Aus dem Forschungsinstitut für Nutztierbiologie (FBN)
Institut für Genetik und Biometrie
Dummerstorf

A SIMILARITY MATRIX AND ITS APPLICATION IN GENOMIC SELECTION FOR HEDGING HAPLOTYPE DIVERSITY

Dissertation
zur Erlangung des Doktorgrades
der Agrar- und Ernährungswissenschaftlichen Fakultät
der Christian-Albrechts-Universität zu Kiel

vorgelegt von
M.Sc. agr. Abdulraheem Arome Musa
aus Kogi LGA, Kogi State, Nigeria

Kiel, 2021

Dekan: Prof. Dr. Karl H. Mühling
1. Berichterstatter: Prof. Dr. Norbert Reinsch
2. Berichterstatter: Prof. Dr. Georg Thaller

Tag der mündlichen Prüfung : 17.11.2021

Die Dissertation wurde mit dankenswerter finanzieller Unterstützung der Bundesanstalt für Landwirtschaft und Ernährung (BLE) im Rahmen des Zuschusses 281B101516 und des Tertiary Education Trust Fund (TETFUND) angefertigt.

“Out of clutter, find Simplicity. From discord, find Harmony. In the middle of difficulty lies Opportunity...”

– ALBERT EINSTEIN

TABLE OF CONTENTS

GENERAL INTRODUCTION

Motivation

In an era of population expansion and climate change, developing novel methods for improving desirable traits in farm animals such as yield and nitrogen emissions reduction is necessary. Genetic improvement of desirable traits in cattle is based on recurrent selection and mating of parents with the best-expected breeding values for their offspring, defined as the average parental breeding values plus a Mendelian sampling term (Kennedy et al., 1988). The availability of genomic data enables the computation of genomic estimated breeding values (GEBVs; Meuwissen et al., 2001) and Mendelian sampling variances (MSVs; Bonk et al., 2016). However, most genomic breeding programs continue to base decisions solely on GEBV due to its effectiveness in delivering high short-term genetic gains while remaining oblivious to the variability in offspring's GEBV caused by Mendelian sampling. While the objective is to improve the mean GEBVs of offspring, MSV can help increase selection response. The continued selection based on GEBV only has sparked concerns about the loss of favorable haplotypes, inbreeding, and its long-term effectiveness in improving complex traits (Schaeffer, 2006; Akdemir and Sánchez, 2016; Lin et al., 2016). As a result, numerous selection criteria for short- or long-term genetic gains, or a combination of the two, have been proposed to address these concerns (Goiffon et al., 2017; Müller et al., 2018; Moeinizade et al., 2019).

MSV has numerous breeding applications (Segelke et al., 2014; Bonk et al., 2016). Recently proposed selection criteria for optimizing the tradeoff between short- and long-term gains in an outbred population are based on an index composed of GEBV and MSV (Santos et al., 2019; Bijma et al., 2020). Compared to selection solely on GEBVs, Bijma et al. (2020) reported that using the index could increase the probability of breeding top-ranking offspring in the Holstein dairy population by 36% and the response to selection by 3.6%. Santos et al. (2019) obtained an additional genetic gain of up to 16% over ten generations when the index was used in a simulation with a small genome. These indices, however, have two drawbacks. First, computing MSV takes a long time. Second, similar parents with high genetic variability potential tend to be selected under a high selection intensity, potentially leading to unintended losses of favorable haplotypes. These drawbacks must be overcome to achieve a higher rate of genetic improvement and avoid unintentional losses of genetic diversity.

The present thesis addresses these possible drawbacks by developing methods and software for rapidly computing MSV and a similarity matrix based on Mendelian sampling values for hedging haplotype diversity. Empirical German Holstein cattle data offer insights into the applicability, relevance, and efficient use of these methods in breeding programs.

Genetic improvement

Genetic improvement for desirable traits in animals is carried out through artificial selection and mating of parents with superior true breeding values (TBVs) (Hill, 2014), expecting their offspring's TBVs will be equal to their parents' average. TBVs are not observed in practice because they are determined by quantitative trait loci (QTL), which are DNA segments that affect a trait (Lynch and Walsch, 1998). Thus, selection and mating decisions are based on estimates of TBVs (commonly known as estimated breeding values, EBVs). Historically, breeders selected parents of the next generation based on phenotypic values, measured as an alternative to their TBVs, known as phenotypic selection. To more accurately estimate TBVs, breeders complemented phenotypic selection with pedigree-based EBVs (Henderson, 1984), known as breeding value selection, and recently with genomic EBVs (Meuwissen et al., 2001), known as genomic selection (GS). If applied with no further constraints, these methods are based on truncation selection because they focus on the top potential parents with regard to the selection criteria before mating. Due to the accelerated genetic gain achieved by shortening the generation interval and improving the accuracy of GEBVs of young candidates, GS has become a popular and preferred tool for breeders (Hayes et al., 2009; Heffner et al., 2009).

Genomic selection

GS derives GEBVs using genome-wide single nucleotide polymorphism (SNP) markers and phenotypic data (Meuwissen et al., 2001). The accuracy of GS models is based on the assumption that all QTL associated with a trait are in linkage disequilibrium with SNP markers (Meuwissen et al., 2001; Goddard and Hayes, 2007). Therefore, the models can explain substantial QTL effects by fitting SNP genotypes. Numerous statistical models for predicting GEBVs have been proposed, with each model differing in its assumptions and treatment of marker effects. Popular GS models such as ridge regression-best linear unbiased prediction (RR-BLUP) and Bayesian methods use a shrinkage

procedure (Heffner et al., 2009). In RR-BLUP, all the marker effects are simultaneously estimated, assuming that they are random and normally distributed with equal variance (Meuwissen et al., 2001). Thus, all marker effects are shrunk equally towards the mean of zero, leading to underestimation of large-effect QTL. In order to overcome this drawback, Bayesian methods assume a separate variance for each marker and that variances follow a prior distribution (Meuwissen et al., 2001; Xu, 2003; Habier et al., 2011). In general, Bayesian estimates of marker effects are close to zero for marker intervals devoid of significant QTL, whereas marker effects in intervals containing significant QTL exhibit virtually no shrinkage.

Because prediction accuracy is directly proportional to genetic gain, it is an important criterion for assessing the performance of GS in breeding programs. Typically, it is calculated as the correlation between GEBVs and TBVs or phenotypic values. Thus, factors affecting GEBVs prediction accuracy will impact GS performance. Extensive GS research identifies a variety of factors that influence GS, including the statistical models used to estimate marker effects (Meuwissen et al., 2001; Heffner et al., 2009; Larkin et al., 2019) and the genotype imputation and phasing methods used (Druet et al., 2014; Wang et al., 2016). Additional factors such as population size, heritability, and marker density are thoroughly discussed by Larkin et al. (2019).

The selection of parents solely based on GEBV has revolutionized plant and animal breeding due to its success in delivering high genetic gain in the short term (Hayes et al., 2009; Heffner et al., 2009; VanRaden et al., 2009). Because the GEBV is calculated as the sum of estimated additive marker effects, it obscures the contributions of small effect markers (QTL), thereby promoting the loss of genetic variability. Because this loss of variability jeopardizes long-term gain, long-term genetic improvements require strategies that maximize short-term gains while minimizing inbreeding and genetic variability loss for the traits under selection (Jannink 2010). Many strategies have been proposed for short- or long-term gains, depending on the breeder's specific objectives.

Other proposed selection criteria

A group of proposals sought to upweight the marker effects of rare favorable alleles to ensure long-term gains (Goddard, 2009; Jannink, 2010; Liu et al., 2015). Optimum contribution selection (OCS) sought to preserve genetic variability by optimizing parental contributions to the next generation

(Wray and Goddard, 1994; Meuwissen, 1997; Woolliams et al., 2015). This is done by maximizing genetic gain under restricted inbreeding or minimizing inbreeding at the desired level of gain. OCS can achieve short-term gains with the appropriate constraint while exhausting genetic variability or achieve modest gains while maintaining genetic variability. It has been used widely in animal (Wang et al., 2017; Wellmann, 2019) and plant (Gaynor et al., 2017; Gorjanc et al., 2018) breeding applications.

Another group of proposals focuses on the GEBV of chromosomal segments. Parents with highly favorable haplotype combinations that are more likely to produce superior progeny are selected as parents in each generation (Kemper et al., 2012; Daetwyler et al., 2015). This approach was improved further by simulating meiosis between parental haplotypes to increase the accuracy of offspring prediction (Müller et al., 2018). By contrast, optimal population value is concerned with a group of parents' GEBV rather than chromosomal segments to produce the best possible progeny from this group after an infinite number of generations (Goiffon et al., 2017). Look-ahead selection is a variant of optimal population value selection that optimizes the tradeoff between short- and long-term gains to maximize genetic gain in the terminal generation (Moeinizade et al., 2019).

The final group of proposals optimizes the tradeoff between short- and long-term gain in an outbred population using an index composed of GEBV and MSV (Santos et al., 2019; Bijma et al., 2020). A similar index has been proposed to develop inbred lines in plants (Schnell and Utz, 1976; Lehermeier et al., 2017). MSV is discussed in detail below, along with its computation and potential breeding applications.

Mendelian sampling variance

MSV is the genetic variability among full-sibs due to the inheritance of random samples of alleles from both parents. The breeding (additive genetic) value bv_{ij} of offspring, according to Kennedy et al. (1988), produced by parents i and j can be expressed as: $bv_{ij} = .5bv_i + .5bv_j + md$, where bv_i and bv_j are the breeding values of sire and dam, respectively; and md is the deviation in bv_{ij} due to Mendelian sampling. Because Mendelian sampling is a stochastically independent process in parents, the variance of additive genetic values of offspring produced by parents i and j is given by

$\sigma^2_{bv_{ij}} = \text{var}(md)$, where $\text{var}(md)$ is the sum of the independent parental contributions $\text{var}(md) = \text{var}(md_i) + \text{var}(md_j)$.

The MSV of offspring produced by parents i and j can be estimated from pedigree information as the sum of the two parental contributions, $\frac{1}{4}\sigma^2_a(1-F_i) + \frac{1}{4}\sigma^2_a(1-F_j)$, where σ^2_a is the additive genetic variance in the base population and F_i (F_j) is the inbreeding coefficient of parent i (j) (Dempfle, 1990). Due to the successful implementation of genomic selection, the MSV can be estimated more precisely through a simulation approach using marker effects, genetic map, and genotypes phased into haplotypes (Bernardo, 2014; Segelke et al., 2014; Mohammadi et al., 2015). An analytical approach was developed to improve MSV prediction that utilizes the same data as the simulation approach but avoids Monte Carlo errors (Bonk et al., 2016; Lehermeier et al., 2017; Santos et al., 2019). While computing MSV using both approaches takes time, the simulation approach takes longer than the analytical approach if the Monte Carlo errors need to be reduced further (Bonk et al., 2016). Therefore, a faster analytical MSV estimation method will facilitate its use in large-scale breeding programs.

Since MSV is derived from marker effects, a genetic map, and phased genotypes, these data could bias estimates of MSV (Heffner et al., 2009; Druet et al., 2014; Lehermeier et al., 2017; Larkin et al., 2019). Furthermore, the assumptions made regarding the recombination rates (Haldane, 1919; Kosambi, 1943) derived from a genetic map could also bias the estimate of MSV (Bauer et al., 2013; Lehermeier et al., 2017). Thus, accurate marker effects, a genetic map, and phased genotypes are required for an unbiased prediction of MSV.

MSV differs between parents due to variation in the heterozygosity and linkage phase of their markers; thus, some parents produce more genetically diverse offspring than others (Segelke et al., 2014; Bonk et al., 2016). The use of MSV depends on the breeder's objectives. For example, a small MSV contributes to phenotypic uniformity among offspring desired by farmers who select bulls of ordinary cows. However, breeding companies that select breeding bulls aim for parents with high MSV to maximize the chance of breeding top-ranking offspring. MSV values are expected to vary between species. However, the effect of certain factors such as sex, selection, and birth year on a species' MSV

values is unknown. The heritability of MSV is also unknown. Thus, understanding MSV heritability and the factors that influence its values may aid in the efficient use of MSV in breeding programs.

