Étude de la consanguinité des bisons d'Europe avec un modèle de Markov caché à multiples classes autozygotes

Auteur : Bertrand, Amandine
Promoteur(s) : Druet, Tom
Faculté : Faculté des Sciences
Diplôme : Master en biochimie et biologie moléculaire et cellulaire, à finalité spécialisée en bioinformatique et modélisation
Année académique : 2016-2017
URI/URL : http://hdl.handle.net/2268.2/3154

Étude de la consanguinité des bisons d'Europe avec un modèle de Markov caché à multiples classes autozygotes

Université de Liège

GIGA-R
Unit of Animal Genomics

Faculté des Sciences
Département des Sciences de la Vie
Unit of Eukaryotic Phylogenomics

Promoteurs Tom Druet ***et*** Denis Baurain

Mémoire présenté par
Amandine Bertrand

En vue de l'obtention du grade de
Master en Biochimie et Biologie Moléculaire et Cellulaire, à finalité spécialisée en Bioinformatique et Modélisation

Année académique 2016-2017

Remerciements

Lors de la réalisation de ce mémoire, j'ai eu la chance d'être bien entourée.

Je voudrais remercier le Dr. Tom Druet de m'avoir accueillie dans son équipe. Son encadrement, ses conseils, ses avis et sa patience m'ont permis d'avancer, tant sur l'apprentissage de la génomique et de la bioinformatique que sur le plan scientifique de façon générale.

Je remercie également le Pr. Denis Baurain de partager son entousiasme pour la bioinformatique, ainsi que pour son temps et ses remarques et conseils constructifs.

Merci aussi à tous les membres des deux équipes dans lesquelles j'ai eu la chance de travailler. L'ambiance au beau fixe ainsi que les réponses aux questions, parfois irréfléchies, ont mis de la couleur sous le ciel couvert de Belgique.

Je tiens également à remercier mes trois amis fous de bioinformatique pour m'avoir écoutée réfléchir tout haut ou, au moins, pour avoir fait semblant, ainsi que toutes les nombreuses personnes qui, de près ou de loin, m'ont soutenue tout au long de cette année.

Enfin, last but not least*, je remercie ma famille. Malgré les épreuves traversées ces deux dernières années, merci d'avoir toujours été là pour me soutenir et m'encourager dans la voie que j'ai choisie. La force de ceux qui ne sont plus là ne s'est pas perdue.*

Résumé du Mémoire

Étude de la consanguinité des bisons d'Europe avec un modèle de Markov caché à multiples classes autozygotes

présenté par Amandine Bertrand et effectué dans le laboratoire de Génomique Animale avec en promoteur le Dr. T. Druet et en collaboration avec le laboratoire de Phylogénomique des Eucaryotes du Prof. D. Baurain durant l'année académique 2016-2017.

Les bisons d'Europe (*Bison bonasus*) ont disparu à l'état sauvage à la fin de la première guerre mondiale. Ils ont survécu dans divers zoos et ont ensuite été réintroduits dans la forêt de Białowieża, en Pologne. Pour étudier les conséquences de ce *bottleneck* et étudier l'état actuel de conservation de l'espèce, une caractérisation de la consanguinité a été réalisée. L'utilisation d'une approche génomique permet de s'affranchir de la généalogie. Le modèle HMM multiclasses utilisé (`ZooRoH`) estime la consanguinité globale et locale réalisée. Il évalue également l'âge des ancêtres (c'est-à-dire à quelle génération du passé ils appartenaient) à l'origine de la consanguinité. Les objectifs de ce travail étaient de porter `ZooRoH` sous l'environnement R en améliorant ses propriétés (entre autres pour l'estimation des paramètres) avant de caractériser la consanguinité chez les bisons d'Europe.

Tout d'abord, la méthode d'estimation des paramètres d'origine (EM) a été remplacée par la méthode L-BFGS-B de la fonction `optim`. Celle-ci nécessite moins d'itérations pour converger et a, en plus, un critère d'arrêt plus adapté. Une reparamétrisation a aussi été réalisée, permettant d'obtenir des paramètres avec les propriétés désirées, ainsi que de rendre les résultats plus facilement interprétables et lisibles. Ces modifications ont été évaluées sur des jeux de données simulées et réelles en comparant les vraisemblances, paramètres et coefficients de consanguinité estimés via 4 modèles (2e, 4p, 4e, 10p). Il a été montré que les modèles 4p et 10p sont plus stables que le modèle 4e. Ainsi, lors de l'étude subséquente de la consanguinité des bisons avec ce programme, le modèle 10p a été utilisé pour allier deux avantages : facilité de comparaison des individus et vue large du passé avec ses 9 classes autozygotes (IBD).

Préalablement à l'estimation de la consanguinité globale des bisons avec `ZooRoH` porté sous R (`ZooRoH/R`), la structure de leur population a été analysée (MDS avec `PLINK`, PCA avec `GCTA` puis `ADMIXTURE`) pour conclure à son homogénéité. Une consanguinité globale de 40% en moyenne dans la population a été estimée, ce qui est très élevé à la fois par rapport au Blanc-Bleu Belge, considéré déjà comme fortement consanguin, et par rapport à une consanguinité attendue d'un accouplement père-fille (25%). En retirant des marqueurs en déséquilibre de liaison, la consanguinité plus ancienne est difficilement détectable. Cependant la consanguinité reste assez récente et semble se stabiliser : les animaux nés plus récemment voient leur consanguinité se déplacer légèrement vers des classes plus anciennes.

Ainsi, ce travail est un premier pas vers la création d'un package R pour `ZooRoH` qui pourra être appliqué à une population au passé démographique inconnu. Il a également permis de confirmer une réintroduction des bisons réussie et de suspecter un événement de purge passé.

Abstract

Study of inbreeding in the European bison with a multiple autozygous classes hidden Markov model

European bison (*Bison bonasus*) disappeared in the wild at the end of the First World War. They survived in several zoos and were subsequently reintroduced into the forest of Białowieża, Poland. In order to study the consequences of this bottleneck and to study the present state of conservation of the species, a characterization of inbreeding was carried out. The use of a genomic approach makes it possible to be free from the use of the genealogy. The multiclass HMM model used (`ZooRoH`) estimates the realized global and local inbreeding. It also evaluates the age of the ancestors (ie, which generation of the past they belonged to) at the origin of the inbreeding. The aim of this work was to bring `ZooRoH` under the R environment by improving its properties (including parameter estimation) before characterizing inbreeding in European bison.

First, the method of estimating the original parameters (EM) was replaced by the L-BFGS-B method of the `optim` function. This requires fewer iterations to converge and has, in addition, a more appropriate stop criterion. Reparametrization was also carried out, making it possible to obtain parameters with the desired properties, as well as to make the results more easily interpretable and readable. These modifications were evaluated on simulated and real data sets by comparing the estimated likelihoods, parameters and coefficients of inbreeding via 4 models (2e, 4p, 4e, 10p). It has been shown that the 4p and 10p models are more stable than the 4e model. Thus, in the subsequent study of the inbreeding of wisent with this program, the 10p model was used to combine two advantages : ease of comparison of individuals and broad view of the past with its 9 autozygous classes (IBD).

Prior to the estimation of the global consanguinity of wisent with `ZooRoH` under R (`ZooRoH/R`), the structure of their population was analyzed (MDS with `PLINK`, PCA with `GCTA` then `ADMIXTURE`) to conclude that it is homogeneous. An overall consanguinity of 40% on average in the population was estimated, which is very high both compared to Belgian Blue Beef, considered already highly inbred, and compared to an expected inbreeding between father and daugther (25%). By removing markers in linkage disequilibrium, older inbreeding is difficult to detect. However, inbreeding remains fairly recent and seems to be stabilizing : more recently born animals see their inbreeding moving slightly towards older classes.

Thus, this work is a first step towards the creation of a R package for `ZooRoH` which can be applied to a population with an unknown demographic past. It also confirmed a successful reintroduction of wisent and suspected a past purge event.

Table des matières

Table des figures

Liste des tableaux

Abréviations

BBB	Blanc-Bleu Belge
DL	Déséquilibre de liaison
EM	Espérance Maximisation (*Expectation Maximization*)
F	Coefficient de consanguinité (globale)
G_i	Taux de la distribution exponentielle associée à la classe i
HD	*High Density*
HMM	Modèle de Markov caché (*Hidden Markov Model*)
IBD	*Identical By Descent*
IBS	*Identical By State*
L-BFGS-B	*Limited-memory Broyden-Fletcher-Goldfarb-Shanno Bound-constraints*
LD	*Low Density*
M_i	Proportion de mélange associée à la classe i
MAE	Erreur absolue moyenne (*Mean Absolute Error*)
MAF	*Minor Allele Frequency*
MDS	*MultiDimensional Scaling*
PCA	*Principal Component Analysis*
RoH	Segment homozygote (*Run of Homozygosity*)
SNP	*Single Nucleotide Polymorphism*

1 Introduction

L'histoire des bisons d'Europe (*Bison bonasus*) est assez particulière. En effet, ces animaux ont disparu à l'état sauvage il y a près d'un siècle, le dernier ayant été abattu en 1927. Pourtant, les bisons ont survécu. Ils ont perduré dans un premier temps grâce à 54 individus hébergés dans divers zoos à travers le monde, avant d'être réintroduits dans la forêt de Białowieża, en Pologne. À l'heure actuelle, plus de 5000 individus vivent en liberté[1,2]. Ceux-ci ont donc des ancêtres communs récents. Cependant, des données de pedigree complets pour étudier la consanguinité avec des méthodes généalogiques font défaut. Avec des marqueurs moléculaires, il est maintenant possible de l'estimer de façon globale, mais aussi locale, ainsi que d'estimer à quand elle remonte.

1.1 Bisons d'Europe

1.1.1 Déclin

L'extinction des bisons d'Europe a fait suite à une lente diminution de la population (encadré 1) depuis le dernier maximum glaciaire, il y a 20000 ans. Cette diminution est le résultat de différents changements ayant eu lieu à cette époque tels que l'expansion de la population humaine, le réchauffement du climat et le changement de la végétation[3]. En effet, les steppes, dans lesquelles ces animaux vivaient, se sont réduites pour être peu à peu remplacées par des forêts.

Encadré 1 : *Population et fréquence allélique*

Pour rappel, une **population** regroupe les individus d'une même espèce ayant la possibilité de se reproduire. Elle suit donc des critères spatiaux, temporels et génétiques[4]. Une population est à l'**équilibre de Hardy-Weinberg** lorsque la reproduction se fait de façon aléatoire et que sa taille est très grande (infinie). Ce modèle très simple implique que les fréquences alléliques sont identiques chez les mâles et les femelles et sont stables de génération en génération. On peut ainsi calculer les fréquences génotypiques[4] à partir de p et q, qui sont les fréquences alléliques des allèles A et a :

$$AA = p^2 \quad Aa = 2pq \quad aa = q^2$$

Cependant, une population est rarement de taille infinie. Ainsi, une **dérive génétique** aléatoire, c'est-à-dire un changement de la fréquence allélique, est régulièrement observée. Cette **différenciation génétique** provoque notamment la fixation ou la disparition de certaines mutations ou de certains allèles au cours des générations[4]. Ainsi, si une population se retrouve divisée spatialement pendant un laps de temps assez long, cette dérive pourra provoquer une spéciation allopatrique, c'est-à-dire dû à l'isolement géographique.

Le territoire occupé par les populations de bisons des steppes (*Bison priscus*) à la fin du Pléistocène recouvrait une grande partie de l'Eurasie. Lors des changements climatiques associés à cette période, ces animaux ont dû s'adapter. Une division a probablement eu lieu dans la popu-

lation, conduisant à la naissance du bison d'Europe actuel (*Bison bonasus*), qui vit plutôt dans les forêts[1,5]. Leur territoire a cependant diminué lentement au cours des siècles, entraînant, entre autres, une nouvelle, lente et constante diminution de la taille de leur population (encadré 2).

Encadré 2 : *Réduction de la taille d'une population*

Plusieurs causes peuvent être à l'origine d'une réduction d'une population sauvage, notamment sa subdivision en sous-populations se retrouvant isolées ou encore un *bottleneck* (goulot d'étranglement). Ces deux phénomènes peuvent se retrouver dans différentes situations.

En effet, certaines populations ont une hiérarchie, n'autorisant des accouplements qu'entre certains individus. Ceci implique une subdivision de la population malgré une localisation du territoire commune. De plus, des sous-groupes peuvent se former dans une population en raison de sa structure, par exemple une colonie, un troupeau, une meute… L'une de ces sous-populations peut se retrouver isolée du reste de la population à la suite d'un événement de migration (par exemple sur une île) ou d'une "catastrophe naturelle" comme un tremblement de terre.

Les causes de *bottleneck* sont souvent liées à celles des divisions de la population étant donné que ce terme fait référence à une période où la taille de la population est petite[4]. Ainsi, si une population voit sa taille fortement diminuer à un moment de son histoire puis, eventuellement, réaugmenter, elle aura subi une période de goulot d'étranglement. C'est typiquement ce qu'il se passe lors de la colonisation d'une île par quelques individus (effet fondateur) mais également chez les populations en voie de disparition.

En plus des catastrophes naturelles, l'isolement d'une partie de la population ou la réduction du nombre d'individus peut être liée à l'activité humaine. En effet, le morcellement du territoire de certaines populations peut être dû, par exemple, à la déforestation, au braconnage ou encore à la chasse intensive. Ces activités ont été des causes importantes de la disparition des bisons sauvages (*Bison bonasus*) en Europe[1,2].

1.1.2 Survie

Le large territoire qu'occupaient les bisons d'Europe il y a plus d'un millénaire a aidé à la formation de sous-espèces. Les deux principales étaient les lignées "Lowland" et "Caucasian" (respectivement *Bison bonasus bonasus* et *Bison bonasus caucasicus*). Elles se répartissaient dans des régions différentes, la première se trouvant plutôt dans les forêts alors que la seconde plutôt dans les montagnes caucasiennes (d'où son appellation)[6]. Mais tous les bisons d'Europe étaient des cibles de choix lors des grandes chasses organisées entre le XI$^{\text{ième}}$ et le XIX$^{\text{ième}}$ siècle, avec une chasse plus intense au XV$^{\text{ième}}$ et XVI$^{\text{ième}}$ siècles, provoquant leur disparition dans de nombreux pays[1,6]. Au XVIII$^{\text{ième}}$ siècle, ils n'en restaient plus qu'en Pologne (et seulement des animaux de la lignée "Lowland"). Lorsque le dernier spécimen sauvage fut tué en 1927, un programme de conservation fut mis en place[1]. À la fin de la première guerre mondiale, il y avait 54 survivants dans divers zoos à travers le monde, descendant eux-mêmes de 12 individus[7], les fondateurs (5 mâles et 7 femelles). Durant l'entre-deux-guerres, des efforts ont été fournis pour garder la race "pure", c'est-à-dire sans croisement avec des bisons d'Amérique (*Bison bison*)[1], mais également

sans croisement avec la lignée caucasienne. Or, l'un des 12 fondateurs était de la sous-espèce *Bisons bonasus caucasicus*. Il y a donc une lignée hybride "Lowland-Caucasian"[8]. À la sortie de la première guerre mondiale, seulement 7 des 54 survivants (4 mâles et 3 femelles) ont été les fondateurs de la lignée "Lowland" considérée comme "pure" (sous-espèce *Bison bonasus bonasus*)[9].

Malgré une nouvelle diminution de leur nombre pendant la deuxième guerre mondiale, des animaux ont ensuite été remis en liberté en 1952 dans la forêt de Białowieża, en Pologne. En 2000, 2900 individus étaient en vie, dont à peu près 1700 en liberté[1]. En 2014, la population vivant en liberté et semi-liberté s'élevait à plus de 5000 individus[2] et le statut de l'espèce a pu être rétrogradé de l'état "en voie de disparition" à celui de "menacé"[7]. Outre la Pologne, des bisons vivraient maintenant, au moins en semi-liberté, en Bielorussie (notamment dans la forêt de Białowieża), Lettonie, Lituanie, Russie, Roumanie et Ukraine[1].

Au vu des événements traversés par cette espèce, il est très probable que des accouplements ont eu lieu entre individus apparentés, entraînant un certain niveau de consanguinité dans le génome de ces animaux. Une étude de celle-ci peut être intéressante. Elle peut en effet permettre une compréhension de l'état actuel de conservation de l'espèce, et ainsi, donner une idée d'un programme de conservation efficace pour d'autres espèces subissant actuellement un morcellement de leur territoire et des subdivisions de leur population pouvant les mener vers une extinction[10].

1.2 Consanguinité

Lorsqu'on parle de consanguinité (encadré 3), il est fréquent de faire la distinction entre des allèles **IBD** et **IBS**. IBD, abréviation pour *Identical By Descent* (c'est-à-dire "identique par descendance"), indique que deux allèles homologues proviennent d'un même ancêtre commun, alors qu'IBS (*Identical By State*, "identique par état") signifie que ces allèles sont les mêmes, sans prise de position quant à leur origine[11]. Ainsi, lorsque des individus ont au moins un ancêtre en commun, c'est-à-dire lorsqu'ils sont **apparentés**, la contribution génétique de cet aïeul à leur descendance sera plus importante. En effet, un allèle donné à l'individu consanguin pourra avoir été transmis à la fois par son père et par sa mère. Il sera donc IBD[4]. Ces termes peuvent également être appliqués à un **segment** d'ADN, c'est-à-dire à une portion du génome d'un individu. Lorsqu'on parle d'un segment hérité, cela signifie que celui-ci aura été transmis tel quel par un parent, sans segmentation de celui-ci par une mutation subséquente et sans événement de crossing-over[12].

Encadré 3 : *Consanguinité : coefficient et population de référence*
La **consanguinité**, c'est-à-dire le résultat de l'accouplement de deux individus apparentés[13], se retrouve aussi bien chez les plantes que chez les animaux. Pour permettre de la quantifier, Wright définit en 1921 les relations de parenté (r) puis, en 1922, le **coefficient de consanguinité** (F)[14,15]. Ce dernier représente la probabilité de tirer deux fois un allèle descendant d'un ancêtre commun

(que ce soit pour un individu ou une population). Il calcule ce coefficient sur base du pedigree d'un individu, ce qui devient la façon traditionnelle de l'estimer. Cet arbre généalogique remonte donc sur plusieurs générations et ce sera la première, la plus ancienne, qui sera considérée comme **population de base**. Celle-ci comprend des individus, supposés tous non apparentés, vivant à un moment précis dans le passé. Lorsqu'il est question de consanguinité, on se réfère toujours à cette population de référence[16], il s'agit donc d'une mesure relative.

1.2.1 Causes

Plusieurs causes peuvent générer de la consanguinité. Dans les populations sauvages, beaucoup sont liées à la réduction du nombre d'individus. En effet, au cours des générations, si leur nombre diminue, les chances d'accouplement entre membres d'une même famille augmentent. Ainsi, la **taille efficace d'une population** (N_e) est un paramètre important dans l'estimation de la variabilité génétique et de son évolution[17,18]. Elle peut être vue comme une mesure du nombre de reproducteurs indépendants les uns des autres. Chez les bisons d'Europe, le *bottleneck* subi entre 1940 et 1945 (diminution de 42% de la population) a entraîné une perte d'un type de chromosome Y dans la population "Lowland". Ainsi, tous les mâles actuels de cette lignée ont le même chomosome Y[1].

Lorsque des animaux sont utilisés dans la production, la sélection de caractères d'intérêt agronomique est fréquente. Pour ce faire, il arrive souvent que le nombre de reproducteurs sélectionnés soit limité, plus particulièrement chez les mâles. Leur patrimoine génétique se retrouvera, dès lors, fortement disséminé dans la population. Ainsi, l'accouplement d'individus apparentés est presque inévitable. C'est notamment le cas chez les bovins (encadré 4).

Encadré 4 : *Blanc-Bleu Belge*

Les races utilisées en Europe sont de l'espèce *Bos taurus*. Le Blanc-Bleu Belge (BBB) en fait partie. Cette lignée a commencé à être sélectionnée au cours du XIX$^{\text{ème}}$ siècle mais ce n'est que dans les années '70s que le terme de "Blanc-Bleu Belge" apparait et est considéré comme race[19]. C'est également à ce moment que s'est développée l'insémination artificielle. Une cinquantaine de taurillons BBB sont sélectionnés chaque année pour la reproduction.[20] Les "meilleurs" étant utilisés plus fréquemment, la proportion de leur génome dans la population sera également plus importante. Ceci a donc entraîné, au bout de quelques générations, l'apparition de segments IBD dans le génome de ces bovins.

Le critère principal de sélection des BBB a été le développement de la masse musculaire (caractère culard), qui s'est accrue fortement ces cinquante dernières années[19]. Une mutation bien connue impliquée dans ce phénomène est celle de la myostatine (*MSTN*), un régulateur négatif du développement de la masse des muscles squelettiques. Ce gène a subi chez ces animaux une délétion de 11 pb induisant une terminaison précoce de la traduction. Les BBB ont donc une protéine tronquée et inactive[21]. Cette mutation serait héritée d'un seul animal vivant dans les années '50s. Le dévelop-

pement musculaire est cependant un caractère multifactoriel complexe et ne dépend pas seulement de cette protéine.

1.2.2 Conséquences

La principale conséquence d'une augmentation de la consanguinité est l'augmentation de l'**homozygotie**[22]. En effet, plus l'ancêtre commun est récent, plus les segments auront de chances d'être longs, sans position hétérozygote provenant d'une mutation subséquente ou d'une fragmentation due à un crossing-over[12]. Ceci implique que des allèles peuvent être aléatoirement perdus ou fixés dans une population où la consanguinité est élevée. De plus, la sélection étant moindre sur une petite population, des mutations délétères se retrouvent plus fréquemment à l'état homozygote. Ces divers facteurs entraînent généralement une réduction de la fitness (encadré 5), proportionnelle à l'augmentation de F, pouvant conduire au phénomène de dépression de consanguinité (*inbreeding depression*)[23].

Cette **dépression de consanguinité** peut être définie comme la réduction de fitness due au faible taux de survie, d'accouplement et/ou de reproduction dans la progéniture d'individus apparentés lorsqu'elle est comparée à la fitness de la progéniture d'individus non apparentés[24]. Elle varie selon les espèces, l'environnement et les populations ayant connu des histoires génétiques et démographiques différentes[23].

Encadré 5 : *Fitness*

La *fitness* est le succès de transmission des gènes d'un individu à la génération suivante. Elle est estimée en mesurant le taux de survie et de reproduction[25]. Les variants alléliques influençant négativement cette fitness peuvent être répartis en deux groupes, ceux ayant un effet important (létal ou sublétal) et ceux ayant un effet modéré à faible (nuisibles). Dans le premier cas, ces allèles seront presque complètement récessifs, sinon ils seraient tellement délétères qu'ils ne pourraient pas survivre dans la population. Dans le deuxième cas, ils ne pourront être que partiellement récessifs car leur effet sur la survie est moindre[24]. Lors d'accouplements consanguins, impliquant un nombre de partenaires possibles limités, des mutations délétères avec un effet moindre sur la sélection vont voir leur fréquence dans la population augmenter. Cependant, même si leurs impacts individuels sont moindres, leur accumulation dans un génome peut mener à une dépression de consanguinité[23].

Plusieurs hypothèses ont été avancées pour expliquer la base génétique de cette dépression. La plus étudiée est celle de la dominance partielle[26], équivalente à l'appellation "variants partiellement récessifs" (encadré 5). Autrement dit, il s'agit de l'accroissement de l'expression d'allèles récessifs délétères. Une autre hypothèse émise est celle de la surdominance, c'est-à-dire la supériorité des hétérozygotes sur les deux types d'homozygotes[26]. Ces deux hypothèses peuvent être difficiles à distinguer car des variants récessifs délétères pourraient être en déséquilibre de liaison, ce qui résulterait en une pseudo-surdominance[24,27].

Une méthode permettant d'estimer la dépression de consanguinité est d'utiliser des données d'un caractère de fitness pour des individus ayant différents taux de consanguinité. Une autre est de comparer la fitness moyenne d'une génération non consanguine avec celle d'une génération consanguine (dont le niveau de consanguinité est connu). Ces estimations montrent que l'impact de la consanguinité est plus important chez les populations vivant dans un environnement naturel. Le sauvetage génétique (*genetic rescue*), c'est-à-dire l'introduction de nouveaux variants dans une population dans laquelle des variants délétères ont été fixés par dérive génétique, semble augmenter dans la plupart des cas la fitness. Cependant, la dépression génétique reste difficile à étudier dans la nature[24].

Celle-ci peut néanmoins être réduite par une **purge**. Il s'agit de l'augmentation d'une sélection de purification des variants récessifs partiellement délétères par consanguinité (dans les cas extrêmes). En effet, celle-ci les expose en augmentant leur homozygotie[24]. Ainsi, la consanguinité peut aider à purger les variants délétères (type perte de fonction) d'une population. Un tel phénomène a par exemple été observé chez les gorilles des montagnes[28]. Cependant, cette purge ne sera efficace, dans une population restreinte, que contre les mutations létales[23]. Comme la purge réduit la fréquence des allèles délétères (exposés car homozygotes), elle réduit également la diminution de fitness liée à la consanguinité[24]. La purge, même dans les cas où elle est efficace, ne peut cependant pas tout contrebalancer[23]. De plus, ce phénomène implique une diminution supplémentaire de la taille de la population.

1.3 Méthodes d'estimation

1.3.1 À partir de la généalogie

L'estimation de la consanguinité individuelle a traditionnellement été calculée à partir du pedigree (F_P, encadré 3). Bien qu'elle soit maintenant en concurrence avec les méthodes utilisant des marqueurs génétiques, elle reste fréquemment utilisée[29].

Pour ne pas risquer d'omettre un ancêtre assez récent pour contribuer à la consanguinité actuelle, plusieurs générations sont nécessaires à la construction de l'arbre généalogique. En effet, la génération la plus ancienne de l'arbre est considérée comme non apparentée (encadré 3). Si dans cette génération il y a deux frères, par exemple, et qu'un accouplement se réalise entre leurs enfants, la consanguinité résultante sera ignorée. Souvent, il est difficile, voire impossible, de remonter assez loin, ce qui entraine une sous-estimation de la proportion du génome qui est IBD[12]. De plus, en ce qui concerne les animaux sauvages, ce pedigree est souvent inconnu ou requiert beaucoup de temps sans qu'on soit pour autant certain de son exactitude ni d'arriver à identifier tous les membres d'une même famille. Aussi, pour certaines espèces, aucun pedigree n'est disponible[24].

Quand on en dispose, ce pedigree permet d'estimer la **consanguinité attendue** d'un individu en évaluant F. Ainsi, un frère et une sœur auront la même. Cependant, la **consanguinité réalisée**

peut être différente. Des frères et soeurs n'auront donc pas exactement la même consanguinité, en raison de la ségrégation aléatoire des allèles et de la recombinaison[29]. Via cette méthode, il n'est pas possible de mesurer cette différence. À cette fin, l'utilisation de marqueurs moléculaires est préférable.

1.3.2 À partir de marqueurs moléculaires

Ce type de mesures s'est développé assez récemment, avec l'avènement des techniques de génotypage et de séquençage. Parmi les différentes méthodes développées, deux grands types utilisant des marqueurs moléculaires se détachent. Elles permettent d'obtenir une estimation de F.

Le premier type est employé en ignorant les positions physiques des marqueurs. Dans ce cas, les mesures de F sont fortement basées sur l'hétérozygotie individuelle (plus elle est faible, plus l'individu est consanguin). L'évaluation la plus simple est l'hétérozygocité à locus-multiples (MLH pour *Multiple-Locus Heterozygosity*), c'est-à-dire la proportion de loci génotypés à travers le génome qui sont hétérozygotes. Cette mesure est considérée comme une estimation de la proportion réelle de loci hétérozygotes (H). On peut ainsi trouver F : $H = H_0(1 - F)$ avec H_0 représentant l'hétérozygotie des individus non-consanguins[12]. Cependant, il a fallu attendre l'utilisation d'un grand nombre de SNPs (*Single Nucleotide Polymorphisms*) pour obtenir une meilleure précision. L'avantage principal de cette méthode est de ne pas avoir besoin d'une carte. Cependant, il est contrebalancé par l'impossibilité d'identifier des segments IBD dans un génome[12]. Ce type de mesure de F permet donc, comme via le pedigree, d'estimer une **consanguinité globale**. Par contre, il permet tout de même d'obtenir une consanguinité réalisée.

Le deuxième type de méthodes repose sur l'utilisation de cartes génétiques. Ici, le coefficient de consanguinité est considéré comme la proportion du génome qui se trouve dans des RoH (*Runs Of Homozygosity*, c'est-à-dire dans des segments homozygotes)[12]. Par conséquent, cette méthode permet de mesurer une **consanguinité locale**. Elle permet également de révéler l'histoire démographique relativement récente de la population étant donné que de plus petits RoH proviennent en moyenne d'un ancêtre plus lointain que de longs RoH[24]. L'identification de ces segments homozygotes peut se faire de différentes façons. `PLINK`[30], par exemple, utilise différents critères pour les identifier : la densité en SNPs, le nombre de positions hétérozygotes admises et la longueur minimum d'un segment. Une autre approche assez répandue est celle utilisant une fenêtre glissante et calculant le rapport de probabilités des génotypes à l'intérieur de cette fenêtre en supposant que le segment est soit IBD soit non IBD[12]. Une approche plus récente permet de trouver des segments IBD via un HMM (*Hidden Markov Model*, ou modèle de Markov caché)[31]. Plusieurs programmes ont été mis au point en utilisant ce cadre statistique, notamment `FEstim`[31] et `ZooRoH`[32].

1.4 Modèles de Markov cachés

Les modèles de Markov cachés, ou HMM pour *Hidden Markov Models* en anglais, constituent un outil puissant et largement utilisé dans plusieurs domaines dont la bioinformatique, par exemple lors de l'étude du contenu en GC, de la prédiction de gènes, de la recherche par homologie de séquences, etc. En effet, ils sont utiles lors de l'étude d'une séquence de caractères car ils permettent de retrouver des caractéristiques sous-jacentes (appelées *états cachés*) de certaines sous-parties[33].

Dans le cas du contenu en GC, un brin d'ADN peut être séparé en régions riches en GC et en régions riches en AT. Ces deux types de régions sont des caratéristiques sous-jacentes à la séquence de bases observées (A, C, G, T).

1.4.1 Principe

Le principe des HMM est donc de parcourir une séquence et de déterminer à quel état caché chaque caractère (ici les bases) appartient. Pour ce faire, le modèle se base sur des probabilités. Les probabilités initiales (π), c'est-à-dire la probabilité de commencer la séquence dans un état particulier, se distinguent des probabilités de transition A, c'est-à-dire la fréquence à laquelle on peut passer d'un état à un autre[34]. Si un modèle simple avec deux états (ici riche en GC et riche en AT) est considéré (figure 1), quatre transitions ($a_{q_1 q_2}$ avec q_1 l'état de départ et q_2 l'état d'arrivée) sont possibles dans l'intervalle entre deux bases : soit on reste dans le même état (GC $\rightarrow$ GC ou AT $\rightarrow$ AT) soit on change (GC $\rightarrow$ AT ou AT $\rightarrow$ GC). Une matrice carrée de dimension deux est ainsi obtenue (figure 2, matrice A).

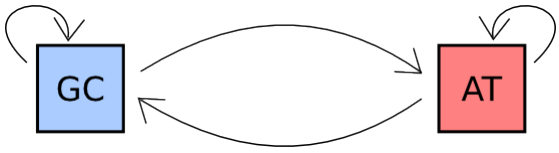


FIG. 1 : Illustration des transitions d'états. Modèle simple à deux états (ici riche en GC et riche en AT). Quatre transitions possibles dans l'intervalle entre deux caractères d'une séquence : soit on reste dans le même état (GC $\rightarrow$ GC ou AT $\rightarrow$ AT) soit on change (GC $\rightarrow$ AT ou AT $\rightarrow$ GC).

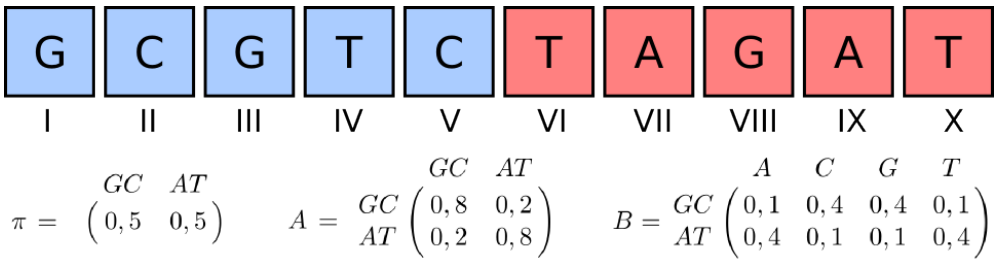
Pour chaque position d'une séquence, plusieurs caractères sont généralement possibles. Dans le cas illustré, on sait que la séquence ADN peut prendre quatre valeurs : soit A, C, G ou T (figure 2). Chacune de ces possibilités de caractère a une probabilité différente selon l'état dans lequel le segment, contenant la base, se trouve. Ainsi, dans l'exemple de la figure 2, la probabilité d'observer le caractère G est différente dans l'état caché GC de l'état AT. On parle de probabilité d'émission B des caractères dans chaque état (b_{kq} avec k la position et q l'état, figure 2)[34].

Ainsi, une probabilité sera obtenue pour une séquence d'états cachés ($Seq = \{q_1, ..., q_K\}$ avec q_i les états à chaque position).

$$P(Seq|\pi, A, B) = \pi_{q_1} \times b_{1q_1} \times \prod_{k=2}^{K} \{a_{q_{k-1}q_k} \times b_{kq_k}\}$$

avec K le nombre de positions dans la séquence.

