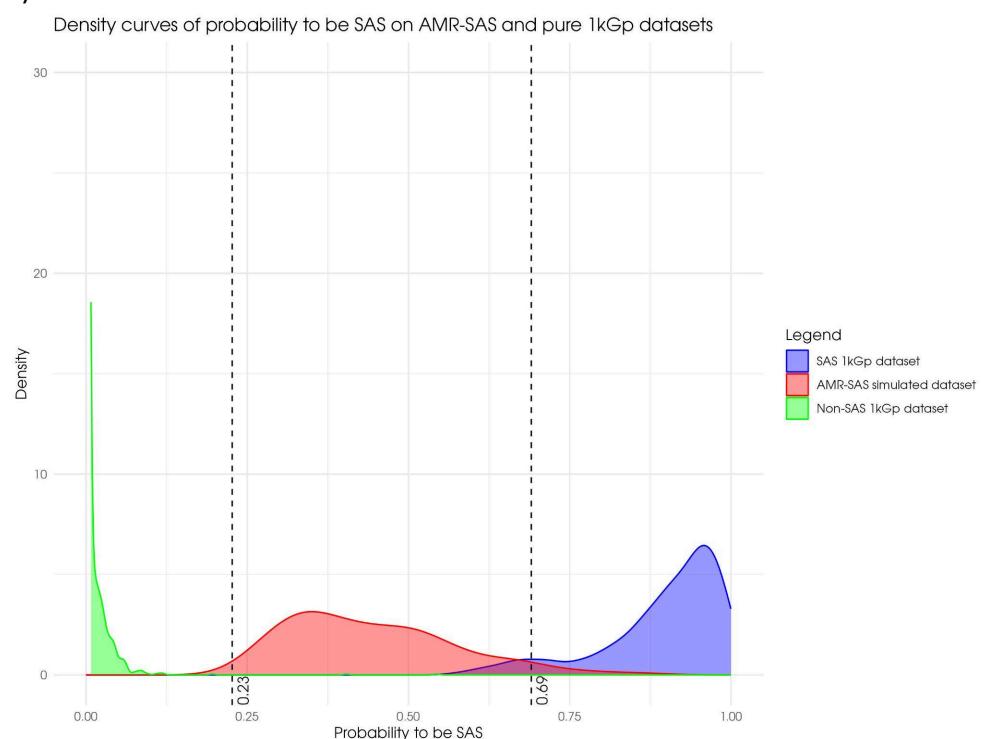


Graphiques de densités des individus d'origine américaine (image A) et Est-asiatique (image B) sur différents ensembles, pour la combinaison d'origines AMR-EAS.

A)