MSV-based breeding decisions

The thesis briefly mentioned the use of recently proposed selection indices to optimize the tradeoff between short- and long-term gains in an outbred population using an index composed of GEBV and MSV earlier (Santos et al., 2019; Bijma et al., 2020). While the GEBV component maximizes short-term gains, the MSV component maximizes genetic variability or the chance of breeding top-ranking offspring. Selection based on these indices resulted in higher genetic gains and variability than selection solely based on the GEBV. According to Bijma et al. (2020), in the Holstein dairy cattle population, the index could increase the probability of breeding top-ranking offspring and response to selection by 36% and 3.6%, respectively. Santos et al. (2019) obtained an additional genetic gain of up to 16% over ten generations in a simulation with a small genome. However, to facilitate the use of these indices in large-scale breeding programs, a more rapid method for estimating MSV is required. Additionally, there is a need to hedge haplotype diversity because these indices tend to select similar parents with a high potential for genetic variability under high selection intensities, potentially resulting in unintended loss of favorable haplotypes.

To hedge haplotype diversity, a measure of similarities between Mendelian sampling values of parents is required. These similarities between parents could be assembled into a matrix where off-diagonal elements represent similarities between parents, and diagonal elements represent a parent's similarity to itself, which equals its MSV. Similar to how covariance matrices of asset prices are used in finance to create diversified portfolios (Markowitz, 1952), such a similarity matrix could be used to optimize the selection of parents with diverse haplotypes. However, such a matrix does not exist.

Therefore, the objectives of this thesis were 1) to introduce a similarity matrix based on Mendelian sampling values of parents for hedging haplotype diversity; 2) to introduce a fast analytical approach for estimating MSV; 3) to develop a software for estimating MSV and similarity matrix; and 4) to determine the heritability of MSV and compute a similarity matrix for a large commercial cattle population and determine the factors influencing the values of the matrix.

Content of the thesis

Chapter 1 conceptualizes the potential benefits of including Mendelian sampling values in breeding decisions. The derivation of similarity matrices and their application for improving genetic gain while maintaining haplotype diversity within each parental generation are conceptualized.

Chapter 2 derives a similarity matrix based on Mendelian sampling values of potential parents and demonstrates its possible breeding applications. An equivalent but computationally fast representation of an analytical method for computing MSV for gametes is derived. A similarity matrix based on Mendelian sampling values of gametes is also derived using the elements of this representation. The concepts were generalized to zygotes and multiple traits. Finally, the applications of the similarity matrix for visualization of population diversity, mate selection and allocation, and hedging haplotype diversity are demonstrated using empirical German Holstein data of 265 cows.

Chapter 3 develops a Python package (*PyMSQ*) to facilitate the widespread use of Mendelian sampling values in breeding programs. The relevance and applicability of MSV and similarity matrix in large-scale breeding programs are demonstrated using empirical and simulated data. Finally, the performance of *PyMSQ* and a previously published Fortran program for estimating MSV was benchmarked. The results showed that *PyMSQ* was up to 240 times faster depending on the number of markers and parents.

Chapter 4 provides answers to questions regarding the similarity matrix. The similarity matrices for four example traits, including longevity and milk urea nitrogen, were computed using an empirical German Holstein population of 71,163 individuals. Heritability for MSV was estimated, and factors affecting the similarity matrix were evaluated. This chapter presents novel information on MSV and the similarities between Mendelian sampling values, which will aid their efficient use in breeding programs.

LITERATURE CITED

Akdemir, D., and J. I. Sánchez. 2016. Efficient breeding by genomic mating. *Front. Genet.* 7:1–12.

Bauer, E., M. Falque, H. Walter, C. Bauland, C. Camisan, L. Campo, N. Meyer, N. Ranc, R. Rincent, W. Schipprack, T. Altmann, P. Flament, A. E. Melchinger, M. Menz, J. Moreno-González, M. Ouzunova, P. Revilla, A. Charcosset, O. C. Martin, and C. C. Schön. 2013. Intraspecific variation

of recombination rate in maize. *Genome Biol.* 14.

Bernardo, R. 2014. Genomewide selection of parental inbreds: Classes of loci and virtual biparental populations. *Crop Sci.* 54:2586–2595.

Bijma, P., Y. C. J. Wientjes, and M. P. L. Calus. 2020. Breeding Top Genotypes and Accelerating Response to Gametic Variance. *Genetics*. 214:91–107.

Bonk, S., M. Reichelt, F. Teuscher, D. Segelke, and N. Reinsch. 2016. Mendelian sampling covariability of marker effects and genetic values. *Genet. Sel. Evol.* 48:1–11.

Daetwyler, H. D., M. J. Hayden, G. C. Spangenberg, and B. J. Hayes. 2015. Selection on optimal haploid value increases genetic gain and preserves more genetic diversity relative to genomic selection. *Genetics*. 200:1341–1348.

Dempfle, L. 1990. Problems in the Use of the Relationship Matrix in Animal Breeding. In: D. Gianola and K. Hammond, editors. Advances in statistical methods for genetic improvement of livestock. 1st ed. Springer, New York. p. 454–473.

Druet, T., I. M. Macleod, and B. J. Hayes. 2014. Toward genomic prediction from whole-genome sequence data: Impact of sequencing design on genotype imputation and accuracy of predictions. *Heredity (Edinb).* 112:39–47.

Gaynor, R. C., G. Gorjanc, A. R. Bentley, E. S. Ober, P. Howell, R. Jackson, I. J. Mackay, and J. M. Hickey. 2017. A two-part strategy for using genomic selection to develop inbred lines. *Crop Sci.* 57:2372–2386.

Goddard, M. 2009. Genomic selection: Prediction of accuracy and maximisation of long term response. *Genetica*. 136:245–257.

Goddard, M. E., and B. J. Hayes. 2007. Genomic selection. *J. Anim. Breed. Genet.* 124:323–330.

Goiffon, M., A. Kusmec, L. Wang, G. Hu, and P. S. Schnable. 2017. Improving response in genomic selection with a population-based selection strategy: Optimal population value selection. *Genetics*. 206:1675–1682.

Gorjanc, G., R. C. Gaynor, and J. M. Hickey. 2018. Optimal cross selection for long-term genetic gain in two-part programs with rapid recurrent genomic selection. *Theor. Appl. Genet.* 131:1953–1966.

Habier, D., R. L. Fernando, K. Kizilkaya, and D. J. Garrick. 2011. Extension of the bayesian alphabet for genomic selection. *BMC Bioinformatics*. 12:186.

Haldane, J. B. S. 1919. The combination of linkage values and the calculation of distances between the loci of linked factors. *J. Genet.* 8:299–309.

Hayes, B. J., P. J. Bowman, A. J. Chamberlain, and M. E. Goddard. 2009. Genomic selection in dairy

cattle: progress and challenges. *J. Dairy Sci.* 92:433–443.

Heffner, E. L., M. E. Sorrells, and J. L. Jannink. 2009. Genomic selection for crop improvement. *Crop Sci.* 49:1–12.

Henderson, C. R. 1984. Applications of Linear Models in Animal Breeding. University of Guelph, Ontario, Canada.

Hill, W. G. 2014. Applications of population genetics to animal breeding, from wright, fisher and lush to genomic prediction. *Genetics*. 196:1–16.

Jannink, J. L. 2010. Dynamics of long-term genomic selection. *Genet. Sel. Evol.* 42.

Kemper, K. E., P. J. Bowman, J. E. Pryce, B. J. Hayes, and M. E. Goddard. 2012. Long-term selection strategies for complex traits using high-density genetic markers. *J. Dairy Sci.* 95:4646–4656.

Kennedy, B. W., L. R. Schaeffer, and D. A. Sorensen. 1988. Genetic Properties of Animal Models. *J. Dairy Sci.* 71:17–26.

Kosambi, D. D. 1943. the Estimation of Map Distances From Recombination Values. *Ann. Eugen.* 12:172–175.

Larkin, D. L., D. N. Lozada, and R. E. Mason. 2019. Genomic selection—considerations for successful implementation in wheat breeding programs. *Agronomy*. 9.

Lehermeier, C., S. Teyssèdre, and C. C. Schön. 2017. Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. *Genetics*. 207:1651–1661.

Lin, Z., N. O. I. Cogan, L. W. Pembleton, G. C. Spangenberg, J. W. Forster, B. J. Hayes, and H. D. Daetwyler. 2016. Genetic Gain and Inbreeding from Genomic Selection in a Simulated Commercial Breeding Program for Perennial Ryegrass. *Plant Genome*. 9:1–12.

Liu, H., T. H. Meuwissen, A. C. Sørensen, and P. Berg. 2015. Upweighting rare favourable alleles increases long-term genetic gain in genomic selection programs. *Genet. Sel. Evol.* 47:19.

Lynch, M., and B. Walsch. 1998. Genetics and Analysis of Quantitative Traits. Sinauer Associates, Sunderland, MA.

Markowitz, H. 1952. Portfolio selection. *J. Finance*. 7:77–91.

Meuwissen, T. H. E. 1997. Maximizing the Response of Selection with a Predefined Rate of Inbreeding. *J. Anim. Sci.* 75:934–940.

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard. 2001. Prediction of total genetic value using genome-wide dense marker maps. *Genetics*. 157:1819–1829.

Moeinizade, S., G. Hu, L. Wang, and P. S. Schnable. 2019. Optimizing selection and mating in

genomic selection with a look-ahead approach: An operations research framework. *G3 Genes, Genomes, Genet.* 9:2123–2133.

Mohammadi, M., T. Tiede, and K. P. Smith. 2015. Popvar: A genome-wide procedure for predicting genetic variance and correlated response in biparental breeding populations. *Crop Sci.* 55:2068–2077.

Müller, D., P. Schopp, and A. E. Melchinger. 2018. Selection on expected maximum haploid breeding values can increase genetic gain in recurrent genomic selection. *G3 Genes, Genomes, Genet.* 8:1173–1181.

Santos, D. J. A., J. B. Cole, T. J. Lawlor, P. M. VanRaden, H. Tonhati, and L. Ma. 2019. Variance of gametic diversity and its application in selection programs. *J. Dairy Sci.* 102:5279–5294.

Schaeffer, L. R. 2006. Strategy for applying genome-wide selection in dairy cattle. *J. Anim. Breed. Genet.* 123:218–223.

Schnell, F. W., and H. F. Utz. 1976. F1-Leistung und Elternwahl in der Züchtung von Selbstbefruchtern. *Bericht über die Arbeitstagung der Vereinigung Österreichischer Pflanzenzüchter, pp. 234–258, Gumpenstein, Österreich*.

Segelke, D., F. Reinhardt, Z. Liu, and G. Thaller. 2014. Prediction of expected genetic variation within groups of offspring for innovative mating schemes. *Genet. Sel. Evol.* 46:42.

VanRaden, P. M., C. P. Van Tassell, G. R. Wiggans, T. S. Sonstegard, R. D. Schnabel, J. F. Taylor, and F. S. Schenkel. 2009. Invited review: reliability of genomic predictions for North American Holstein bulls. *J. Dairy Sci.* 92:16–24.

Wang, Y., J. Bennewitz, and R. Wellmann. 2017. Novel optimum contribution selection methods accounting for conflicting objectives in breeding programs for livestock breeds with historical migration. *Genet. Sel. Evol.* 49.

Wang, Y., G. Lin, C. Li, and P. Stothard. 2016. Genotype Imputation Methods and Their Effects on Genomic Predictions in Cattle. *Springer Sci. Rev.* 4:79–98.

Wellmann, R. 2019. Optimum contribution selection for animal breeding and conservation: The R package optiSel. *BMC Bioinformatics*. 20:1–13.

Woolliams, J. A., P. Berg, B. S. Dagnachew, and T. H. E. Meuwissen. 2015. Genetic contributions and their optimization. *J. Anim. Breed. Genet.* 132:89–99.

Wray, N. R., and M. E. Goddard. 1994. Increasing long-term response to selection. *Genet. Sel. Evol.* 26:431–451.

Xu, S. 2003. Estimating polygenic effects using markers of the entire genome. *Genetics*. 163:789–801.