La figure 2 illustre l'idée sur laquelle fonctionne un HMM. Ainsi, à chaque position, la base est A, C, G ou T. Derrière ces caractères se trouve une information cachée : la base fait partie d'un segment riche en GC ou riche en AT. Pour trouver ce renseignement dissimulé, le HMM va se baser sur des probabilités : une probabilité de l'état de départ (π), une probabilité de transition d'état (A), une probabilité de l'information observée selon l'état (B).



$$\pi = \begin{array}{c} \\ \end{array} \begin{array}{cc} GC & AT \\ \end{array} \begin{pmatrix} 0,5 & 0,5 \end{pmatrix} \qquad A = \begin{array}{c} \\ GC \\ AT \end{array} \begin{array}{c} \begin{array}{cc} GC & AT \end{array} \\ \begin{pmatrix} 0,8 & 0,2 \\ 0,2 & 0,8 \end{pmatrix} \end{array} \qquad B = \begin{array}{c} \\ GC \\ AT \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 0,1 & 0,4 & 0,4 & 0,1 \\ 0,4 & 0,1 & 0,1 & 0,4 \end{pmatrix} \end{array}$$

FIG. 2 : Illustration du principe d'un HMM. À chaque position (carré), la base est soit A, soit C, soit G, soit T. Les états cachés sont soit "riche en GC" (bleu, positions de I à V), soit "riche en AT" (rouge, de VI à X). Pour trouver la probabilité d'être dans un état caché à une position donnée dans une séquence d'états cachés (Seq), **(1)** jusqu'à la position I, π et B doivent être pris en compte pour l'état riche en GC $\pi_{(GC)} \times b_{G(GC)} = 0,5 \times 0,4 = 0,2$; **(2)** on étend ensuite le modèle jusqu'à la position II, A et B doivent être pris en compte pour chaque état, ici on vient de la position I considérée comme riche en GC donc la probabilité d'aller dans l'état riche en GC est de $A_{GC \to GC} \times B_{C(GC)} = 0,8 \times 0,4 = 0,32$. Le principe du HMM est de considérer toutes les possibilités de séquences d'états cachés.

1.4.2 Estimation de la vraisemblance via une procédure *forward*

Comme écrit ci-dessus, le HMM permet de calculer la probabilité d'observer la séquence de caractères selon les probabilités π, A et B. Cette probabilité est appelée vraisemblance (*likelihood*). Elle permet de savoir à quel point le modèle correspond à la "réalité". Pour ce faire, toutes les séquences d'états cachés possibles devraient être prises en compte (c'est-à-dire toutes les combinaisons possibles d'états cachés sur la séquence de caractères). Cependant, ce calcul deviendrait vite très long (il suffit d'essayer toutes les possibilités avec l'exemple de la figure 2 qui ne contient que dix bases pour s'en rendre compte). Il est de l'ordre de $2K \times Q^K$ calculs, avec K correspondant au nombre de positions dans la séquence et Q au nombre d'états possibles[34]. En effet, à chacune des K positions, il y a Q états pouvant être atteints (il y a donc Q^K séquences possibles d'états) et, pour chaque séquence d'états, il y a $2K$ opérations à réaliser (voir équation précédente). Une procédure de calcul plus efficace peut cependant être utilisée : la procédure *forward*.

Comme l'indique son nom ("vers l'avant" en français), le principe est de partir du début de la séquence vers la fin. Pour une position donnée, disons la deuxième pour commencer, la probabilité conditionnelle aux observations antérieures (avec la présente) d'arriver à un des états possibles à partir de tous les états de la position précédente va être calculée (ici, la première position). Ce processus est répété de la même façon de position en position jusqu'à la dernière de la séquence observée. Étant donné que pour chaque état d'arrivée, des probabilités vont être calculées pour venir de chaque état de départ, le nombre de calculs sera égal au carré du nombre d'états (Q) et ce, pour chaque pas d'une base à une autre (figure 3). L'ordre du calcul est donc réduit à Q^2K opérations[34].

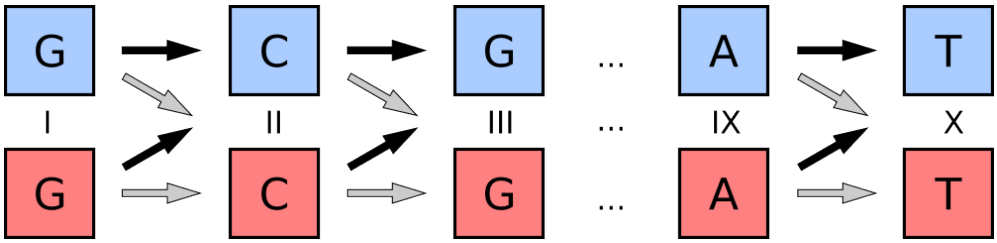


FIG. 3 : Illustration du principe d'une procédure *forward*. Pour chaque état de la deuxième position, la probabilité de venir de chacun des états de la première position est calculée. Ici la probabilité d'arriver dans le premier état (GC, bleu) à la position II est calculée en sommant les probabilités des flèches noires et la probabilité d'arriver dans le deuxième état (AT, rouge) est obtenue en sommant les probabilités des flèches grises. L'opération est répétée pour arriver à la position III à partir de la position II et ainsi de suite jusqu'à la fin de la séquence.

Ainsi, la variable *forward* α_{kq}, c'est-à-dire la probabilité obtenue à une position donnée k pour l'état q, tient compte de toutes les positions précédentes. À la dernière, pour un état donné (α_{Kq}), on obtient ainsi la probabilité d'être dans cet état étant donné toute la séquence. Pour obtenir la vraisemblance du modèle ($P(O|\pi, A, B)$), c'est-à-dire la probabilité d'avoir toute la séquence (observée) de caractères (O) selon le modèle, il suffit de sommer les probabilités obtenues à chaque état de la dernière position ($\sum_{q=1}^{Q} \alpha_{Kq}$). Les trois étapes (initiation, induction, terminaison) sont[34] :

$$\alpha_{1q} = \pi_q \times b_{1q}$$

$$\alpha_{kq_2} = \sum_{q_1=1}^{Q} \alpha_{k-1q_1} \times a_{q_1q_2} \times b_{kq_2}$$

$$P(O|\pi, A, B) = \sum_{q=1}^{Q} \alpha_{Kq}$$

Par exemple, à la figure 3 (avec les probabilités de la figure 2), pour arriver au premier état caché (riche en GC) de la position II, on doit calculer la probabilité d'y arriver à partir de la position I à l'état GC ($\alpha_{I(GC)} = \pi_{(GC)} \times B_{G(GC)} = 0,5 \times 0,4 = 0,2$) et de la position I à l'état AT ($\alpha_{I(AT)} = 0,05$). On a donc respectivement :

$$(1) \quad P(O_1, O_2 | q_1 = GC, q_2 = GC, \pi, A, B) = \alpha_{I(GC)} \times a_{GC \to GC} \times b_{C(GC)}$$
$$= 0,2 \times 0,8 \times 0,4 = 0,064$$
$$(2) \quad P(O_1, O_2 | q_1 = AT, q_2 = GC, \pi, A, B) = \alpha_{I(AT)} \times a_{AT \to GC} \times b_{C(GC)}$$
$$= 0,05 \times 0,2 \times 0,4 = 0,004$$

On obtient ainsi une probabilité d'être GC à la position II, en tenant compte des probabilités de la position I, de $\alpha_{II(GC)} = (1) + (2) = 0,064 + 0,004 = 0,068$ (addition des flèches noires de la figure 3). La même opération est réalisée pour arriver à l'état AT de la position II (cette fois en suivant et en additionnant les flèches grises).

1.4.3 Estimation de la probabilité d'un état à une position via une procédure *forward/backward*

1.4.3.1 Procédure *backward*

Lorsque la procédure *forward* est terminée, on obtient la vraisemblance. Celle-ci illustre à quel point notre modèle est vraisemblable, correspond à nos données. Si elle est bonne, on aimerait alors pouvoir connaître les probabilités de chaque état caché à chaque position. Avec la procédure *forward*, on n'a que les probabilités d'être dans un état *en tenant compte de toute la séquence précédant la base d'intérêt*. Pour obtenir une probabilité d'être dans un état ou un autre pour une base précise, une autre procédure doit être utilisée, la procédure *backward*. Elle fonctionne de la même manière que la procédure *forward* sauf que, comme son nom l'indique ("en arrière"), le principe est de partir de la dernière position et de revenir vers la première (figure 4).

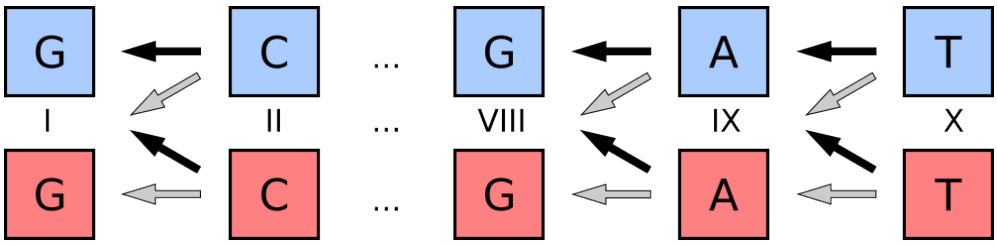


FIG. 4 : Illustration du principe d'une procédure *backward*. Pour chaque état de l'avant-dernière position, la probabilité d'aller vers chacun des états de la dernière position est calculée. Ici la probabilité de venir du premier état (GC, bleu) à la position IX est calculée en sommant les probabilités des flèches noires et la probabilité de venir du deuxième état (AT, rouge) est obtenue en sommant les probabilités des flèches grises. L'opération est répétée pour venir de la position VIII vers la position IX et ainsi de suite jusqu'au début de la séquence.

1.4.3.2 Combinaison des procédures *forward* et *backward*

Grâce à la combinaison *forward/backward*, il est donc possible d'obtenir la probabilité d'être dans un état précis à une position donnée. En effet, si la probabilité d'arriver à un caractère donné à partir du début est connue ainsi que celle d'arriver à ce même caractère depuis la fin, il suffit

de multiplier ces deux probabilités et de diviser par la probabilité de la séquence d'observations (c'est-à-dire la vraisemblance).

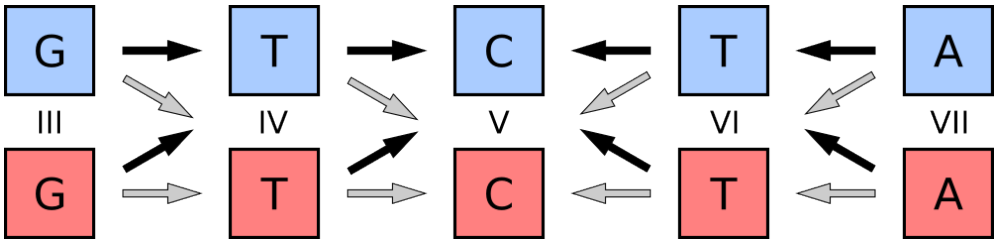


FIG. 5 : Illustration du principe d'une combinaison *forward/backward*.

En prenant pour exemple la figure 5, la probabilité d'être à la base V avec la procédure *forward* tient compte de toutes les probabilités des positions précédentes ($\alpha_{V(GC)}$). De même, avec la procédure *backward*, on obtient $\beta_{V(GC)}$, β_{kq} étant la variable *backward*, c'est-à-dire la probabilité d'être à l'état q à la position k en sachant toutes les observations déjà parcourues (de K jusque k). En sachant qu'on avait obtenu la probabilité de la séquence, $P(O|\pi, A, B)$, on peut donc calculer $P_{V(GC)} = \frac{\alpha_{V(GC)} \times \beta_{V(GC)}}{P(O|\pi,A,B)}$.

1.4.4 Estimation de l'appartenance à un état via l'algorithme de Viterbi

Jusqu'à présent, des probabilités d'être dans un état ou un autre ont été calculées via la procédure *forward/backward*. Cependant, on peut vouloir obtenir la séquence d'états cachés la plus probable et ne pas rester avec des probabilités à chaque position. Dans ce cas-là, l'algorithme de Viterbi peut être utilisé.

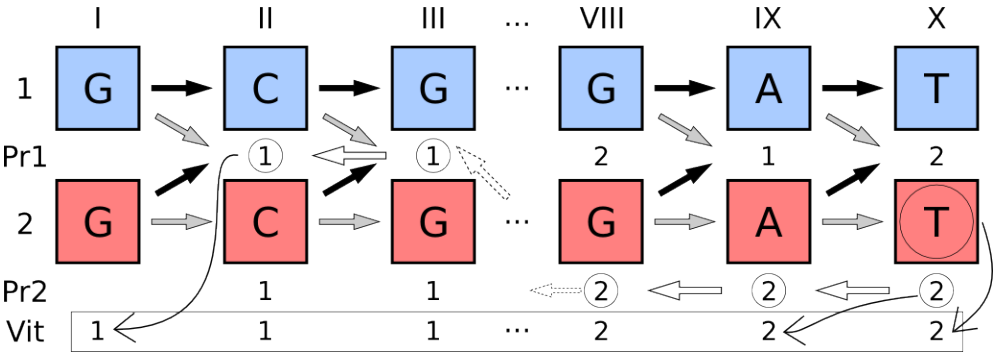


FIG. 6 : Illustration de l'algorithme de Viterbi. Comme dans le *forward*, la probabilité d'arriver dans l'état 1 pour la position II est calculée à partir de l'état 1 et de l'état 2 de la position I (flèches noires). Ici, les deux probabilités sont comparées. La plus grande est enregistrée comme état de provenance Pr1 (le plus probable est de venir de l'état 1 de la position I). À la dernière position (X), les probabilités d'être dans l'état 1 ou 2 sont comparées. La plus grande est considérée comme l'état de cette position par Viterbi (état 2). On regarde ensuite quel est l'état de provenance le plus probable (état 2 à la position IX) et ainsi de suite jusqu'à la première position (flèches blanches). La séquence d'états la plus probable pour chaque position est obtenue (Vit).

Le principe de l'algorithme de **Viterbi** est de garder en mémoire, pour chaque position et chaque

état, quel est l'état de provenance le plus probable, c'est-à-dire celui maximisant la probabilité[33]. En pratique, la séquence est parcourue du début vers la fin (comme la procédure *forward*) en enregistrant, pour chaque position et pour chaque état, l'état de la position précédente pour lequel la probabilité est la plus grande. Une fois toute la séquence parcourue, on observe à la dernière position quel est l'état le plus probable. Pour cet état, on regarde quel est l'état de provenance enregistré comme étant le plus probable pour y arriver de la position précédente. On se rend donc dans cet état à l'avant-dernière position et on regarde à nouveau quel est l'état de provenance le plus probable et on remonte la séquence de cette façon (figure 6). Le résultat final sera la séquence d'état les plus probables par lesquels on sera revenu à la première position.

1.4.5 Estimation des paramètres

Lors des exemples précédents expliquant le principe du HMM, les différents paramètres (π, A et B) étaient connus à l'avance. Cependant, on ne peut pas être certain, dans un cas réel, de connaître, par exemple, les probabilités de transition (A) d'un état caché à l'autre et, si on a une idée de l'ordre de grandeur qu'elles peuvent prendre, on est rarement certain des valeurs optimales. La vraisemblance va être utilisée pour déterminer de meilleures valeurs pour ces paramètres. En effet, comme écrit plus haut, elle permet de savoir si notre modèle (c'est-à-dire l'ensemble des paramètres utilisés) est bon ou pas. Elle va ainsi nous permettre d'optimiser les paramètres par le biais de méthodes dites d'optimisation. C'est le cas de la méthode d'**Espérance Maximisation** (EM, ou *Expectation Maximization*)[33].

1.4.5.1 Algorithme d'Espérance Maximisation

L'EM est un algorithme itératif qui ajuste les paramètres (qui peuvent donc être *a priori* aléatoires ou estimés) de manière à mieux expliquer les données. Il va chercher à obtenir la maximisation de la vraisemblance (cette méthode considère donc que plus la vraisemblance est grande, meilleur est le modèle). À chaque itération, l'EM va calculer les probabilités à toutes les positions selon les paramètres puis il va utiliser cette séquence pour réestimer les paramètres. Donc dans un premier temps, il va utiliser les paramètres à estimer pour calculer, à l'aide d'un *forward*/*backward*, les probabilités d'appartenance à chaque classe pour chaque position. Ensuite, il utilise ces probabilités calculées pour compter le nombre de transitions d'un état à l'autre par exemple (si l'on cherche A) pour estimer des nouvelles probabilités de transition. Il va répéter ces deux phases en boucle jusqu'à ce que son critère d'arrêt soit atteint. Celui-ci peut être soit un nombre d'itérations fixé à l'avance soit une différence de vraisemblance (ou entre paramètres) donnée entre deux itérations successives (convergence).

Ainsi, à chaque itération, la vraisemblance sera meilleure étant donné que l'EM optimise les paramètres à chaque fois.[33] La première étape d'une itération est l'étape E (*expectation*) tandis que la deuxième est l'étape M (*maximization*)[35].

1.4.6 Programmes utilisant un HMM pour étudier la consanguinité

Dans le cas de la consanguinité, un chromosome étudié aura des caratéristiques sous-jacentes aux caractères "homozygotes" et "hétérozygotes". En effet, il peut être divisé en segments IBD et non IBD, c'est-à-dire en parties héritées d'un ancêtre commun aux deux parents et en parties héritées d'autres ancêtres.

Cette utilisation des HMM pour identifier des segments IBD n'est pas encore très répandue. Le premier programme à l'utiliser a été `FEstim`[31] qui estime la consanguinité individuelle sur base de données génomiques et qui n'a pas besoin de connaître à l'avance les liens de parenté, donc la généalogie. Il modélise les marqueurs génotypés via un HMM qui dépend de F et du taux de changement IBD/nIBD par cM (probabilités de transition A). Ces deux paramètres sont estimés en maximisant la vraisemblance avec GEMINI[36]. Il fonctionne avec une classe IBD et une non IBD.

Plus récemment, des modèles similaires dédiés aux données de séquence ont été développés, notamment `BCFtools/ROH`[37] et `ngsF-HMM`[38] qui, contrairement à `FEstim`, utilisent un algorithme L-BFGS-B pour estimer les paramètres. De plus, F est obtenu à partir d'un algorithme de Viterbi.

D'un point de vue biologique, se restreindre à une seule classe de consanguinité ne semble cependant pas assez réaliste. En général, plusieurs évènements de consanguinité peuvent avoir eu lieu par le passé (c'est le cas des bisons ayant subi plusieurs *bottleneck* importants). Pour mieux comprendre l'histoire d'une population ainsi que celle d'un individu en particulier, plusieurs classes de consanguinité peuvent être envisagées, certaines remontant plus loin dans le temps que d'autres.

Un tel modèle a été proposé récemment. `ZooRoH`[32] est en effet un programme basé sur un HMM pouvant fonctionner avec plusieurs classes autozygotes (IBD) remontant à différents moments dans le passé. Il permet d'obtenir la consanguinité globale d'un individu ainsi que la probabilité de consanguinité associée à chaque classe. De plus, la probabilité de consanguinité locale peut également être obtenue. Pour l'estimation des paramètres, ce programme utilise un algorithme EM qui est peut-être moins efficace que GEMINI et L-BFGS-B. Il n'y a malheureusement pas de critère de convergence, le nombre d'itérations doit être choisi à l'avance. De plus, contrairement à `ngsF` et `BCFtools/ROH`, la définition de segments IBD n'est pas intuitive mais naturelle. Enfin, l'EM peut parfois fusionner les classes, ce qui nous en donne finalement moins qu'attendu, et les mélanger, c'est-à-dire que des classes plus récentes peuvent être intercallées entre des classes plus anciennes. Ce programme nécessite donc quelques améliorations.

2 Objectifs

Comme le suggèrent les observations rapportées dans l'introduction, une amélioration de `ZooRoH` est envisageable dans le contexte de l'étude de la consanguinité des bisons. De plus, un nombre croissant de scientifiques utilisent R pour leurs analyses de données. Dans le cadre de ce travail, deux objectifs principaux sont visés :

1. Porter `ZooRoH` sous l'environnement R et essayer d'améliorer l'estimation des paramètres ;
2. Caractériser la consanguinité chez les bisons d'Europe.

Différentes stratégies seront développées pour atteindre ces objectifs. Le portage sous R de `ZooRoH`, écrit en fortran, sera d'abord entrepris en permettant l'utilisation des fonctions d'origine du HMM dans cet environnement. Ensuite, quatre approches seront envisagées pour améliorer son efficacité.

1) Changer la méthode d'estimation des paramètres afin d'améliorer la vitesse de convergence.
2) Reparamétriser les paramètres afin de garantir leurs propriétés et de les ordonner.
3) Incorporer un algorithme de Viterbi et comparer ses propriétés avec celles du *forward/backward*.
4) Paralléliser les calculs des différents individus.

Dans le but de juger de l'efficacité de ces modifications, plusieurs modèles (ayant notamment un nombre de classes différent) pourront être utilisés pour comparer les vraisemblances obtenues, les paramètres estimés, ainsi que la consanguinité globale et locale estimées.

En ce qui concerne la caractérisation de la consanguinité chez les bisons d'Europe, une analyse de la structure de la population sera d'abord réalisée avec les programmes `PLINK`, `GCTA` et `ADMIXTURE` pour détecter si elle est homogène. Ensuite, une analyse de la consanguinité locale et globale sera réalisée avec `ZooRoH` porté sous R lors de l'objectif précédent. Celui-ci permettra d'estimer la proportion consanguine de leur génome, ainsi que la mesure dans laquelle cette consanguinité résulte d'événements passés plus ou moins récents. Finalement, ce modèle pourra aider à savoir si cette consanguinité se stabilise depuis leur réintroduction dans la nature.

3 Matériel et méthodes

3.1 Modélisation de la consanguinité

La modélisation de la consanguinité proposée dans le programme `ZooRoH` (développé par Tom Druet et Mathieu Gautier) considère le génome comme une mosaïque de segments IBD et non IBD. Ces états des segments sont les états cachés du HMM. Différents modèles, variant notamment par le nombre d'états cachés, ont été appliqués lors de ce travail.

3.1.1 Modèle à une seule classe consanguine

Le modèle le plus simple a deux états cachés : une classe autozygote (IBD) et une classe non IBD. Dans ce cas, le génome est considéré comme une succession de segments IBD et non IBD hérités de la même génération ancestrale. Les probabilités de transition (A) sont :

$$
\begin{aligned}
P(q_1 \rightarrow q_1) &= e^{-rG} + (1 - e^{-rG})M \\
P(q_2 \rightarrow q_1) &= (1 - e^{-rG})M \\
P(q_2 \rightarrow q_2) &= e^{-rG} + (1 - e^{-rG})(1 - M) \\
P(q_1 \rightarrow q_2) &= (1 - e^{-rG})(1 - M)
\end{aligned}
$$

avec q_1 indiquant l'état IBD, q_2 l'état non IBD, r le taux de recombinaison entre les deux marqueurs concernés, G le taux de la distribution exponentielle associée à la longueur des segments IBD et non IBD (équivalent plus ou moins à la taille de la boucle de consanguinité en nombre de générations lorsque r est défini en Morgan) et M et $(1 - M)$ les proportions de mélange associées aux deux états. Le taux de recombinaison dépend de la distance entre les marqueurs :

$$
r = \frac{Pos_k - Pos_{k-1}}{Morgan}
$$

avec Pos les positions (en pb) des marqueurs sur le chromosome et $Morgan$ la distance (en pb) équivalente à 1 M soit 100000000.

Étant donné que les segments sont considérés comme hérités d'une seule génération passée, le même G est utilisé dans les deux classes. De plus, les probabilités initiales (π) du HMM sont ici de M et $(1 - M)$.

3.1.2 Probabilités d'émission

Trois génotypes peuvent être observés pour des marqueurs bi-alléliques : homozygote pour l'allèle A, homozygote pour l'allèle a ou hétérozygote. Les probabilités d'émission (B) de ces caractères pour l'état non IBD sont :

$$
\begin{aligned}
P(AA|nIBD) &= p^2 \\
P(Aa|nIBD) &= 2p(1-p) \\
P(aa|nIBD) &= (1-p)^2
\end{aligned}
$$

Les probabilités d'émission pour l'état IBD sont :

$$
\begin{aligned}
P(AA|IBD) &= (1-\epsilon) \times p \\
P(Aa|IBD) &= \epsilon \\
P(aa|IBD) &= (1-\epsilon) \times (1-p)
\end{aligned}
$$

avec ϵ représentant la probabilité d'erreur pouvant résulter du génotypage ou d'une mutation récente et p et $(1-p)$ les fréquences alléliques. Les probabilités d'émission pour la classe non IBD correspondent aux proportions de Hardy-Weinberg. Les probabilités d'émission pour les classes non IBD tiennent compte d'erreurs possibles en définissant une probabilité pour le caractère hétérozygote non nulle mais faible. Dans le cas de segments IBD, les fréquences génotypiques homozygotes seront équivalentes aux fréquences alléliques en tenant cependant compte du taux d'erreur ϵ.

3.1.3 Modèle à multiples classes consanguines

Un modèle avec une seule classe consanguine revient à supposer que tous les segments IBD sont hérités d'un seul ancêtre commun ou de plusieurs ancêtres communs provenant de la même génération ancestrale. Cependant, la consanguinité au sein d'un individu résulte d'événements passés multiples. C'est pourquoi plusieurs classes de consanguinité ont été envisagées (augmentant dès lors le nombre d'états cachés du HMM). Les probabilités de transition (A) ont ainsi été légèrement adaptées :

$$
\begin{aligned}
P(q_i \to q_i) &= e^{-rG_i} + (1 - e^{-rG_i})M_i \\
P(q_i \to q_j) &= (1 - e^{-rG_i})M_j
\end{aligned}
$$

avec q les états (IBD ou non), r le taux de recombinaison entre les deux marqueurs concernés, G_i le taux de la distribution exponentielle associée à chaque classe et M_i les proportions de mélange des différents états ainsi que les probabilités initiales. Le taux de recombinaison dépend à nouveau de la distance entre les marqueurs.

3.1.4 Paramètres du modèle à estimer

En supposant que les fréquences alléliques (p et $(1-p)$) soient connues pour chaque marqueur, ainsi que le taux d'erreur ϵ et la carte génétique (et donc le taux de recombinaison r), deux types de paramètres doivent être estimés : les taux G_i des distributions exponentielles associées à chaque classe et les proportions de mélange M_i des classes IBD et non IBD.

Pour le modèle à une seule classe IBD, deux paramètres seront donc à estimer : G et M. Par contre, pour les modèles à plusieurs classes consanguines, le nombre de paramètres à estimer sera de $2Q-1$, Q étant le nombre d'états cachés. En effet, pour chaque classe (consanguine et non consanguine), il faudra estimer G_i et M_i mais, puisque $\sum M_i = 1$, on épargne l'estimation d'un paramètre. Ce type de modèle sera nommé par la suite Qe, Q étant le nombre d'états considérés et e l'abréviation de "estimé".

Il est également possible de travailler avec des taux prédéfinis G_i. Ce type de modèle permet à la fois de réduire le coût computationnel et à la fois de pouvoir comparer plus facilement deux individus. Ainsi, seules les proportions de mélange devront être estimées, réduisant par conséquent le nombre de paramètres à estimer à $Q-1$. Ces modèles seront nommés Qp, pour "prédéfinis".

3.1.5 Estimation de la consanguinité

3.1.5.1 Consanguinité locale

Le HMM permet d'estimer, grâce aux algorithmes *forward/backward*, les probabilités de la contribution de chaque classe de consanguinité à la consanguinité locale réalisée à chaque marqueur. Pour obtenir la probabilité de contribution d'un locus à la consanguinité, une somme sur les probabilités des classes IBD est réalisée.

$$\widehat{F}(k) = \sum_{q=1}^{Q_{IBD}} F_q(k)$$

avec $\widehat{F}(k)$ la contribution locale estimée au locus k, Q_{IBD} le nombre de classes IBD et $F_q(k)$ la contribution locale estimée pour l'état q au locus k.

3.1.5.2 Consanguinité globale

Pour obtenir la contribution globale d'une classe sur l'ensemble du génome pour un individu pour chaque classe, une moyenne est calculée sur toutes les positions pour cette classe.

$$\widehat{F}_{q_i} = \frac{\sum_{k=1}^{K} F_{q_i}(k)}{K}$$

avec $\widehat{F}_{q_i}$ la contribution globale de l'état q_i, K le nombre total de loci, $F_{q_i}(k)$ la contribution locale estimée pour l'état q_i au locus k.

Ensuite, la consanguinité globale réalisée d'un individu est obtenue en sommant ces contributions par classe (voir par exemple les lignes 151-154 du script `Optim_4_est_contr_repar_ord_wz_paral.R` en annexe 7.4.2 page IV- 5).

$$\widehat{F} = \sum_{q=1}^{Q_{IBD}} \widehat{F}_{q_i}$$

avec $\widehat{F}$ la consanguinité globale réalisée estimée d'un individu.

3.1.5.3 Identification de segments IBD

Le *forward/backward* permet d'estimer les probabilités locales et globales. L'algorithme de Viterbi permet de classer chaque marqueur dans une classe et donc de les définir comme étant IBD ou non IBD. Ceci permet d'identifier des segments IBD, ainsi que leurs longueurs, contrairement au *forward/backward* (pour lequel il faudrait définir un seuil de probabilité).

Les segments obtenus de cette façon ont été enregistrés sous la forme de couples de positions début-fin (voir script `Vit_em_fb_gamma_2e_xy.R` en Annexe 7.4.4 page IV- 15, lignes 274-377).

3.2 Méthode d'optimisation

Dans ce travail, lors du portage de ces modèles sous R, divers aspects ont été pris en compte, notamment la façon d'estimer les paramètres et la consanguinité.

3.2.1 Portage des librairies

Les fonctions d'origine de `ZooRoH`, écrites en fortran (`.f90`), ont été compilées via la commande `R CMD SHLIB -o zoolik.so zoolik.f90`. Les librairies partagées résultantes (`.so`) sont appelées dans les versions R via `dyn.load("zoolik.so")`. Ces fonctions reprennent les algorithmes *forward*, *forward/backward*, Baum-Welch (EM) et Viterbi.

3.2.2 Estimation des paramètres

La méthode d'estimation originale de `ZooRoH` était un EM (algorithme de Baum-Welch). Dans le cadre de l'optimisation et du portage du programme sous R, la méthode utilisée est une fonction du package "stats" : `optim` (abrévié de *General-purpose optimization*). Cette fonction propose plusieurs méthodes basées sur des algorithmes de type Nelder-Mead, quasi-Newton et de gradients-conjugués[39]. `Optim` a été appelé avec la méthode "L-BFGS-B" pour pouvoir utiliser des bornes dans les paramètres à estimer (comme pour l'EM). Il s'agit d'une méthode de quasi-Newton

(plus précisément *Limited-memory Broyden-Fletcher-Goldfarb-Shanno Bound-constraints*) qui a été originellement écrite en Fortran[39] puis ré-implémentée pour le package en Python.

Un algorithme de quasi-Newton permet, comme l'EM, d'estimer les paramètres de manière à optimiser la vraisemblance. La méthode travaillant avec des dérivées, le chemin vers les nouveaux paramètres se déroulera par "bonds" plutôt que "petit pas par petit pas" comme l'EM. La convergence de cette méthode est quadratique[40].

Pour fonctionner, `optim` a seulement besoin d'un objet reprenant les paramètres à estimer et de la fonction renvoyant la vraisemblance à optimiser. Cette fonction (appelée `getlik` dans les modèles en annexes) se base sur une procédure *forward* comme expliqué dans l'introduction : $P(O|\pi, A, B) = \sum_{q=1}^{Q} \alpha_{K_q}$. En sortie, plusieurs éléments sont donnés par `optim` dont, évidemment, les paramètres optimisés et la vraisemblance finale obtenue avec ces paramètres.

3.2.3 Reparamétrisation

Dans le cas d'`optim`, le modèle a été reparamétrisé de telle sorte que les paramètres aient automatiquement les propriétés requises. Les proportions de mélange M_i doivent être comprises entre 0 et 1 et sommer à 1 et les taux G_i doivent avoir des valeurs positives (nombre de générations). La reparamétrisation a également permis d'ordonner les taux des distributions exponentielles. Cette ordination des taux (donc la première classe aura le taux le plus petit et la dernière classe IBD aura le taux le plus grand) permet de faciliter l'interprétation des résultats : on remonte de plus en plus dans le temps en allant de la première à la dernière classe IBD. Cette reparamétrisation est donc distincte pour les deux classes de paramètre à estimer. Ces nouveaux paramètres ont été inspirés par Zucchini[41].

$$(1) \quad \eta_i = \begin{cases} log(G_i - G_{i-1}) & (i = 2, ..., Q-1) \\ log(G_i) & (i = 1 \ ou \ i = Q) \end{cases}$$

$$(2) \quad \tau_i = log\left(\frac{M_i}{M_Q}\right) \qquad (i = 1, ..., Q-1)$$

avec G_i représentant le taux, M_i représentant les proportions de mélange de la classe i et Q le nombre de classes, la dernière étant la classe non IBD.

Pour retrouver les paramètres d'origine, la transformation inverse est réalisée comme suit :

$$(1) \quad G_i = \begin{cases} G_{i-1} + e^{\eta_i} & (i = 2, ..., Q-1) \\ e^{\eta_i} & (i = 1 \ ou \ i = Q) \end{cases}$$

$$(2) \quad M_i = \begin{cases} \dfrac{e^{\tau_i}}{1 + \sum_{j=2}^{Q} e^{\tau_j}} & (i = 1, ..., Q-1) \\ 1 - \sum_{j=1}^{Q-1} M_j & (i = Q) \end{cases}$$

Ces paramètres d'origine sont alors donnés aux fonctions du HMM écrites en fortran.

Une reparamétrisation alternative de G a également été testée. Celle-ci n'ordonne pas les taux et se réalise simplement en prenant le logarithme : $\eta_i = log(G_i) \quad (i = 1, ..., Q)$ et, pour la retransformation des données, en prenant l'exponentielle : $G_i = e^{\eta_i} \quad (i = 1, ..., Q)$.