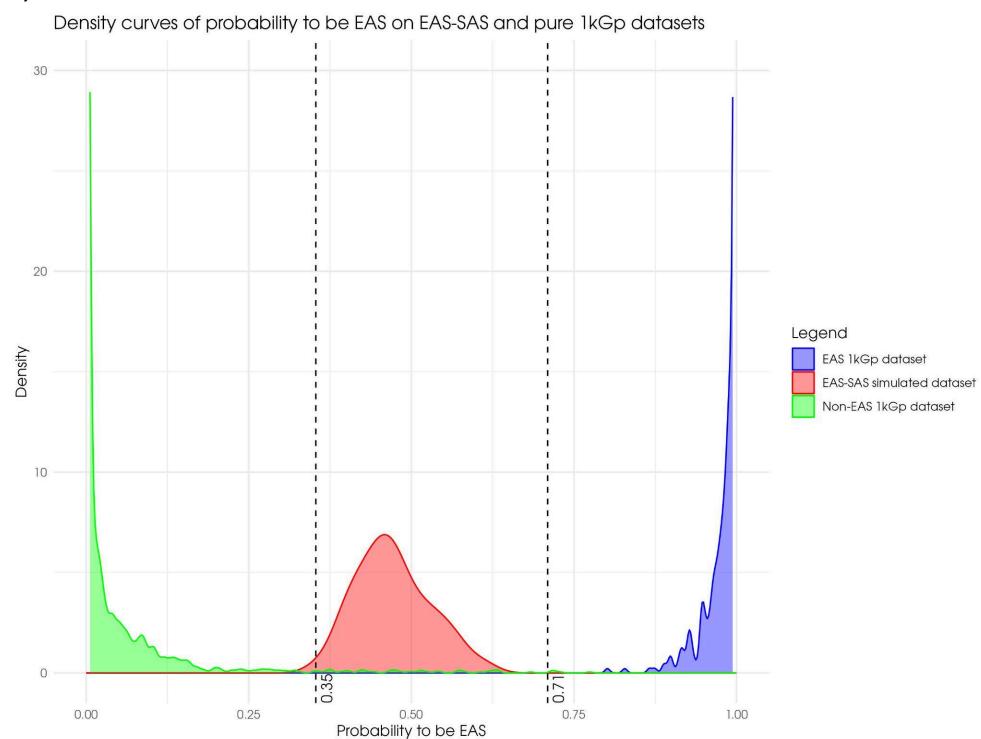


B)

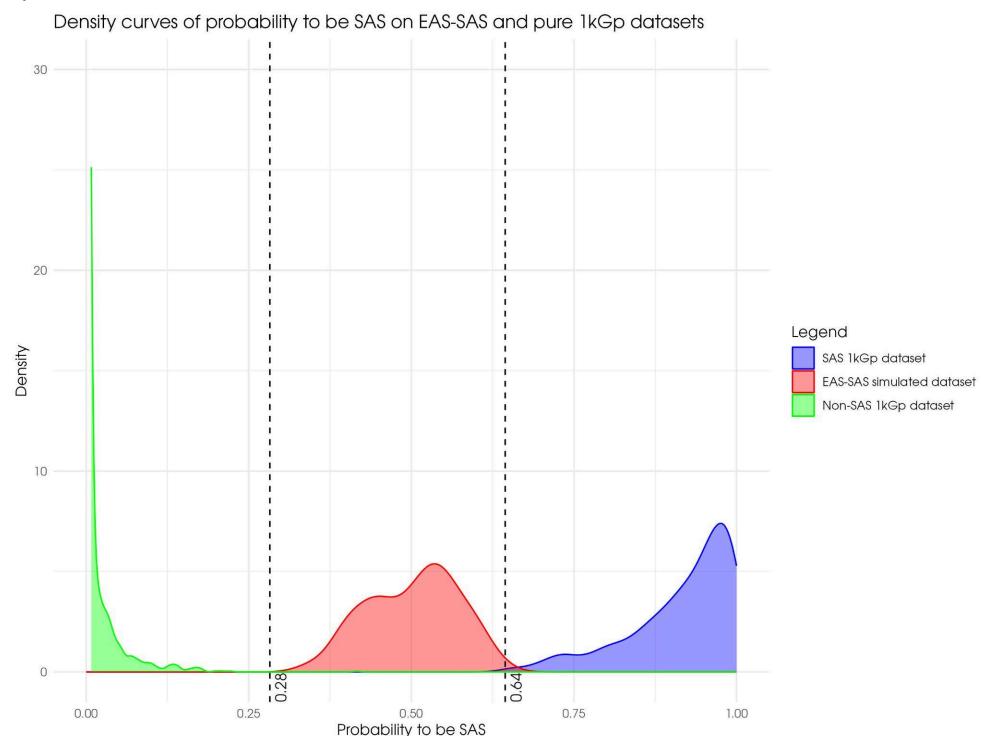


Graphiques de densités des individus d'origine américaine (image A) et Sud-asiatique (image B) sur différents ensembles, pour la combinaison d'origines AMR-SAS.

A)



B)



Graphiques de densités des individus d'origine Est-asiatique(image A) et Sud-asiatique (image B) sur différents ensembles, pour la combinaison d'origines EAS-SAS.