CHAPTER 1

Neue Kriterien für die genomische Selektion

N. Reinsch und **A. A. Musa**

Forschungsinstitut für Nutztierbiologie (FBN), Institut für Genetik und Biometrie, Wilhelm-Stahl-Allee 2, 18196 Dummerstorf

Originally published in *Züchtungskunde* 2021; 93(3): 201-209

The version of record is available online at:

https://www.zuechtungskunde.de/Archiv/Neue-Kriterien-fuer-die-genomische-Selektion,QUlEPTY4OTIwMjUmTUlEPTY5MTU4.html

Neue Kriterien für die genomische Selektion

ZUSAMMENFASSUNG

Selektionskandidaten zeigen ein unterschiedliches Ausmaß von Variation hinsichtlich der additiv-genetischen Werte ihrer Gameten. Diese Varianz lässt sich in vielen Zuchtprogrammen aus vorhandenen Markerinformationen für alle genotypisierten Tiere berechnen. Hierzu werden spezielle Matrizen eingesetzt, welche die Rekombinationsraten zwischen Markern wiederspiegeln. In der Züchtung verspricht man sich durch Beachtung dieser Varianzen verbesserte Möglichkeiten besonders hervorragende Zuchttiere zu erhalten, sowie einen erhöhten Zuchtfortschritt. Mit Hilfe der gleichen Matrizen lassen sich außerdem Ähnlichkeiten von potentiellen Eltern ableiten. Daraus ergeben sich in der Anpaarungsplanung neue Möglichkeiten zum Erhalt einer möglichst großen Vielfalt von Haplotypen in jeder Elterngeneration.

Schlüsselwörter: Gametische Varianz, Kopplungsungleichgewicht, Haplotypen, Optimale Anpaarung

SUMMARY

New criteria in genomic selection

Additive-genetic values of gametes show different degrees of variability in selection candidates. This gametic variance can be predicted for all genotyped individuals from marker information available in many breeding programs. To this end, special matrices are employed, reflecting recombination rates between markers. If breeding organizations consider these gametic variances in selection decisions, they may achieve higher probabilities for top ranking animals and more genetic progress. The same matrices allow for a measure of similarity between potential parents, useful for preserving a higher diversity of haplotypes in each parent generation.

Keywords: gametic variance, linkage disequilibrium, haplotypes, optimized matings

1 EINLEITUNG

Klassisches Selektionskriterium in der Tierzüchtung ist der geschätzte (Gesamt-) Zuchtwert. Die Schätzung beruht dabei auf den vorhandenen genomischen und phänotypischen Informationen, sowie dem Pedigree. Zur Zucht ausgewählt werden demgemäß in jeder Selektionsrunde die Kandidaten mit den besten Zuchtwerten, wenn Abschnittsselektion angewandt wird (Abb. 1a). Um den Inzuchtanstieg zu begrenzen kann mit Hilfe von *optimum contribution selection* (Meuwissen, 1997; Abb. 1b) dafür eine obere Schranke gesetzt werden.

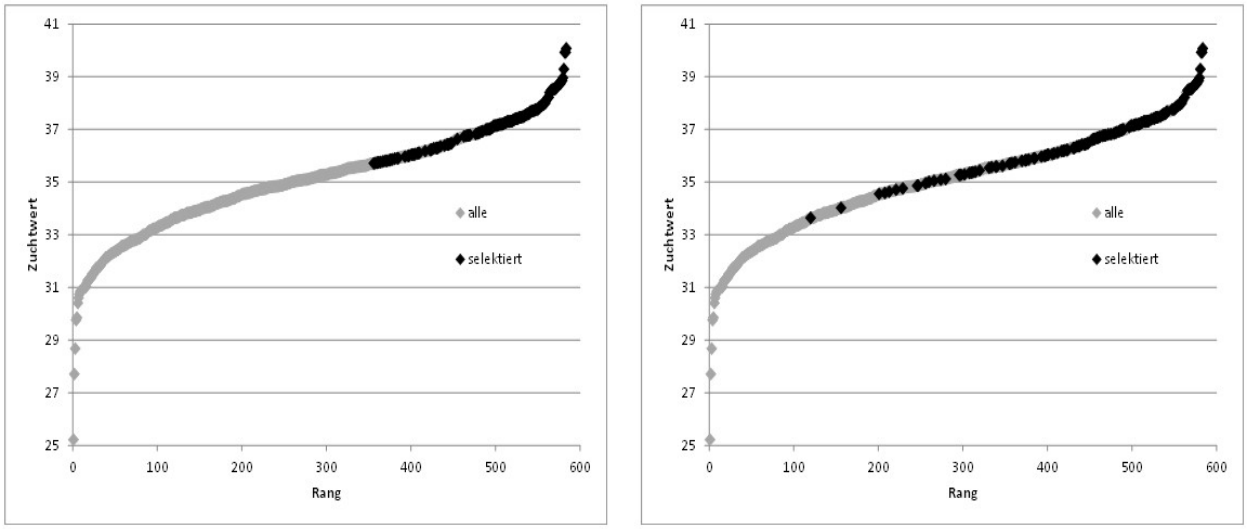


Abb. 1. Beide Kurven zeigen die geschätzten Zuchtwerte für alle Selektionskandidaten aus einer Generation in der Dummerstorfer Mäuselinie DU6 für das Merkmal Körpermasse am 42. Lebenstag. Die Zuchtwerte sind nach ihrer Höhe rangiert, ein hoher Rang (Abszisse) zeigt einen hohen Zuchtwert an (Ordinate). Links: Bei Abschnittsselektion werden die Tiere mit den besten Zuchtwerten und höchsten Rangziffern zur Zucht ausgewählt (dunkle Symbole). Rechts: Wird die Verwandtschaft der Selektionskandidaten berücksichtigt (*optimum contribution selection*), so werden einige Tiere in der Reihenfolge übersprungen, um den Inzuchtanstieg zu begrenzen.

Both curves represent the estimated breeding values of all selection candidates from one generation in the Dummerstorf mouse line DU6. The trait is body mass at day 42. Estimated breeding values are ranked according to size, a high rank (abscissa) indicating a high estimate (ordinate). Left: Under truncation selection, animals with the highest ranking estimates are selected as parents of the next generation (dark symbols). Right: By considering relationships between candidates in optimum contribution selection, some of the higher ranking candidates are left out in order to restrict the future increase in inbreeding.

In der nächsten Generation entspricht der erwartete Zuchtwert der Nachkommen dann dem durchschnittlichen Zuchtwert ihrer Eltern. Der Zuchtwert eines einzelnen Nachkommen weicht von seinem Elterndurchschnitt ab um die Mendelsche Zufallsabweichung (*Mendelian sampling term*). Wenn beide Eltern nicht ingezüchtet sind, dann tragen der Elterndurchschnitt und diese Abweichung gleich viel zur gesamten Varianz der Nachkommenzuchtwerte bei. Die Zufallshälfte des Zuchtwertes sorgt also für die genetischen Unterschiede zwischen den Geschwistern einer Familie.

Sind nur Pedigreeinformationen verfügbar, dann existieren keine weiteren Hinweise auf Unterschiede für die Größe der Zufallsstreuung in verschiedenen Familien, abgesehen vom möglicherweise unterschiedlichen Inzuchtgrad der Eltern. Anders in Zuchtprogrammen mit genomischer Selektion, wo Markerdaten angesammelt werden. Mit deren Hilfe ist eine Vorausschätzung der innerfamiliären Zufallsstreuung möglich, auch völlig ohne Nachkommen. Dies kann geschehen für einzelne Selektionskandidaten (Zufallsstreuung der additiven Werte der Gameten des Kandidaten; gametische Varianz) ebenso wie für Elternpaare (Zufallsstreuung der Zuchtwerte potentieller Nachkommen).

Im Folgenden soll ein Überblick darüber gegeben werden, wie unterschiedliche Mendelsche Zufallsabweichungen von Selektionskandidaten zukünftig in die Züchtung Eingang finden könnten.

2 GENETISCHE ZUFALLSSTREUUNG UND KOVARIANZEN VON MARKERN

2.1 Marker geben Auskunft über die Zufallsstreuung von Nachkommen

Die Abbildung 2 zeigt ein Beispiel mit zwei Tieren, die beide für zwei Marker typisiert sind. Beide Tiere sind außerdem heterozygot (Allele 1 und 2) an beiden Markern, haben also den gleichen Genotyp. Weiter wird angenommen, dass an beiden Markern das 1er-Allel einen additiven Effekt von +1 und das 2er Allel einen additiven Effekt von -1 hat. Beide Tiere erzeugen Gameten, die einen von drei möglichen additiven Werten (Summe der Effekte aller Marker auf einem Gameten) annehmen können: +2, 0 oder -2 (Abb. 2). Unterschiedlich sind jedoch die Häufigkeiten dieser verschiedenen Gameten, die Häufigkeitsverteilung hängt ab von der Rekombinationsrate zwischen den Markern. Als Konsequenz ergibt sich für Tier Eins eine Varianz der additiven Gametenwerte von 0,32 und für Tier Zwei eine Varianz von nur 0.08. Die Mendelsche Zufallsvarianz der Gameten von Tier Zwei ist also nur halb so groß wie die von Tier Eins. Aus einer Paarung der beiden Tiere ergeben sich

Nachkommen, deren Zuchtwerte eine Mendelsche Zufallsabweichung von 0,40 (0,32+0,08) aufweisen.

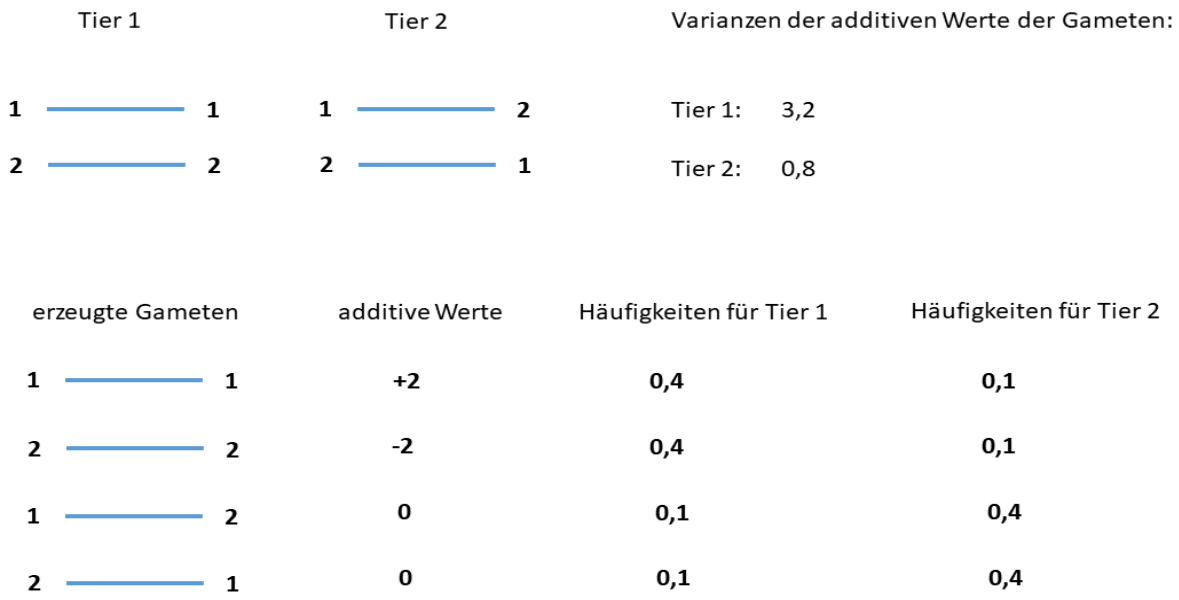


Abb. 2. Das Beispiel zeigt zwei Tiere mit gleichem doppelt heterozygotem Genotyp für zwei Marker. Die Marker sind unterschiedlich in Haplotypen angeordnet. Es wird angenommen, dass alle 1er-Allele einen Effekt von +1 haben und 2er Allele einen von -1. Beide Tiere erzeugen die gleichen Gametentypen, aber in unterschiedlicher Häufigkeit. Daraus ergeben sich unterschiedliche Varianzen von 0,32 für die additiven Werte der Gameten von Tier 1 und von 0,8 für die Gameten von Tier 2.

There are two animals in the example with an identical double heterozygous genotype. Alleles at the two markers are, however, differently arranged in haplotypes. An additive effect of +1 is assumed for each 1-allele and effect of -1 for each 2-allele. Both animals produce the same kinds of gametes, yet with different frequency distributions. In consequence, the gametic variances differ and are 0.32 for animal 1 and 0.8 for animal 2.