3.2.4 Contraintes appliquées à l'estimation des paramètres

L'EM de `ZooRoH` utilise des limites contraignant les taux à rester dans un certain intervalle de valeurs. Les mêmes contraintes ont été appliquées en entrée de la fonction lors de l'estimation des paramètres avec `optim`. Selon le jeu de données (voir 3.3.1), des bornes différentes ont été définies. Celles-ci ont été choisies selon la densité de marqueurs. En effet, si le taux était trop élevé, les segments IBD identifiés deviendraient plus courts que la distance moyenne entre deux SNPs. Ils contiendraient donc 1 SNP ou moins. La distance moyenne entre deux marqueurs (en cM) est la taille du génome divisée par le nombre de marqueurs.

La taille attendue d'un segment IBD de la classe avec un taux G_i est $1/G_i$ (espérance de l'exponentielle). Ainsi, on choisi le dernier G_i, c'est-à-dire le plus grand, correspondant à peu près à l'écart entre deux SNPs.

3.2.5 Parallélisation des modèles

Afin d'améliorer les performances des modèles, une parallélisation a été réalisée en utilisant le package `doMC` de R. Une fois chargé, il suffit de spécifier le nombre de cœurs à utiliser. Lorsqu'une boucle `foreach` est utilisée, ici pour les différents individus du jeu de données, les tâches du corps de la boucle s'exécutent en parallèle sur le nombre de cœurs défini (voir les scripts des modèles en annexe 7.4.1, 7.4.2 et 7.4.3 pages IV- 1, IV- 5 et IV- 10).

3.3 Comparaison des méthodes développées pour caractériser la consanguinité avec des HMMs

Les méthodes d'estimation, de reparamétrisation et de contrainte des paramètres ont été évaluées sur des données simulées dont les paramètres sont connus (voir 3.3.1.1) et sur des données réelles. À cet effet, les vraisemblances obtenues ainsi que les paramètres et la consanguinité estimés ont été comparés.

3.3.1 Données

Différents jeux de données ont été utilisés lors de ce travail : des jeux de données simulées ainsi que plusieurs jeux de données réelles. Dans cette partie, le premier de ceux-ci reprenait un pedigree de 36 chevaux et le deuxième des taureaux Blanc-Bleu Belge génotypés à haute densité (> 700000 SNPs).

3.3.1.1 Données simulées

Pour chaque jeu de paramètres, 500 individus ayant 25 chromosomes de 100 cM avec 25000, 250000 et 2500000 SNPs (densités de 10, 100 et 1000 SNPs/Mb) ont été simulés. Des segments IBD et non IBD, ayant une longueur exponentiellement distribuée selon les paramètres simulés, ont été distribués aléatoirement selon une probabilité M et $1-M$ (correspondant aux proportions de mélange simulées)[32].

3.3.1.2 Chevaux

36 chevaux ont été génotypés par Neogen GeneSeek en utilisant la puce Illumina Equine SNP70. Vingt-deux de ces animaux étaient BR1 (*brindle 1*, c'est-à-dire porteur d'un phénotype particulier au niveau du pelage)[42], les quatorze autres étant des contrôles. Tous les chevaux BR1 descendaient d'une seule génitrice (UKH22). Leur arbre généalogique se trouve en annexe (figure S.1). Dans ce pedigree, plusieurs individus sont consanguins, notamment UKH08 et UKH21 (accouplement père-fille et mère-fils) ou encore UKH36 (même arrière-grand-père côté maternel et paternel). Ils représentent donc un bel exemple pour étudier la consanguinité. Les données utilisées comprennent 39400 SNPs pour une taille de génome de 2437 Mb[43].

3.3.1.3 Blanc-Bleu Belge

634 taureaux Blanc-Bleu Belge (BBB) ont été génotypés avec la puce Illumina BovineHD BeadChip. De ces données, des marqueurs ont été retirés (en deux temps) pour arriver à une densité à peu près similaire à une puce 50K et à une puce LD, respectivement. Les SNPs conservés ont été sélectionnés pour leur présence sur la puce Illumina BovineSNP50 BeadChip et sur la puce Illumina BovineLD BeadChip. Trois jeux de données ont donc été obtenus à partir de ces animaux génotypés, correspondant à trois densités de marqueurs différentes, nommées dans ce travail HD, 50K et LD, ayant chacune respectivement 613005, 34008 et 6885 SNPs pour une taille de génome de 2650 Mb[44].

3.3.2 Modèles appliqués

Quatre modèles ont été appliqués (2e, 4p, 4e et 10p) afin d'étudier les propriétés des différentes méthodes sous différentes conditions.

Pour le modèle 2e, la valeur de départ de l'estimation de G a été fixée à *50* et celles de M à *0,05* et *0,95*, respectivement pour la classe IBD et non IBD (script `Optim_2_est_contr_repar_wz_paral.R` en annexe 7.4.1 page IV- 1).

Les modèles 4p et 4e ont les mêmes valeurs de départ pour M : *0,02* pour les trois classes IBD et *0,94* pour la classe non IBD. Les valeurs prédéfinies de G pour le modèle 4p servent de points de départ pour l'estimation dans le modèle 4e. Elles sont de *8*, *64*, *256* et *256*, respectivement pour les trois classes IBD et la classe non IBD (scprits `Optim_4_predef_repar_paral.R` et `Optim_4_est_contr_repar_ord_wz_paral.R` en annexe 7.4.3 et 7.4.2 pages IV- 10 et IV- 5).

Le modèle 10p a comme valeurs prédéfinies de G $2^{1,2,3,\ldots,9}$ pour les neuf classes consanguines et $512\ (= 2^9)$ pour la classe non IBD. Ces valeurs permettent de capturer des segments IBD associés à un large nombre de générations passées tout en limitant le nombre de classes. Le point de départ de l'estimation des M est de *0,01* pour les neuf premières classes et de *0,91* pour la dernière (script `Optim_10_predef_repar_paral.R` en annexe 7.4.3 page IV- 14).

3.3.3 Précision de l'estimation des paramètres

Pour évaluer la précision des estimations obtenues par les différentes méthodes, les jeux de données simulées ont été utilisés. Ainsi, sur base de ces valeurs simulées, l'erreur absolue moyenne (*Mean Absolute Error*, MAE) a pu être calculée pour chaque paramètre estimé (taux G_i des distributions exponentielles, proportions de mélange M_i, ainsi que la consanguinité globale F) :

$$MAE(\lambda) = \frac{\sum_{n=1}^{N} \left| \widehat{\lambda}_n - \lambda_n \right|}{N}$$

avec λ le paramètre évalué, N le nombre d'individus simulés, $\widehat{\lambda}_n$ la valeur du paramètre estimé pour l'individu n, λ_n la valeur simulée du paramètre pour l'individu n.

Les MAEs ont été calculées pour la version portée sous R, pour `ZooRoH` et pour `FEstim`.

3.3.4 Comparaison de l'estimation des paramètres sur données réelles

Deux jeux de données ont été utilisés : le pedigree équin et les 634 taureaux BBB (avec le génotypage 50K). Cette approche a permis une comparaison des résultats obtenus sur des populations distinctes en terme de consanguinité (et de taille). Ces deux jeux de données ont cependant une densité de marqueurs similaire.

Différentes versions de la méthode d'estimation des paramètres ont été comparées dans cette partie. Tout d'abord, l'estimation de l'impact de la reparamétrisation (ordonnée) sur les résultats, avec et sans contraintes, a été étudiée en comparant les résultats des différentes versions. Ensuite,

une comparaison entre `ZooRoH` et la version sous R avec la reparamétrisation ordonnée contrainte a été réalisée.

Pour comparer ces différents aspects de la version sous R, plusieurs critères ont été choisis. Tout d'abord, la vraisemblance permet d'estimer si les résultats obtenus d'une certaine façon sont meilleurs, moins bons ou équivalents à ceux obtenus d'une autre manière. Pour évaluer le temps d'exécution des différentes versions, le nombre d'itérations a été utilisé. Ensuite, les paramètres estimés ont également été comparés : les taux pour les modèles 2e et 4e (dans lesquels ils sont estimés) et les proportions de mélange pour les modèles 4p et 10p. Enfin, l'impact sur la consanguinité globale estimée, ainsi que sur la contribution de chaque classe à celle-ci, a été examiné.

Propriétés des modèles

La version sous R avec la reparamétrisation ordonnée et les contraintes a alors été utilisée pour étudier l'impact de la densité en marqueurs du jeu de données sur l'estimation de la consanguinité. Pour ce faire, les trois jeux de données BBB ont été utilisés, avec les mêmes critères de comparaison des résultats.

Enfin, les résultats des quatre modèles ont été comparés avec cette même version sous R. Ici, seul le jeu de données BBB 50K a été utilisé.

3.3.5 Estimation de la consanguinité locale : *forward/backward* vs Viterbi

Pour estimer la précision de l'algorithme de Viterbi par rapport à celle obtenue via les algorithmes *forward/backward*, les données simulées ont à nouveau été utilisées et les MAEs de la consanguinité locale calculées.

À titre illustratif, les segments IBD identifiés avec l'algorithme de Viterbi sur les données BBB 50K ont été comparés à la probabilité de consanguinité locale.

3.4 Étude de la consanguinité chez les bisons d'Europe

Après la validation des procédures d'estimation des paramètres et d'identification des segments IBD, le modèle a été appliqué pour caractériser la consanguinité chez les bisons d'Europe. Une étude de la structure de la population a été préalablement réalisée via les programmes `PLINK`, `GCTA` et `ADMIXTURE` puis la consanguinité a été estimée via notre modèle. Les résultats ont été comparés en appliquant différents filtres pour sélectionner les marqueurs. Enfin, un aperçu de l'évolution de la consanguinité globale a été réalisé afin de déterminer si elle continue à s'accumuler.

3.4.1 Données

159 bisons ont été génotypés sur des puces 770K bovines. Parmi ces animaux, 108 viennent de la forêt Bialowieza du côté polonais et trois du côté bielorusse. Les autres viennent de réserves et

de zoos, majoritairement polonais. Les échantillons ont été prélevés pour 43 d'entre eux dans les années 90s. Toutes ces données ont préalablement été filtrées pour ne conserver que les SNPs qui se trouvaient sur des autosomes, qui avaient un *call rate* plus grand que 0,95 et qui avaient une *p-value* supérieure à *0,001* lors du test pour l'équilibre de Hardy-Weinberg (HWE). Les données résultantes contiennent 21786 SNPs.

3.4.2 Analyse de la structure de la population

Pour détecter une possible structure dans la population de bisons, des analyses ont été réalisées via les programmes `PLINK`, `GCTA` et `ADMIXTURE`.

PLINK

`PLINK` est un programme gratuit, en open-source. Il permet de réaliser une large gamme d'analyses génomiques[30,45]. Étant donné que ce programme a été utilisé pour l'étude des bisons, l'option `--cow` a été appliquée lors de chaque commande pour ajuster le nombre de chromosomes (par défaut, le nombre de chromosomes est celui de l'Homme).

Dans le cadre de cette analyse de la structure de la population, il a été utilisé pour réaliser une analyse MDS (*MultiDimensional Scaling*) de la population de bisons. Ce type d'analyse permet de visualiser des données dans un espace de dimensions inférieur à celui des données originales. Pour réduire ce nombre de dimensions, le principe est de se baser sur les distances entre les éléments (ici les individus) pour les rapprocher et les regrouper[46].

Pour réaliser l'analyse MDS des bisons, l'option `--mds-plot` a été utilisée avec l'option `--cluster`. La commande exacte est en annexe 7.3.1 page III- 1.

GCTA

Genome-wide Complex Trait Analysis (`GCTA`) est, comme son nom l'indique, un programme permettant de réaliser diverses analyses génomiques[47]. Il a été utilisé pour réaliser une analyse en composantes principales (PCA pour *Principal Component Analysis*). Ce type d'analyse consiste à réduire le nombre de variables lorsqu'elles sont corrélées. Les nouvelles variables formées sont appelées *composantes principales* et ne sont pas corrélées entre elles. Cette analyse statistique permet de visualiser des données en éliminant le plus possible les informations redondantes[46].

Pour ce faire, `GCTA` travaille sur base d'une matrice génétique de parenté (GRM pour *Genetic Relationship Matrix*). Ce type de matrice estime la relation existant entre deux individus sur base des génotypes pour l'ensemble des SNPs. Pour créer cette matrice à partir du fichier `.bed` de `PLINK`, l'option `--make-grm` a été utilisée. De plus, l'option `--autosome` a également été utilisée pour spécifier que cette estimation devait se faire à partir des SNPs situés sur les autosomes. Ensuite, la matrice a été utilisée pour réaliser l'analyse PCA. À cette fin, l'option `--grm` a été utilisée, pour spécifier la matrice voulue, en plus de l'option `--pca`. Le nombre de composantes a été fixé à deux. Les commandes exactes sont en annexe 7.3.1 page III- 1.

ADMIXTURE

ADMIXTURE est un programme permettant d'estimer une structure à partir de génotypes d'individus non apparentés. Il a été créé en s'inspirant des programmes *structure* et EIGENSTRAT, développés précédemment[48].

L'utilisation d'ADMIXTURE a pour but de déterminer le nombre de populations ancestrales (K) ayant conduit à la population actuelle et d'examiner dans quelles proportions ces différentes populations contribuent à la population actuelle. Pour trouver le nombre K qui correspondrait le plus à nos données, l'option `--cv` (pour *cross-validation*)[49] a été utilisée pour K allant de 1 à 6. La commande exacte est en annexe 7.3.1 page III- 1.

3.4.3 Analyse de la consanguinité

Après avoir étudié la structure de la population, la consanguinité globale de chaque individu a été à son tour caractérisée. La couverture du génome en marqueurs a aussi été évaluée (densité, distribution de l'écart entre SNPs), et ce après avoir appliqué différents filtres aux marqueurs (DL, MAF). Enfin, une comparaison de la consanguinité chez des groupes de bisons nés à différentes périodes a été réalisée. La consanguinité globale et locale, ainsi que les segments consanguins ont été estimés via la version R. Le modèle utilisé principalement a été le 10p.

Sélection des SNPs : MAF et DL

Il est commun d'appliquer un seuil minimum pour la MAF et le déséquilibre de liaison (DL) des marqueurs à utiliser dans des analyses génomiques. Pour rappel, le terme *déséquilibre de liaison* est utilisé lorsque des allèles à différents loci sont associés de façon non aléatoire[50]. La consanguinité a été estimée après avoir réalisé différentes éliminations de marqueurs (avec différents seuils). Pour les DL, PLINK a été utilisé. Une première commande a été réalisée avec l'option `--indep-pairwise`. Les tailles de fenêtre utilisées sont de *2*, *5* et *10* SNPs. Le pas réalisé lors de chaque glissement a été fixé à 1 SNP. Les seuils de tolérance acceptés sont de *0,9*, *0,7* et *0,5*. Ainsi, à chaque étape, les SNPs se trouvant dans une fenêtre sont analysés en calculant pour chaque paire de SNPs le déséquilibre de liaison. Si ce DL est plus grand que le seuil fixé, l'un des SNPs de la paire est retiré. Enfin, pour obtenir un nouveau fichier sans ces SNPs en déséquilibre de liaison, une deuxième commande est réalisée avec l'option `--extract` suivie du nom du fichier de sortie (`plink.prune.in`) de la première commande. Les commandes exactes sont en annexe 7.3.1 page III- 1.

Évolution de la consanguinité globale

Dans le but de déterminer s'il y a eu une évolution dans la consanguinité globale des bisons au cours des générations, une partition du jeu de données a été effectuée. Les bisons nés avant 1995 ont vu leur consanguinité estimée et comparée à celle des bisons nés après 2005.

4 Résultats

4.1 Caractérisation de la modélisation et de l'optimisation

Les nouveaux éléments du programme porté sous R, ainsi que son bon fonctionnement dans cet environnement, ont été évalués. Cette optimisation du programme a été réalisée dans le but de pouvoir, ensuite, caractériser la consanguinité d'un individu avec un HMM à plusieurs classes. Ces classes représentent des événements de consanguinité ayant eu lieu dans un passé plus ou moins proche selon le taux G_i y étant associé.

4.1.1 Estimation des paramètres selon trois méthodes sur données simulées

Précédemment, il a été montré que les programmes `ZooRoH` et `FEstim` obtiennent une estimation identique des paramètres dans la majorité des cas[32]. Les quelques différences remarquées sont causées par les contraintes appliquées à `ZooRoH` et qui ne le sont pas à `FEstim`. Ces contraintes limitent l'estimation du taux G_i dans un intervalle de valeurs précises. Ici, une comparaison de la précision de l'estimation des paramètres par la version sous R, par la version originale de `ZooRoH` et par `FEstim` sur les jeux de données simulées (voir section 3.3.1.1) a été réalisée. Les MAEs des résultats obtenus sur ces données permettent de se faire une idée de l'écart par rapport aux valeurs vraies.

Pour chaque méthode d'estimation, on remarque (tableau 1 page 28) que les MAEs sont sensiblement identiques entre les méthodes dans plus de la moitié des cas. En ce qui concerne l'estimation des taux G_i, 20 sur les 33 sont strictement identiques pour les 3 méthodes comparées. Des différences sont cependant visibles lorsque les proportions de mélange M_i simulées sont petites. Ceci reflète la limite imposée à `ZooRoH` dans l'estimation de ces taux, empêchant les G_i de monter trop haut, et qui ne se retrouve pas dans le programme `FEstim` ni dans cette version sous R testée. En ce qui concerne les différences de MAEs observées pour les proportions de mélange M_i, les quelques différences visibles entre les méthodes sont minimes.

Globalement, on remarque également que la plupart des valeurs de ces MAEs représentent de petits écarts relativement aux "valeurs vraies" (*0,34* pour une valeur vraie de *2* et *16,67* pour une valeur vraie de *256*) même si elles sont plus grandes lorsqu'on va vers les taux plus élevés. De plus, les consanguinités estimées sont bonnes (les MAEs se situent entre 10^{-3} et 10^{-4}).

De ces observations, il peut être mis en avant que ces trois méthodes obtiennent des résultats similaires en ce qui concerne l'estimation de la consanguinité ainsi que des paramètres comparés, nécessitant peut-être l'ajout d'une contrainte pour la méthode L-BFGS-B. Cependant, ici seul le modèle 2e a été utilisé car il s'agit du seul modèle équivalent à `FEstim`. Cette première étude a permi de s'assurer que la version portée sous R (du modèle 2e) estimait correctement les paramètres et la consanguinité.

TAB. 1 : MAEs des paramètres estimés et du coefficient de consanguinité obtenus avec les trois méthodes pour différents paramètres simulés (Sim.). Une MAE meilleure par rapport aux deux autres méthodes est en **gras**. Une MAE moins bonne par rapport aux deux autres méthodes est en *italique*.

Taux G				Proportions M				Consanguinité F	
Sim.	ZooRoH	L-BFGS-B	FEstim	Sim.	ZooRoH	L-BFGS-B	FEstim	ZooRoH	L-BFGS-B
2.0	0.34	0.34	0.34	0.507	0.0325	0.0325	0.0325	0.0005	0.0005
3.0	0.43	0.43	0.43	0.249	0.0287	0.0287	0.0287	0.0005	0.0005
3.9	0.57	0.57	0.57	0.124	0.0194	0.0194	0.0194	0.0005	0.0005
6.4	**2.41**	*4.24*	3.41	0.031	0.0093	0.0093	0.0093	**0.0004**	0.0005
6.0	1.02	1.02	1.02	0.062	0.0124	0.0124	0.0124	0.0005	0.0005
6.1	0.65	0.65	0.65	0.126	0.0188	0.0188	0.0188	0.0007	0.0007
7.9	19.18	*75.57*	**18.76**	0.008	0.0038	*0.0040*	0.0038	**0.0002**	0.0006
8.6	**6.05**	*23.57*	6.61	0.014	**0.0056**	0.0057	0.0057	**0.0003**	0.0005
8.4	**2.07**	*3.33*	2.22	0.030	0.0082	*0.0083*	0.0082	0.0005	0.0005
7.9	1.12	1.12	1.12	0.063	0.0114	0.0114	0.0114	0.0006	0.0006
8.1	0.82	0.82	0.82	0.126	0.0148	0.0148	0.0148	0.0008	0.0008
16.0	**10.13**	*18.30*	11.33	0.009	0.0034	0.0034	*0.0035*	**0.0005**	0.0006
16.7	**4.02**	*4.18*	4.08	0.019	0.0054	0.0054	0.0054	0.0007	0.0007
16.0	1.99	1.99	1.99	0.049	0.0080	0.0080	0.0080	0.0009	0.0009
16.0	1.35	1.35	1.35	0.099	0.0112	0.0112	*0.0113*	0.0011	0.0011
16.0	1.18	1.18	1.18	0.149	0.0131	0.0131	0.0131	0.0012	0.0012
16.0	1.01	1.01	**1.00**	0.199	0.0160	0.0160	0.0160	0.0012	0.0012
34.3	**11.93**	*15.13*	14.42	0.010	0.0028	0.0028	*0.0029*	0.0009	0.0009
32.4	**6.13**	6.23	6.23	0.019	0.0037	0.0037	0.0037	0.0011	0.0011
32.3	3.62	3.62	3.62	0.049	0.0062	0.0062	0.0062	0.0014	0.0014
32.1	2.26	2.26	2.26	0.100	0.0085	0.0085	0.0085	0.0017	0.0017
32.1	1.87	1.87	1.87	0.151	0.0098	0.0098	0.0098	0.0018	0.0018
32.2	1.66	1.66	1.66	0.200	0.0109	0.0109	0.0109	0.0018	0.0018
65.7	**17.64**	18.52	*20.40*	0.010	0.0025	**0.0024**	*0.0026*	0.0016	0.0016
66.1	**11.15**	11.24	*11.34*	0.020	0.0033	0.0033	0.0033	0.0017	0.0017
64.4	6.17	6.17	6.17	0.050	0.0046	0.0046	0.0046	0.0021	0.0021
64.2	4.06	4.06	4.06	0.099	0.0063	0.0063	0.0063	0.0024	0.0024
64.0	3.36	3.36	3.36	0.150	0.0071	0.0071	0.0071	0.0027	0.0027
64.3	3.02	3.02	3.02	0.200	0.0087	0.0087	0.0087	0.0027	0.0027
128.1	11.79	11.79	11.79	0.050	0.0044	0.0044	0.0044	0.0030	0.0030
128.0	8.03	8.03	8.03	0.101	0.0058	0.0058	0.0058	0.0037	0.0037
256.8	26.71	**26.68**	*35.94*	0.050	0.0049	**0.0048**	*0.0067*	0.0043	**0.0042**
256.3	16.67	*16.68*	16.67	0.100	0.0055	0.0055	0.0055	0.0046	0.0046

4.1.2 Estimation des paramètres après portage sur données réelles

La méthode d'optimisation utilisée dans le programme `ZooRoH` est un algorithme EM (*Expectation Maximization*). Comme expliqué à la section 3.2.2, cette façon d'optimiser les paramètres a été modifiée, dans cette version sous R, pour une autre de la fonction `optim` (package "stats") qui repose sur un algorithme de quasi-Newton. De plus, une reparamétrisation a été effectuée.

Pour connaître l'impact de l'utilisation de la méthode L-BFGS-B (`optim`) sur l'estimation des paramètres et du coefficient de consanguinité, les résultats ont été comparés à ceux obtenus avec `ZooRoH` via les quatre modèles détaillés dans le matériel et méthodes (section 3.3.2). Les jeux de données des chevaux et des BBB 50K ont été utilisés et les vraisemblances, nombres d'itérations, consanguinités globales et paramètres ont été comparés. Les résultats suivants concernent les BBB.

4.1.2.1 Estimation de la nécessité de la reparamétrisation pour L-BFGS-B

Dans le but de contraindre les classes, la reparamétrisation a été la première étape de l'optimisation de la version sous R. L'estimation de son utilité a été réalisée en comparant l'utilisation de la méthode L-BFGS-B avec et sans cette modification des paramètres donnés à la fonction `optim`. Sans cette reparamétrisation, le programme s'arrête à cause de problèmes numériques. L'ajout d'une contrainte via la fonction `optim` a alors été réalisé pour essayer d'obtenir des résultats sans reparamétrisation. À nouveau, le programme s'arrête fréquemment suite à des problèmes numériques (14 individus sur les 105 premiers BBB n'ont pas pû être estimés pour le modèle 2e). Sur les individus pour lesquels un résultat a été obtenu, c'est-à-dire pour lesquels le programme ne s'est pas arrêté, des analyses ont été réalisées sur ce modèle. Cependant, il est apparu que les contraintes données via la fonction `optim` ne fonctionnent pas pour les proportions de mélange M_i (elles ne sont pas dans l'intervalle $0 < M_i < 1$) et fonctionnent mal pour les taux G_i. La reparamétrisation semble donc nécessaire à l'obtention de résultats conformes aux valeurs attendues (avec $\sum M_i = 1$ et $G_i > 0$). Cependant, elle ne permet pas de mettre une limite supérieure à l'estimation des taux G_i.

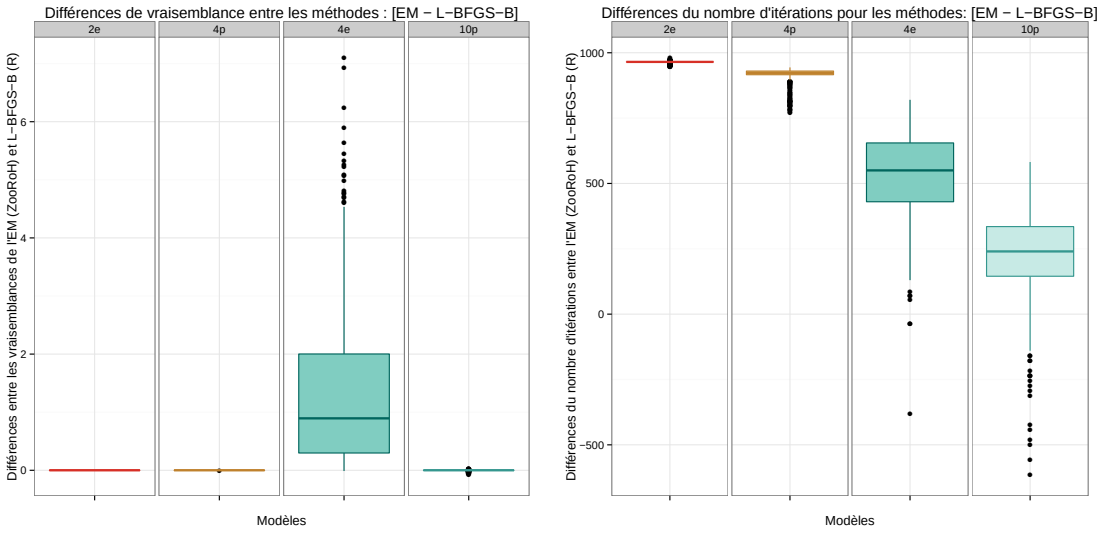
4.1.2.2 Estimation des paramètres avec et sans contraintes pour L-BFGS-B

L'ajout de contraintes est nécessaire afin de garantir l'obtention de paramètres souhaitables. Celles-ci sont majoritairement comprises dans la reparamétrisation mais les taux de la distribution exponentielle devraient avoir un seuil au-delà duquel la densité en marqueurs serait insuffisante pour obtenir une estimation correcte. En effet, la densité de marqueurs conditionne la taille des segments pouvant être identifiés. Les mêmes contraintes que celles de `ZooRoH` ont ainsi été appliquées dans la version sous R. Le taux minimum est toujours fixé à 1. Pour le jeu de données 50K des BBB, le taux maximum a été fixé à 1024. Seul les modèles 2e et 4e sont concernés par cette modification.

Le modèle 2e ne montre aucune différence entre l'ajout ou non de contraintes, que ce soit concernant la vraisemblance, le nombre d'itérations ou encore le taux G_i estimé de l'exponentielle (figures S.2a, S.2b et S.2c). Le modèle 4e montre quelques petites différences mais obtient en médiane les mêmes valeurs (figures S.2a, S.2b et S.2d).

Ainsi, l'ajout de contraintes n'influence pas les valeurs obtenues avec le modèle 2e. Concernant le modèle 4e, sans influencer grandement les résultats, les contraintes garantissent une certaine robustesse au programme, l'empêchant de donner des valeurs de taux G_i trop grandes. La suite sera donc réalisée avec l'ajout de ces limites.

4.1.2.3 Estimation des paramètres : comparaison EM et L-BFGS-B

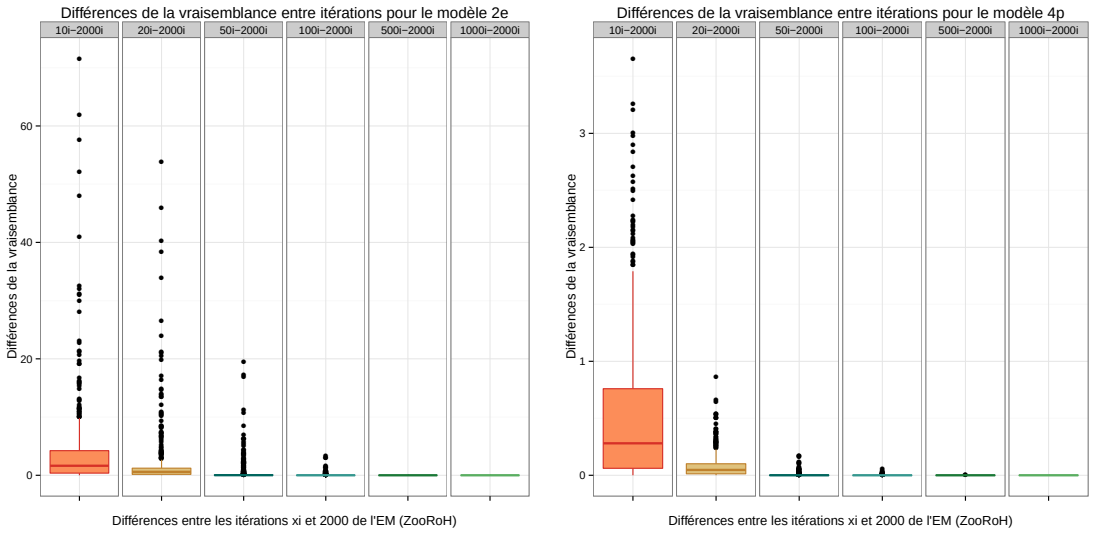


(a) Différence de vraisemblance. (b) Différence du nombre d'itérations.

FIG. 7 : Différences entre l'EM (ZooRoH) et L-BFGS-B (R) pour les 4 modèles sur les BBB (50K). Seul le modèle 4e montre une différence de vraisemblance (7a). Le modèle 10p dépasse parfois les 1000 itérations fixées de l'EM pour la méthode L-BFGS-B (7b).

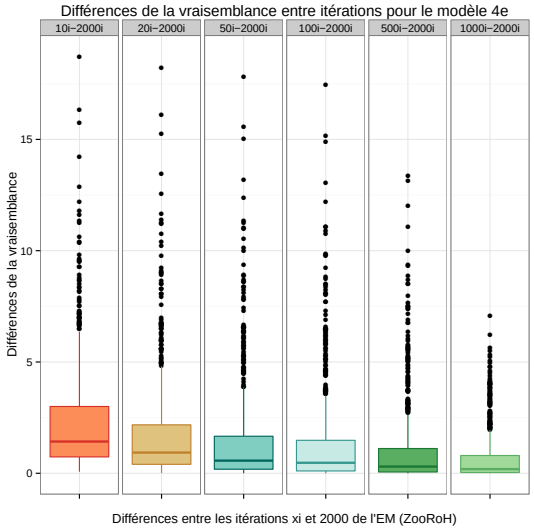
Lors de cette comparaison, différents éléments ont été observés. Tout d'abord, des vraisemblances différentes n'ont été mises en évidence que pour le modèle 4e. Celles-ci semblent légèrement à l'avantage de la méthode L-BFGS-B (figure 7a). De plus, étant donné que le nombre d'itérations de `ZooRoH` est fixé au départ, la version portée sous R en réalise moins. On remarque cependant que ce nombre augmente avec le nombre de paramètres à estimer, c'est-à-dire les modèles 2e, 4p, 4e puis 10p (figure 7b). Le modèle 10p se rapproche des 1000 itérations et dépasse dans certains cas ce nombre pré-défini pour `ZooRoH`.

Il est cependant à noter que la vraisemblance obtenue par `ZooRoH` converge avant cette limite des 1000 itérations, excepté pour le modèle 4e (figures 8a, 8b, 8c et 8d). On observe que l'EM converge moins vite que le L-BFGS-B lors de l'estimation des taux G_i (2e, 4e) mais que c'est l'inverse lorsque ce paramètre est prédéfini (4p, 10p). En effet, le modèle 2e converge à peu près

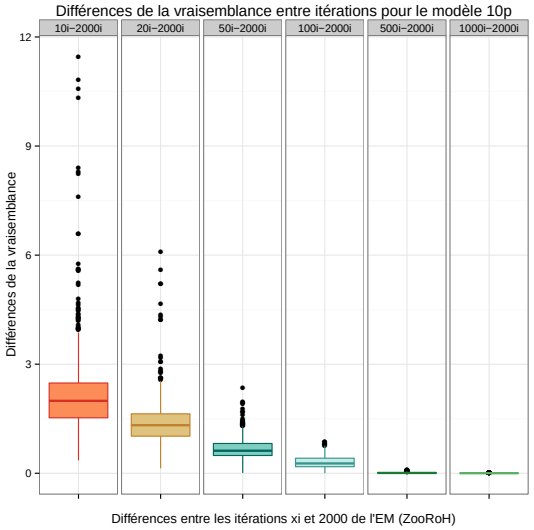


(a) Modèle 2e.

(b) Modèle 4p.



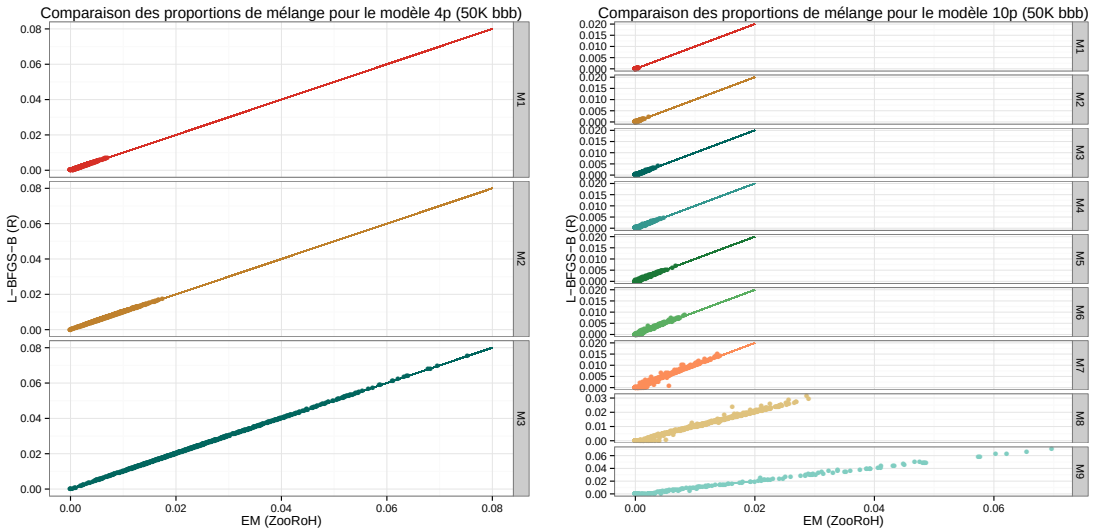
(c) Modèle 4e.



(d) Modèle 10p.

FIG. 8 : Différences de vraisemblance entre la xième itération et la 2000ième de l'EM (ZooRoH) sur les BBB (50K). L'EM semble converger entre la 100ième et 500ième itération pour les modèles 2e et 10p (8a et 8d) et aux environs de la 100ième pour le modèle 4p (8b). Il ne converge pas pour le modèle 4e (8c).

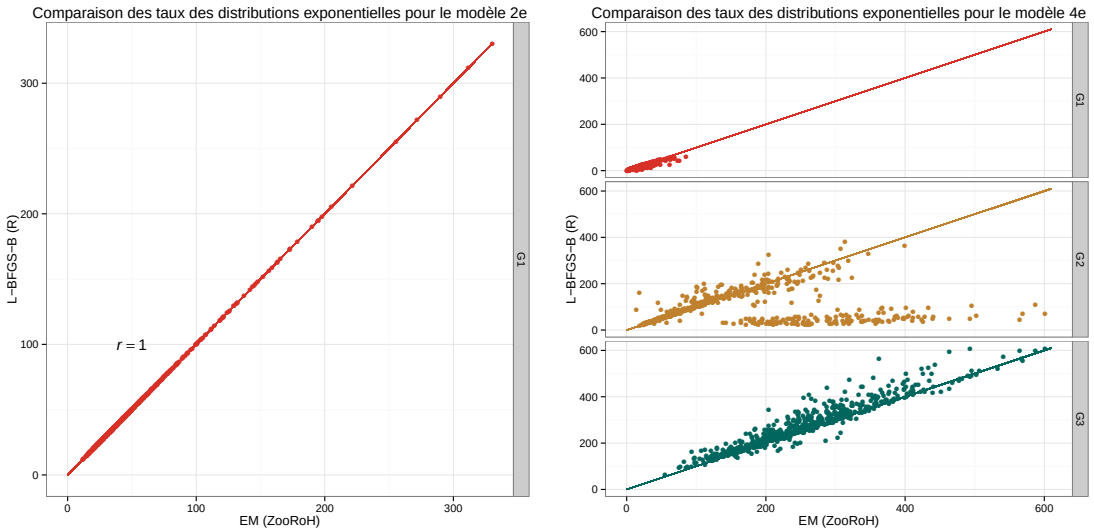
à 40 itérations pour L-BFGS-B et aux alentours de la 100$^{\text{ième}}$ itération pour `ZooRoH` et le modèle 4e converge vers les 500 itérations pour L-BFGS-B et n'a pas convergé à la 1000$^{\text{ième}}$ itération pour l'EM (figures 7b et 8). Par contre, le modèle 4p converge entre les itérations 50 et 100 pour l'EM alors qu'il converge entre 56 et 231 itérations pour `optim` et le modèle 10p converge vers la 500$^{\text{ième}}$ itération pour `ZooRoH` alors que le L-BFGS-B est largement au-delà. En ce qui concerne les proportions de mélange M_i, pour les modèles 4p et 10p (figure 9), on remarque qu'elles sont très similaires entre les deux méthodes d'estimation des paramètres. En effet, les corrélations de Pearson (r) se situent au-dessus de *0,99*, à l'exception de la première classe du modèle 10p qui s'en rapproche (*0,9881*).



(a) Modèle 4p. (b) Modèle 10p.

FIG. 9 : Comparaison des proportions de mélange pour les modèles Qp sur les BBB (50K) entre l'EM (ZooRoH) et L-BFGS-B (R). Les corrélations pour le modèle 4p (9a) sont de *0,9999*, *0,9997* et *1*. Pour le modèle 10p (9b), elles sont de *0,9881*, *0,9946*, *0,9967*, *0,9953*, *0,994*, *0,9927*, *0,9928*, *0,9959* et *0,9986*.

Enfin, on remarque que les taux G_i estimés pour le modèle 4e sont assez différents. Les corrélations pour les trois classes de consanguinité sont respectivement de *0,9356*, *0,2385* et *0,9501* (figure 10b) chez les BBB 50K. Le modèle 2e a une corrélation de *1* avec des valeurs identiques entre les deux versions du programme (figure 10a).



(a) Modèle 2e. (b) Modèle 4e.

FIG. 10 : Comparaison des taux des distributions exponentielles pour les modèles Qe sur les BBB (50K) entre l'EM (ZooRoH) et L-BFGS-B (R). Les corrélations pour le modèle 4e (10b) sont de *0,9356*, *0,2385* et *0,9501*.

De ces analyses, on remarque que, pour une vraisemblance similaire, on obtient des estimations de paramètres différentes. Cependant, si les taux G_i estimés changent, les proportions M_i as-

sociées changeront également. Il se peut donc que les méthodes soient équivalentes en ce qui concerne l'estimation du coefficient de consanguinité. Quant au nombre d'itérations, les modèles *Qe* semblent converger plus vite avec la version portée sous R. Pour les modèles *Qp*, le nombre d'itérations est plus élevé. Cependant, une itération du L-BFGS-B comprend un *forward* alors qu'une itération de l'EM comprend une combinaison *forward*/*backward*, réalisant ainsi deux fois plus de calculs. Du point de vue de l'estimation des paramètres, la version portée sous R semble donc avantageuse par rapport à l'EM de `ZooRoH`.

4.1.2.4 Consanguinité globale : comparaison EM et L-BFGS-B

Comme expliqué à la section 3.2, une modification de la méthode d'estimation des paramètres, ainsi que l'ajout de la reparamétrisation, a été apportée lors du portage de `ZooRoH` sous R. Leur impact sur l'estimation de la consanguinité globale a été analysé. On remarque que seul le modèle 4e montre une très légère différence (figure 11a). En ce qui concerne les contributions de chaque classe à cette consanguinité, le modèle 10p (figure 11d) montre de légères différences entre les deux méthodes. Le modèle 4e (figure 11c) diffère également, mais avec une plus grande variabilité. Les deux méthodes estiment de la même façon les contributions de chaque classe à la consanguinité totale avec le modèle 4p (figure 11b). De plus, le modèle 2e n'ayant qu'une seule classe IBD, celle-ci représente la consanguinité totale (figure 11a).

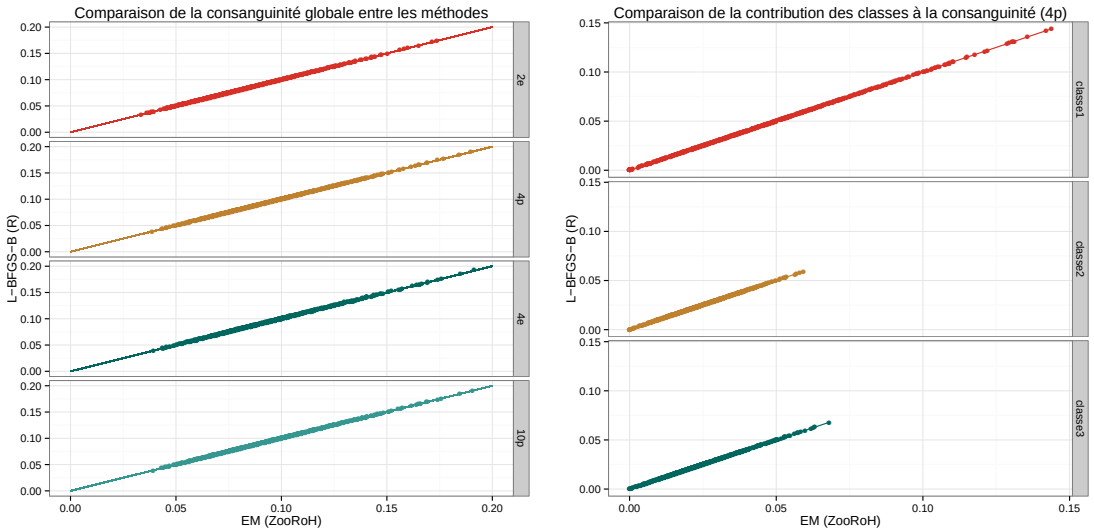
La consanguinité totale évaluée, ainsi que la contribution de chaque classe à celle-ci, semble la même avec les deux versions du programme. Il est à noter que la différence observée pour le modèle 4e pourrait être due au fait que ce modèle n'avait pas encore convergé avec l'EM lorsque l'estimation a été arrêtée, dû à la limite des 1000 itérations.

4.1.3 Effet de la densité en marqueurs sur l'estimation de la consanguinité globale

L'effet de la densité sur l'estimation de la consanguinité globale a été étudié en comparant les résultats obtenus sur les trois jeux de données BBB (LD, 50K et HD).

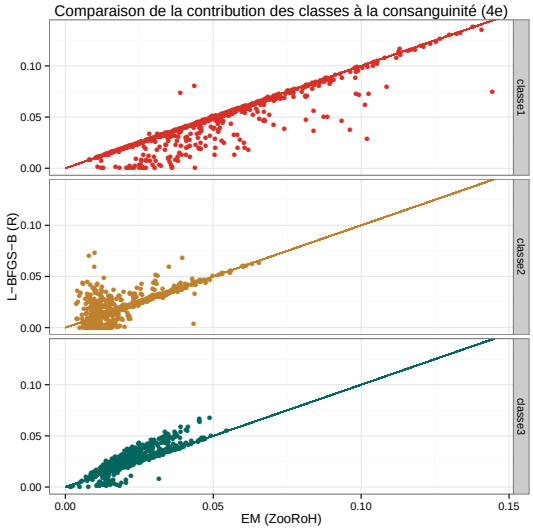
Lorsque les consanguinités sont comparées, on remarque qu'une plus faible densité de marqueurs donne une plus faible estimation de la consanguinité totale (figures 12a, S.3a, S.3b et S.3c). Par contre, les corrélations pour les 4 modèles sont élevées : plus grandes que *0,94* entre LD et 50K, entre *0,95* et *0,98* entre 50K et HD et entre *0,91* et *0,94* entre LD et HD. De plus, les consanguinités estimées pour chaque densité de marqueurs sont identiques entre L-BFGS-B (R) et EM (`ZooRoH`, figures 11a, S.4a et S.4b, corrélations de 1). On remarque également à la figure 12b que la diminution de la consanguinité estimée est perdue au niveau des classes les plus anciennes. Ainsi, si l'on cherche de la consanguinité récente avec une faible densité de marqueurs, elle sera globalement bien estimée.

Il est également à noter que le nombre d'itérations ne semble pas être influencé par la densité :

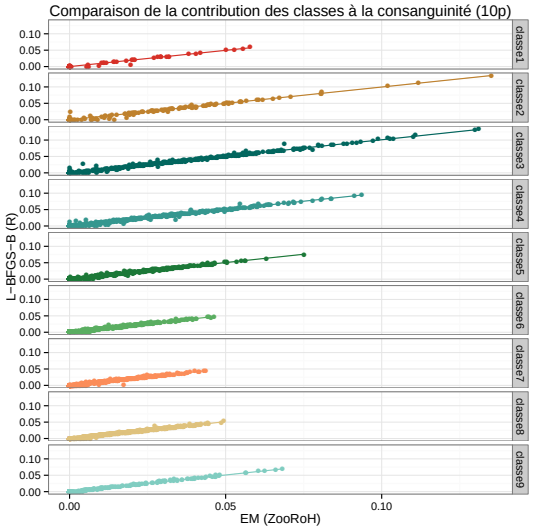


(a) Consanguinité totale.

(b) Modèle 4p (contribution des classes).



(c) Modèle 4e (contribution des classes).

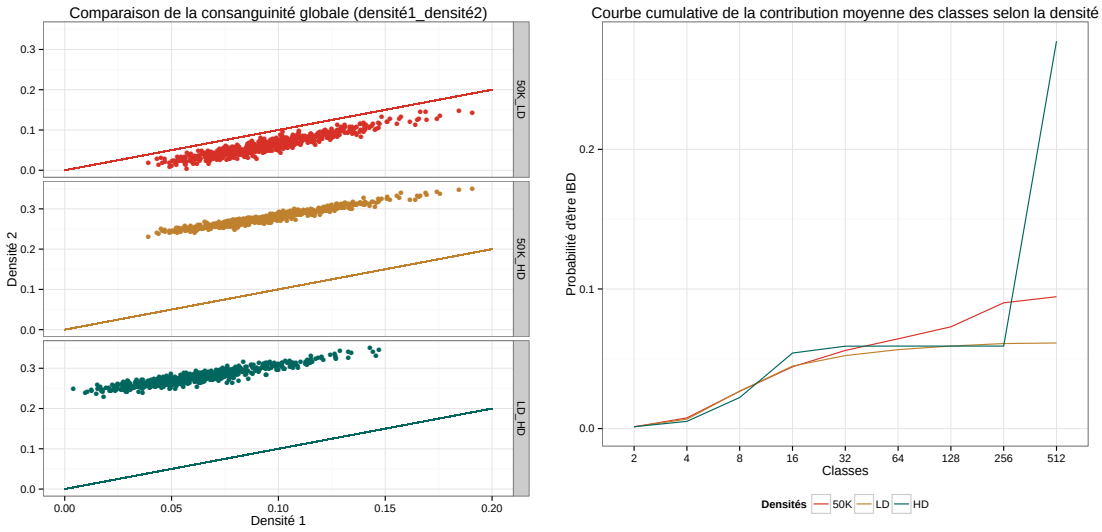


(d) Modèle 10p (contribution des classes).

FIG. 11 : Comparaison entre EM (ZooRoH) et L-BFGS-B (R) de la consanguinité estimée chez les BBB 50K. **(11a)** Comparaison de la consanguinité totale. Les corrélations sont de *1* pour les modèles 2e, 4p et 10p et de *0,9998* pour le modèle 4e. **(11b)** Comparaison de la contribution de chaque classe à la consanguinité totale pour le modèle 4p. Les corrélations sont de *1* pour les trois classes. **(11c)** Comparaison de la contribution de chaque classe à la consanguinité totale pour le modèle 4e. Les corrélations sont de *0,9199*, *0,7097* et *0,8399*. **(11d)** Comparaison de la contribution de chaque classe à la consanguinité totale pour le modèle 10p. Les corrélations sont de *0,9933*, *0,995*, *0,9964*, *0,9954*, *0,9945*, *0,993*, *0,9927*, *0,9959* et *0,9986*.

c'est avec le jeu de données 50K que les modèles 2e, 4p et 4e font le moins d'itérations alors que le modèle 10p en fait le plus (figures S.5a, S.5b, S.5c, S.5d).

Par conséquent, une faible densité de marqueurs sous-estime systématiquement l'estimation de la consanguinité ancienne mais un niveau de consanguinité élevé reflètera bien une consanguinité importante.



(a) Consanguinité totale. (b) Contribution moyenne des classes.

FIG. 12 : Comparaison de l'estimation de la consanguinité selon la densité du jeu de données. **(12a)** Comparaison de la consanguinité totale entre les trois densités de marqueurs chez les BBB (modèle 10p). Les résultats obtenus pour une plus faible densité en marqueurs (LD par rapport aux deux autres, 50K par rapport au HD) ont des valeurs plus petites. Les corrélations sont de : *0,9442*, *0,9701* et *0,9323*. **(12b)** Courbe cumulative de la contribution moyenne des classes selon la densité. Lorsque la densité diminue, la consanguinité ancienne est sous-estimée.

4.1.4 Effet des modèles sur l'estimation de la consanguinité globale

Les résultats obtenus via les différents modèles ont été comparés (ici sur les BBB 50K). On remarque que les consanguinités totales sont bien corrélées d'un modèle à l'autre (entre *0,9808* et *0,9996*) mais que les consanguinités obtenues ne sont pas exactement les mêmes : le modèle 2e sous-estime la consanguinité totale par rapport aux autres modèles (exemple à la figure 13, les autres sont en annexe S.6a, S.6b, S.6c) comme attendu suite au développement de `ZooRoH`[32].

Comme expliqué à la section 1.4.6, ceci illustre qu'une classe IBD ne semble pas suffisante pour modéliser des événements complexes de consanguinité passée. De plus, on remarque que les deux modèles les plus corrélés sont également ceux considérant un passé de consanguinité plus complexe : les modèles 4e et 10p.

4.1.5 Estimation de la consanguinité locale : *forward/backward* vs Viterbi

La modélisation nous permet également d'estimer le coefficient de consanguinité locale, c'est-à-dire la probabilité de consanguinité associée à chaque SNP. Cependant, il peut être utile d'obtenir des segments IBD plutôt que des probabilités. L'obtention de segments IBD et non IBD est possible via l'algorithme de Viterbi.

Les résultats de l'algorithme de Viterbi ont été comparés à ceux du *forward/backward*. Pour ce faire, les MAEs de la consanguinité locale à toutes les positions et celles dans les segments IBD

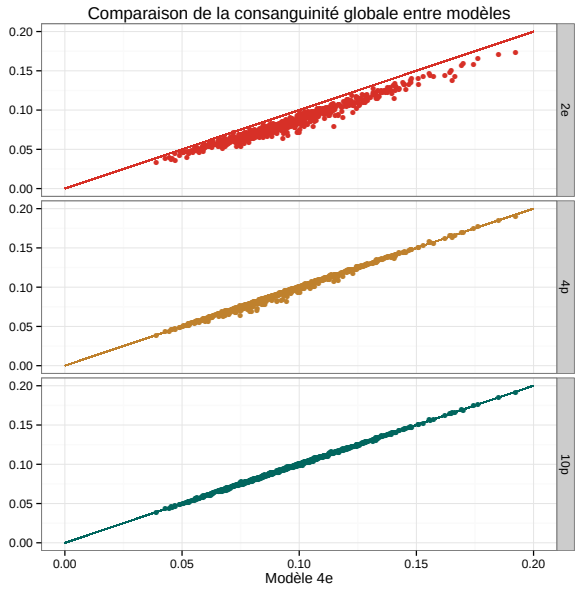


FIG. 13 : Comparaison de la consanguinité globale entre le modèle 4e et les autres. Le modèle 2e la sous-estime par rapport aux autres. Les corrélations sont respectivement de *0,9818*, *0,9969* et *0,9996*.

ont été calculées pour ces deux approches. Il est à noter que les concepts ne sont pas tout-à-fait identiques étant donné que le *forward/backward* donne des probabilités alors que le Viterbi détermine si un SNP est IBD ou pas. On remarque (tableau 2) que lorsque les taux G_i simulés sont faibles, les MAEs sont légèrement meilleures pour l'algorithme de Viterbi. Par contre, lorsque les taux G_i augmentent, la combinaison *forward/backward* fait mieux. Les faibles différences entre les résultats obtenus avec l'algorithme de Viterbi et le *forward/backward* peuvent être dues à une estimation de *0,99* alors que l'algorithme de Viterbi mettra toujours 1. Ceci montre que les probabilités sont meilleures pour l'estimation de la consanguinité lorsque peu d'informations sont accessibles, c'est-à-dire lorsque le nombre de SNPs par segment diminue. Le *forward/backward* fera donc mieux lorsque le taux G_i est élevé et lorsque la densité en marqueur est faible.

Pour ce qui est de l'estimation de la consanguinité globale, on remarque que la consanguinité estimée par *forward/backward* est toujours la même que celle réalisée dans la simulation. Par contre, celle estimée par Viterbi est presque systématiquement soit sous-estimée soit surestimée. Ceci illustre certainement la nuance apportée par le *forward/backward* donnant des probabilités par rapport au Viterbi estimant si un SNP est IBD ou pas.

De plus, on peut comparer sur les BBB les segments obtenus et la probabilité d'être consanguin le long des chromosomes. On voit que les deux types de mesures ne sont pas les mêmes. En effet, l'algorithme de Viterbi ne coupe pas le deuxième segment consanguin (figure 14) alors que la probabilité descend à environ 75%. Par contre, pour deux pics de probabilité avoisinant les 90%, l'un est considéré comme segment consanguin tandis que l'autre pas. De plus, les pics aux alentours de 60% ne sont pas non plus considérés comme consanguins par l'algorithme de Viterbi alors que selon le *forward/backward* il est plus probable qu'il le soit par rapport à non IBD.

TAB. 2 : MAEs de l'estimation de la consanguinité avec *forward/backward* et Viterbi. "MAE tot" représente les MAEs calculées sur tous les SNPs et "MAE seg" représente les MAEs calculées au niveau des segments IBD. Une MAE meilleure par rapport à l'autre méthode est en **gras**.

Consanguinité F			*forward/backward*		Viterbi	
Simulée	*for/back*	Viterbi	MAE tot	MAE seg	MAE tot	MAE seg
0.502	0.502	0.503	0.002	0.002	0.002	**0.000**
0.252	0.252	0.253	0.003	0.006	0.003	**0.002**
0.128	0.128	0.129	0.003	0.010	**0.002**	**0.004**
0.032	0.032	0.032	0.001	0.019	0.001	**0.010**
0.063	0.063	0.064	0.002	0.018	0.002	**0.009**
0.124	0.124	0.126	0.004	0.016	**0.003**	**0.008**
0.008	0.008	0.008	0.001	0.032	**0.000**	**0.021**
0.015	0.015	0.015	0.001	0.029	0.001	**0.019**
0.031	0.031	0.031	0.002	0.027	**0.001**	**0.017**
0.065	0.065	0.066	0.003	0.024	0.003	**0.014**
0.127	0.127	0.128	0.005	0.021	**0.004**	**0.011**
0.010	0.010	0.010	0.001	0.065	0.001	**0.055**
0.020	0.020	0.020	0.003	0.062	**0.002**	**0.050**
0.050	0.050	0.050	0.006	0.055	**0.004**	**0.042**
0.099	0.099	0.100	0.010	0.050	**0.008**	**0.036**
0.149	0.149	0.151	0.013	0.044	**0.011**	**0.030**
0.200	0.200	0.203	0.016	0.040	**0.013**	**0.026**
0.010	0.010	0.009	0.003	**0.160**	**0.002**	0.169
0.020	0.020	0.018	0.006	**0.141**	**0.004**	0.142
0.049	0.049	0.047	0.012	0.123	**0.009**	**0.120**
0.100	0.100	0.098	0.021	0.103	**0.017**	**0.094**
0.150	0.150	0.149	0.028	0.092	**0.023**	**0.078**
0.199	0.199	0.200	0.033	0.083	**0.027**	**0.066**
0.010	0.010	0.007	0.006	**0.326**	**0.005**	0.385
0.020	0.020	0.015	0.012	**0.291**	**0.009**	0.347
0.050	0.050	0.040	0.024	**0.243**	**0.019**	0.281
0.100	0.100	0.088	0.041	**0.206**	**0.033**	0.225
0.151	0.151	0.139	0.054	**0.179**	**0.044**	0.184
0.200	0.200	0.191	0.064	0.161	**0.053**	**0.157**
0.050	0.050	0.026	0.044	**0.439**	**0.033**	0.572
0.100	0.100	0.065	0.074	**0.368**	**0.059**	0.466
0.050	0.050	0.007	0.066	**0.669**	**0.046**	0.887
0.100	0.100	0.029	0.113	**0.569**	**0.086**	0.786

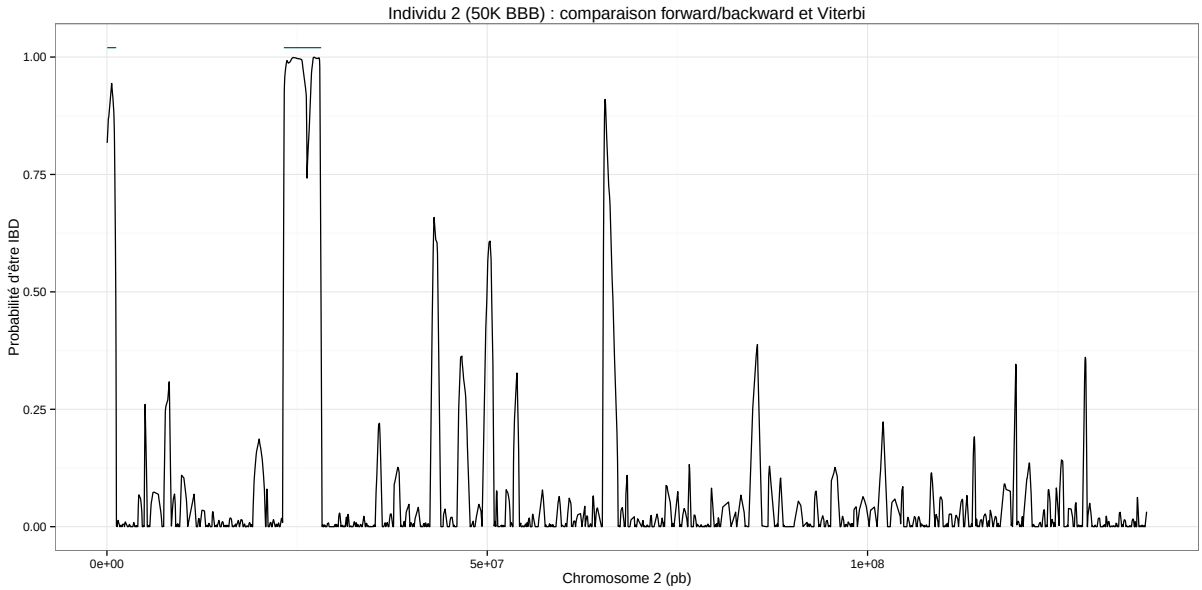


FIG. 14 : Consanguinité locale et segments consanguins (chromosome 2, individu 2, BBB 50K, modèle 4e). Les segments trouvés par Viterbi (lignes au-dessus) restent continus lors d'une probabilité locale plus faible sur une petite distance entre deux endroits aux probabilités plus élevées obtenues via *forward/backward*.

4.2 Étude de la consanguinité chez les bisons d'Europe

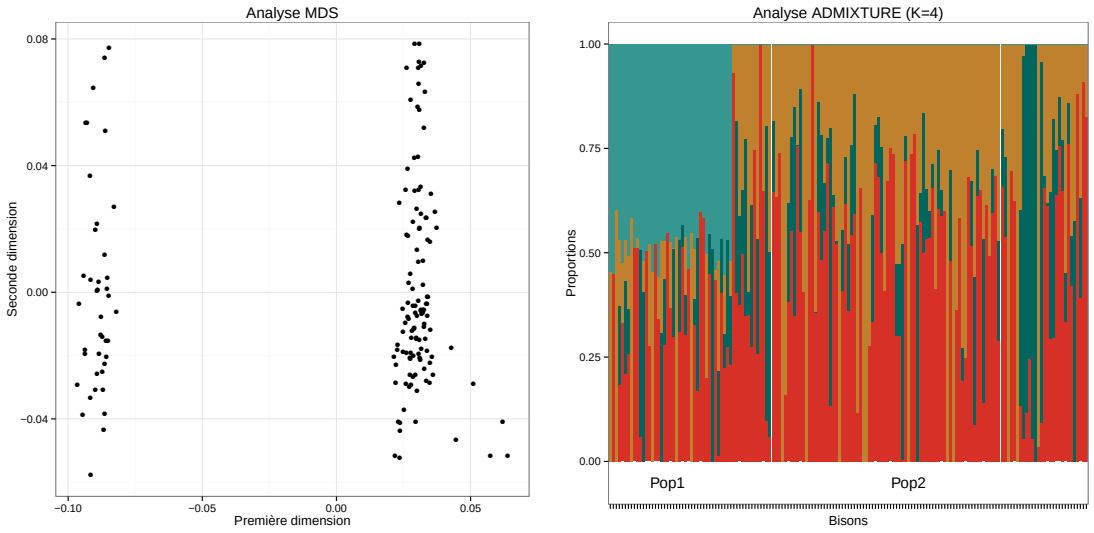
Suite à la caractérisation de l'optimisation de la modélisation, l'application de ce modèle à une population sauvage, les bisons d'Europe, a été possible. Tout d'abord, la structure de la population a été analysée en réalisant des études statistiques multivariées (MDS, PCA, `ADMIXTURE`). Ensuite, une analyse de la consanguinité a été réalisée. Préalablement à celle-ci, la répartition des marqueurs le long du génome a été vérifiée. L'impact des paramètres de sélection des SNPs (selon des critères de déséquilibre de liaison ou de MAF) a également été analysé. Enfin, l'évolution du niveau de la consanguinité a été réalisée en comparant des animaux nés à différentes périodes.

La consanguinité a été analysée via les mêmes modèles que ceux utilisés lors du portage du programme sous R. Cependant, étant donné qu'il a été mis en évidence précédemment que les résultats obtenus sont similaires, seuls les résultats obtenus chez les bisons avec le modèle à 10 classes avec les taux G_i prédéfinis seront montrés. Ce choix se justifie par la facilité de comparaison des différents paramètres (étant donné que les catégories sont fixées par individu et donc identiques) mais aussi parce qu'elle couvre de nombreuses générations d'ancêtres (par rapport à quatre classes, les segments IBD pourront être associés à un état remontant plus loin dans le passé ou étant plus précis).

4.2.1 Structure de la population

Trois types d'analyses ont été réalisées en parallèle : une analyse MDS (via le programme `PLINK`), une analyse PCA (via le programme `GCTA`) et un clustering de sous-populations (via le programme

ADMIXTURE). Lorsque les résultats de l'analyse MDS sont analysés (figure 15a), on remarque que deux sous-populations sont mises en évidence. Celles-ci ont été confirmées avec l'analyse PCA (figure S.8). 41 bisons sont dénombrés dans la petite sous-population et 118 dans la plus grande (annexe 7.2.1.1). Lors de l'analyse avec ADMIXTURE, en considérant d'abord les individus de la petite sous-population, il est également clairement visible qu'il y a une subdivision de la population d'origine (figure 15b).



(a) Analyse MDS. (b) Analyse ADMIXTURE.

FIG. 15 : Analyse structurale de la population. L'analyse MDS (15a) montre une nette séparation de la population en deux groupes distincts. Il y a 41 individus dans le groupe de gauche contre 118 dans celui de droite. Lors de l'analyse avec ADMIXTURE (15b), les individus des deux sous-populations trouvées dans (15a) sont mis côte à côte. Un pattern différent est également clairement visible entre les deux groupes.

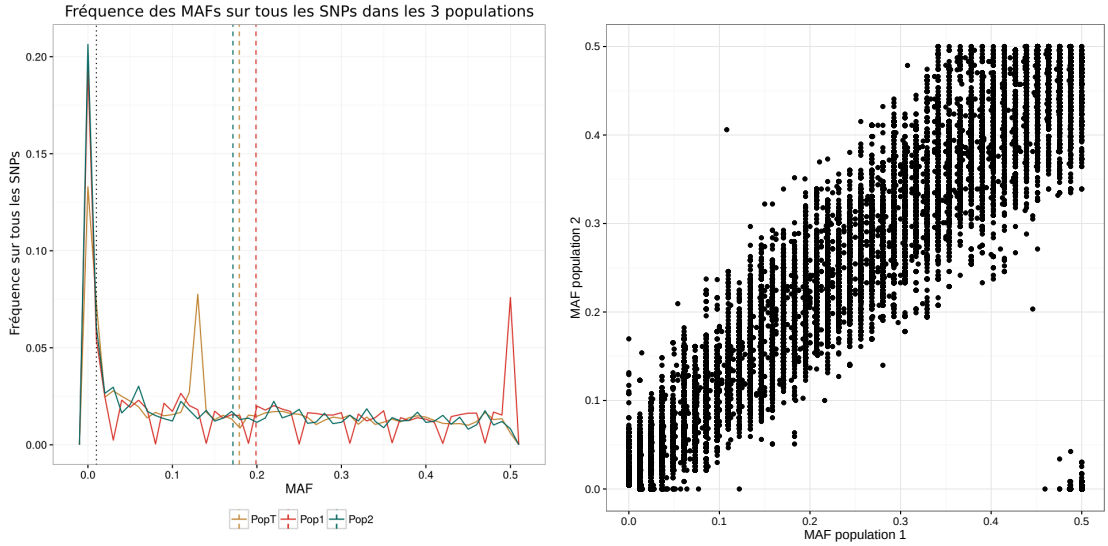
Trois hypothèses ont été envisagées concernant ces deux sous-populations. Tout d'abord, les bisons pourraient venir d'endroits différents (et donc appartenir à des populations ou lignées différentes). Ensuite, les bisons pourraient avoir des âges différents. Enfin, le génotypage des bisons pourrait ne pas avoir été réalisé pour tous de la même façon et en même temps.