Wie das Beispiel in Abb. 2 zeigt, müssen drei verschiedene Arten von Information zusammengeführt werden, um die Mendelsche Zufallsvarianz für einzelne Selektionskandidaten zu berechnen: gute Schätzwerte für die einzelnen Markereffekte, Markergenotypen, die anhand ihrer Kopplungsphasen in Haplotypen zerlegt werden können, sowie die Kenntnis der Rekombinationsraten zwischen den Markern (genetische Karte). Insbesondere für Zuchtprogramme in den wichtigen Milchviehrassen, wo genomische Selektion schon seit etwa einem Jahrzeht angewandt wird, sind gute Schätzwerte für

Markereffekte vorhanden und die Zerlegung von Markergenotypen in Haplotypen wird unter Nutzung von Genotypen von Verwandten und Kenntnissen über Haplotypenhäufigkeiten in der Population zuverlässig routinemäßig durchgeführt. Eine genetische Karte für Holstein-Rinder, die auf umfangreichen aktuellen Daten aus dem Rechenzentrum vit beruht, wurde erst kürzlich veröffentlicht (Qanbari and Wittenburg, 2020). Auf der Datenseite sind die Voraussetzungen für die Berechnung von Mendelschen Zufallsabweichungen also in vielen wichtigen Populationen gegeben.

Bei der praktischen Umsetzung mit ca. 40 000 Markern oder mehr ergibt sich allerdings die Schwierigkeit, dass anders als in kleinen Beispielen nicht mehr alle möglichen Gameten mit ihren Wahrscheinlichkeiten und additiven Werten abgeleitet werden können, um daraus dann die Mendelsche Zufallsvarianz zu berechnen. Die Anzahl dieser Gameten wäre einfach sehr viel zu groß. Als Ausweg wurden zunächst Simulationen eingesetzt, um aus einer ausreichend großen Stichprobe aus allen möglichen Gameten (z.B. 10 000) die Mendelsche Zufallsstreuung näherungsweise zu berechnen (Cole and VanRaden, 2011; Segelke et al., 2014). Eine exakte Berechnung ist möglich mit Hilfe einer LD-Matrix, die das Kopplungsungleichgewicht (*linkage disequilibrium, LD*) von Markern innerhalb von Familien darstellt. Diese Matrix kann auch als Kovarianzmatrix von Markereffekten aufgefasst werden (Bonk et al., 2016). Die Anzahl der Zeilen und Spalten der LD-Matrix ist gleich der Anzahl der informativen Marker. Sie spiegelt die Rekombinationswahrscheinlichkeiten zwischen allen Markern wider und wird aus der genetischen Karte berechnet. Der bei der Simulation wegen der begrenzten Stichprobengröße nicht ganz zu vermeidende Fehler kann bei Anwendung der LD-Matrix komplett ausgeschaltet werden (Bonk et al., 2016). Ein Programm zur Aufstellung von LD-Matrizen wurde vor kurzem veröffentlicht (De Abreu Santos et al., 2020), wobei eine lineare Approximation für die Kovarianz zwischen je zwei Markern eingesetzt wird.

In mehreren Arbeiten (Segelke et al., 2014; Bonk et al., 2016; Santos et al., 2019) konnte gezeigt werden, dass für verschiedene Merkmale die Mendelschen Zufallsvarianzen von Gameten eine deutliche Schwankungsbreite aufweisen. Selektionskandidaten unterscheiden sich also hinsichtlich der Zufallsvarianz ihrer Gameten. Das gleiche gilt für Elternpaare hinsichtlich der Zufallsvarianz der Zuchtwerte ihrer Nachkommen. Dies hat dazu geführt, dass verschiedene Ansätze vorgeschlagen wurden, um zusätzlich zum geschätzten Zuchtwert die Mendelsche Zufallsvarianz bei der Selektion und/oder bei der Auswahl von Paarungspartnern mit ins Kalkül zu ziehen.

2.2 Anwendung in der Züchtung – neue Selektionskriterien

Mehrere Autoren haben darauf hingewiesen, dass Tiere mit gleichem Zuchtwert, aber unterschiedlicher gametischer Varianz sich in der Wahrscheinlichkeit unterscheiden, mit der sie Nachkommen mit einem ausgesprochenen Spitzenzuchtwert erzeugen (Segelke et al., 2014; Bonk et al., 2016; Santos et al., 2019). Bei gleichwertigen Anpaarungspartnern wären also etwa Bullen mit höherer gametischer Varianz von Vorteil für Zuchtorganisationen (Abb. 3). Um diese Wahrscheinlichkeit zu berechnen, muss von einer vorher festgelegten Schwelle ausgegangen werden, ab der man von einem „Spitzennachkommen“ (oder „Spitzengameten“) sprechen möchte.

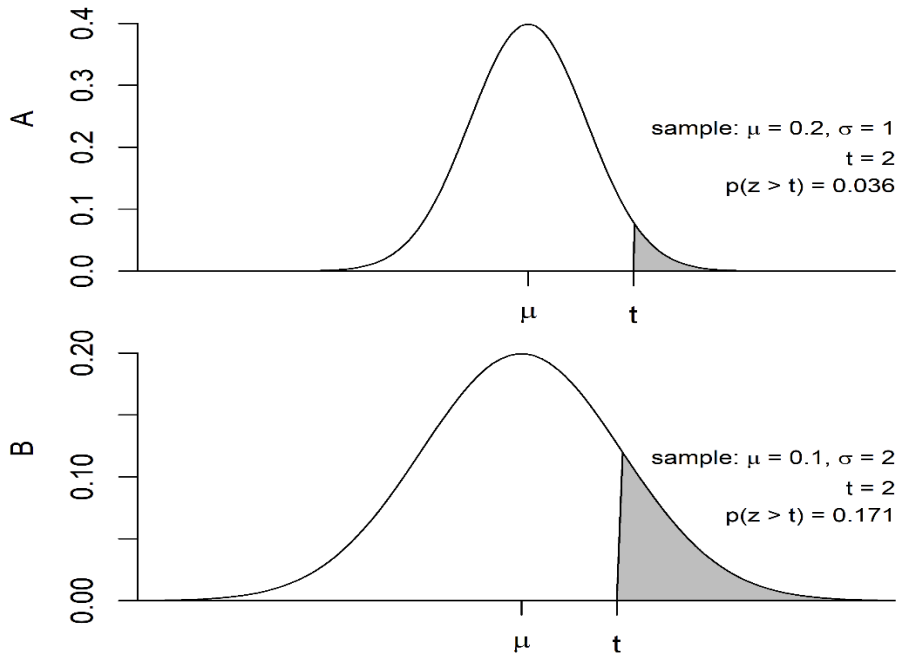


Abb. 3. Verteilungen der additiven Werte für die Gameten zweier hypothetischer Bullen. Der obere Bulle besitzt zwar einen etwas höheren Mittelwert (Zuchtwert), wegen der geringeren Variabilität seiner Gameten aber eine geringere Wahrscheinlichkeit Spitzengameten mit einem additive Wert über der Schwelle zu erzeugen. Der zweite Bulle (unten) kann durch die höhere Variabilität seiner Gameten seinen etwas niedrigeren Zuchtwert kompensieren und besitzt eine höhere Wahrscheinlichkeit für Spitzengameten und ebensolche Nachkommen.

Distributions of additive values for gametes of two hypothetical bulls. The first bull (upper panel) has a slightly higher mean (breeding value), but due to its lower gametic variance, the probability of a top gamete above the threshold is lower compared to the distribution for gametes from the second bull (lower panel). The second bull thus compensates for its lower breeding value by its larger gametic variability.

Ähnliche Überlegungen führten zum Vorschlag einen Index aus den Schätzwerten für den Zuchtwert und der gametische Standardabweichung als Selektionskriterium in der Tierzüchtung zu nutzen, wie es auch für die Pflanzenzucht diskutiert wurde (siehe Bijma et al., 2020 und die dort angegebene Literatur). In einer Simulationsstudie konnte bei sehr starken Selektionsintensitäten durch einen solchen Index die Häufigkeiten von Spitzenbullen erheblich (um über 30%) gesteigert werden im Vergleich zu einer Selektion nur nach Zuchtwert. Der Selektionserfolg in der Gesamtpopulation stieg ebenfalls leicht an (ca. 3%). Beide Verbesserungen können zu niedrigen Kosten erreicht werden, wenn die nötigen Informationen bereits vorhanden sind, so wie es für die wichtigen Milchviehrassen der Fall ist.

Das Ausmaß des erreichbaren Zusatznutzens hängt dabei davon ab, wie sehr das Ausmaß der gametischen Variabilität sich zwischen den Individuen unterscheidet. Wie gut solche Schwankungen erfasst werden können, wird dabei auch durch die Genauigkeit beeinflusst, mit der Markereffekte geschätzt werden können (Santos et al., 2019).

2.3 Ähnlichkeit von Eltern hinsichtlich der genetischen Streuung ihrer Nachkommen

Hat man die Wahl zwischen mehreren Eltern (z.B. Bullen), die sich im durchschnittlichen Zuchtwert ihrer Nachkommen und auch in ihren Streuungen nicht sehr unterscheiden, so stellt sich die Frage welche der Bullen in welchem Ausmaß eingesetzt werden sollten. Dazu wäre es hilfreich, ein Maß für die Ähnlichkeit der Streuungen in den einzelnen Halbgeschwisterfamilen zu haben. Ein solches Ähnlichkeitsmaß kann wieder mit Hilfe der oben erwähnten LD-Matrizen berechnet werden. Die Ähnlichkeit ist umso höher, je mehr heterozygote Marker mit großen Effekten und gleicher Kopplungsphase bei beiden Bullen vorhanden sind (Musa and Reinsch, 2020). Zwei identische Zwillinge hätten dann die größtmögliche Ähnlichkeit, bei völligem Fehlen einer Übereinstimmung wäre die Ähnlichkeit Null. Analog zur Umwandlung einer Kovarianz in eine Korrelation lassen sich Ähnlichkeiten standardisieren, so dass die standardisierten Ähnlichkeiten im Bereich zwischen Null und Eins liegen. Abb. 3 zeigt eine Ähnlichkeitsmatrix für das Merkmal tägliche Milchmenge und zehn Tiere in ihrer ursprünglichen und standardisierten Form. Als Anwendungen solcher Ähnlichkeitsmatrizen kommen einerseits die Quantifizierung und Visualisierung von Ähnlichkeiten

in Betracht, zum anderen die optimale Auswahl von Anpaarungen aus einer größeren Menge von möglichen Anpaarungen.

2.4 Optimale Anpaarungsportfolios

Die Ähnlichkeitsmatrix versetzt uns in die Lage eine bestimmte Anzahl von Anpaarungen auszuwählen, die einerseits einen möglichst hohen Zuchtwert der Nachkommen erwarten lässt (oder einen möglichst günstigen Wert einer anderen gewählten Zielfunktion) und auf der anderen Seite keine größere Ähnlichkeit zwischen den angepaarten Tieren in Kauf nimmt. Damit zielt man darauf ab, in jeder Anpaarungsrunde möglichst unterschiedliche Haplotypen im Spiel zu halten, ohne interessante Varianten unabsichtlich auszuschließen, weil die entsprechenden Eltern nicht angepaart werden.

Die optimale Auswahl von Anpaarungen zu einem Anpaarungsportfolio (ähnlich einem Wertpapierportfolio) kann mit Hilfe der ursprünglich aus der Finanzmathematik bekannten Portfoliotheorie erfolgen. Nur solche Anpaarungsportfolios werden in Betracht gezogen, die auf der sog. Effizienzlinie liegen (Abb. 4). Damit ist gewährleistet, dass es kein Anpaarunsportfolio mit gleichwertiger Ähnlichkeit zwischen den Eltern aber höherem durchschnittlichen Nachkommenzuchtwert gibt, oder, anders ausgedrückt, für einen bestimmten Nachkommenzuchtwert die geringsmögliche Ähnlichkeit erreicht wird. Zur Berechnung dieser Effizienzlinie wird die Ähnlichkeitsmatrix eingesetzt, wodurch man den beschriebenen „diversifizierenden“ Effekt erzielt und die Ähnlichkeit der Eltern hinsichtlich ihrer merkmalsrelevanten Haplotypenabschnitte möglichst klein halten kann.

Die Analogie zur bestmöglichen Auswahl von Wertpieren für ein Anlageportfolio ist natürlich rein formal. Um bei Investitionsentscheidungen die Wahrscheinlichkeit von gleichzeitigen (insbesondere negativen) Kursbewegungen von allen ausgewählten Wertpapieren zu minimieren wird eine Kovarianzmatrix für deren Kurse ermittelt und in die Optimierung eingesetzt. Der diversifizierende Effekt führt dann zur Auswahl von gering korrelierten Papieren, deren Kurse sich vergleichsweise unabhängig entwickeln (zumindest lassen Daten aus der Vergangenheit dies vermuten). Die Ähnlichkeitsmatrix für Anpaarungen hat als Diagonalelemente innerfamiliäre Streuungen (Mendelsche Zufallsvarianzen) für alle in Betracht gezogenen Anpaarungen, alle anderen Elemente lassen sich wie Kovarianzen interpretieren. Daher kann die Ähnlichkeitsmatrix in der

Anpaarungsplanung ihre diversifizierende Wirkung auf die elterliche Haplotypen entfalten, ganz wie eine Kovarianzmatrix für Wertpapierkurse (Abb. 5).