En ce qui concerne la provenance des animaux, 7 sur les 41 de la petite sous-population ne viennent pas de la forêt de Bialowieza polonaise. Les autres se répartissent dans diverses zones de cette forêt, pas nécessairement connexes (figure S.7).

En ce qui concerne les âges, il apparaît que les individus de la petite sous-population sont nés entre 1984 et 2007 (à l'exception de quatre animaux dont on ne connait pas la date de naissance, 7.2.1.1).

La dernière hypothèse a été envisagée en analysant la fréquence allélique (`calcul_maf_v2.R` section 7.4.5). La figure 16a présente des pics différents entre les deux groupes. La petite sous-population présente notamment un pic à 0,5. Pour comprendre d'où vient cette différence, une analyse des SNPs concernés a été réalisée.

Il a été remarqué que les 1504 SNPs concernés (figure 16b) étaient systématiquement hétérozygotes dans la petite sous-population et systématiquement homozygotes dans la seconde. Ces SNPs sont répartis tout le long du génome de façon aléatoire (figure S.9a). Sous cet angle, il pourrait s'agir d'erreurs de génotypages (lors du *genotype calling*). Celui-ci a pu être réalisé en plusieurs fois sur ces différents animaux entraînant des regroupements différents. Ceci a été confirmé par certains animaux qui ont été génotypés deux fois. Dans l'un des deux groupes ils étaient homozygotes pour ces SNPs et dans l'autre ils étaient hétérozygotes.



(a) Distribution des MAFs. (b) Comparaison des MAFs.

FIG. 16 : Analyse des MAFs dans les deux sous-populations. **(16a)** Les lignes verticales montrent les positions des moyennes. La petite sous-population (pop1) montre un pic à 50% non visible dans la deuxième sous-population. **(16b)** Les points en bas à droite (1504) montrent les SNPs pour lesquels la MAF est de *0,5* pour la petite sous-population et de *0* pour la grande sous-population.

Cette structure de la population serait donc un artéfacte dû au génotypage. Dans cette optique, ces marqueurs ont été retirés du jeu de données et les mêmes analyses ont été recommencées. Cette fois, l'analyse MDS, PCA, `ADMIXTURE` et des MAFs ne montrent plus cette subdivision de la population (figures S.10).

On peut donc conclure de cette analyse structurale que la population de bisons de notre jeu de données est relativement homogène. Nous n'avons donc bien qu'une seule population.

Les analyses suivantes sont réalisées sur le jeu de données nettoyé (donc sans les 1504 SNPs).

4.2.2 Distribution des marqueurs et consanguinité locale

Étant donné que les bisons ont été génotypés sur une puce prévue au départ pour les bovins, la distribution des marqueurs le long du génome (carte physique) et leur MAF doivent être vérifiées pour s'assurer d'une couverture suffisante. En effet, si certains SNPs ne sont pas polymorphes

chez ces animaux par rapport aux bovins d'élevage, des parties du génome peuvent ne pas être couvertes pour l'estimation de la consanguinité.

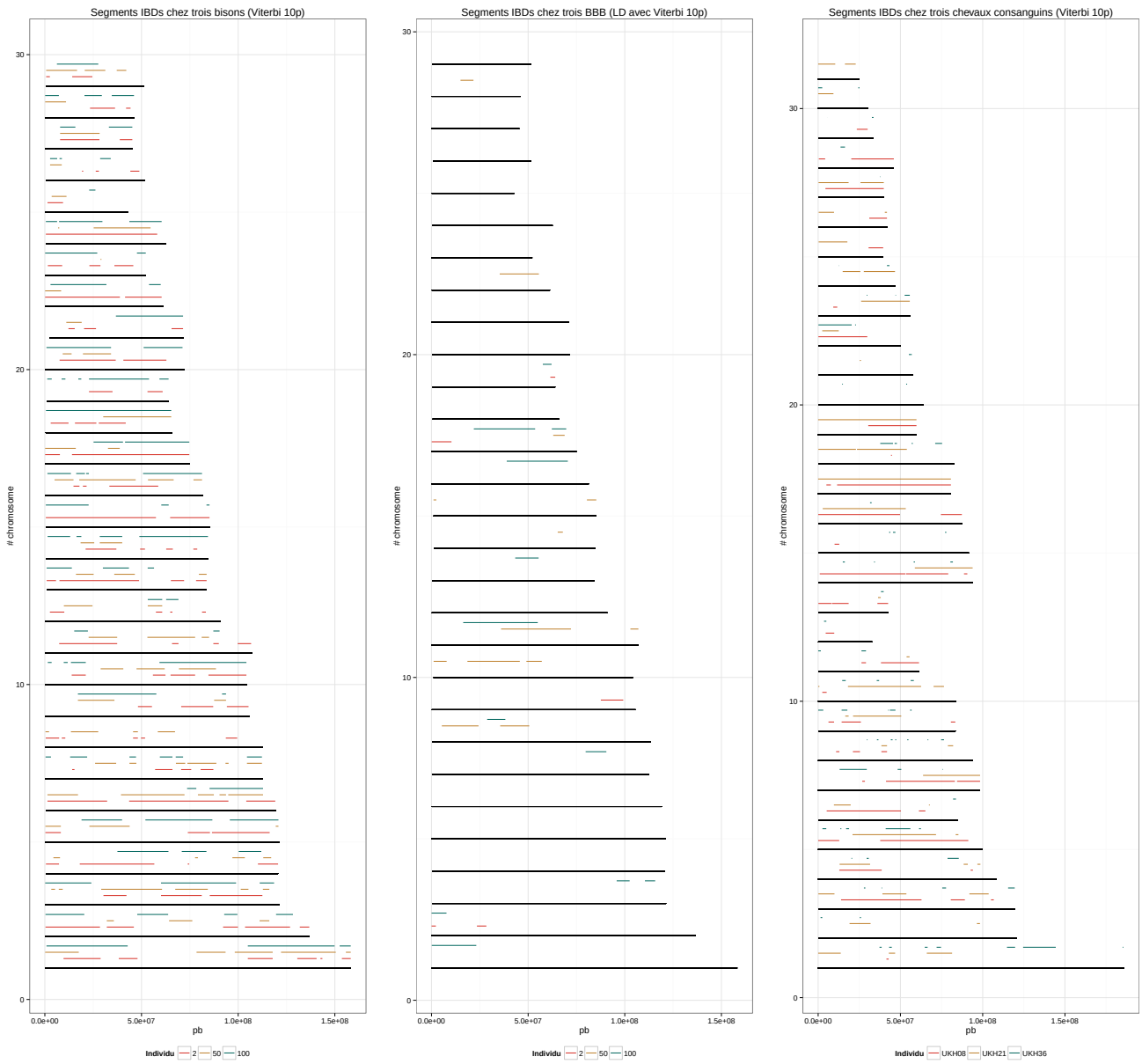
FIG. 17 : Position le long des chromosomes des SNPs utilisés aux MAFs *0*, *0,01* et *0,05*. Les positions découvertes à partir de *0,01* sont rouges, celles découvertes à partir de *0,05* sont vertes.

La première observation lors de l'analyse de la distribution des marqueurs a été que, malgré un élagage conséquent dans le nombre de SNPs avec une MAF de *0,01* (tableau 3), celui-ci est réalisé de façon assez uniforme sur le long du génome. On peut cependant remarquer que certaines petites parties ne sont plus couvertes, notamment le début du chromosome 27 (figure 17). Une MAF de *0,05* retire également des marqueurs sur tout le génome mais en obtenant plus de régions qui ne sont plus couvertes (par exemple la fin du chromosome 6).

TAB. 3 : Nombre de SNPs retenus selon la MAF et le déséquilibre de liaison appliqués. Les DL sont notés sous la forme "fenêtre-seuil" (comme expliqué à la section 3.4.3).

	DL appliqué			
MAF	Aucun	2-0,9	5-0,9	10-0,5
0,00	20282	14895	10731	6131
0,01	16028	11407	7971	3936
0,05	13928	9872	6879	3277

Lorsque les segments consanguins identifiés avec l'algorithme de Viterbi (à une MAF de *0,01*)



(a) Bisons. (b) BBB (LD). (c) Chevaux.

FIG. 18 : Segments IBD estimés par Viterbi sur le modèle 10p. **(18a)** Chez les bisons 2, 50 et 100. **(18b)** Chez les BBB 2, 50 et 100 à la densité LD. **(18c)** Chez les chevaux UKH08, UKH21 et UKH36 ayant une consanguinité attendue de 25% pour les deux premiers et de 3% pour le dernier.

sont illustrés pour quelques individus (figure 18a), on remarque que certains chromosomes sont presque entièrement considérés comme IBD, notamment les chromosomes 22 et 24 pour l'individu 2 ou encore le chromosome 18 pour l'individu 100. Lors d'une comparaison avec les résultats obtenus via les algorithmes *forward/backward* (figure 19 pour le chromosome 24), on remarque que pour l'individu 100, le dernier segment IBD détecté par Viterbi comprend une partie moins probable. Il s'agit ici d'un effet de bord, le segment IBD s'arrêtant un peu plus loin. Par contre, l'algorithme de Viterbi ne tient pas compte des petits pics de probabilité (notamment la fin du chromosome de l'individu 50). À titre de comparaison, on remarque également chez les chevaux

consanguins (figure 18c) des chromosomes presque entièrement consanguins mais les segments sont globalement moins nombreux. Chez les BBB par contre (figure 18b) on ne voit presque pas de segments IBD.

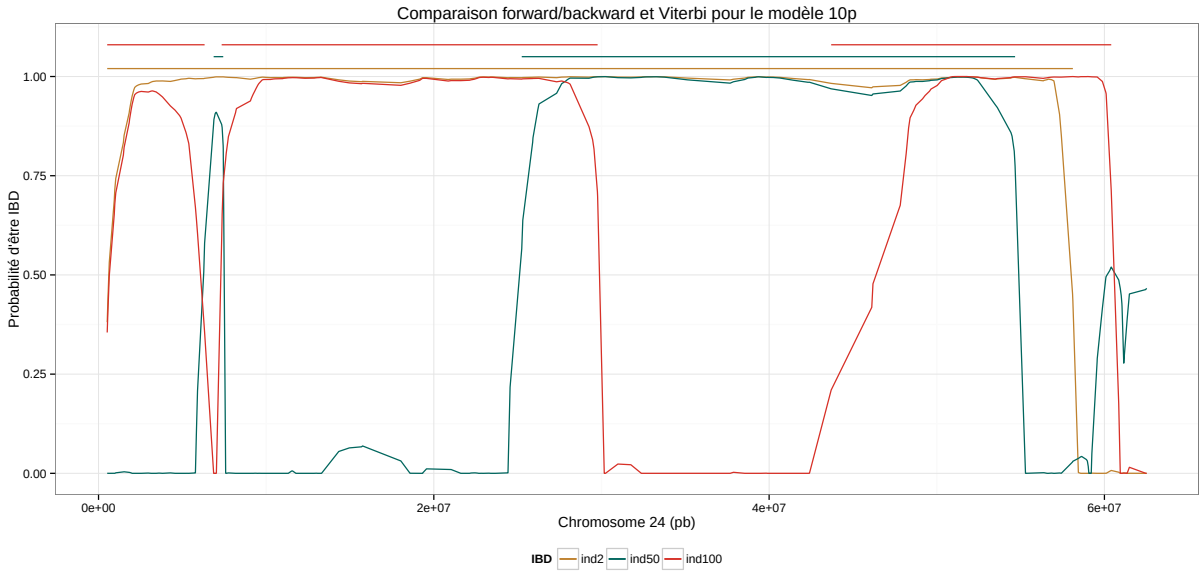
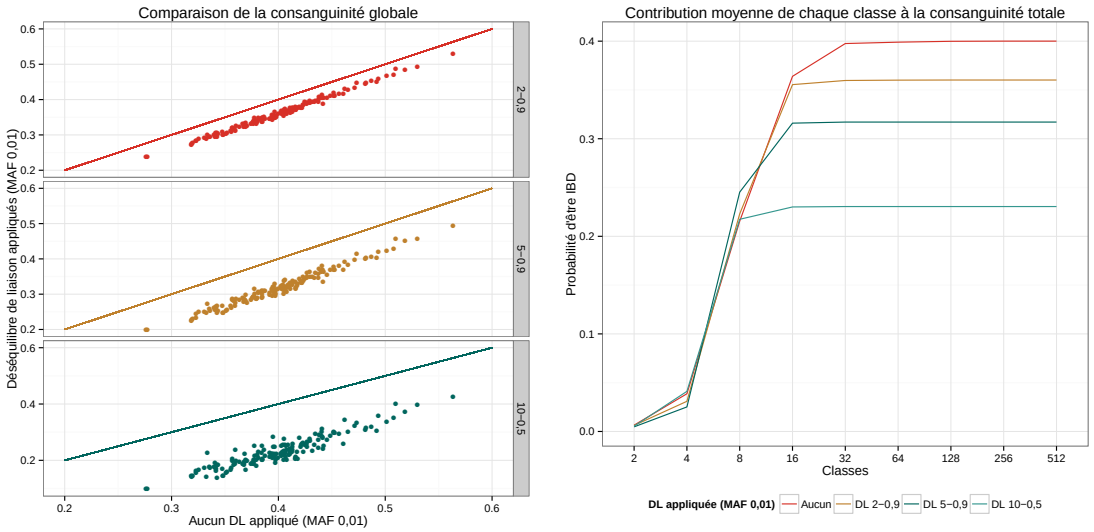


FIG. 19 : Analyse de la consanguinité locale des individus 2, 50 et 100. Comparaison de la probabilité d'être IBD (*forward*/*backward*) et des segments obtenus avec l'algorithme de Viterbi pour le modèle 10p pour 3 individus. L'individu 2 montre une grande probabilité d'être IBD sur presque tout le chromosome (ici le 24) bien associé à un segment détecté par Viterbi. Les segments obtenus pour les indivius 50 et 100 sont plus courts mais tout de même associés à de fortes probabilités.

L'impact du déséquilibre de liaison a ensuite été analysé via le programme `PLINK`. Lorsque le nombre de marqueurs retiré selon les paramètres fixés est analysé, on remarque que beaucoup de marqueurs sont éliminés très rapidement (tableau 3). La consanguinité a ensuite été à nouveau calculée sur ces nouveaux jeux de données et il a été constaté que la consanguinité totale estimée est moindre lorsqu'on retire des marqueurs (en accord avec l'étude sur la densité de marqueurs réalisée précédemment) mais que les corrélations restent bonnes (figure 20a). En observant la contribution des classes à la consanguinité totale (20b), on remarque que jusqu'à la classe 8, toutes les estimations sont les mêmes, quel que soit le critère de DL utilisé. À partir de la classe 16, des différences sont observées pour les critères retirant le plus de marqueurs et à partir de la 32, le critère retirant le moins de marqueurs estime à son tour une consanguinité plus faible que sur le jeu de marqueurs sans DL appliqué. Ainsi, l'impact des critères de DL, appliqué sur le jeu de données, sur l'estimation de la consanguinité globale est le même que lorsqu'on compare des jeux de données à différentes densités (section 4.1.3). Ceci n'est pas étonnant si l'on observe la figure 21. En effet, on remarque que la distance entre SNPs a tendance à diminer lorsqu'on ajoute de plus grandes contraintes sur le DL.



(a) Consanguinité totale. (b) Contribution moyenne des classes.

FIG. 20 : Analyse de la consanguinité globale selon les critères de DL appliqués. **(20a)** Comparaison de la consanguinité totale en ayant retiré des marqueurs en déséquilibre de liaison (MAF = *0,01*). Les consanguinités estimées sont moindres. Les corrélations sont de *0,9953*, *0,9836* et *0,9304*. **(20b)** Courbe cumulative de la contribution moyenne de chaque classe à la consanguinité totale pour une MAF de 0,01 et différents critères de DL.

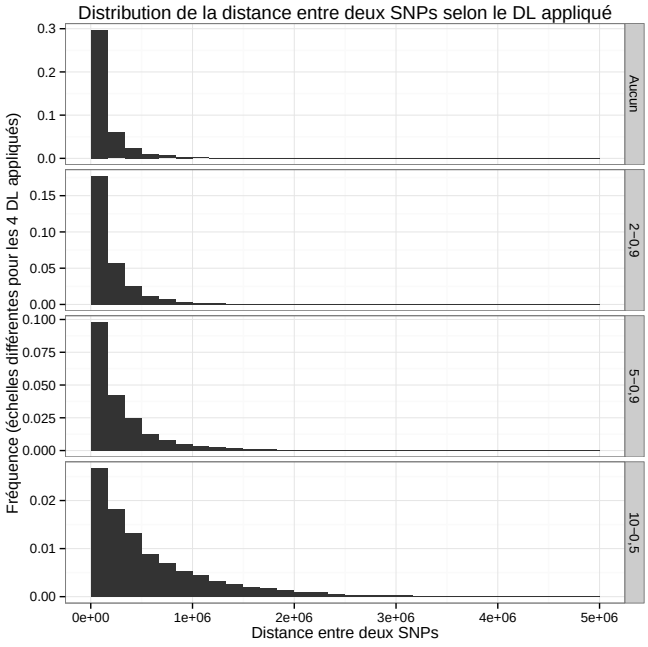
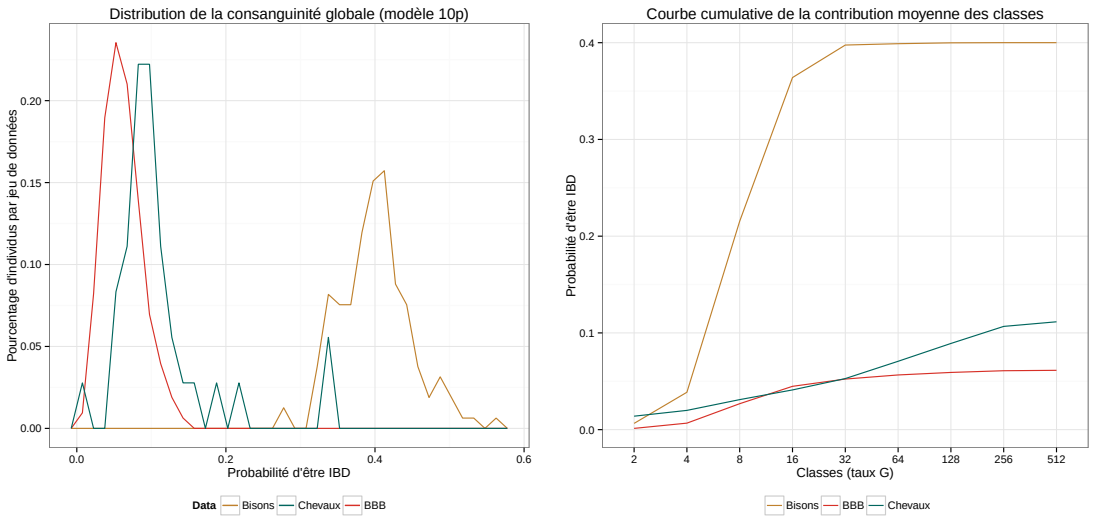


FIG. 21 : Distribution des distances entre SNPs selon le DL appliqué (ici pour une MAF de *0,01*).

4.2.3 Distribution de la consanguinité globale

La distribution de la consanguinité totale a été examinée dans la population de bisons. On remarque (figure 22a) qu'elle est relativement homogène sur tous les individus avec un seul pic à environ 40%. Cette consanguinité est très élevée. À titre de comparaison, la consanguinité

trouvée chez les BBB présente un pic à 6%. De plus, le maximum de consanguinité chez les chevaux est à 34% et concerne les individus UKH08 et UKH21 dans le pédigree (c'est-à-dire les accouplements père-fille et mère-fils qui avaient une consanguinité attendue de 25%). Ainsi, les bisons sont plus consanguins qu'un accouplement parent-enfant. Cependant, la figure 22b montre que cette consanguinité est répartie différemment dans ces différentes populations animales. En effet, la consanguinité des bisons montre une grande augmentation entre les classes 4 et 16 (et donc une grande proportion de sa consanguinité qui proviendrait de ces classes) alors que celle des chevaux montre une proportion plus ou moins bien répartie jusqu'à la classe 256. Plus de précision sur cette répartition dans les classes peut se voir à la figure 23a. On observe bien que la consanguinité des bisons se répartit principalement dans les classes 8 et 16 alors que chez les deux autres espèces on ne voit pas de classe prédominante.

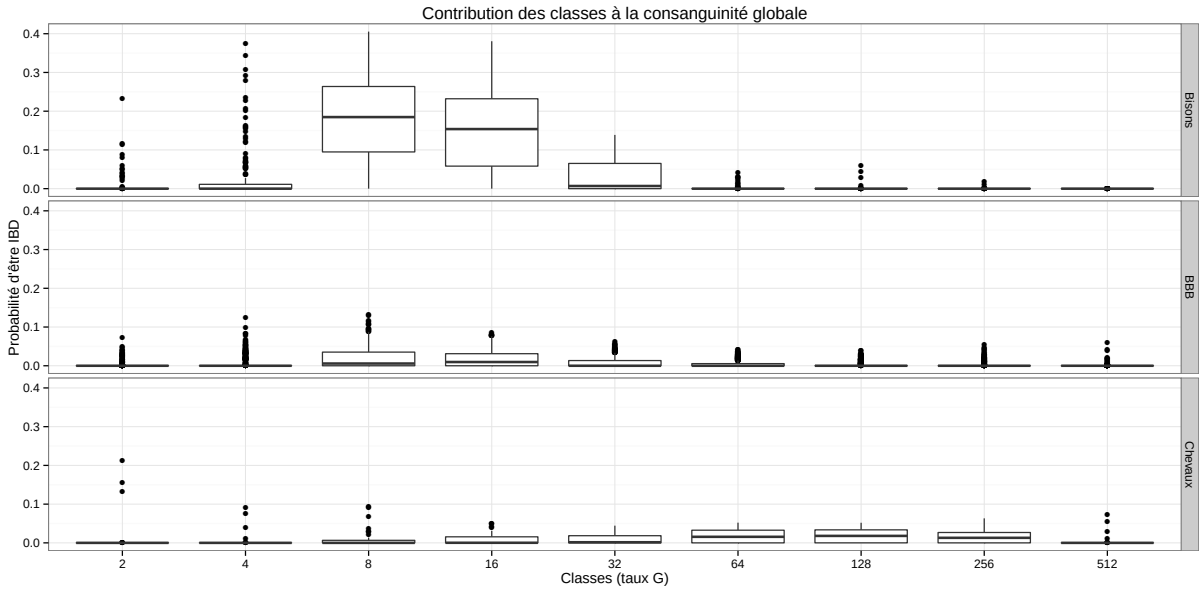


(a) Distribution de la consanguinité totale. (b) Contribution des classes.

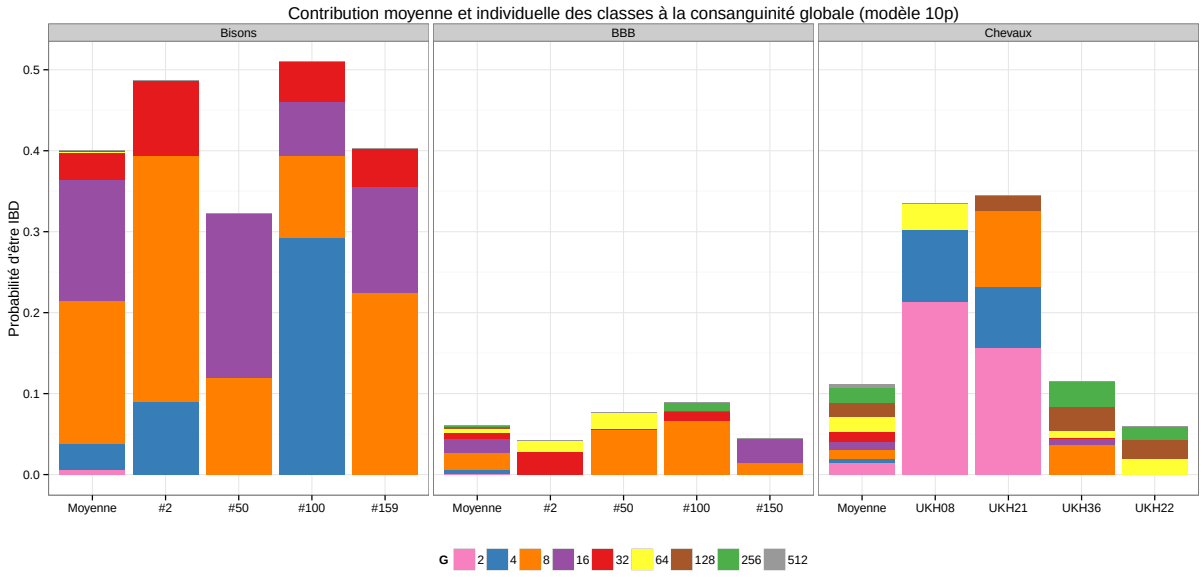
FIG. 22 : Niveau de la consanguinité totale. **(22a)** Distribution de la consanguinité totale dans la population chez les bisons, les BBB (LD) et les chevaux. **(22b)** Courbe cumulative de la contribution de chaque classe en moyenne dans la population à la consanguinité totale chez les bisons, les BBB (LD) et les chevaux.

La consanguinité individuelle de certains bisons, ici pris au hasard, a également été observée. À la figure 23b, sur les quatre bisons, on remarque une grande proportion de consanguinité assez récente : les classes 4, 8 et 16 y sont fortement représentées. La classe 32 est également importante. Comparativement à la consanguinité que l'on peut retrouver dans les deux autres espèces, on voit que la consanguinité est élevée. En effet, les BBB (à la densité LD pour pouvoir comparer à une densité en marqueurs similaires) sont beaucoup moins consanguins et les contributions des classes semblent plus variées d'un individu à l'autre. Si l'on compare maintenant le niveau de consanguinité des bisons avec celui d'invidus fortement consanguins (car résultant d'accouplements parents-enfants : UKH08 et UKH21), avec une densité en marqueurs plus élevée, on remarque que la consanguinité reste plus faible. Cependant, la proportion de la classe 2 est

très grande comparativement aux autres individus où elle est inexistante. Ainsi, la consanguinité chez les bisons est plus importante que lors d'un accouplement parents-enfants mais elle ne se répartit pas de la même façon. En effet, la consanguinité est plus récente lors d'un croisement parents-enfants que chez les bisons où elle se répartit dans des classes un peu plus anciennes.



(a) Distribution des classes dans les trois populations.



(b) Proportion des classes dans des individus provenant des trois populations.

FIG. 23 : Contribution de chaque classe à la consanguinité totale. **(23a)** De haut en bas : Bisons, BBB (LD) et Chevaux. Distribution des classes dans ces trois populations. **(23b)** De gauche à droite : Bisons, BBB (LD) et Chevaux. La contribution moyenne des classes dans chaque population (Moyenne) est représentée ainsi que la contribution des classes dans 4 individus. Chez les bisons, les 2, 50, 100 et 159. Chez les BBB, les 2, 50, 100 et 150. Chez les chevaux, UKH08 et UKH21 résultent d'accouplements père-fille et mère-fils. UKH36 a une consanguinité attendue de *0,03*. UKH22 est la fondatrice du pedigree.

4.2.4 Évolution de la consanguinité au fil des ans

Suite à ces analyses, une question supplémentaire s'est posée. Est-il possible, avec cette version de `ZooRoH` sous R, de détecter une diminution de la consanguinité récente chez les bisons nés plus récemment par rapport aux bisons plus âgés ? Pour répondre à cette question, la population a été divisée en deux : les bisons nés avant 1995 d'une part et ceux nés après 2005 d'autre part. Le choix de ne pas prendre une même date pour séparer les animaux nés avant de ceux nés après repose sur l'hypothèse que la longueur des segments consanguins attendus est souvent une moyenne approximative de ce que l'on trouve dans la nature. En effet, comme expliqué plus haut, un frère et une sœur n'auront pas exactement la même consanguinité réalisée. Il y a donc une certaine continuité dans la distribution de la consanguinité entre deux générations successives.

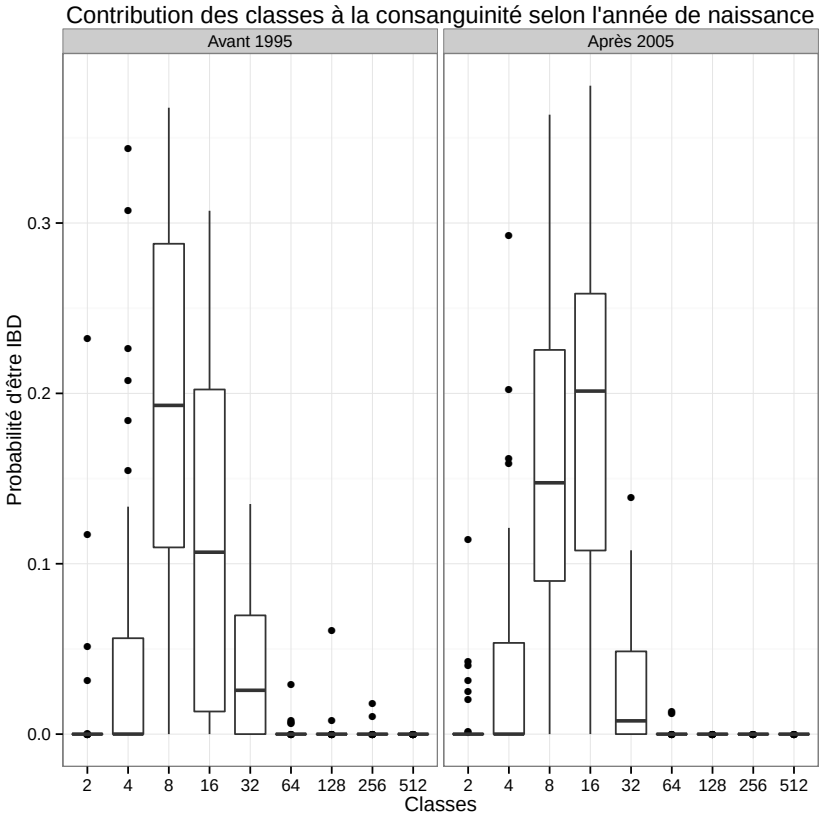


FIG. 24 : Répartition de la consanguinité dans chaque classe pour les bisons né avant 1995 et après 2005. Un déplacement est observé entre les classes 16 et 32 entre les bisons nés avant 1995 et ceux nés après 2005.

Lorsque l'on observe la répartition de la consanguinité dans chaque classe (figure 24), on remarque un déplacement entre les classes 8 et 16. En effet, une plus grande proportion de la consanguinité est associée au taux 8 pour les animaux nés avant 1995 alors qu'il s'agit du taux 16 pour les individus nés après 2005. Ceci illustre une consanguinité devenant plus ancienne dans le génome des animaux plus jeunes.

5 Discussion

Les résultats obtenus dans ce mémoire ont permis à la fois d'optimiser une technique existante de mesure de la consanguinité réalisée individuelle et de caractériser la consanguinité d'une espèce réintroduite après son extinction. En effet, les bisons d'Europe ont subi un important *bottleneck* entraînant une consanguinité élevée : seulement 12 animaux ont contribué à la restauration de la population et ce n'est que récemment que l'IUCN a relevé leur niveau de "en voie de disparition" à "menacé". L'étude de leur consanguinité actuelle permet ainsi de mettre en évidence l'efficacité du programme de conservation mis en place entre les deux guerres. Étant donné que la population est passée à travers un *bottleneck* majeur (suivi d'un second), un modèle qui estime le coefficient de consanguinité tout en déterminant quelles générations d'ancêtres ont contribué à celui-ci semble approprié.

Dans ce cadre, le programme `ZooRoH` a été porté sous R. Ceci amène la possibilité d'utiliser de nouvelles méthodes d'optimisation comme le L-BFGS-B utilisé dans d'autres programmes (notamment `BCFtools/ROH`[37] et `ngsF-HMM`[38]). Cette méthode peut éventuellement obtenir une meilleure estimation des paramètres au niveau du nombre d'itérations ou de la vraisemblance ou encore permettre de réaliser moins de calculs (*forward* par rapport au *forward/backward*). Il s'agit également d'une occasion de vérifier si l'EM de `ZooRoH` converge bien, ce qui n'a pas été étudié auparavant. Le portage permet aussi de reparamétriser pour s'assurer que les propriétés des paramètres estimés soient corrects tout en ordonnant les classes, ce que ne permet pas l'EM. On peut également paralléliser plus facilement sur R qu'avec le fortran. De plus, cet environnement est plus accessible et convivial et pourrait permettre dans le futur d'ajouter des options graphiques.

5.1 Portage de `ZooRoH` sous R

Concernant la réalisation du premier objectif de ce mémoire, divers éléments ont été mis en avant dans les résultats. Cependant, certaines comparaisons n'ont pas été montrées, notamment celles avec ou sans la parallélisation, car les résultats étaient identiques. Il faut souligner que le fait de réaliser l'estimation de plusieurs animaux en parallèle permet de réduire considérablement le temps pendant lequel le programme tourne, et ce, de façon automatique. *A contrario*, les fonctions d'origine (*forward/backward*, Viterbi) ont été gardées en fortran car leur implémentation sous R augmente considérablement le temps de calcul.

5.1.1 Méthode d'estimation des paramètres et reparamétrisation

Une reparamétrisation a permis d'obtenir automatiquement des paramètres avec les propriétés souhaitées ($0 < M_i < 1$ avec $\sum M_i = 1$ et $G_i > 0$) ainsi que de les ordonner. Pour ce faire, elle exprime les nouveaux paramètres sur une nouvelle échelle en fonction de leurs valeurs d'origine. L'ordination permet automatiquement une bonne lisibilité des résultats sans pour autant

influencer l'estimation de la consanguinité globale. Sans la reparamétrisation, il est nécessaire de mettre des contraintes à la fonction `optim` (méthode d'estimation des paramètres L-BFGS-B) pour qu'elle ne s'arrête pas systématiquement à cause de problèmes numériques mais des valeurs aberrantes sont toutefois obtenues.

`Optim` s'est révélé être un meilleur choix qu'une autre fonction du package "stats" permettant l'estimation des paramètres. En effet, lors de tests réalisés avec la fonction `nlm` (pour Non-Linear Minimization[51]), celle-ci s'arrêtait également dû à des problèmes numériques (dans ce cas-ci, même avec la reparamétrisation). Cette fonction utilise également un algorithme de type Newton. Un autre désavantage de cette méthode était qu'elle ne permet pas d'imposer des contraintes sur les paramètres à estimer. Ce phénomène d'arrêt fréquent du programme était d'autant plus marqué que la densité de marqueurs était faible. Étant donné que le jeu de données des bisons contenait peu de marqueurs, cette fonction `nlm` a été abandonnée complètement.

Après la mise au point de la reparamétrisation, l'estimation des paramètres par la méthode L-BFGS-B a été comparée à l'EM de la version originale de `ZooRoH`. Celui-ci est implémenté pour réaliser un nombre d'itérations choisi. Le portage du programme sur R a donc été l'occasion d'étudier les propriétés de convergence de l'EM. Il a été montré que 1000 itérations étaient suffisantes pour trois des quatre modèles appliqués lors de l'évaluation de ce portage. Même s'il nécessite moins d'itérations pour converger que la méthode L-BFGS-B pour les modèles 4p et 10p, deux éléments doivent être pris en compte. Le premier est qu'une itération de l'EM comprend deux fois plus de calculs qu'une itération par la fonction `optim` étant donné qu'on a une combinaison *forward/backward* dans le premier cas contre une procédure *forward* seulement dans le second. Ensuite, même si l'EM allait effectivement plus vite que la méthode L-BFGS-B, il faudrait changer le critère d'arrêt pour intégrer des critères de convergence dans l'EM sur les différences entre paramètres ou entre vraisemblances successives.

En résumé, nous avons mis en évidence que les paramètres estimés par les deux méthodes étaient similaires tout comme les vraisemblances obtenues dans la majorité des cas. La méthode L-BFGS-B semble donc une bonne alternive à l'EM, permettant le plus souvent une convergence plus rapide en termes d'itérations et donc une réduction du temps de calcul. De plus, la reparamétrisation apporte robustesse et lisibilité aux résultats. Il serait cependant intéressant de réaliser des mesures de performance plus systématiques même si la parallélisation permet de réduire considérablement le temps de travail. De plus, il s'agit d'un premier pas vers la formation d'un package R pouvant, peut-être, générer automatiquement des graphiques des résultats d'intérêt.

5.1.2 Estimation de la consanguinité

Tout comme les paramètres estimés entre les deux versions du programme `ZooRoH`, l'estimation de la consanguinité globale ne montre une différence que pour le modèle 4e, qui n'a pas encore convergé avec l'EM. En effet, les paramètres n'étant pas optimaux, la consanguinité estimée

résultante ne l'est pas non plus. Les contributions de chaque classe autozygote (IBD) à cette consanguinité sont semblables pour les autres modèles. De plus, le coefficient de consanguinité estimé à travers toutes les classes est identique selon les méthodes, que ce soit globalement ou localement.

Les consanguinités estimées sur le jeu de données des chevaux sont plus élevées que la consanguinité attendue par leur pedigree (environ *0,335*, *0,344* et *0,115* pour UKH08, UKH21 et UKH36 pour lesquels on attend *0,25*, *0,25* et *0,03*, section 3.3.1.2). La fondatrice du pedigree (UKH22, figure S.1) a une consanguinité globale estimée de *0,060*. Ceci illustre qu'un pedigree ne capture que la fraction la plus récente de la consanguinité. Dans le cas présent, le pedigree est particulièrement réduit car seuls les individus génotypés nécessaires à la cartographie de la maladie ont été conservés. En termes de comparaison de paramètres, les mêmes tendances ont été observées sur le pedigree équin que sur la population BBB.

Lors de ce mémoire, nous avons également comparé l'estimation du coefficient de consanguinité local avec la combinaison *forward/backward* (comme dans `ZooRoH` et `FEstim`[31]) et avec l'algorithme de Viterbi (comme `ngsF-HMM`[38] et `BCFTools`[37]). En effet, la combinaison des algorithmes *forward/backward* permet d'obtenir la probabilité à laquelle le SNP est IBD. Elle offre donc une caractérisation plus fine que l'algorithme de Viterbi qui assigne à chaque SNP une classe IBD ou non IBD et permet donc l'obtention de segments consanguins (et non consanguins). Bien que l'algorithme de Viterbi était légèrement plus efficace pour identifier les segments IBD relativement longs (donc avec un taux G_i petit), l'approche par le *forward/backward* est plus précise pour estimer le coefficient de consanguinité global et se comporte mieux lorsque le nombre de SNPs par segment IBD diminue (c'est-à-dire avec une consanguinité plus ancienne ou une densité de marqueurs réduite).

Le nombre de marqueurs est d'ailleurs réduit dans le jeu de données des bisons (un peu plus de 20000). La réduction de la densité en marqueurs utilisée a un effet important sur le coefficient de consanguinité estimée (sa valeur absolue) mais l'on observe des corrélations élevées avec des coefficients estimés avec plus de marqueurs. De plus, l'évaluation de la consanguinité est également un peu influencée par le nombre de classes choisi. En effet, le modèle 2e la sous-estime légèrement. Les autres semblent équivalents, en particulier le 4e et le 10p. Cependant, un nombre de classes plus élevé permettra d'obtenir une estimation plus fine de l'âge (nombre de générations jusqu'à l'ancêtre commun) des segments. C'est pourquoi le modèle 10p a été utilisé pour étudier les bisons. En effet, les modèles Qp présentent plusieurs avantages sur les modèles Qe, parmi lesquels une bonne convergence et une comparaison plus aisée entre individus.

5.2 Étude de la consanguinité chez les bisons d'Europe

Après avoir éliminé des SNPs mal génotypés, la population des bisons échantillonnés dans le jeu de données est relativement homogène. Ainsi, les bisons provenant de la partie polonaise de la

forêt de Bialowieza ne paraissent pas avoir de pattern particulier par rapport à ceux qui n'y vivent pas. De plus, il ne semble pas y avoir non plus de subdivision de la population à l'intérieur de ce territoire. Le jeu de données a donc l'air de ne contenir que des bisons de la lignée "Lowland", sans "contamination" par la ligne hybride "Lowland-Caucasian".

Le faible nombre de SNP compris dans ces données (après filtration) montre que les puces bovines ne sont pas adaptées à l'analyse du génome de ces animaux. En effet, seulement 20K des 770K sont utilisables. Ceci n'est pas surprenant : le bison d'Europe (*Bison bonasus*) descendrait d'un croisement secondaire ayant eu lieu il y a 120000 ans entre le bison des steppes (*Bison priscus*), maintenant éteint, et l'auroch (*Bos primigenius*), l'ancêtre des bovins d'élevage actuels[52]. Ainsi, le génome des bisons d'Europe ne contient qu'un peu plus de 10% du génome des aurochs. De plus, il ne serait pas inattendu que des SNPs se soient fixés dans le génome de ces bisons, notamment à la suite du large *bottleneck* subi il y a plus ou moins un siècle avec son extinction à l'état sauvage. Cependant, leur diversité génétique devrait être revenue à un niveau raisonnable, grâce à l'expansion de la population, comparé à celui des bovins d'élevage selon une étude réalisée en 2016[3]. Cette étude a également estimé la divergence de ces deux lignées animales. Elle aurait eu lieu il y a 1700000 à 850000 ans. Une autre étude de 2016[2] schématise également ce croisement secondaire, c'est-à-dire l'apport de gènes du genre *Bos* dans le genre *Bison*, ainsi que la divergence entre les deux lignées, même s'il n'y a pas de notion de temps (figure 25).

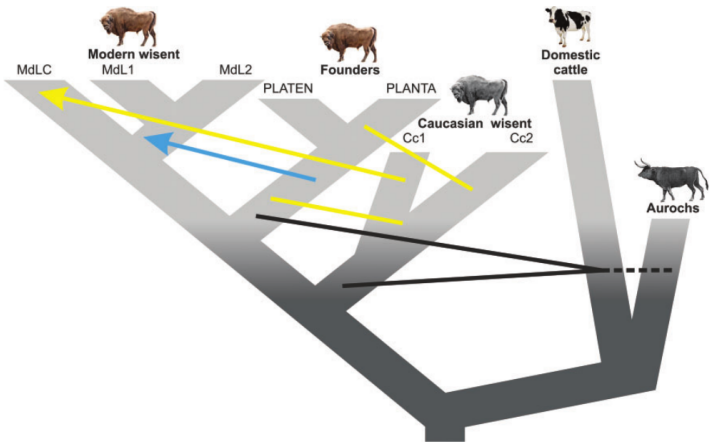


FIG. 25 : Croisements ayant eu lieu entre les différentes lignées de bisons, les aurochs et le bétail actuel[2]. Les lignes indiquent le flux des gènes. Les noires sont les croisements entre aurochs (ou bétail) et bisons. Les jaunes sont les croisements entre la lignée "Caucasian" et les fondateurs des "Lowland" ou "Lowland-Caucasian" actuels. La bleue représente le flux génique entre les fondateurs et la lignée actuelle "Lowland".

La consanguinité globale estimée dans la population étudiée est d'environ 40%. Bien que ce niveau soit extrêmement élevé, comme illustré dans les résultats, celui-ci est légèrement plus faible que les résultats rapportés dans la littérature : 44 ou 50% selon les sources[1,8]. La consanguinité étant estimée via des pedigrees, cette petite différence n'est pas surprenante. De plus, la faible densité de marqueurs disponibles ne permet pas de capturer toute la consanguinité. Lorsqu'on examine les contributions des classes à la consanguinité globale, la classe 8 domine de peu la

16. La classe 32 semble également plus importante que les autres. Ainsi, la consanguinité serait toujours assez récente. Puisque ces classes sont plus ou moins équivalentes à la taille des boucles de consanguinité en nombre de générations, cela correspond à des ancêtres présents il y a 4 à 16 générations (c'est-à-dire les G_i divisés par deux). Si l'on considère l'intervalle de temps estimé entre deux générations à 6 ans[1], ces ancêtres vivaient il y a 24 à 96 ans (classes 8 à 32) avec une majorité entre 24 et 48 ans (classes 8 et 16). Ces limites supérieures correspondent plus ou moins aux *bottlenecks* subis et représentent une consanguinité très récente. L'estimation de l'âge des segments IBD reste cependant qualitative (indicative) et le recouvrement des distributions des segments IBD pourrait rendre cette estimation plus difficile en sous-estimant la consanguinité plus ancienne.

En prenant en compte l'évolution de la consanguinité entre les animaux nés avant 1995 et ceux nés après 2005, le déplacement entre les classes 8 et 16 montre une légère diminution de la consanguinité récente. En d'autres termes, pour les individus actuels, l'ancêtre commun des segments IBD est plus éloigné que pour des animaux nés il y a 20 ans. Cela indique que l'apport de nouvelle consanguinité semble diminuer. Ainsi, le niveau de consanguinité des bisons baisse lentement même si la proportion consanguine du génome reste très élevée. Cette évolution de la consanguinité, ainsi que l'expansion du nombre d'individus vivant en liberté, suggère une réintroduction réussie. Cette espèce représente un bon exemple pour étudier le développement de programmes de conservation d'espèces en voie de disparition. Il faut souligner que lors de la mise en place du programme de conservation des bisons d'Europe, à la sortie de la première guerre mondiale, des efforts ont été réalisés pour enregistrer les pedigrees des animaux vivants dans les zoos. C'est ainsi que le EBPB (European Bison Pedigree Book) fut créé dans les années '30s[1]. Certainement grâce à ce programme, la lignée "Lowland" ne subit pas d'effet de dépression de consanguinité connu. Il se pourrait donc que, comme les gorilles des montagnes[28], les bisons aient subi une purge, une conséquence de la consanguinité menant à de meilleures chances de survie.

6 Conclusion

Au cours de ce mémoire, on a réussi à porter sous R le programme `ZooRoH`. Cela a permis d'utiliser de nouvelles procédures d'estimation des paramètres (meilleure vraisemblance, moins d'itérations, …). Il a toutefois été nécessaire de proposer une reparamétrisation originale du modèle, qui permet également d'ordonner les classes. Ce portage sous R rend la parallélisation de la méthode plus aisée et constitue un premier pas vers un package R. De la sorte, la méthode sera accessible à un plus grand nombre de chercheurs et des fonctionnalités supplémentaires pourront être intégrées (par exemple de visualisation des données).

D'autres aspects méthodologiques ont été abordés, comme la comparaison des algorithmes *forward/backward* et de Viterbi pour l'estimation du coefficient de consanguinité et l'identification des segments IBD. Ces résultats aideront au choix de la méthode selon les applications.

Le bison d'Europe illustre l'intérêt de la méthode qui permet d'obtenir une image de l'histoire démographique récente en estimant le coefficient de consanguinité. Son étude confirme que ce modèle permettra d'étudier des espèces dont l'histoire n'est pas connue.

La consanguinité estimée chez les bisons est extrêmement élevée, ce qui est la conséquence de leur extinction à l'état sauvage et des *bottlenecks* successifs subis. Ce coefficient (estimé à 40%) est beaucoup plus élevé que celui retrouvé chez les bovins ou encore chez les individus résultant d'un accouplement entre parents et enfants. Une légère diminution de la consanguinité récente est cependant observée. Cette observation permet, outre le nombre d'individus, de considérer leur réintroduction comme réussie et leur consanguinité comme stabilisée. De plus, aucune dépression de consanguinité n'est observée, suggérant qu'une purge des variants délétères a eu lieu.

7 Annexes

7.1 Comparaison des méthodes développées pour caractériser la consanguinité avec des HMMs

7.1.1 Pedigree des chevaux

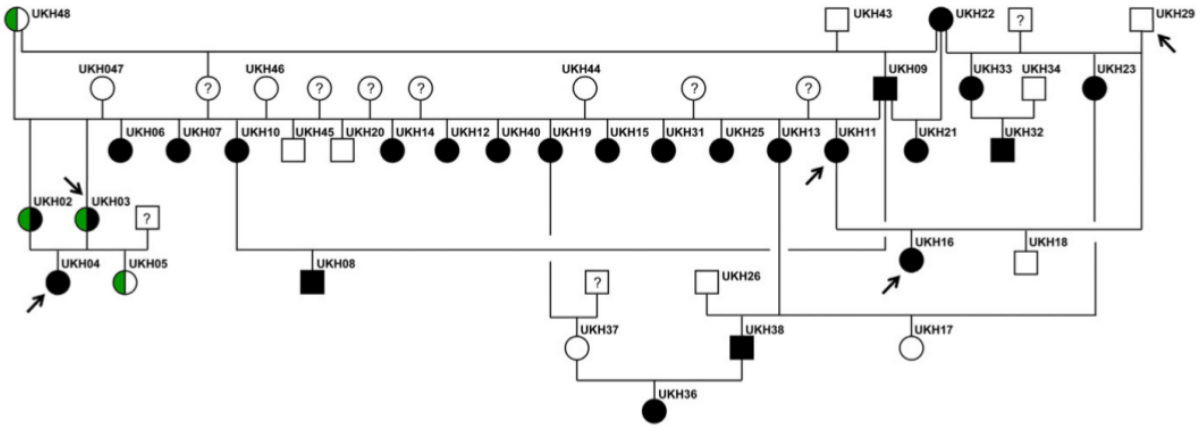
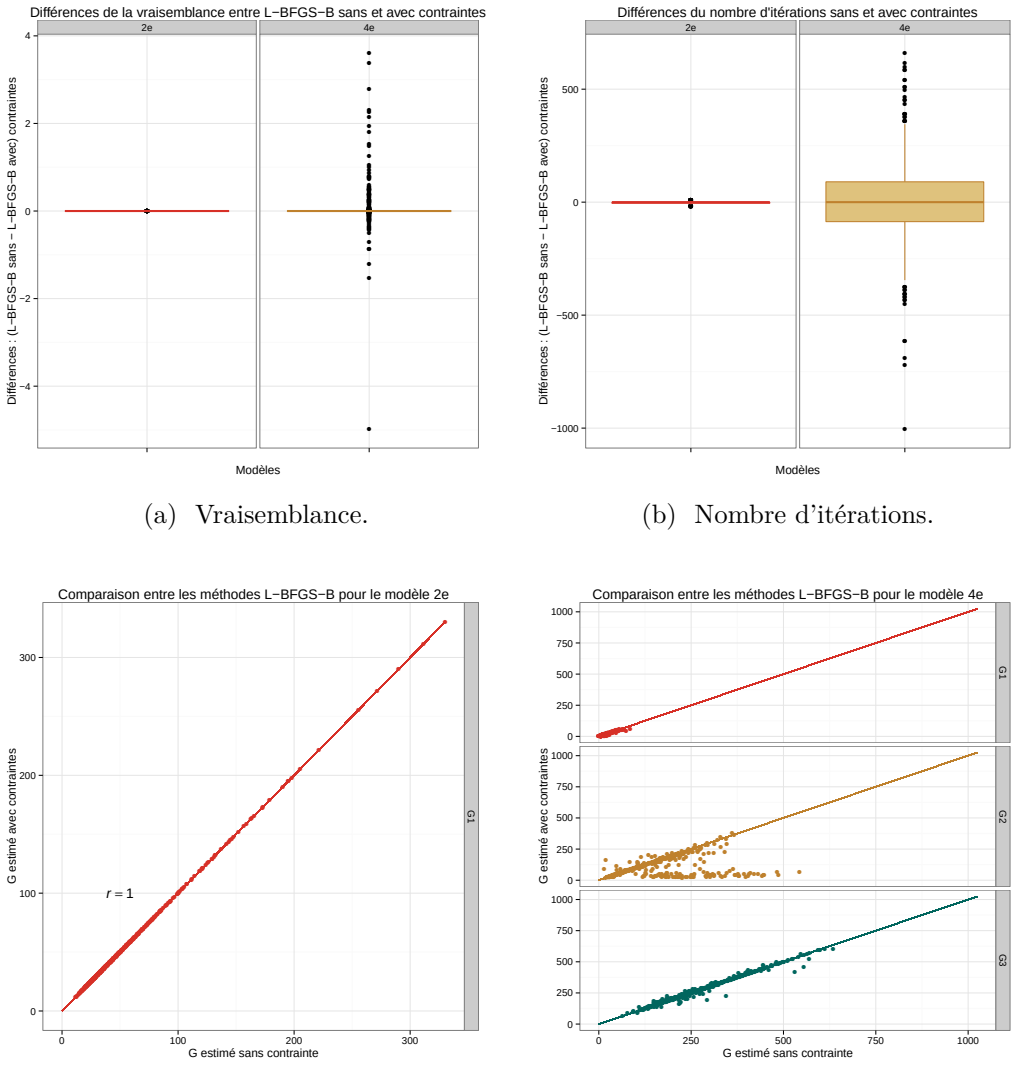


FIG. S.1 : Arbre généalogique des chevaux *BR1*[42].

7.1.2 Estimation des paramètres avec et sans contraintes pour L-BFGS-B



(a) Vraisemblance.

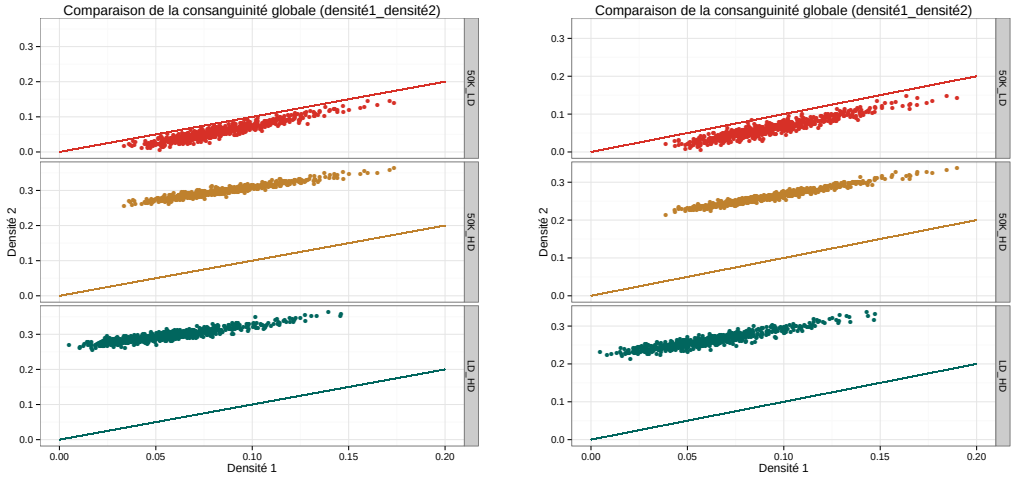
(b) Nombre d'itérations.

(c) Comparaison des taux G (modèle 2e).

(d) Comparaison des taux G (modèle 4e).

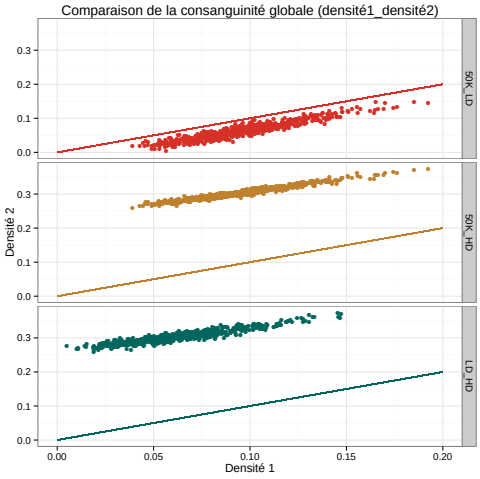
FIG. S.2 : Influences des contraintes sur L-BFGS-B. **(S.2a, S.2b)** Différences entre L-BFGS-B non contraint et L-BFGS-B avec contraintes sur les données BBB 50K pour les modèles 2e et 4e concernant la vraisemblance (S.2a) et le nombre d'itérations (S.2b). Les valeurs positives montrent un nombre plus grand sans contraintes tandis que les valeurs négatives montrent un nombre plus grand pour L-BFGS-B contraint. **(S.2c)** Comparaison de la consanguinité globale estimée avec L-BFGS-B avec et sans contraintes pour le modèle 2e. **(S.2d)** Comparaison des taux G estimés avec L-BFGS-B avec et sans contraintes chez les BBB 50K pour le modèle 4e. Les corrélations obtenues pour les trois classes IBD sont respectivement de *0,9624*, *0,4681* et *0,9923*.

7.1.3 Effet de la densité du jeu de données sur l'estimation de la consanguinité globale



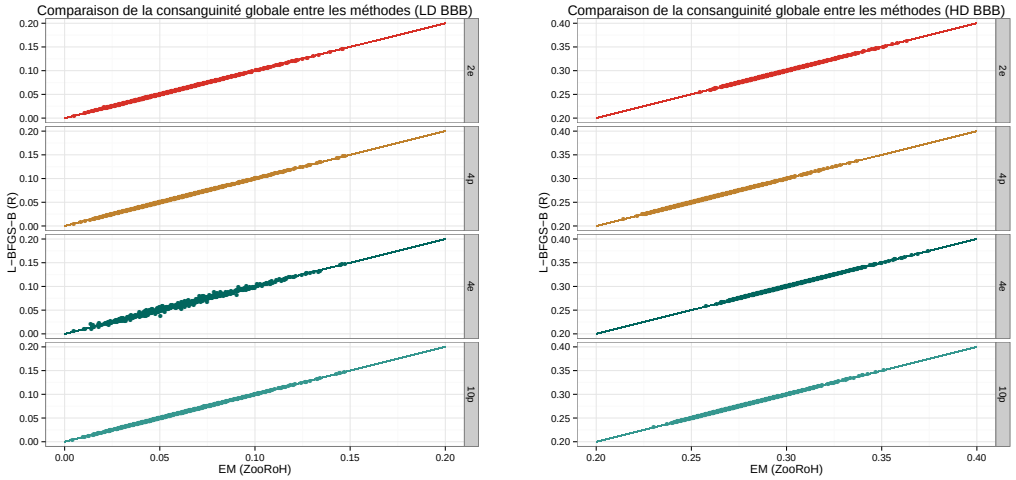
(a) Modèle 2e.

(b) Modèle 4p.



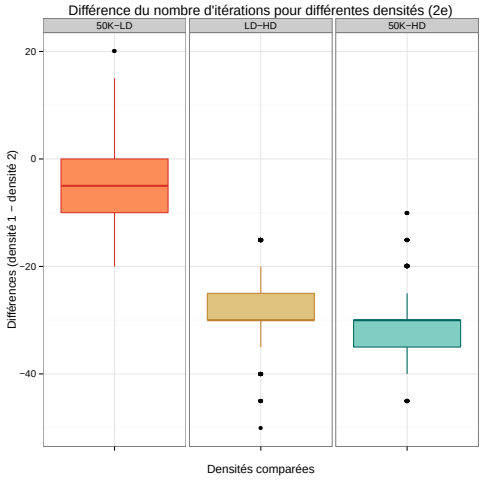
(c) Modèle 4e.

FIG. S.3 : Influence de la densité sur l'estimation de la consanguinité. Comparaison de la consanguinité globale entre les trois densités pour les modèles (S.3a) 2e, (S.3b) 4p et (S.3c) 4e.

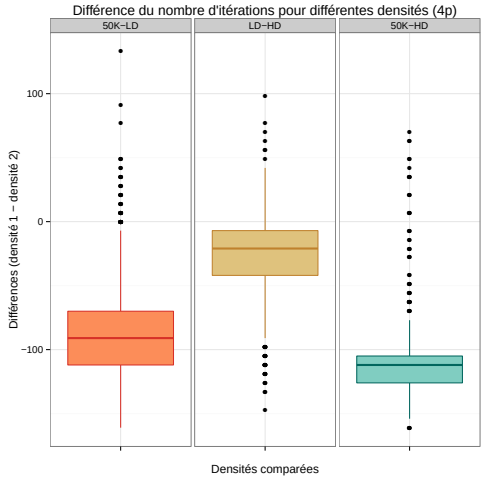


(a) Densité LD (b) Densité HD

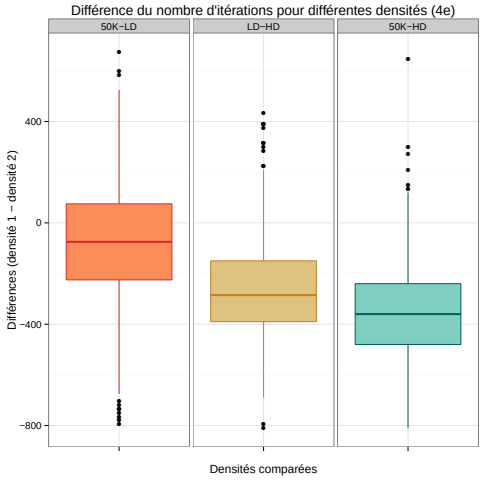
FIG. S.4 : Comparaison de la consanguinité globale entre les méthodes. **(S.4a)** Comparaison de la consanguinité globale entre les méthodes (densité LD). Corrélations : *1*, *1*, *0,9958* et *1*. **(S.4b)** Comparaison de la consanguinité globale entre les méthodes (densité HD). Les corrélations sont toutes égales à *1*.



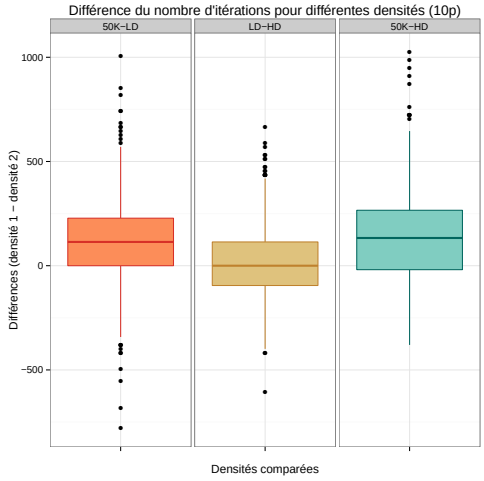
(a) Modèle 2e.



(b) Modèle 4p.



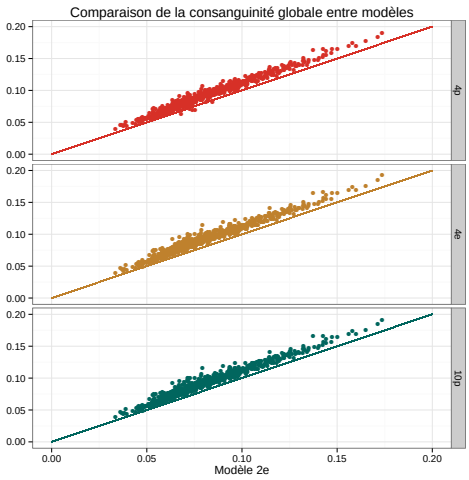
(c) Modèle 4e.



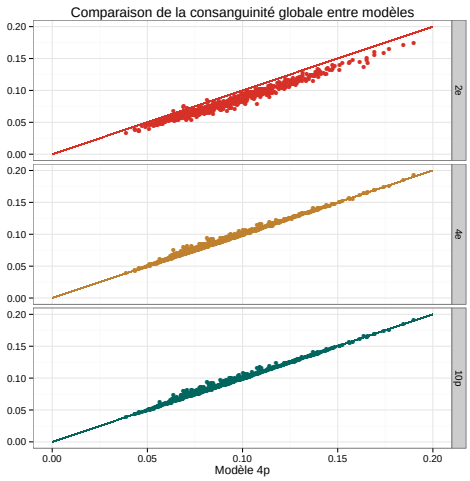
(d) Modèle 10p.

FIG. S.5 : Différence du nombre d'itérations entre les trois densités pour les 4 modèles.

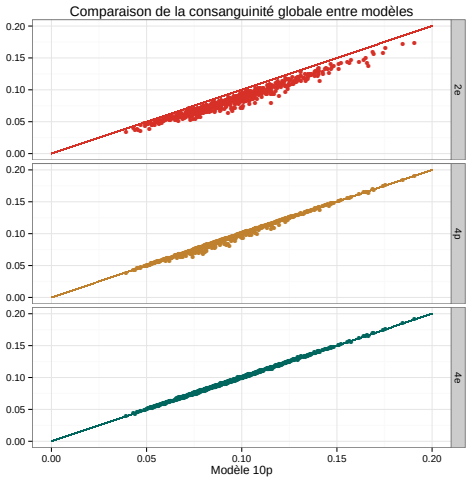
7.1.4 Effet des modèles sur l'estimation de la consanguinité globale



(a) Modèle 2e.



(b) Modèle 4p.



(c) Modèle 10p.

FIG. S.6 : Comparaison de la consanguinité globale **(S.6a)** entre le modèle 2e et les autres. Les corrélations sont de *0,9855*, *0,9818* et *0,9808*. **(S.6b)** entre le modèle 4p et les autres. Les corrélations sont de *0,9855*, *0,9969* et *0,9967*. **(S.6c)** entre le modèle 10p et les autres. Les corrélations sont de *0,9808*, *0,9967* et *0,9996*.

7.2 Étude de la consanguinité chez les bisons d'Europe

7.2.1 Structure de la population

7.2.1.1 Données

Individus faisant partie de la petite sous-population :

1, 7, 8, 11, 13, 14, 15, 17, 18, 19, 20, 21, 23, 24, 25, 29, 31, 42, 45, 46, 47, 48, 51, 53, 56, 58, 59, 60, 88, 98, 101, 103, 106, 107, 108, 114, 115, 122, 123, 126, 134

Répartition des animaux de la forêt de Bialowieza polonaise dans la petite sous-population.

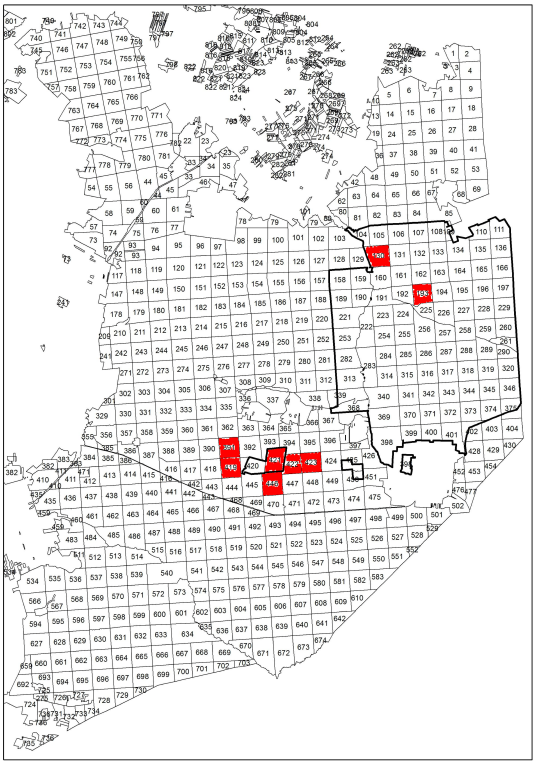


FIG. S.7 : Répartition des animaux de la forêt de Bialowieza polonaise dans la petite sous-population.