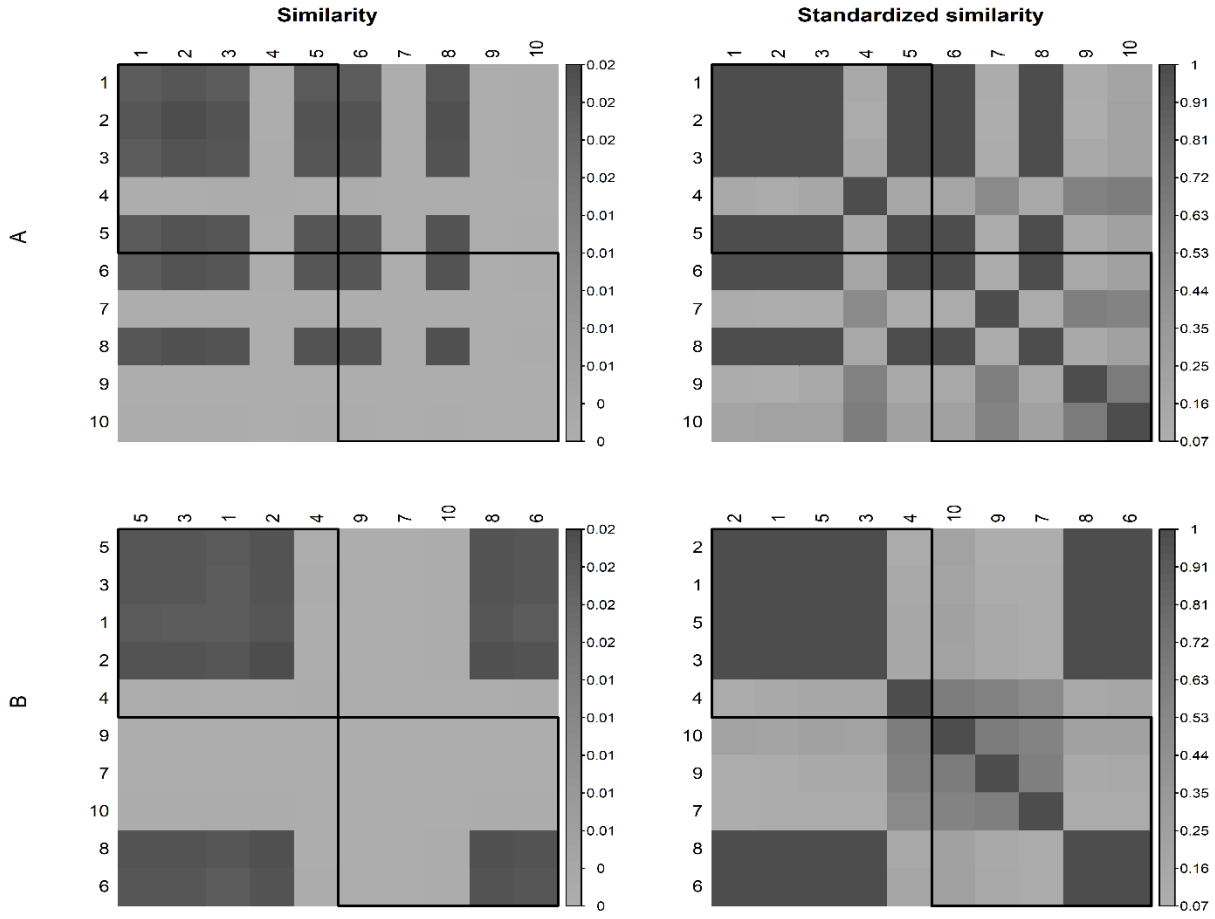


Abb. 4. Eine Ähnlichkeitsmatrix für zehn Tiere (oben links) aus zwei Halbgeschwisterfamilien (begrezt durch Trennlinien). Die Diagonalelemente geben die Varianz der Gameten für jedes Tier an, die Nebendiagonalelemente lassen sich als gemeinsame gametische Varianz zweier Tiere interpretieren. Standardisierte Ähnlichkeiten (oben rechts; alle Diagonalelemente sind Eins) lassen feinere Unterschiede in den Ähnlichkeiten besser hervortreten. Werden die Tiere innerhalb der Familien nach Ähnlichkeitsgruppen sortiert (unten links und rechts), so lässt sich erkennen, dass starke Ähnlichkeiten auch über Familien hinweg auftreten.

A similarity matrix for ten animals (upper left) drawn from two half-sib families (separated by lines). Diagonal elements are identical to Mendelian sampling variances for each animal, off-diagonals can be interpreted as a share of variance that is common to two animals. Standardized similarities (upper right; all diagonal elements equal one) make smaller differences between similarities more visible. Clustering animals by similarity within half-sib families (lower right and left) highlight strong similarities across families.

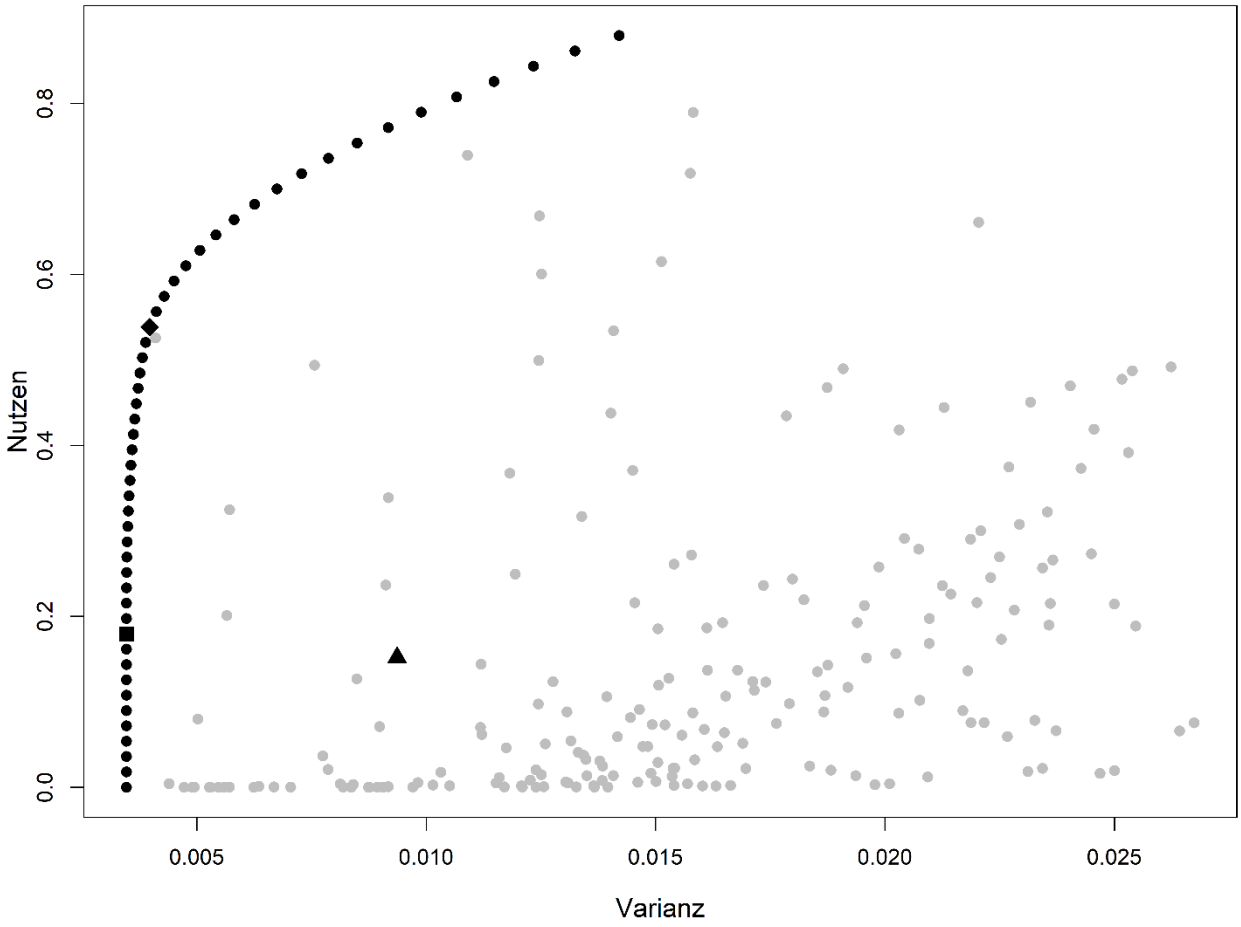


Abb. 5. Jeder graue Punkt repräsentiert eine Kombination (Portfolio) von Anpaarungen, die aus allen möglichen Paarungen in einer Selektionsrunde ausgewählt werden können. Bei gleichem erwarteten Nutzen (z.B. dem durchschittlichen Zuchtwert der resultierenden Nachkommen) unterscheiden sich diese Portfolios hinsichtlich ihrer Gesamtvarianz, die sich aus der Ähnlichkeitsmatrix ergibt. Optimale Portfolios liegen auf der Effizienzlinie (schwarze Symbole) und verbinden eine bestimmte Gesamtvarianz mit dem höchstmöglichen Nutzen. Dies ist gleichbedeutend mit der geringsmöglichen Ähnlichkeit der Eltern hinsichtlich ihrer Markerhaplotypen bei bestimmten Nutzen.

Each grey dot represents a combination (portfolio) of a limited number of matings chosen from all possible matings in a single round of selection. Portfolios with the same value of the goal function (e.g., average breeding value of all offspring) differ in their variance, which can be derived from the similarity matrix. Optimum portfolios are located on the efficient frontier (black symbols) and combine a certain value of the goal function with the smallest possible variance. This is equivalent to the smallest possible haplotype similarity of all parents in a portfolio for a given value of the goal function.

3 FAZIT

Mit Hilfe von LD-Matrizen lassen sich aus vorhandenen Markerinformationen genetische Streuungen von Gameten und Nachkommen berechnen. Indices, die Zuchtwerte und solche Streuungen kombinieren lassen einen erhöhten Zuchtfortschritt und eine höhere Wahrscheinlichkeit von herausragenden Zuchttieren erwarten. LD-Matrizen können außerdem dazu verwendet werden, die Auswahl von Anpaarungen so zu optimieren, dass unter den Elterntieren eine möglichst hohe Vielfalt an Haplotypen erhalten bleibt.

LITERATUR

De Abreu Santos, D. J., J. B. Cole, G. E. Liu, P. M. Vanraden, and L. Ma. 2020. Gamevar.f90: A software package for calculating individual gametic diversity. *BMC Bioinformatics*. 21:3–7.

Bijma, P., Y. C. J. Wientjes, and M. P. L. Calus. 2020. Breeding Top Genotypes and Accelerating Response to Gametic Variance. *Genetics*. 214:91–107.

Bonk, S., M. Reichelt, F. Teuscher, D. Segelke, and N. Reinsch. 2016. Mendelian sampling covariability of marker effects and genetic values. *Genet. Sel. Evol.* 48:1–11.

Cole, J. B., and P. M. VanRaden. 2011. Use of haplotypes to estimate Mendelian sampling effects and selection limits. *J. Anim. Breed. Genet.* 128:446–455.

Meuwissen, T. H. E. 1997. Maximizing the Response of Selection with a Predefined Rate of Inbreeding. *J. Anim. Sci.* 75:934–940.

Musa, A. A., and N. Reinsch. 2020. Similarities between Mendelian sampling distributions and their possible application in breeding. In: Book of Abstracts of the 71st the Annual Meeting of the European Federation of Animal Science (EAAP) held on December 1-4, virtual meeting, Portugal. pp 484.

Qanbari, S., and D. Wittenburg. 2020. Male recombination map of the autosomal genome in German Holstein. *Genet. Sel. Evol.* 52:73.

Santos, D. J. A., J. B. Cole, T. J. Lawlor, P. M. VanRaden, H. Tonhati, and L. Ma. 2019. Variance of gametic diversity and its application in selection programs. *J. Dairy Sci.* 102:5279–5294.