Dates de naissance des individus de la petite sous-population :

NA, 2002, 2003, NA, 2004, 2004, NA, 2000, 2002, 2002, 2001, 2001, 2005, 2000, 2004, 1999, 2000, 1997, 1997, 1984, 1989, 1997, 1991, 1997, 2002, 2005, 1997, 1994, 1995, NA, 2005, 1996, 2002, 2003, 2002, 2007, 2004, 2003, 2003, 2003, 2003

7.2.1.2 Résultats supplémentaires

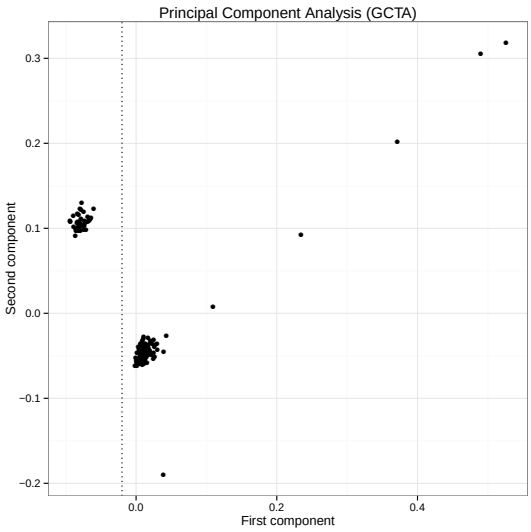
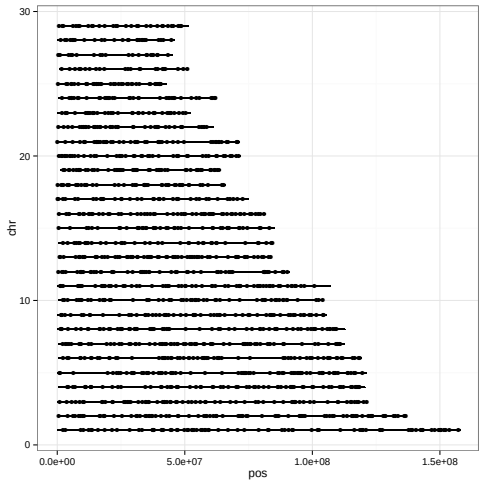
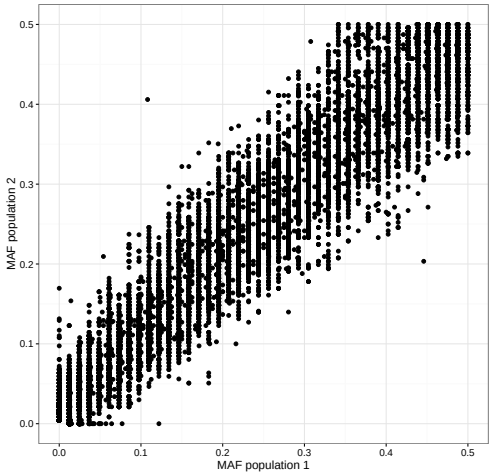


FIG. S.8 : Analyse PCA.

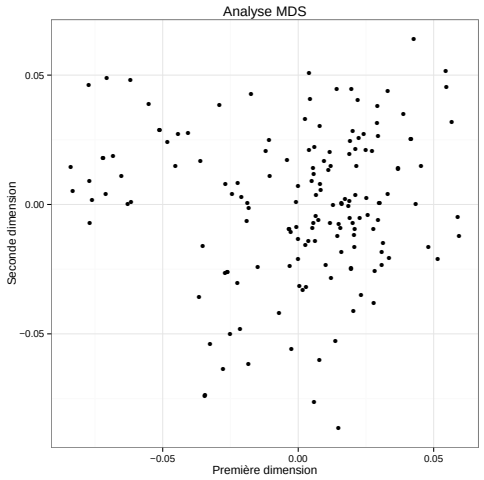


(a) Position des SNPs retirés.

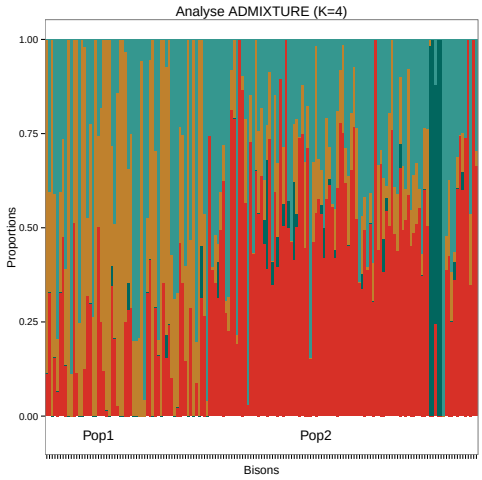


(b) Comparaison des MAFs sans les 1504 SNPs.

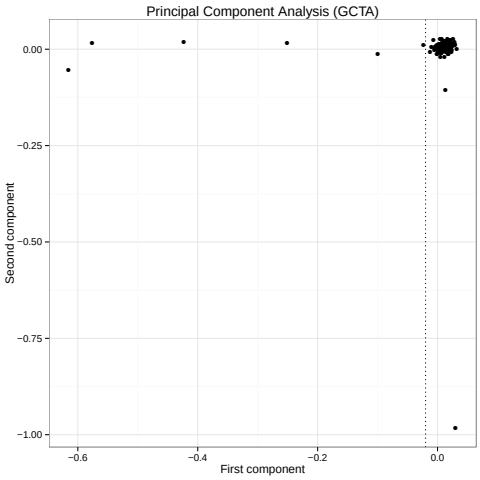
FIG. S.9 : 1504 SNPs retirés.



(a) Analyse MDS



(b) Analyse ADMIXTURE



(c) Analyse PCA

FIG. S.10 : Analyse structurale de la population. L'analyse MDS (S.10a) ne montre plus de séparation nette de la population. Lors de l'analyse avec ADMIXTURE (S.10b), les individus des deux sous-populations précédemment trouvées dans (15a) sont mis côte à côte. Le pattern n'est plus clairement visible entre les deux groupes. Il semble cependant y avoir quelques individus se distingant de la partie principale de la population comme le montre l'analyse PCA (S.10c).

7.3 Scripts Bash

7.3.1 Analyse de la structure de la population des bisons d'Europe

Commande utilisée pour l'analyse MDS du jeu de données des bisons avec `PLINK` :

```
p-link --noweb --file bbo --cluster --mds-plot 2 --cow
```

Commandes utilisées pour l'analyse du déséquilibre de liaison avec `PLINK`. Le deuxième paramètre a toujours été mis à 1. Comme expliqué plus haut, le premier a soit été fixé à 2, 5 ou 10 et le dernier soit à 0.9, 0.7 ou 0.5.

```
p-link --noweb --file bbo_less_v3 --indep-pairwise 2 1 0.9 --cow
p-link --noweb --file bbo_less_v3 --extract plink.prune.in --out bbo_less_v3_prune209 --recode --make-bed --cow
```

Commandes utilisées pour l'analyse PCA sur le jeu de données des bisons avec `GCTA` :

```
gcta64 --bfile bbo_less_v3 --make-grm --autosome --out bbo_less_v3_gcta
gcta64 --grm bbo_less_v3_gcta --pca 2 --out bbo_less_v3_gcta
```

Commandes utilisées pour l'analyse avec `ADMIXTURE` (avec cross-validation) :

```
for K in 1 2 3 4 5 6; do admixture --cv bbo_less_v3.bed $K | tee log${K}_less_v3.out; done
grep -h CV log*_less_v3.out | cut -d' :' -f2 | cut -d' ' -f2 > cv_log_out_val_less_v3.txt
grep -h CV log*_less_v3.out | cut -d'=' -f2 | cut -d')' -f1 > cv_log_out_k_less_v3.txt
paste cv_log_out_k_less_v3.txt cv_log_out_val_less_v3.txt > cv_log_out_tab_less_v3.txt
```

7.4 Scripts R

7.4.1 Modèle à deux classes

Programme `Optim_2_est_contr_repar_wz_paral.R`

```
dyn.load("zoolik.so")
dyn.load("zooFB.so")
library(tools)
library(foreach)
library(doMC)
registerDoMC(3)


allelefreq <- function(genoline) {
  sum(genoline[genoline<3]) / (2*length(genoline[genoline<3]))
}

getlik <- function(par,pem) {
  #print(paste("getlik début"))
  as=c(exp(par[1]),exp(par[1]))
  f=1/(1+exp(-par[2]))
  Fs=c(f, (1-f))
  countfct <<- countfct+1
  pemission <- pem
  oneiter <- .Fortran("zoolik",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
                      as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
                      as.integer(bp),as.double(0))
  loglik <- oneiter[11][[1]][1]
  #print(paste("getlik fin",loglik))
  return(-loglik)
}

fb <- function(out,pemission) {
  as=c(exp(out$par[1]),exp(out$par[1]))
  f=1/(1+exp(-out$par[2]))
  Fs=c(f, (1-f))
  onefb <- .Fortran("zooFB",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
                    as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
                    as.integer(bp),as.double(0),matrix(as.double(0),K,nsnps))
}

funpem <- function(freqs,genos,ID,ff){
  #print(paste("funpem",ID))
  geno <- genos[(freqs >= 0.01 & freqs <= 0.99), ID]

  #### COMPUTE EMISSION PROB.
  pemission<-matrix(0, length(ff), 2)

  pemission[geno==0,1]=(1-ff[geno==0])**2
  pemission[geno==1,1]=2*ff[geno==1]*(1-ff[geno==1])
  pemission[geno==2,1]=ff[geno==2]**2
  pemission[geno==9,1]=1.00

  gerr=0.001
  pemission[geno==0,2]=(1-ff[geno==0])*(1-gerr)
  pemission[geno==1,2]=gerr
  pemission[geno==2,2]=ff[geno==2]*(1-gerr)
  pemission[geno==9,2]=1.00

  #print(paste("funpem fin",ID))
  return(pemission)
}

indiv <- function(ID,pemission){
  #print(paste("indiv",ID))

  #### Number of iteration (optim)
  countfct <<- 0

```

```
#### Optimisation of the parameters via the likelihood (optim)
#print(paste("indiv avant optim getlik",ID))
out <- optim(par=par, getlik,pem=pemission, method="L-BFGS-B",lower=c(log(www),-Inf),
             upper=c(log(zzz),5), control = list(trace=TRUE))
#print(paste("indiv après optim getlik",ID))

#### Outfile : iterations
output_count <- matrix(0,1,2)

output_count[1,1] <- ID
output_count[1,2] <- countfct
#print(output_count)

write.table(output_count, file = paste0("Opt_2_est_contr_repar_paral_count_", name,"_",zzz, ".txt"),
            append=TRUE, sep = "\t",quote=F,col.names=FALSE)
#print("count")

#### Outfile : Likelihood and optimized parameters
r <- 1/(1+exp(-out$par[2]))
v <- out$value

output_optim_2_1_Fs <- matrix(0, 1, (2*K+3))

output_optim_2_1_Fs[1, 1] <- ID
output_optim_2_1_Fs[1, 2] <- round(v, 6)
output_optim_2_1_Fs[1, 3] <- round(exp(out$par[1]),4)
output_optim_2_1_Fs[1, 4] <- round(r,7)
output_optim_2_1_Fs[1, 5] <- round(exp(out$par[1]),4)
output_optim_2_1_Fs[1, 6] <- round((1-r),7)
output_optim_2_1_Fs[1, 7] <- countfct

write.table(output_optim_2_1_Fs, file = paste0("Opt_2_est_contr_repar_paral_out_", name,"_",zzz,".txt"),
            append=TRUE, sep = "\t",quote=F,col.names=FALSE)
#print("out")

#### Outfile : Age
output_age <- matrix(0,1,K+1)

output_age[1, 1] <- ID
output_age[1, 2] <- round(exp(out$par[1]),4)
output_age[1, 3] <- round(exp(out$par[1]),4)

write.table(output_age, file = paste0("Opt_2_est_contr_repar_paral_age_",name,"_",zzz,".txt"),
            append=TRUE,sep = "\t", quote = F,col.names=FALSE)
#print("age")

#### Outfile : mixing proportions
output_mixing <- matrix(0,1,K+1)

output_mixing[1, 1] <- ID
output_mixing[1, 2] <- round(r,7)
output_mixing[1, 3] <- round((1-r),7)

write.table(output_mixing, file = paste0("Opt_2_est_contr_repar_paral_mixing_",name,"_",zzz,".txt"),
            append=TRUE,sep = "\t", quote = F,col.names=FALSE)
#print("mixing")

#### Forward/backward to obtain the inbreeding
#print(paste("indiv avant fb",ID))
out2<-fb(out,pemission)
#print(paste("indiv après fb",ID))

#### Outfile : inbreeding per class and global inbreeding
FG<-matrix(0,1,3)

FG[1,1]<-ID
FG[1,2]<-mean(out2[[12]][1,])
FG[1,3]<-sum(FG[1,(2 :(ncol(FG)-1))])

write.table(FG, file = paste0("Opt_2_est_contr_repar_paral_inbreeding_", name,"_",zzz,".txt"),
            append=TRUE,sep = "\t",quote=F,col.names=FALSE)
#print("inbreeding")

#### Outfile : resume (likelihood, parameters, inbreeding, iterations)
```

```
  output_onefile <- matrix(0,1,6)

  output_onefile[1, 1] <- ID
  output_onefile[1, 2] <- round(v, 6)
  output_onefile[1, 3] <- round(exp(out$par[1]),4)
  output_onefile[1, 4] <- round(r,7)
  output_onefile[1, 5] <- FG[1,2]
  output_onefile[1, 6] <- countfct

  write.table(output_onefile, file = paste0("Opt_2_est_contr_repar_paral_",name,"_",zzz,".txt"),
              append=TRUE,sep = "\t",quote=F,col.names=FALSE)
  #print("tout")
}

#### Read the geno file (terminal) that must have 5 informative columns and the limits (www and zzz)
argv <- commandArgs(TRUE)
data <- read.table(argv[1])
name <- file_path_sans_ext(basename(argv[1]))
www <- as.numeric(argv[2])
zzz <- as.numeric(argv[3])

########################################################
### Compute informations from the data (geno file) ###
########################################################

#### Number of individuals and their informations for the markers
NIND = length(data[1,])-5
genos <- as.matrix(data[,6 :(NIND+5)])

#### Compute allelic frequencies
freqs <- apply(genos, 1, allelefreq)

#### Keep markers with MAF >= 0.01
chr = data$V1[(freqs >= 0.01 & freqs <= 0.99)]
bp = data$V3[(freqs >= 0.01 & freqs <= 0.99)]
ff = freqs[(freqs >= 0.01 & freqs <= 0.99)]

#### Number of chromosomes
nchr=max(chr)
#chrs=seq(1 :nchr)

#### Bounds of the chromosomes (line numbers = #marker)
chrbound=matrix(0, nchr, 2)
chrbound[1,1]=which.min((bp[chr == 1]))
chrbound[1,2]=which.max((bp[chr == 1]))

for (i in 2 :nchr) {
  chrbound[i,1]=which.min(bp[chr == i])+chrbound[(i-1),2]
  chrbound[i,2]=which.max(bp[chr == i])+chrbound[(i-1),2]
}

#### Total number of SNPs
nsnps=chrbound[nchr,2]

########################################################
###               Set the parameters                 ###
########################################################

#### Number of classes
K=2

#### Class IBD (1) or not (0)
isF <- c(replicate((K-1),1),0)

#### Matrice de transition
T=matrix(1,K,K)

#### Number of generation
as=c(50,50)

#### Mixing proportion
Fs=c(0.05, 0.95)

#### Re-parametrization
```

```
lf=log(0.05/(1-0.05))
par <- c(log(50),lf)

#########################################################
###           Compute for each individual          ###
#########################################################

countfct <- 0

#### Colnames of each file to write
write.table(matrix(c("ind", "iter"),1,2),
            file = paste0("Opt_2_est_contr_repar_paral_count_", name,"_",zzz, ".txt"),
            sep = "\t",quote=F,col.names=FALSE)
write.table(matrix(c("ind","likelihood", "age", "mixing","age", "mixing", "iter"),1,7),
            file = paste0("Opt_2_est_contr_repar_paral_out_", name,"_",zzz,".txt"),
            sep = "\t",quote=F,col.names=FALSE)
write.table(matrix(c("ind","age1","age2"),1,3),
            file = paste0("Opt_2_est_contr_repar_paral_age_",name,"_",zzz,".txt"),
            sep = "\t", quote = F,col.names=FALSE)
write.table(matrix(c("ind","mixing1","mixing2"),1,3),
            file = paste0("Opt_2_est_contr_repar_paral_mixing_",name,"_",zzz,".txt"),
            sep = "\t", quote = F,col.names=FALSE)
write.table(matrix(c("ind","G1","Gtot"),1,3),
            file = paste0("Opt_2_est_contr_repar_paral_inbreeding_", name,"_",zzz,".txt"),
            sep = "\t",quote=F,col.names=FALSE)
write.table(matrix(c("ind","likelihood", "age", "mixing","Gtot", "iter"),1,6),
            file = paste0("Opt_2_est_contr_repar_paral_",name,"_",zzz,".txt"),
            sep = "\t",quote=F,col.names=FALSE)

#### For each individual
foreach(ID = 1 :5) %dopar% {
#foreach(ID = 1 :NIND) %dopar% {
    print(paste("ind",ID))
    pemission <- funpem(freqs,genos,ID,ff)
    indiv(ID,pemission)
}
```

7.4.2 Modèle à plusieurs classes avec estimation des taux

Programme `Optim_4_est_contr_repar_ord_wz_paral.R`

Les modifications apportées à ce modèle étant éparpillées par rapport au précédent, il est ici retranscrit dans son entièreté. Cependant, les fonctions `allelefreq` et `funpem` sont strictement identiques ainsi que le début du corps du programme (lignes 153-192 pour le programme précédent et lignes 187-219 pour celui-ci).

Ce programme peut être adapté facilement à tout modèle à plusieurs classes avec estimation des taux. Il suffit, en plus de changer les noms des fichiers de sortie, de changer la ligne 225 (nombre de classe), la 234 (point de départ de l'estimation des taux G_i) et la 237 (point de départ de l'estimation des proportions de mélange M_i). Ces modifications pourront facilement être adaptables à un futur package R lors de sa création.

```
1  dyn.load("zoolik.so")
2  dyn.load("zooFB.so")
3  library(tools)
4  library(foreach)
5  library(doMC)
6  registerDoMC(3)
7
8
9  allelefreq <- function(genoline) {
10   sum(genoline[genoline<3]) / (2*length(genoline[genoline<3]))
11 }
12
13 getlik <- function(par,pem) {
14   #print(paste("getlik début"))
15   as=array(0,K)
16   as[1]=exp(par[1])
17   for (j in 2 :(K-1)){as[j]=as[j-1]+exp(par[j])}
18   as[K]=exp(par[K])
19
20   Fs=array(0,K)
21   D=1
22   for (j in 1 :(K-1)){D=D+exp(par[K+j])}
23   for (j in 1 :(K-1)){Fs[j]=exp(par[K+j])/D}
24   Fs[K]=1-sum(Fs[1 :(K-1)])
25   countfct <<- countfct+1
26   pemission <- pem
27   oneiter <- .Fortran("zoolik",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
28                       as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
29                       as.integer(bp),as.double(0))
30   loglik <- oneiter[11][[1]][1]
31   #print(paste("getlik fin",loglik))
32   return(-loglik)
33 }
34
35 fb <- function(out,pemission) {
36   as[1] <- exp(out$par[1])
37   for(j in 2 :(K-1)){as[j] <- as[j-1]+exp(out$par[j])}
38   as[K] <- exp(out$par[K])
39   Fs[1 :(K-1)] <- 1/(1+exp(-out$par[(K+1) :(K+K-1)]))
40   Fs[K]=1-sum(Fs[1 :(K-1)])
41
42   onefb <- .Fortran("zooFB",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
43                     as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
44                     as.integer(bp),as.double(0),matrix(as.double(0),K,nsnps))
45 }
46
47 funpem <- function(freqs,genos,ID,ff){
```

```
  #print(paste("funpem",ID))
  geno <- genos[(freqs >= 0.01 & freqs <= 0.99), ID]

  #### COMPUTE EMISSION PROB.
  pemission<-matrix(0, length(ff), 2)

  pemission[geno==0,1]=(1-ff[geno==0])**2
  pemission[geno==1,1]=2*ff[geno==1]*(1-ff[geno==1])
  pemission[geno==2,1]=ff[geno==2]**2
  pemission[geno==9,1]=1.00

  gerr=0.001
  pemission[geno==0,2]=(1-ff[geno==0])*(1-gerr)
  pemission[geno==1,2]=gerr
  pemission[geno==2,2]=ff[geno==2]*(1-gerr)
  pemission[geno==9,2]=1.00

  #print(paste("funpem fin",ID))
  return(pemission)
}

indiv <- function(ID,pemission){
  #print(paste("indiv",ID))

  #### Number of iteration (optim)
  countfct <<- 0

  #### Optimisation of the parameters via the likelihood (optim)
  #print(paste("indiv avant optim getlik",ID))
  out <- optim(par=par, getlik,pem=pemission, method="L-BFGS-B", lower=c(rep(log(www),K),rep(-Inf,K)),
               upper=c(rep(log(zzz),K),rep(+Inf,K)), control = list(trace=TRUE))
  #print(paste("indiv après optim getlik",ID))

  #### Outfile : iterations
  output_count <- matrix(0,1,2)

  output_count[1,1] <- ID
  output_count[1,2] <- countfct
  #print(output_count)

  write.table(output_count, file = paste0("Opt_4_est_contr_repar_ord_paral_count_", name,"_",zzz, ".txt"),
              append=TRUE, sep = "\t",quote=F,col.names=FALSE)
  #print("count")

  #### Outfile : Likelihood and optimized parameters
  r <- 1/(1+exp(-out$par[(K+1) :(K+K-1)]))
  v <- out$value
  G <- vector()
  G[1] <- exp(out$par[1])
  for(j in 2 :(K-1)){G[j] <- G[(j-1)]+exp(out$par[j])}
  G[K] <- exp(out$par[K])

  output_optim_4_1_Fs <- matrix(0, 1, (2*K+3))

  output_optim_4_1_Fs[1, 1] <- ID
  output_optim_4_1_Fs[1, 2] = round(v, 6)
  for(i in seq(3,(2*K+1),2)){
      output_optim_4_1_Fs[1, i] <- round(G[(i-1)/2],4)
  }
  for(i in seq(4,(2*K),2)){
      output_optim_4_1_Fs[1, i] <- round(r[(i-2)/2],7)
  }
  output_optim_4_1_Fs[1, (2*K+2)] <- round(1-sum(r[1 :(K-1)]),7)
  output_optim_4_1_Fs[1, (2*K+3)] <- countfct

  write.table(output_optim_4_1_Fs, file = paste0("Opt_4_est_contr_repar_ord_paral_out_", name,"_",zzz,".txt"),
              append=TRUE, sep = "\t",quote=F,col.names=FALSE)
  #print("out")

  #### Outfile : Age
  output_age <- matrix(0,1,K+1)

  output_age[1, 1] <- ID
  for(i in 1 :K){
```

```
122         output_age[1,(i+1)] <- round(G[i],4)
123     }
124 
125     write.table(output_age, file = paste0("Opt_4_est_contr_repar_ord_paral_age_",name,"_",zzz,".txt"),
126                 append=TRUE,sep = "\t", quote = F,col.names=FALSE)
127     #print("age")
128 
129     #### Outfile : mixing proportions
130     output_mixing <- matrix(0,1,K+1)
131 
132     output_mixing[1, 1] <- ID
133     for(i in 1 :(K-1)){
134         output_mixing[1,(i+1)] <- round(r[i],7)
135     }
136     output_mixing[1,(K+1)] <- round(1-sum(r[1 :(K-1)]),7)
137 
138     write.table(output_mixing, file = paste0("Opt_4_est_contr_repar_ord_paral_mixing_",name,"_",zzz,".txt"),
139                 append=TRUE,sep = "\t", quote = F,col.names=FALSE)
140     #print("mixing")
141 
142     #### Forward/backward to obtain the inbreeding
143     #print(paste("indiv avant fb",ID))
144     out2<-fb(out,pemission)
145     #print(paste("indiv après fb",ID))
146 
147     #### Outfile : inbreeding per class and global inbreeding
148     FG<-matrix(0,1,K+1)
149 
150     FG[1,1]<-ID
151     for(i in 1 :(K-1)){
152         FG[1,(i+1)] <- mean(out2[[12]][i,])
153     }
154     FG[1,(K+1)] <- sum(FG[1,(2 :K)])
155 
156     write.table(FG, file = paste0("Opt_4_est_contr_repar_ord_paral_inbreeding_", name,"_",zzz,".txt"),
157                 append=TRUE,sep = "\t",quote=F,col.names=FALSE)
158     #print("inbreeding")
159 
160     #### Outfile : resume (likelihood, parameters, inbreeding, iterations)
161     output_onefile <- matrix(0,1,(2*K+4))
162 
163     output_onefile[1, 1] <- ID
164     output_onefile[1, 2] <- round(v, 6)
165     for(i in seq(3,(2*K+1),2)){
166         output_onefile[1, i] <- round(G[(i-1)/2],4)
167     }
168     for(i in seq(4,(2*K),2)){
169         output_onefile[1, i] <- round(r[(i-2)/2],7)
170     }
171     output_onefile[1, (2*K+2)] <- round(1-sum(r[1 :(K-1)]),7)
172     output_onefile[1, (2*K+3)] <- FG[1,(K+1)]
173     output_onefile[1, (2*K+4)] <- countfct
174 
175     write.table(output_onefile, file = paste0("Opt_4_est_contr_repar_ord_paral_",name,"_",zzz,".txt"),
176                 append=TRUE,sep = "\t",quote=F,col.names=FALSE)
177     #print("tout")
178 }
179 
180 #### Read the geno file (terminal) that must have 5 informative columns and the limits (www and zzz)
181 argv <- commandArgs(TRUE)
182 data <- read.table(argv[1])
183 name <- file_path_sans_ext(basename(argv[1]))
184 www <- as.numeric(argv[2])
185 zzz <- as.numeric(argv[3])
186 
187 ########################################################
188 ### Compute informations from the data (geno file) ###
189 ########################################################
190 
191 #### Number of individuals and their informations for the markers
192 NIND = length(data[1,])-5
193 genos <- as.matrix(data[,6 :(NIND+5)])
194 
195 #### Compute allelic frequencies
```

```
freqs <- apply(genos, 1, allelefreq)

#### Keep markers with MAF >= 0.01
chr = data$V1[(freqs >= 0.01 & freqs <= 0.99)]
bp = data$V3[(freqs >= 0.01 & freqs <= 0.99)]
ff = freqs[(freqs >= 0.01 & freqs <= 0.99)]

#### Number of chromosomes
nchr=max(chr)
#chrs=seq(1 :nchr)

#### Bounds of the chromosomes (line numbers = #marker)
chrbound=matrix(0, nchr, 2)
chrbound[1,1]=which.min((bp[chr == 1]))
chrbound[1,2]=which.max((bp[chr == 1]))

for (i in 2 :nchr) {
  chrbound[i,1]=which.min(bp[chr == i])+chrbound[(i-1),2]
  chrbound[i,2]=which.max(bp[chr == i])+chrbound[(i-1),2]
}

#### Total number of SNPs
nsnps=chrbound[nchr,2]

#########################################################
###               Set the parameters                  ###
#########################################################

#### Number of classes
K=4

#### Class IBD (1) or not (0)
isF <- c(replicate((K-1),1),0)

#### Matrice de transition
T=matrix(1,K,K)

#### Number of generation
as=c(8,64,256,256)

#### Mixing proportion
Fs=c(0.02, 0.02, 0.02, 0.94)

#### Re-parametrization
par<-array(0,2*K-1)
par[1]=log(as[1])
for(j in 2 :(K-1)){par[j]=log(as[j]-as[j-1])}
par[K]=log(as[K])

for (j in 1 :(K-1)){par[K+j]=log(Fs[j]/Fs[K])}

#########################################################
###           Compute for each individual           ###
#########################################################

countfct <- 0

#### Colnames of each file to write
write.table(matrix(c("ind", "iter"),1,2),
            file = paste0("Opt_4_est_contr_repar_ord_paral_count_", name,"_",zzz, ".txt"),
            sep = "\t",quote=F,col.names=FALSE)
vecout <- vector()
for(i in seq(2,2*K,2)){
    vecout[(i-1)] <- paste0("age",((i-2)/2+1))
    vecout[i] <- paste0("mixing",((i-2)/2+1))
}
vecage <- vector()
vecmix <- vector()
for(i in 1 :K){
    vecage[i]<-paste0("age",i)
    vecmix[i]<-paste0("mixing",i)
}
vecinb <- vector()
for(i in 1 :(K-1)){vecinb[i] <- paste0("Inb",i)}
```

```
vecinb[K] <- "InbTot"
#print(vecout)
#vecout <- rep(c("age", "mixing"),K)
write.table(matrix(c("ind","likelihood", vecout, "iter"),1,(2*K+3)),
            file = paste0("Opt_4_est_contr_repar_ord_paral_out_", name,"_",zzz,".txt"),
            sep = "\t",quote=F,col.names=FALSE)
write.table(matrix(c("ind",vecage),1,(K+1)),
            file = paste0("Opt_4_est_contr_repar_ord_paral_age_",name,"_",zzz,".txt"),
            sep = "\t", quote = F,col.names=FALSE)
write.table(matrix(c("ind",vecmix),1,K+1),
            file = paste0("Opt_4_est_contr_repar_ord_paral_mixing_",name,"_",zzz,".txt"),
            sep = "\t", quote = F,col.names=FALSE)
write.table(matrix(c("ind",vecinb),1,K+1),
            file = paste0("Opt_4_est_contr_repar_ord_paral_inbreeding_", name,"_",zzz,".txt"),
            sep = "\t",quote=F,col.names=FALSE)
write.table(matrix(c("ind","likelihood", vecout,"InbTot", "iter"),1,(2*K+4)),
            file = paste0("Opt_4_est_contr_repar_ord_paral_",name,"_",zzz,".txt"),
            sep = "\t",quote=F,col.names=FALSE)

#### For each individual
foreach(ID = 1 :5) %dopar% {
#foreach(ID = 1 :NIND) %dopar% {
    print(paste("ind",ID))
    pemission <- funpem(freqs,genos,ID,ff)
    indiv(ID,pemission)
}
```

7.4.3 Modèles à plusieurs classes avec les taux prédéfinis

Programme `Optim_4_predef_repar_paral.R`

À nouveau, les modifications apportées à ce modèle sont éparpillées dans le programme par rapport au précédent. Il est donc retranscrit dans son entièreté. Cependant, les fonctions `allelefreq` et `funpem` sont toujours strictement identiques ainsi que le début du corps du programme.

```
dyn.load("zoolik.so")
dyn.load("zooFB.so")
library(tools)
library(foreach)
library(doMC)
registerDoMC(3)


allelefreq <- function(genoline) {
  sum(genoline[genoline<3]) / (2*length(genoline[genoline<3]))
}

getlik <- function(par,pem) {
  #print(paste("getlik début"))

  Fs=array(0,K)
  D=1
  for (j in 1 :(K-1)){D=D+exp(par[j])}
  for (j in 1 :(K-1)){Fs[j]=exp(par[j])/D}
  Fs[K]=1-sum(Fs[1 :(K-1)])
  countfct <<- countfct+1
  pemission <- pem
  oneiter <- .Fortran("zoolik",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
                      as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
                      as.integer(bp),as.double(0))
  loglik <- oneiter[11][[1]][1]
  #print(paste("getlik fin",loglik))
  return(-loglik)
}

fb <- function(out,pemission) {
  Fs[1 :(K-1)] <- 1/(1+exp(-out$par[1 :(K-1)]))
  Fs[K]=1-sum(Fs[1 :(K-1)])

  onefb <- .Fortran("zooFB",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
                    as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
                    as.integer(bp),as.double(0),matrix(as.double(0),K,nsnps))
}

funpem <- function(freqs,genos,ID,ff){
  #print(paste("funpem",ID))
  geno <- genos[(freqs >= 0.01 & freqs <= 0.99), ID]

  #### COMPUTE EMISSION PROB.
  pemission<-matrix(0, length(ff), 2)

  pemission[geno==0,1]=(1-ff[geno==0])**2
  pemission[geno==1,1]=2*ff[geno==1]*(1-ff[geno==1])
  pemission[geno==2,1]=ff[geno==2]**2
  pemission[geno==9,1]=1.00

  gerr=0.001
  pemission[geno==0,2]=(1-ff[geno==0])*(1-gerr)
  pemission[geno==1,2]=gerr
  pemission[geno==2,2]=ff[geno==2]*(1-gerr)
  pemission[geno==9,2]=1.00

  #print(paste("funpem fin",ID))
  return(pemission)
}

```

```
indiv <- function(ID,pemission){
  #print(paste("indiv",ID))

  #### Number of iteration (optim)
  countfct <<- 0

  #### Optimisation of the parameters via the likelihood (optim)
  #print(paste("indiv avant optim getlik",ID))
  out <- optim(par=par, getlik,pem=pemission, method="L-BFGS-B", control = list(trace=TRUE))
  #print(paste("indiv après optim getlik",ID))

  #### Outfile : iterations
  output_count <- matrix(0,1,2)

  output_count[1,1] <- ID
  output_count[1,2] <- countfct
  #print(output_count)

  write.table(output_count, file = paste0("Opt_4_predef_paral_count_", name, ".txt"),
              append=TRUE, sep = "\t",quote=F,col.names=FALSE)
  #print("count")

  #### Outfile : Likelihood and optimized parameters
  r <- 1/(1+exp(-out$par[1 :(K-1)]))
  v <- out$value

  output_optim_4_1_Fs <- matrix(0, 1, (2*K+3))

  output_optim_4_1_Fs[1, 1] <- ID
  output_optim_4_1_Fs[1, 2] = round(v, 6)
  for(i in seq(3,(2*K+1),2)){
      output_optim_4_1_Fs[1, i] <- as[(i-1)/2]
  }
  for(i in seq(4,(2*K),2)){
      output_optim_4_1_Fs[1, i] <- round(r[(i-2)/2],7)
  }
  output_optim_4_1_Fs[1, (2*K+2)] <- round(1-sum(r[1 :(K-1)]),7)
  output_optim_4_1_Fs[1, (2*K+3)] <- countfct

  write.table(output_optim_4_1_Fs, file = paste0("Opt_4_predef_paral_out_", name,".txt"),
              append=TRUE, sep = "\t",quote=F,col.names=FALSE)
  #print("out")

  #### Outfile : Age
  #output_age <- matrix(0,1,K+1)
  #
  #output_age[1, 1] <- ID
  #for(i in 1 :K){
  #    output_age[1,(i+1)] <- round(G[i],4)
  #}
  #
  #write.table(output_age, file = paste0("Opt_4_predef_paral_age_",name,".txt"),
               append=TRUE,sep = "\t", quote = F,col.names=FALSE)
  ##print("age")

  #### Outfile : mixing proportions
  output_mixing <- matrix(0,1,K+1)

  output_mixing[1, 1] <- ID
  for(i in 1 :(K-1)){
      output_mixing[1,(i+1)] <- round(r[i],7)
  }
  output_mixing[1,(K+1)] <- round(1-sum(r[1 :(K-1)]),7)

  write.table(output_mixing, file = paste0("Opt_4_predef_paral_mixing_",name,".txt"),
              append=TRUE,sep = "\t", quote = F,col.names=FALSE)
  #print("mixing")

  #### Forward/backward to obtain the inbreeding
  #print(paste("indiv avant fb",ID))
  out2<-fb(out,pemission)
  #print(paste("indiv après fb",ID))

  #### Outfile : inbreeding per class and global inbreeding
```

```
  FG<-matrix(0,1,K+1)

  FG[1,1]<-ID
  for(i in 1 :(K-1)){
      FG[1,(i+1)] <- mean(out2[[12]][i,])
  }
  FG[1,(K+1)] <- sum(FG[1,(2 :K)])

  write.table(FG, file = paste0("Opt_4_predef_paral_inbreeding_", name,".txt"),
              append=TRUE,sep = "\t",quote=F,col.names=FALSE)
  #print("inbreeding")

  #### Outfile : resume (likelihood, parameters, inbreeding, iterations)
  output_onefile <- matrix(0,1,(2*K+4))

  output_onefile[1, 1] <- ID
  output_onefile[1, 2] <- round(v, 6)
  for(i in seq(3,(2*K+1),2)){
      output_onefile[1, i] <- as[(i-1)/2]
  }
  for(i in seq(4,(2*K),2)){
      output_onefile[1, i] <- round(r[(i-2)/2],7)
  }
  output_onefile[1, (2*K+2)] <- round(1-sum(r[1 :(K-1)]),7)
  output_onefile[1, (2*K+3)] <- FG[1,(K+1)]
  output_onefile[1, (2*K+4)] <- countfct

  write.table(output_onefile, file = paste0("Opt_4_predef_paral_",name,".txt"),
              append=TRUE,sep = "\t",quote=F,col.names=FALSE)
  #print("tout")
}

#### Read the geno file (terminal) that must have 5 informative columns
argv <- commandArgs(TRUE)
data <- read.table(argv[1])
name <- file_path_sans_ext(basename(argv[1]))

########################################################
### Compute informations from the data (geno file) ###
########################################################

#### Number of individuals and their informations for the markers
NIND = length(data[1,])-5
genos <- as.matrix(data[,6 :(NIND+5)])

#### Compute allelic frequencies
freqs <- apply(genos, 1, allelefreq)

#### Keep markers with MAF >= 0.01
chr = data$V1[(freqs >= 0.01 & freqs <= 0.99)]
bp = data$V3[(freqs >= 0.01 & freqs <= 0.99)]
ff = freqs[(freqs >= 0.01 & freqs <= 0.99)]

#### Number of chromosomes
nchr=max(chr)
#chrs=seq(1 :nchr)

#### Bounds of the chromosomes (line numbers = #marker)
chrbound=matrix(0, nchr, 2)
chrbound[1,1]=which.min((bp[chr == 1]))
chrbound[1,2]=which.max((bp[chr == 1]))

for (i in 2 :nchr) {
  chrbound[i,1]=which.min(bp[chr == i])+chrbound[(i-1),2]
  chrbound[i,2]=which.max(bp[chr == i])+chrbound[(i-1),2]
}

#### Total number of SNPs
nsnps=chrbound[nchr,2]

########################################################
###               Set the parameters                ###
########################################################

```

```
#### Number of classes
K=4

#### Class IBD (1) or not (0)
isF <- c(replicate((K-1),1),0)

#### Matrice de transition
T=matrix(1,K,K)

#### Number of generation
as=c(8,64,256,256)

#### Mixing proportion
Fs=c(0.02, 0.02, 0.02, 0.94)

#### Re-parametrization
par<-array(0,K-1)

for (j in 1 :(K-1)){par[j]=log(Fs[j]/Fs[K])}

#########################################################
###           Compute for each individual          ###
#########################################################

countfct <- 0

#### Colnames of each file to write
write.table(matrix(c("ind", "iter"),1,2), file = paste0("Opt_4_predef_paral_count_", name, ".txt"),
            sep = "\t",quote=F,col.names=FALSE)
vecout <- vector()
for(i in seq(2,2*K,2)){
    vecout[(i-1)] <- paste0("age",((i-2)/2+1))
    vecout[i] <- paste0("mixing",((i-2)/2+1))
}
#vecage <- vector()
vecmix <- vector()
for(i in 1 :K){
   # vecage[i]<-paste0("age",i)
    vecmix[i]<-paste0("mixing",i)
}
vecinb <- vector()
for(i in 1 :(K-1)){vecinb[i] <- paste0("Inb",i)}
vecinb[K] <- "InbTot"
#print(vecout)
#vecout <- rep(c("age", "mixing"),K)
write.table(matrix(c("ind","likelihood", vecout, "iter"),1,(2*K+3)),
            file = paste0("Opt_4_predef_paral_out_", name,".txt"), sep = "\t",quote=F,col.names=FALSE)
#write.table(matrix(c("ind",vecage),1,(K+1)), file = paste0("Opt_4_predef_paral_age_",name,".txt"),
              #sep = "\t", quote = F,col.names=FALSE)
write.table(matrix(c("ind",vecmix),1,K+1), file = paste0("Opt_4_predef_paral_mixing_",name,".txt"),
            sep = "\t", quote = F,col.names=FALSE)
write.table(matrix(c("ind",vecinb),1,K+1), file = paste0("Opt_4_predef_paral_inbreeding_", name,".txt"),
            sep = "\t",quote=F,col.names=FALSE)
write.table(matrix(c("ind","likelihood", vecout,"InbTot", "iter"),1,(2*K+4)),
            file = paste0("Opt_4_predef_paral_",name,".txt"),sep = "\t",quote=F,col.names=FALSE)

#### For each individual
foreach(ID = 1 :5) %dopar% {
#foreach(ID = 1 :NIND) %dopar% {
    print(paste("ind",ID))
    pemission <- funpem(freqs,genos,ID,ff)
    indiv(ID,pemission)
}
```

Programme `Optim_10_predef_repar_paral.R`

En plus des noms des fichiers de sortie, les petites modifications apportées au script précédent seront faciles à adapter lors de la création d'un package R. Les voici :

```
206 ######################################################
207 ###               Set the parameters               ###
208 ######################################################
209
210 #### Number of classes
211 K=10
212
213 #### Class IBD (1) or not (0)
214 isF <- c(replicate((K-1),1),0)
215
216 #### Matrice de transition
217 T=matrix(1,K,K)
218
219 #### Number of generation
220 as=c(2,4,8,16,32,64,128,256,512,512)
221
222 #### Mixing proportion
223 Fs=c(0.01, 0.01, 0.01,0.01, 0.01, 0.01,0.01, 0.01, 0.01, 0.91)
224
225 #### Re-parametrization
226 par<-array(0,K-1)
227
228 for (j in 1 :(K-1)){par[j]=log(Fs[j]/Fs[K])}
```

7.4.4 Viterbi

Programme `Vit_em_fb_gamma_2e_wz_paral.R`

Par rapport au programmes précédents, certaines fonctions ont été ajoutées. La fonction `vit` appelle l'algorithme de Viterbi (écrite en fortran). La fonction `postvit` permet de traiter les résultats de `vit` pour obtenir des positions de début et de fin de segments. La fonction `expmax` a également été ajoutée pour appeler l'EM sous R. Enfin, quelques lignes ont été ajoutées dans la fonction `indiv` pour appeler ces différentes fonctions.

```
3   dyn.load("zooEM_v2.so")
4   dyn.load("zooViterbi.so")


37  expmax <- function(par,n,pemission) {
38    #print("em")
39    as=c(exp(par[1]),exp(par[1]))
40    f=1/(1+exp(-par[2]))
41    Fs=c(f, (1-f))
42    oneiter <- .Fortran("zooEM_v2",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
43                        as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
44                        as.integer(bp),as.double(0),matrix(as.double(0),K,nsnps),as.integer(1),as.integer(1),
45                        as.integer(n),as.double(www),as.double(zzz))
46    #print("n=1")
47    loglik <- oneiter[11][[1]]
48    emas <- oneiter[8][[1]]
49    emfs <- oneiter[9][[1]]
50    gamma <- oneiter[12][[1]]
51    #print(gamma[1,])
52    #print(as)
53    liste <- list(lik=-loglik,fs=emfs,as=emas,pibd=gamma[1 :(K-1),])
54    #print(liste)
55    return(liste)
56  }
57
58  vit <- function(out,pemission) {
59    #print("vit")
60    as=out[1 :K]
61    Fs=out[(K+1) :(K+K)]
62
63    states <- replicate(nsnps,0)
64    oneiter <- .Fortran("zooViterbi",as.integer(K), as.integer(nchr),as.integer(nsnps),as.double(pemission),
65                        as.integer(chrbound),as.integer(isF),as.integer(T),as.double(as),as.double(Fs),
66                        as.integer(bp),as.integer(states))
67    states <- oneiter[11][[1]]
68    return(states)
69  }


93  postvit <- function(vec,ID){
94    #print("postvit")
95    ne14 <- which(vec !=K)
96    #print(ne14)
97    vec2 <- vector()
98    for(i in 1 :(length(ne14)-1)){
99      if(length(ne14)==0){break}
100     if(i==1){
101       #print("début")
102       vec2[i] <- 0
103     }
104     if(ne14[i]==(ne14[i+1]-1)){
105       #print("segment")
106       vec2[i+1] <- 1
107     }
108     else{
109       #print("new")
110       #print(i+1)
```

```
    vec2[i+1] <- 0
  }
}
posi <- which(vec2==0)
#print(length(posi))
start <- vector()
end <- vector()
for(i in 1 :length(posi)){
  if(length(posi)==1){
    start[i] <- ne14[posi[i]]
    end[i] <- ne14[length(ne14)]
    break
  }
  if(i==1){
    start[i] <- ne14[posi[i]]
    #print(ne14[posi[i]])
    next
  }
  end[i-1]<-ne14[posi[i]-1]
  #print(ne14[posi[i]-1])
  start[i] <- ne14[posi[i]]
  #print(ne14[posi[i]])
  if(i==length(posi)){
    end[i] <- ne14[length(ne14)]
    #print(ne14[length(ne14)])
  }
}
if(length(ne14)==0){
  #start <- 0
  #end <- 0
  frag2 <- data.frame(ind=ID,chrom=NA,starts=NA,ends=NA,len=NA)
  write.table(frag2,file = paste0("gamma_vit_em_2e_v2_",name,"_",zzz,".txt"),
              append = TRUE,sep="\t",quote=FALSE,col.names = FALSE)
  next
}
#print(chrbound[1 :3,2])

ends <- vector()
starts <- vector()
for(i in 1 :length(end)){
  if(which(chrbound[,2]>start[i])[1] == 1){
    if(which(chrbound[,2]>start[i])[1]==
          (which(chrbound[,1]<end[i])[length(which(chrbound[,1]<end[i]))]-1)){
      #print("1 if")
      starts <- c(starts,start[i],chrbound[which(chrbound[,2]>start[i])[2],1])
      ends <- c(ends,chrbound[which(chrbound[,2]>start[i])[1],2],end[i])
    }
    else if(which(chrbound[,2]>start[i])[2]==
              (which(chrbound[,1]<end[i])[length(which(chrbound[,1]<end[i]))]-1)){
      #print("1 else if")
      starts <- c(starts,start[i],chrbound[which(chrbound[,2]>start[i])[2],1],
                  chrbound[which(chrbound[,2]>start[i])[3],1])
      ends <- c(ends,chrbound[which(chrbound[,2]>start[i])[1],2],
                chrbound[which(chrbound[,2]>start[i])[2],2],end[i])
    }
    else{
      #print("1 else")
      ends <- c(ends,end[i])
      starts <- c(starts,start[i])
    }
  }
  else if((which(chrbound[,2]>start[i])[1])==
          (which(chrbound[,2]<end[i])[length(which(chrbound[,2]<end[i]))])){
    #print("else if 1")
    starts <- c(starts,start[i],chrbound[which(chrbound[,2]>start[i])[2],1])
    ends <- c(ends,chrbound[which(chrbound[,2]>start[i])[1],2],end[i])
  }
  else if(is.na(which(chrbound[,2]>start[i])[2])==FALSE && (which(chrbound[,2]>start[i])[2])==
                  (which(chrbound[,2]<end[i])[length(which(chrbound[,2]<end[i]))])){
    #print("else if 2")
    starts <- c(starts,start[i],chrbound[which(chrbound[,2]>start[i])[2],1],
                  chrbound[which(chrbound[,2]>start[i])[3],1])
    ends <- c(ends,chrbound[which(chrbound[,2]>start[i])[1],2],
                  chrbound[which(chrbound[,2]>start[i])[2],2],end[i])
```

```
    }
    else{
      #print("else")
      ends <- c(ends,end[i])
      starts <- c(starts,start[i])
    }
  }
  chromo <- vector()
  for(i in 1 :length(starts)){chromo[i] <-which(chrbound[,2]>starts[i])[1]}

  #print(start)
  #print(bp[start])
  #print(end)
  #print(end-start)
  frag2 <- data.frame(ind=ID,chrom=chromo,starts=bp[starts],ends=bp[ends],len=(bp[ends]-bp[starts]))
  #print(frag2)

  write.table(frag2,file = paste0("gamma_vit_em_2e_v2_",name,"_",zzz,".txt"),append = TRUE,sep="\t",
              quote=FALSE,col.names = FALSE)
}

indiv <- function(ID,pemission){
  #print(paste("indiv",ID))

  #### EM
  iter <- 1000
  outem <- expmax(par,iter,pemission)

  #### Viterbi EM
  param <- c(outem$as,outem$fs)
  vec <- vit(param,pemission)

  #### Outfile : out Vit (EM)
  vecvit <- data.frame(ind=ID,chrom=chr, posi=bp, vi=vec)
  write.table(vecvit,file = paste0("gamma_vit_brut_em_2e_v2_",name,"_",zzz, ".txt"),append = TRUE,
              sep="\t",quote=FALSE,col.names = FALSE)
  postvit(vec, ID)

  #### Number of iteration (optim)
  countfct <<- 0

  #### Optimisation of the parameters via the likelihood (optim)
  #print(paste("indiv avant optim getlik",ID))
  out <- optim(par=par, getlik,pem=pemission, method="L-BFGS-B",lower=c(log(www),-Inf),
              upper=c(log(zzz),5), control = list(trace=TRUE))
  #print(paste("indiv après optim getlik",ID))

  #### Viterbi optim
  r <- 1/(1+exp(-out$par[2]))
  params <- c(exp(out$par[1]),exp(out$par[1]),r,(1-r))
  opt_vec <- vit(params,pemission)

  #### Outfile : out Vit (opt)
  opt_vecvit <- data.frame(ind=ID,chrom=chr, posi=bp, vi=opt_vec)
  write.table(opt_vecvit,file = paste0("gamma_vit_brut_opt_2e_v2_",name,"_",zzz,".txt"),
              append = TRUE,sep="\t",quote=FALSE,col.names = FALSE)
  postvit(opt_vec,ID)


  gamfb <- out2[[12]][1 :(K-1),]
  gamem <- outem$pibd

  #chroms <- vector()
  #for(i in 1 :nrow(chrbound)){chroms<-c(chroms,replicate((chrbound[i,2]-chrbound[i,1]),i))}
  #print(bp)
  #print(chr)
  datafb <- data.frame(ind=ID,chrom=chr, posi=bp, G1=gamfb)
  dataem <- data.frame(ind=ID,chrom=chr, posi=bp,G1=gamem)
  write.table(datafb,file = paste0("gamma_fb_2e_v2_",name,"_",zzz,".txt"),
              append = TRUE, sep="\t",quote=FALSE,col.names = FALSE)
  write.table(dataem,file = paste0("gamma_em_2e_v2_",name,"_",zzz,".txt"),
              append = TRUE, sep="\t",quote=FALSE,col.names = FALSE)
}
```

7.4.5 Analyse des fréquences alléliques

Programme `calcul_maf_v2.R`

```
library(ggplot2)

frequencies <- function(pedfile){
  freqA <- vector()
  freqB <- vector()
  freqAB <- vector()
  for(i in (1 :((ncol(pedfile)-6)/2))){
    print(i)
    AA <- 0
    AB <- 0
    BA <- 0
    BB <- 0

    for(j in 1 :length(colnames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)])))){
      if(colnames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)]))[j]=="1"){
        for(k in 1 :length(rownames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)])))){
          if(rownames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)]))[k]=="1"){
            AA <- table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)])[k,j]
            print("AA")
            print(AA)
          }
          else if(rownames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)]))[k]=="2"){
            BA <- table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)])[k,j]
            print("BA")
            print(BA)
          }
        }
      }
      else if(colnames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)]))[j]=="2"){
        for(k in 1 :length(rownames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)])))){
          if(rownames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)]))[k]=="1"){
            AB <- table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)])[k,j]
            print("AB")
            print(AB)
          }
          else if(rownames(table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)]))[k]=="2"){
            BB <- table(pedfile[,(6+2*i-1)],pedfile[,(6+2*i)])[k,j]
            print("BB")
            print(BB)
          }
        }
      }
    }


    freqA[i] <- (2*AA+AB+BA)/(2*(AA+AB+BA+BB))
    freqB[i] <- (2*BB+AB+BA)/(2*(AA+AB+BA+BB))
    freqAB[i] <- (AB+BA)/(AA+AB+BA+BB)


  }
  tab <- data.frame(freqA,freqB,freqAB)
  return(tab)
}

ped_cow <- read.table("plink/cow/bbo.ped")

datamds <- read.table("plink/cow/plink.mds",header = TRUE)
ind1 <- datamds[which(datamds[,4]<0),2]

ped_cow_pop1 <- ped_cow[ind1,]
ped_cow_pop2 <- ped_cow[-ind1,]



freqPopTot <- frequencies(ped_cow)
freqPop1 <- frequencies(ped_cow_pop1)
freqPop2 <- frequencies(ped_cow_pop2)
```

```
69
70  write.table(freqPopTot,"plink/cow/freqPopTot_v2.txt", quote = FALSE,sep = "\t")
71  write.table(freqPop1,"plink/cow/freqPop1_v2.txt", quote = FALSE,sep = "\t")
72  write.table(freqPop2,"plink/cow/freqPop2_v2.txt", quote = FALSE,sep = "\t")
73
74  maf1 <- vector()
75  maf2 <- vector()
76  mafT <- vector()
77  for(i in 1 :nrow(freqPopTot)){
78    if(freqPop1[i,1] < freqPop1[i,2]){
79      maf1[i] <- freqPop1[i,1]
80    }
81    else{
82      maf1[i] <- freqPop1[i,2]
83    }
84    if(freqPop2[i,1] < freqPop2[i,2]){
85      maf2[i] <- freqPop2[i,1]
86    }
87    else{
88      maf2[i] <- freqPop2[i,2]
89    }
90    if(freqPopTot[i,1] < freqPopTot[i,2]){
91      mafT[i] <- freqPopTot[i,1]
92    }
93    else{
94      mafT[i] <- freqPopTot[i,2]
95    }
96  }
97
98  freqPop1[c(14,24),]
99  freqPop2[c(14,24),]
100 mafs <- data.frame(maf1,maf2,mafT)
101 mafs[c(14,24),]
102
103 write.table(mafs,"plink/cow/mafs_v2.txt", quote = FALSE,sep = "\t")
104
105 ######### graphs ############
106
107 #### frequency ####
108 pdf("plink/cow/maf_admixture_3pop_v3.pdf")
109 graph2 <- ggplot(data = mafs, aes(x=mafs$mafT,y=..count../sum(..count..)))
110 graph2+
111   theme_bw()+
112   geom_freqpoly(binwidth=0.01,aes(color="PopT"))+
113   geom_freqpoly(binwidth=0.01,aes(x=mafs$maf1,color="Pop1"))+
114   geom_freqpoly(binwidth=0.01,aes(x=mafs$maf2,color="Pop2"))+
115   geom_vline(data=summ,aes(xintercept=mean(mafs$mafT),color="PopT"),linetype="dashed")+
116   geom_vline(data=summ_pop1,aes(xintercept=mean(mafs$maf1),color="Pop1"),linetype="dashed")+
117   geom_vline(data=summ_pop2,aes(xintercept=mean(mafs$maf2),color="Pop2"),linetype="dashed")+
118   geom_vline(xintercept = 0.01,linetype="dotted")+
119   labs(title=paste("Frequency of the different MAF over all the SNPs in the 3 populations"))+
120   #labs(title=paste("Subpopulation 1"))+
121   xlab("MAF")+
122   #xlab("")+
123   ylab("Frequency over all the SNPs")+
124   #ylab("")+
125   scale_colour_manual("", values = clr, breaks = c("PopT","Pop1","Pop2"))+
126   theme(legend.position="bottom")
127 dev.off()
128
129
130 #### maf - maf ####
131 pdf("plink/cow/maf_maf_tot_v3.pdf")
132 graphmaf <- ggplot(data = mafs,aes(x=maf1,y=maf2))
133 graphmaf+
134   theme_bw()+
135   geom_point()+
136   labs(title=paste(""))+
137   xlab("MAF population 1")+
138   ylab("MAF population 2")
139 dev.off()
140
141 View(mafs)
142
```

```
######### graph without fixed ############

mafs045 <- mafs[which(mafs$maf1>0.45),]
mafs04505 <- mafs045[which(mafs045$maf2<0.05),]
sans <- mafs[-as.numeric(rownames(mafs04505)),]

pdf("plink/cow/maf_maf_sans_v3.pdf")
graphsans <- ggplot(data = sans,aes(x=maf1,y=maf2))
graphsans+
  theme_bw()+
  geom_point()+
  labs(title=paste(""))+
  xlab("MAF population 1")+
  ylab("MAF population 2")
dev.off()


############### a retirer ###################

aretirer <- as.numeric(rownames(mafs04505))

#### frequency ####
mafs_sans <- mafs[-aretirer,]

pdf("plink/cow/maf_admixture_3pop_sans_v3.pdf")
graph3 <- ggplot(data = mafs_sans, aes(x=mafs_sans$mafT,y=..count../sum(..count..)))
graph3+
  theme_bw()+
  geom_freqpoly(binwidth=0.01,aes(color="PopT"))+
  geom_freqpoly(binwidth=0.01,aes(x=mafs_sans$maf1,color="Pop1"))+
  geom_freqpoly(binwidth=0.01,aes(x=mafs_sans$maf2,color="Pop2"))+
  geom_vline(data=summ,aes(xintercept=mean(mafs_sans$mafT),color="PopT"),linetype="dashed")+
  geom_vline(data=summ_pop1,aes(xintercept=mean(mafs_sans$maf1),color="Pop1"),linetype="dashed")+
  geom_vline(data=summ_pop2,aes(xintercept=mean(mafs_sans$maf2),color="Pop2"),linetype="dashed")+
  geom_vline(xintercept = 0.01,linetype="dotted")+
  labs(title=paste("Frequency of the different MAF over all the SNPs in the 3 populations"))+
  #labs(title=paste("Subpopulation 1"))+
  xlab("MAF")+
  #xlab("")+
  ylab("Frequency over all the SNPs")+
  #ylab("")+
  scale_colour_manual("", values = clr, breaks = c("PopT","Pop1","Pop2"))+
  theme(legend.position="bottom")
dev.off()

######## _less_v3.gen ########

bbogeno <- read.table("data/5col/bisons.gen")
bbogeno1 <- read.table("data/5col/bisons_pop1.gen")
bbogeno2 <- read.table("data/5col/bisons_pop2.gen")

bbogeno1_2 <- bbogeno1[-aretirer,]
bbogeno2_2 <- bbogeno2[-aretirer,]
bbogenoT_2 <- bbogeno[-aretirer,]

write.table(bbogeno1_2,file = "data/5col/bisons_pop1_less_v3.gen", quote = FALSE,col.names = FALSE, row.names = FALSE)
write.table(bbogeno2_2,file = "data/5col/bisons_pop2_less_v3.gen", quote = FALSE,col.names = FALSE, row.names = FALSE)
write.table(bbogenoT_2,file = "data/5col/bisons_less_v3.gen", quote = FALSE,col.names = FALSE, row.names = FALSE)

snpsremove <- paste0("MAR00X",aretirer)

write.table(snpsremove,file = "plink/cow/snpless_v3.txt", quote=FALSE,col.names = FALSE,row.names = FALSE)

map_cow <- read.table("plink/cow/bbo.map")

vectmp <- vector()
for(i in 1 :length(aretirer)){
  vectmp <- c(vectmp,2*aretirer[i]-1+6,2*aretirer[i]+6)
}
pedsans <- ped_cow[,-vectmp]
mapsans <- map_cow[-aretirer,]

write.table(pedsans, "plink/cow/bbo_less_v3.ped", quote=FALSE,col.names = FALSE,row.names = FALSE)
write.table(mapsans, "plink/cow/bbo_less_v3.map", quote=FALSE,col.names = FALSE,row.names = FALSE)
```

Bibliographie

1. Pucek, Z. *European bison : status survey and conservation action plan.* (IUCN, 2004).

2. Węcek, K. *et al.* Complex Admixture Preceded and Followed the Extinction of Wisent in the Wild. *Molecular Biology and Evolution* msw254 (2016). doi :10.1093/molbev/msw254

3. Gautier, M. *et al.* Deciphering the Wisent Demographic and Adaptive Histories from Individual Whole-Genome Sequences. *Molecular Biology and Evolution* **33,** 2801–2814 (2016).

4. Hartl, D. L. & Clark, A. G. *Principles of Population Genetics.* (2007).

5. Markova, A. *et al.* Changes in the Eurasian distribution of the musk ox (Ovibos moschatus) and the extinct bison (Bison priscus) during the last 50 ka BP. *Quaternary International* **378,** 99–110 (2015).

6. Krasińska, M. & Krasiński, Z. A. *European Bison.* (Springer Berlin Heidelberg, 2013). Available at : http://link.springer.com/10.1007/978-3-642-36555-3. (Accessed : 17th May 2017)

7. IUCN. *Bison bonasus : Olech, W. (IUCN SSC Bison Specialist Group) : The IUCN Red List of Threatened Species 2008 : e.T2814A9484719.* (International Union for Conservation of Nature, 2008). Available at : http://www.iucnredlist.org/details/2814/0. (Accessed : 10th July 2017)

8. Tokarska, M. *et al.* Genetic status of the European bison Bison bonasus after extinction in the wild and subsequent recovery : European bison conservation genetics. *Mammal Review* **41,** 151–162 (2011).

9. Tokarska, M. *et al.* Effectiveness of microsatellite and SNP markers for parentage and identity analysis in species with low genetic diversity : the case of European bison. *Heredity* **103,** 326–332 (2009).

10. Caballero, A. *et al.* Management of genetic diversity of subdivided populations in conservation programmes. *Conservation Genetics* **11,** 409–419 (2010).

11. Powell, J. E., Visscher, P. M. & Goddard, M. E. Reconciling the analysis of IBD and IBS in complex trait studies. *Nature Reviews Genetics* **11,** 800–805 (2010).

12. Kardos, M. *et al.* Genomics advances the study of inbreeding depression in the wild. *Evolu-*

tionary Applications **9,** 1205–1218 (2016).

13. Simm, G. *Genetic improvement of cattle and sheep.* (1998).

14. Wright, S. Coefficients of inbreeding and relationship. (1922).

15. Wright, S. Systems of mating. (1921).

16. Kardos, M., Luikart, G. & Allendorf, F. Measuring individual inbreeding in the age of genomics : marker-based measures are better than pedigrees. *Heredity* **115,** 63–72 (2015).

17. Charlesworth, B. Fundamental concepts in genetics : Effective population size and patterns of molecular evolution and variation. *Nature Reviews Genetics* **10,** 195–205 (2009).

18. Lynch, M. *et al.* Genetic drift, selection and the evolution of the mutation rate. *Nature Reviews Genetics* **17,** 704–714 (2016).

19. Sartelet, A. Contribution à la gestion des défauts génétiques dans la race Blanc-Bleu Belge. (Université de Liège, 2013).

20. Herd Book Blanc Bleu Belge. Available at : https://www.hbbbb.be/fr/. (Accessed : 2nd August 2017)

21. Cieploch, A. *et al.* Genetic disorders in beef cattle : a review. *Genes & Genomics* **39,** 461–471 (2017).

22. Charlesworth, D. & Charlesworth, B. Inbreeding depression and its evolutionary consequences. *Annual review of ecology and systematics* **18,** 237–268 (1987).

23. Keller, L. F. & Waller, D. M. Inbreeding effects in wild populations. *Trends in Ecology & Evolution* **17,** 230–241 (2002).

24. Hedrick, P. W. & Garcia-Dorado, A. Understanding Inbreeding Depression, Purging, and Genetic Rescue. *Trends in Ecology & Evolution* **31,** 940–952 (2016).

25. Savolainen, O., Lascoux, M. & Meril, J. Ecological genomics of local adaptation. *Nature Reviews Genetics* **14,** 807–820 (2013).

26. Leroy, G. Inbreeding depression in livestock species : review and meta-analysis. *Animal Genetics* **45,** 618–628 (2014).

27. Charlesworth, D. & Willis, J. H. The genetics of inbreeding depression. *Nature Reviews Genetics* **10,** 783–796 (2009).

28. Xue, Y. *et al.* Mountain gorilla genomes reveal the impact of long-term population decline

and inbreeding. *Science* **348,** 242–245 (2015).

29. Wang, J. Pedigrees or markers : Which are better in estimating relatedness and inbreeding coefficient ? *Theoretical Population Biology* **107,** 4–13 (2016).

30. Purcell, S. *et al.* PLINK : A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. *The American Journal of Human Genetics* **81,** 559–575 (2007).

31. Leutenegger, A.-L. *et al.* Estimation of the inbreeding coefficient through use of genomic data. *The American Journal of Human Genetics* **73,** 516–523 (2003).

32. Druet, T. & Gautier, M. A model-based approach to characterize individual inbreeding at both global and local genomic scales. *Molecular Ecology* (2017). doi :10.1111/mec.14324

33. Cristianini, N. & Hahn, M. W. *Introduction to computational genomics : a case studies approach.* (Cambridge University Press, 2007).

34. Rabiner, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *IEEE* **77,** (1989).

35. Dempster, A., Laird, N. & Rubin, D. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* **39,** 1–38 (1977).

36. Leutenegger, A.-L. *et al.* Using genomic inbreeding coefficient estimates for homozygosity mapping of rare recessive traits : application to Taybi-Linder syndrome. *The American journal of human genetics* **79,** 62–66 (2006).

37. Narasimhan, V. *et al.* BCFtools/RoH : a hidden Markov model approach for detecting autozygosity from next-generation sequencing data. *Bioinformatics* **32,** 1749–1751 (2016).

38. Vieira, F. G., Albrechtsen, A. & Nielsen, R. Estimating IBD tracts from low coverage NGS data. *Bioinformatics* **32,** 2096–2102 (2016).

39. R : General-purpose Optimization. Available at : https://stat.ethz.ch/R-manual/R-devel/library/stats/html/optim.html. (Accessed : 26th June 2017)

40. Haelterman, R. Analytical study of the least squares quasi-Newton method for interaction problems. (Ghent University, 2009). Available at : https://biblio.ugent.be/publication/720660. (Accessed : 28th March 2017)

41. Zucchini, W. & MacDonald, I. L. *Hidden Markov Models for Time Series : An Introduction Using R.* (2009).

42. Murgiano, L. *et al.* An Intronic *MBTPS2* Variant Results in a Splicing Defect in Horses with

Brindle Coat Texture. *G3; GenesGenomesGenetics* **6,** 2963–2970 (2016).

43. Equus caballus (ID 145) - Genome - NCBI. Available at : https://www.ncbi.nlm.nih.gov/genome?term=equus%20caballus. (Accessed : 15th August 2017)

44. Bos taurus (ID 82) - Genome - NCBI. Available at : https://www.ncbi.nlm.nih.gov/genome?term=bos%20taurus. (Accessed : 15th August 2017)

45. Purcell, S. PLINK. Available at : http://zzz.bwh.harvard.edu/plink/.

46. Zuur, A. F., Ieno, E. N. & Smith, G. M. *Analysing ecological data*. (Springer, 2007).

47. Yang, J. *et al.* GCTA : A Tool for Genome-wide Complex Trait Analysis. *The American Journal of Human Genetics* **88,** 76–82 (2011).

48. Alexander, D., Novembre, J. & Lange, K. Fast model-based estimation of ancestry in unrelated individuals. *Genome Research* **19,** 1655–1664 (2009).

49. Alexander, D. H., Novembre, J. & Lange, K. Admixture 1.23 Software Manual. (2013). Available at : https://www.biomath.ucla.edu/software/admixture/admixture-manual.pdf. (Accessed : 4th May 2017)

50. Excoffier, L. & Heckel, G. Computer programs for population genetics data analysis : a survival guide. *Nature Reviews Genetics* **7,** 745–758 (2006).

51. stats. R : Non-Linear Minimization. Available at : https://stat.ethz.ch/R-manual/R-devel/library/stats/html/nlm.html. (Accessed : 25th August 2017)

52. Soubrier, J. *et al.* Early cave art and ancient DNA record the origin of European bison. *Nature Communications* **7,** 13158 (2016).