Segelke, D., F. Reinhardt, Z. Liu, and G. Thaller. 2014. Prediction of expected genetic variation within groups of offspring for innovative mating schemes. *Genet. Sel. Evol.* 46:42.

CHAPTER 2

A similarity matrix for hedging haplotype diversity among parents in genomic selection

A. A. Musa and N. Reinsch

Research Institute for Farm Animal Biology (FBN), Institute of Genetics and Biometry, 18196 Dummerstorf, Germany

Submitted

A similarity matrix for hedging haplotype diversity among parents in genomic selection

ABSTRACT

Background: Mendelian sampling variability (MSV), determined by the heterozygosity and linkage phase of the parental haplotypes, quantifies the chance of producing offspring with high breeding values. The recently proposed criteria for genomic selection combine the expected breeding values of offspring with MSV. These criteria, however, tend to select similar parents with high variability potential. In order to avoid this tendency and subsequently hedge diversity, a measure of haplotype similarity is required.

Results: In the first step of our derivation, all potential gametes from the two parents are matched into pairs based on segregating marker patterns. Subsequently, a similarity measure between parents is defined as the absolute value of the covariance between the additive values of the matching gametes on a chromosome. High similarity indicates that the parents share many heterozygous markers, with large effects on a trait in the same linkage phase. A similarity matrix, which has absolute covariances as off-diagonal elements and MSVs on the diagonal, summarizes all pairwise similarities among a group of parents. The similarity of a parent with itself is equal to its MSV. A previously defined linkage-disequilibrium matrix of genotype indicators enables all the matrix elements to be computed, even when there are tens of thousands of markers. The concept generalizes to multiple chromosomes, an aggregate genotype with multiple traits, and similarities between zygotes.

Conclusions: The similarity matrix helps select an optimal subset of matings from a larger possible set while balancing parental haplotype diversity with expected genetic gain in the next generation. However, the effects on genetic gain and variability over successive generations remain to be investigated in a breeding program with recurrent selection.

INTRODUCTION

Owing to the prospect of accelerating genetic gain by shortening the generation interval and increasing the accuracy of estimated breeding values of young candidates, genomic selection (GS) has become a popular and essential tool for breeders (Meuwissen et al., 2001; Heffner et al., 2009; Hickey et al., 2017). In classical GS, the selection criterion is the genomic estimated breeding value (GEBV), computed as the sum of all the estimated additive effects of all alleles that make up an individual's genotype (Meuwissen et al., 2001). Selecting young candidates as parents based on their GEBV maximizes the population mean of the next generation. Its continued use, however, also promotes inbreeding as a side effect (Schaeffer, 2006; Liu et al., 2015; Rutkoski et al., 2015; Lin et al., 2016). Imposing constraints on the coancestry of selected parents (Meuwissen, 1997; Meuwissen and Sonesson, 1998; Sonesson et al., 2012; Woolliams et al., 2015) acts as a countermeasure, leading to more genetic progress in the long term.

Proposed selection criteria other than GEBVs promise more substantial genetic gain over a longer sequence of generations with recurrent selection. These criteria may also better preserve genetic variability in the breeding population. A group of these selection criteria focuses on the additive values (GEBVs) of predefined chromosomal sections rather than individuals to achieve highly favorable combinations of chromosomal segments following generations of repeated segregation and selection. As Kemper et al. (2012) presented, genotype building considers the best of two haplotype blocks in a group of diploid individuals. Individuals that are more likely to produce superior progeny when crossed with each other are selected in each generation. Selection based on optimal haploid values (OHVs) (Daetwyler et al., 2015) (i.e., the sum of maximal segmental GEBVs) is a related proposed method by which individuals with the highest OHVs are chosen as parents. Similarly, when optimal population value selection (Goiffon et al., 2017) is applied, a set of parents is selected such that the best possible future offspring has the maximum additive value. Modification with a predefined fixed planning horizon is associated with look-ahead selection (Moeinizade et al., 2019), which maximizes the expected GEBV of the best progeny in the terminal generation.

The second group of proposed selection criteria also aims to provide an optimal tradeoff between short- and long-term gains but does not use chromosome segmentation. Instead, they share the common principle of an index that is the sum of the expected GEBV of offspring plus a constant multiplied by the within-family standard deviation of the GEBVs of future offspring. As with expected

GEBVs, these standard deviations are predicted from parental alleles and additive marker effects, either by simulating the segregation of parental haplotypes into gametes (Cole and VanRaden, 2011; Bernardo, 2014; Segelke et al., 2014; Mohammadi et al., 2015) or applying an analytical formula (Bonk et al., 2016; Santos et al., 2019). The usefulness criterion (Schnell and Utz, 1976; Lehermeier et al., 2017) and expected maximum haploid breeding value (Müller et al., 2018) are examples of plant breeding with doubled haploid lines as offspring. Bijma et al. (2020) investigated various indices and found greater genetic gain in simulated diploid animals than for the conventional genomic selection on GEBVs. In general, the constant in the standard deviation terms of these indices can be interpreted as selection intensities among future offspring, obtained either as an integral over the upper tail of the standard normal distribution or from order statistics (Müller et al., 2018) if only a limited number of offspring are considered.

As part of such indices, the within-family variation (Mendelian sampling variance, MSV) of additive genetic values quantifies the chance of obtaining offspring with high breeding values (Segelke et al., 2014; Bonk et al., 2016; Müller et al., 2018; Bijma et al., 2020). Therefore, parents with heterozygous genotypes at marker loci with large positive effects in coupling are more likely to be selected. However, breeders may be interested in keeping many varieties of haplotypes in play over succeeding generations to avoid unintentional losses of genetic variability and to allow new favorable haplotypes to emerge through recombination in later generations. The latter is essentially the same aim ingrained in the first group of selection criteria mentioned above by design. Combinations of GEBV and Mendelian standard deviation, on the other hand, lack this inherent mechanism.

As a solution, we propose a similarity matrix among parents in which off-diagonal elements reflect pairwise haplotype similarities and diagonal elements reflect a parent's similarity with itself, which equals the MSV of its gametes. Similar to how covariance matrices of asset prices are used in finance to create diversified portfolios (Markowitz, 1952), a similarity matrix of this kind can be used in breeding to select parents with diverse haplotypes.

Drawing upon previous results by Bonk et al. (2016), we first introduce a computationally faster representation of their method for predicting the MSVs of gametes with the help of a covariance matrix of genotype indicators reflecting within-family linkage disequilibrium (LD). Second, we derive a measure of pairwise haplotype similarity for single chromosomes that rely on the same LD matrix. The concept is then extended to multiple chromosomes, multiple traits, and zygotes. Finally, we

illustrate potential applications for selecting an optimized subset of parents from a larger group of candidates.

METHODS

In this section, we first review the method of Bonk et al. (2016) for calculating MSV and then present an equivalent but computationally faster representation. The elements of this representation are also useful for computing our similarity measure with large numbers of markers and for as many individuals as needed for potential applications to optimize mating decisions.

Mendelian Sampling Variance of Gametes

An analytical approach for estimating the MSV of gametes was presented by Bonk et al. (2016). Previous attempts (e.g., Segelke et al., 2014; Mohammadi et al., 2015) estimated MSV through simulation. Both approaches use recombination rates, marker effects, and phased genotypes of parents as information and, thus, are equivalent, except for Monte Carlo errors. The exact formula for MSV is related to the GEBV bv_i of individual i:

$$bv_i = \boldsymbol{c}_i'\boldsymbol{m} ,$$

where $\boldsymbol{c}_i'$ is a row vector of genotype indicators (1, 0, and -1 for respective genotypes AA, AB or BA, and BB at each locus) and $\boldsymbol{m}$ is a vector of population additive marker effects, most commonly allele substitution effects. Then, the MSV of the additive genetic values g_i of the gametes produced by a potential parent i can be computed, according to Bonk et al. (2016), as follows:

$$var\left(g_i\right) = \boldsymbol{m}'\boldsymbol{R}_i\boldsymbol{m} . \qquad (1)$$

Here, $\boldsymbol{R}_i$ is a parent-specific covariance matrix of additive genotype indicators in $\boldsymbol{c}_i'$ and is derived from the recombination rates and linkage phase of the parent i (see formula (4) below). $\boldsymbol{R}_i$ is a block-diagonal matrix with one block per chromosome. The diagonal elements of $\boldsymbol{R}_i$ for heterozygous loci are all equal (0.25). Signs of off-diagonal elements depend on the linkage phases of pairs of markers (Bonk et al., 2016; Santos et al., 2019). All rows and columns of $\boldsymbol{R}_i$ corresponding to homozygous markers are set to zero because they do not contribute to MSV.

The MSV of additive genetic values g_{ij} of zygotes from parents i and j, then, is the sum of the two independent contributions (Bonk et al., 2016) of the paternal and maternal gametes:

$$var\left(g_{ij}\right) = \boldsymbol{m}'\boldsymbol{R}_i\boldsymbol{m} + \boldsymbol{m}'\boldsymbol{R}_j\boldsymbol{m} \,. \tag{2}$$

Note that in a scenario with only additive effects, the MSVs for additive genetic values remain unaffected if half the differences between homozygotes are used as additive marker effects instead of allele substitution effects (Bonk et al., 2016; page 10).

Equivalent representations

We propose computing the MSVs of gametes produced by a parent i in an equivalent manner to that described above:

$$var\left(g_i\right) = \boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_i \tag{3}$$

where $\boldsymbol{m}_i$ is a parent-specific vector of additive marker effects incorporating the linkage phase of the markers, and $\boldsymbol{R}$ is a population covariance matrix derived from only recombination rates, thus reflecting the expected within-family linkage disequilibrium of markers. Using an appropriate mapping function, the elements ρ_{kl} of $\boldsymbol{R}$ can be expressed in terms of distances instead of recombination rates. Assuming Haldane's mapping function (Haldane, 1919), the element ρ_{kl} of $\boldsymbol{R}$ is calculated as follows:

$$\rho_{kl} = \frac{\exp\left(-2d_{kl}\right)}{4}, \tag{4}$$

where d_{kl} is the genetic distance between two markers in Morgan. The formula in Santos et al. (2019) assumes a linear relationship between recombination rate and genetic distance. Each block of matrix $\boldsymbol{R}$ corresponds to a chromosome and has an autoregressive variance structure with variable marker distances and 0.25 on its diagonal. Marker alleles are assumed to be biallelic. The standard allele at each locus k to which the population additive marker effect m_k has been assigned is coded as 1; otherwise, the allele code is 2. For each marker k, a phase-indicator variable is set to $\delta_{ik} = 1$ if the standard allele is observed on the first haplotype of individual i; otherwise, it is set to $\delta_{ik} = -1$. For any homozygous genotype, the indicator is set to zero $\delta_{ik} = 0$. Then, each additive marker effect m_k

in $\boldsymbol{m}$ is converted into an element m_{ik} in $\boldsymbol{m}_i$ by multiplying m_k by the respective indicator δ_{ik}, $m_{ik} = \delta_{ik} m_k$. Thus, the sign of the respective additive marker effect is retained as in $\boldsymbol{m}$ if the standard allele is found on the first haplotype; otherwise, the sign is the opposite, provided that the marker genotype is heterozygous. As homozygous loci do not contribute to MSV, they are assigned a value of zero m_{ik} as an entry in $\boldsymbol{m}_i$. See Appendices 1 and 2 for an illustration and proof of the equivalence of equations (1) and (3).

It is simple to add vector $\boldsymbol{m}_i$ to parents, as only single markers have to be explored, which is in contrast with the previous representation in which all possible marker-pairs on an individual's haplotype need to be explored to determine each element's sign in the individual LD-matrix $\boldsymbol{R}_i$. Instead, the population covariance matrix $\boldsymbol{R}$ (Eq. (3)) has to be set up only once, making the computation of MSVs faster than when using the previously presented formulas (Bonk et al., 2016; Santos et al., 2019; De Abreu Santos et al., 2020). The benefits become increasingly evident as the numbers of markers and individuals increase. However, it is important that the new representation will later be useful in the derivation of our similarity measure. From the above definition, it is also immediately clear that a swap of haplotypes results in $-\boldsymbol{m}_i$ (i.e., the same vector with opposite signs). The resulting MSV, in any case, remains unaffected regardless of which haplotype is chosen as the first one.

Multiple traits and the aggregate genotype

Breeders rarely evaluate and select parents based on a single trait but instead make decisions based on multiple traits—therefore, they are interested not only in the MSV for each trait but also their covariance. The MSV $\boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_i$ for a single trait of parent i extends to the vector $\boldsymbol{g}_i$ of multiple traits,

$$var\left(\boldsymbol{g}_i\right) = \boldsymbol{M}_i \boldsymbol{R} \boldsymbol{M}_i', \tag{5}$$

where each row of matrix $\boldsymbol{M}_i$ is a vector $\boldsymbol{m}_{it}'$ of additive marker effects for each trait t of the parent i. The diagonal elements of $\boldsymbol{M}_i \boldsymbol{R} \boldsymbol{M}_i'$ are the MSVs of all traits, whereas the off-diagonal elements are the Mendelian sampling covariances (MSCs) between traits (Bonk et al., 2016). Finally, by assuming a known vector of index weight $\boldsymbol{a}$ for all traits, the MSV for the aggregate genotype becomes

$$var\left(\boldsymbol{a}'\boldsymbol{g}_i\right) = \boldsymbol{a}' \cdot (\boldsymbol{M}_i \boldsymbol{R} \boldsymbol{M}_i') \cdot \boldsymbol{a}\,. \tag{6}$$

Similarities between Gametes from Two Parents

Conditional covariance between matching gametes

To derive our similarity measure, we consider two parents with a single chromosome and all the gametes they produce. First, we choose an arbitrary order of haplotypes from both parents and assign a segregation pattern to each possible gamete. Table 1 shows a demonstration using three biallelic markers and eight distinct segregation patterns. For example, pattern $A-B-A$ implies that alleles from the first parental haplotype (A) are transmitted at the first and third loci while an allele from the second parental haplotype (B) is transmitted at the second locus. Recombination rates between adjacent markers (assumed to be $\theta_{1,2} = .1$ and $\theta_{2,3} = .2$) give the probabilities of gametes with a particular segregation pattern. For each of the first parent's gametes, there is a matching gamete from the second parent with an identical segregation pattern and probability of occurrence. Additive values of matching gametes are, however, different and depend on which alleles (1 or 2) are present on the first (A) and second (B) haplotype of each parent.

In our example, the first parent has the ordered parental haplotype $1-1-1/2-2-2$, while the second parent has the ordered parental haplotype $1-2-1/2-1-2$. Consequently, the additive genetic value of the double recombinant $A-B-A$ segregation pattern is either 0.41 or 0.61, depending on whether it descends from the first or second parent (Table 1). Finally, we use the probability distribution of segregation patterns to compute the MSV for each parent and covariance between the additive genetic values of the matching gametes from both parents.

The variance of the genetic values g_1 from the gametes produced by the first parent in the example is calculated as follows:

$$E\left(g_1^2\right) = \frac{1}{2}\sum_s p_s g_1^s = \frac{1}{2}\cdot 0.9\cdot 0.8\cdot 0.61^2 + \ldots + \frac{1}{2}\cdot 0.9\cdot 0.8\cdot (-0.61)^2 = 0.3461$$

$$E\left(g_1\right) = \frac{1}{2}\sum_s p_s g_1^s = \frac{1}{2}\cdot 0.9\cdot 0.8\cdot 0.61 + \ldots + \frac{1}{2}\cdot 0.9\cdot 0.8\cdot \left(-0.61\right) = 0$$

$$var\left(g_1\right) = E\left(g_1^2\right) - \left(E\left(g_1\right)\right)^2 = 0.3461 - 0 = 0.3461\,,$$

where $\frac{1}{2}$ is the probability of transmitting the first allele from either the first or second haplotype, s is the index of the segregation pattern, p_s is the probability of pattern s, and g_1^s is the additive value of the corresponding gamete in the first parent. Following the same procedure, the MSV for the second parent is calculated as $var(g_2) = E(g_2^2) - (E(g_2))^2 = 0.1837$.

Table 1: Segregation patterns (sequence of alleles from the first and second parental haplotypes), probabilities of occurrence, and genetic values of pairs of gametes produced by the first and second parents with matching recombination patterns

Segregation pattern/ Gametes produced	Probability	Genetic value	
		First parent	Second parent
$A-A-A$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}(m_1+m_2+m_3)=0.61$	$\frac{1}{2}(m_1-m_2+m_3)=0.41$
$A-A-B$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}(m_1+m_2-m_3)=0.59$	$\frac{1}{2}(m_1-m_2-m_3)=0.39$
$A-B-A$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}(m_1-m_2+m_3)=0.41$	$\frac{1}{2}(m_1+m_2+m_3)=0.61$
$A-B-B$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}(m_1-m_2-m_3)=0.39$	$\frac{1}{2}(m_1+m_2-m_3)=0.59$
$B-A-A$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}(-m_1+m_2+m_3)=-0.39$	$\frac{1}{2}(-m_1-m_2+m_3)=-0.59$
$B-A-B$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}(-m_1+m_2-m_3)=-0.41$	$\frac{1}{2}(-m_1-m_2-m_3)=-0.61$
$B-B-A$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}(-m_1-m_2+m_3)=-0.59$	$\frac{1}{2}(-m_1+m_2+m_3)=-0.39$
$B-B-B$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}(-m_1-m_2-m_3)=-0.61$	$\frac{1}{2}(-m_1+m_2-m_3)=-0.41$

A represents the alleles on the first haplotype, and B represents alleles on the second haplotype. The first parent has ordered haplotypes of $1-1-1/2-2-2$, while the second parent has ordered haplotypes of $1-2-1/2-1-2$. Recombination rates between adjacent markers are assumed to be $\theta_{1,2}=.1$ and $\theta_{2,3}=.2$, and vector of population additive marker effects is assumed to be $\boldsymbol{m}'=[1 \quad .2 \quad .02]$. Pattern $A-A-B$, for example, has alleles $1-1-2$ in the first parent and $1-2-2$ in the second parent. Note that genetic value represents the value transmitted to the offspring.

Depending on the chosen order of haplotypes, the covariance between the genetic values of the gametes produced by both parents is equal to the covariance between the additive values of matching gametes sharing identical segregation patterns and probabilities of occurrence:

$$E(g_1, g_2) = \frac{1}{2}\sum_s p_s g_1^s g_2^s = \frac{1}{2}\cdot 0.9\cdot 0.8\cdot 0.61\cdot 0.41 + \ldots + \frac{1}{2}\cdot 0.9\cdot 0.8\cdot(-0.61)\cdot(-0.41) = 0.2449$$

$$ccov(g_1, g_2) = E(g_1, g_2) - E(g_1)E(g_2) = 0.2449\,.$$

Enumerating all segregation patterns and probabilities of occurrence becomes difficult with hundreds to thousands of markers. However, using the previously mentioned equivalent matrix expressions, it becomes easy to compute the variances even when dealing with tens of thousands of markers (Bonk et al., 2016). Similarly, there is an equivalent matrix formula for the conditional covariance between the additive values of matching gametes. Relying on parent-specific vectors $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$, we define as the bilinear form

$$ccov(g_i, g_j) = \boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_j\,. \qquad (7)$$

In our example (Table 1), the first haplotypes of both parents give rise to parent-specific vectors of marker effects $\boldsymbol{m}_1' = [1 \quad .2 \quad .02]$ and $\boldsymbol{m}_2' = [1 \quad -.2 \quad .02]$ (see Additional File 1 (A.1-A.3) for numerical examples). Looking at the matrix expression, it becomes apparent that if both parents have markers with identical additive effects linked in the same manner on their first and second haplotypes, the MSVs in both families and the conditional covariance will be the same size. If turned into a correlation, the resulting value of one would then indicate the perfect equality of the ordered parental haplotypes and, hence, the equal distributions of their gametes' additive values. We emphasize that expectations are based on the univariate conditional distribution of segregation patterns rather than the bivariate joint distribution of gametes from two parents. The latter would result in a zero covariance because of the independence of the Mendelian sampling processes in different individuals.

The sign of the conditional covariance changes if the haplotype order for one parent is swapped (either $\boldsymbol{m}_i$ is altered to $-\boldsymbol{m}_i$ or $\boldsymbol{m}_j$ is altered to $-\boldsymbol{m}_j$). The average conditional covariance over all four possible haplotype orders for a single chromosome is zero. The conversion of this conditional covariance into a unique similarity measure valid for any possible order of parental haplotypes is shown below.

Similarities are independent of the order of haplotypes

In deriving the conditional covariance between parents i and j, the chosen order of haplotypes in the parents is arbitrary. For example, consider another parent with the order of haplotypes swapped to that of the second parent (i.e., $2-1-2/1-2-1$). By swapping the haplotypes, the parent-specific marker effects vector becomes $-\boldsymbol{m}_2 = \begin{bmatrix} -1 & 0.2 & -0.02 \end{bmatrix}$, and so the parent-specific marker effects vector for the second parent with opposite signs. The conditional covariance between the first parent and this parent, then, is $\boldsymbol{m}_1' \boldsymbol{R} \left(-\boldsymbol{m}_2 \right) = -0.2449$, which is the negative of the conditional covariance of both candidates. From this example, it follows that the respective correlation between any two individuals with an identical genotype, but with the order of haplotypes swapped, is $\boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_j \,/\, \sqrt{\boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_i \cdot \boldsymbol{m}_j' \boldsymbol{R} \boldsymbol{m}_j} = -1$, as $\boldsymbol{m}_i = -\boldsymbol{m}_j$ in this case. Therefore, to make the similarity between parents i and j independent of the chosen order of haplotypes, it should be calculated for a single chromosome as follows:

$$s_{i,j} = \left| \boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_j \right| . \tag{8}$$

For a genome with several chromosomes, the sum of absolute contributions from all chromosomes is calculated as

$$s_{i,j} = \sum_c \left| \left(\boldsymbol{m}_i^c \right)' \boldsymbol{R}^c \boldsymbol{m}_j^c \right| , \tag{9}$$

where $\boldsymbol{R}^c$ is the diagonal block of $\boldsymbol{R}$ pertaining to chromosome c, and $\boldsymbol{m}_i^c$ ($\boldsymbol{m}_j^c$) represents the parent-specific additive marker effects vector for parent i (j) on the same chromosome.

An individual's similarity to itself (i.e., $i = j$) equals that individual's MSV, which is always positive and independent of haplotype order. Therefore, it is possible to assemble a trait-specific similarity matrix $\boldsymbol{S}$ with MSVs for each parent on the diagonal and pairwise similarities $s_{i,j}$ as off-diagonal elements for any set of individuals.

Multiple traits and the aggregate genotype

When considering multiple traits, there is also a similarity between potential parents i and j with respect to the aggregate genotype. Each element s_{ij} of the respective similarity matrix is given by the following equation:

$$s_{ij} = \sum_c \left| \boldsymbol{a}' \cdot \left(\boldsymbol{M}_{i,c} \boldsymbol{R}_c \left(\boldsymbol{M}_{j,c} \right)' \right) \cdot \boldsymbol{a} \right|.$$

Mendelian Sampling Variance of Zygotes

An equivalent expression for MSV

We propose computing the MSV of zygotes produced by parents i and j in the same way as described in Eq. (2):

$$var\left(g_{ij}\right) = \boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_i + \boldsymbol{m}_j' \boldsymbol{R} \boldsymbol{m}_j, \tag{10}$$

where, again, $\boldsymbol{R}$ is the population covariance matrix reflecting the expected within-family linkage disequilibrium of markers, and $\boldsymbol{m}_i$ ($\boldsymbol{m}_j$) represents the parent-specific additive marker effects vector for parent i (j).

The extension to additive values for multiple traits (in vector $\boldsymbol{g}_{ij}$) is presented by the Mendelian covariance matrix,

$$var\left(\boldsymbol{g}_{ij}\right) = \boldsymbol{M}_i \boldsymbol{R} \boldsymbol{M}_i' + \boldsymbol{M}_j \boldsymbol{R} \boldsymbol{M}_j', \tag{11}$$

where each row $\boldsymbol{m}_{it}$ and $\boldsymbol{m}_{jt}$ of $\boldsymbol{M}_i$ and $\boldsymbol{M}_j$, respectively, are vectors of the marker effects for trait t of parents i and j. Again, the result has zygotic MSVs for all traits on the diagonal and MSCs between traits as off-diagonals.

Similarities between Zygotes of Two Pairs of Parents

Distributions of segregation patterns

The makeup of haplotypes that zygotes inherit from male and female parents can also be characterized by the recombination patterns of parental haplotypes. As above, A and B represent alleles from the first and second haplotypes of a parent, respectively. Then, a zygote with the segregation pattern

$A-B/B-B$ combines a recombinant haplotype from the first parent and a non-recombinant one from the second. Two pairs of parents can produce two populations of zygotes which match into pairs by their recombination pattern, depending on the haplotype orders chosen for the parents. The zygotes of each pair share the same probability of occurrence, but their genetic values may differ depending on their parents' genetic makeup.

From the probability distribution of segregation patterns, we can compute the conditional covariance among the genetic values of zygotes produced by their parents.

Consider two linked biallelic marker loci in the zygotes produced by four parents, i, j, u, and v, with haplotypes $1-1/2-2$, $1-1/2-2$, $1-1/1-2$, and $1-1/2-2$, respectively. Assuming a vector of population additive marker effects $\boldsymbol{m}=\begin{bmatrix}1 & 0.2\end{bmatrix}$ and a recombination rate of $\theta=0.1$ (0.1116 Haldane Morgan), the matching common segregation patterns, probabilities of occurrence, and genetic values of zygotes produced by parent pairs ij and uv are summarized in Table 2.

Table 2: Segregation patterns, probabilities of occurrence, marker haplotypes, and genetic values of zygotes produced by two pairs of parents (ij and uv) with matching recombination patterns

Segregation pattern/ Zygotes produced		Probability	Marker haplotypes of zygotes		Genetic value of zygotes	
			ij	uv	Parents ij	Parents uv
$A-A$	$A-A$	$\frac{1}{4}\cdot(1-\theta)^2$	$1-1/1-1$	$1-1/1-2$	1.2	0.6
	$A-B$	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$1-1/1-2$	$1-1/1-2$	1	0.6
	$B-A$	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$1-1/2-1$	$1-1/2-2$	0.2	–0.4
	$B-B$	$\frac{1}{4}\cdot(1-\theta)^2$	$1-1/2-2$	$1-1/2-2$	0	–0.4
$A-B$	$A-A$	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$1-2/1-1$	$1-2/1-2$	1	0.4
	$A-B$	$\frac{1}{4}\cdot\theta^2$	$1-2/1-2$	$1-2/1-2$	0.8	0.4
	$B-A$	$\frac{1}{4}\cdot\theta^2$	$1-2/2-1$	$1-2/2-2$	0	–0.6
	$B-B$	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$1-2/2-2$	$1-2/2-2$	–0.2	–0.6
$B-A$	$A-A$	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$2-1/1-1$	$1-1/1-2$	0.2	0.6
	$A-B$	$\frac{1}{4}\cdot\theta^2$	$2-1/1-2$	$1-1/1-2$	0	0.6
	$B-A$	$\frac{1}{4}\cdot\theta^2$	$2-1/2-1$	$1-1/2-2$	–0.8	–0.4
	$B-B$	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$2-1/2-2$	$1-1/2-2$	–1	–0.4
$B-B$	$A-A$	$\frac{1}{4}\cdot(1-\theta)^2$	$2-2/1-1$	$1-2/1-2$	0	0.4
	$A-B$	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$2-2/1-2$	$1-2/1-2$	–0.2	0.4
	$B-A$	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$2-2/2-1$	$1-2/2-2$	–1	–0.6
	$B-B$	$\frac{1}{4}\cdot(1-\theta)^2$	$2-2/2-2$	$1-2/2-2$	–1.2	–0.6

A represents the alleles on the first haplotype, and B represents alleles on the second haplotype of a zygote parent. Parents i, j, u, and v have ordered haplotypes $1-1/2-2$, $1-1/2-2$, $1-1/1-2$, and $1-1/2-2$, respectively. A recombination rate of $\theta = .1$ and a vector of population additive marker effects $\boldsymbol{m} = [1 \quad 0.2]$ are assumed. Pattern $A-B/B-B$, for example, has alleles $1-1/2-2$ in parent pair ij and $1-2/2-2$ in parent pair uv. Note that genetic value represents the value transmitted to the offspring.

Mendelian variance of gametic values of zygotes

The variance of the genetic values g_{ij} of the zygotes produced by parent pair ij can be derived from the probability distribution of segregation patterns as follows:

$$var\left(g_{ij}\right) = E\left(g_{ij}^2\right) - E\left(g_{ij}\right)^2$$

$$E\left(g_{ij}^2\right) = \sum_s p_s \left(g_{ij}^s\right)^2 = 0.2025 \cdot 1.2^2 + \ldots + 0.2025 \cdot \left(-1.2\right)^2 = 0.68$$

$$E\left(g_{ij}\right) = \sum_s p_s g_{ij}^s = 0.2025 \cdot 1.2 + \ldots + 0.2025\left(-1.2\right) = 0$$

$$var\left(g_{ij}\right) = E\left(g_{ij}^2\right) - \left(E\left(g_{ij}\right)\right)^2 = 0.68 .$$

Following the same procedure, the variance of the genetic values from zygotes produced by parent pair uv is $var\left(g_{uv}\right) = E\left(g_{uv}^2\right) - \left(E\left(g_{uv}\right)\right)^2 = 0.26 - 0 = 0.26$.

Conditional covariance between Mendelian values of zygotes

Similar to the gametic case, the conditional covariance $ccov(g_{ij}, g_{uv})$ between the zygotes of parent pairs ij and uv can be derived from the probability distribution of joint segregation patterns (probabilities p_s) as follows:

$$ccov\left(g_{ij}, g_{uv}\right) = E\left(g_{ij} g_{uv}\right) - E\left(g_{ij}\right) E\left(g_{uv}\right)$$

$$E\left(g_{ij} g_{uv}\right) = \sum_s p_s g_{ij}^s g_{uv}^s = 0.2025 \cdot 1.2 \cdot 0.6 + \ldots + 0.2025 \cdot \left(-1.2\right) \cdot \left(-0.6\right) = 0.34$$

$$ccov\left(g_{ij}, g_{uv}\right) = 0.34 - 0 = 0.34 .$$

Matrix expressions

Equivalently—and much more conveniently for many markers—this covariance can be expressed in matrix notation. If $\boldsymbol{m}_i$, $\boldsymbol{m}_j$, $\boldsymbol{m}_u$, and $\boldsymbol{m}_v$ are the marker effect vectors of parents i, j, u, and v, then the similarity between the Mendelian sampling values of zygotes produced by parent pairs ij and uv is

$$ccov\left(g_{ij}, g_{uv}\right) = \boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_u + \boldsymbol{m}_j' \boldsymbol{R} \boldsymbol{m}_v . \qquad (12)$$

Numerical examples of the derivation of MSV and conditional covariance between the zygotes of parent pairs ij and uv are provided in Additional File 1 (B).

Similarities between zygotes are independent of the order of haplotypes

When the haplotypes of any parent (i, j, u, or v) are swapped, the corresponding $\boldsymbol{m}$ vector becomes negative (e.g., $\boldsymbol{m}_i'\boldsymbol{R}(-\boldsymbol{m}_u)$). In order to remove the arbitrariness generated by choosing a particular order of haplotypes, we retain the absolute values of both contributions as similarity $s_{ij,uv}$ (between families with indices ij and uv) in the case of a single chromosome:

$$s_{ij,uv} = \left|\boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_u\right| + \left|\boldsymbol{m}_j'\boldsymbol{R}\boldsymbol{m}_v\right|. \tag{13}$$

Eq. (13) operates independently of the order of haplotypes for all four parents. For multiple chromosomes, we maintain this independence by taking the sum of absolute values chromosome by chromosome and then summing them:

$$s_{ij,uv} = \sum_c \left(\left| \left(\boldsymbol{m}_i^c\right)' \boldsymbol{R}^c \boldsymbol{m}_u \right| + \left| \left(\boldsymbol{m}_j^c\right)' \boldsymbol{R}^c \boldsymbol{m}_v \right| \right). \tag{14}$$

Finally, these similarities $s_{ij,uv}$ define the off-diagonal elements of a similarity matrix $\boldsymbol{S}$ between pairs of parents (families) with respect to the makeup of the haplotypes in the zygotes of these families. The MSV of the respective family (which is equal to the similarity of a family with itself) represents the diagonal elements $s_{ij,ij}$. In effect, this similarity measure for zygotes is the sum of the similarities of the sire and dam gametes between the two families. Possible modifications are discussed below.

MSV for multiple traits and the aggregate genotype

For multiple traits, the matrix of MSVs and MSCs $var\left(\boldsymbol{g}_{ij}\right)$ for a parent pair ij (Eq. (11)) for multiple chromosomes can also be expressed by marker effects that are specific for individuals:

$$var\left(\boldsymbol{g}_{ij}\right) = \sum_c \left[\boldsymbol{M}_i^c \boldsymbol{R}^c \left(\boldsymbol{M}_i^c\right)' + \boldsymbol{M}_j^c \boldsymbol{R}^c \left(\boldsymbol{M}_j^c\right)' \right] = \boldsymbol{V}_{ij} \tag{15}$$

Since $\boldsymbol{V}_{ij}$ is independent of the chosen order of haplotypes, the MSV for the aggregate genotype is unequivocally given by the following quadratic form:

$$var\left(\boldsymbol{a}'\boldsymbol{g}_{ij}\right)=\boldsymbol{a}'\boldsymbol{V}_{ij}\boldsymbol{a}\,. \tag{15}$$

Similarities for multiple traits and the aggregate genotype

For two pairs of parents ij and uv, the similarity measure $s_{ij,uv}$ for the aggregate genotype, which does not depend on the haplotype order of any of the four involved parents i, j, u, and v, becomes

$$s_{ij,uv}=\sum_{c}\left[\left|\boldsymbol{a}'\cdot\left(\boldsymbol{M}_i^c\boldsymbol{R}^c\left(\boldsymbol{M}_u^c\right)'\right)\cdot\boldsymbol{a}\right|+\left|\boldsymbol{a}'\cdot\boldsymbol{M}_j^c\boldsymbol{R}^c\left(\boldsymbol{M}_v^c\right)'\cdot\boldsymbol{a}\right|\right].$$

Then, the MSVs of the aggregate genotype define the diagonal elements, and the similarities $s_{ij,uv}$ define the off-diagonal elements of the similarity matrix $\boldsymbol{S}$ between pairs of parents.

Monoecious species and order of parents in dioecious organisms

In monoecious plants, parents i and j may swap their function as male (pollen donor) and female parents when producing offspring. The MSVs of the resulting zygotes, then, remain the same, assuming equal recombination rates in male and female parents. With sex-specific covariance matrices $\boldsymbol{R}_m$ and $\boldsymbol{R}_f$ reflecting maps of males and females, the MSVs for parents ij and ji are

$$var\left(g_{ij}\right)=\boldsymbol{m}_i'\boldsymbol{R}_m\boldsymbol{m}_i+\boldsymbol{m}_j'\boldsymbol{R}_f\boldsymbol{m}_j \text{ and } var\left(g_{ji}\right)=\boldsymbol{m}_i'\boldsymbol{R}_f\boldsymbol{m}_i+\boldsymbol{m}_j'\boldsymbol{R}_m\boldsymbol{m}_j\,.$$

For two pairs of parents ij and ji there are, in principle, four different conditional covariances, depending on which individual acts as pollen donor within each pair:

$$ccov\left(g_{ij},g_{uv}\right)=\boldsymbol{m}_i'\boldsymbol{R}_m\boldsymbol{m}_u+\boldsymbol{m}_j'\boldsymbol{R}_f\boldsymbol{m}_v,\ ccov\left(g_{ji},g_{vu}\right)=\boldsymbol{m}_j'\boldsymbol{R}_m\boldsymbol{m}_v+\boldsymbol{m}_i'\boldsymbol{R}_f\boldsymbol{m}_u,$$

$$ccov\left(g_{ij},g_{vu}\right)=\boldsymbol{m}_i'\boldsymbol{R}_m\boldsymbol{m}_v+\boldsymbol{m}_j'\boldsymbol{R}_f\boldsymbol{m}_u, \text{ and } ccov\left(g_{ji},g_{uv}\right)=\boldsymbol{m}_j'\boldsymbol{R}_m\boldsymbol{m}_u+\boldsymbol{m}_i'\boldsymbol{R}_f\boldsymbol{m}_v\,.$$

If all four kinds of parentage have equal probabilities and $\boldsymbol{R}=\frac{1}{2}\left(\boldsymbol{R}_m+\boldsymbol{R}_f\right)$, then the average of these four conditional covariances can be simplified as

$$\frac{1}{2}\left(\boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_u+\boldsymbol{m}_j'\boldsymbol{R}\boldsymbol{m}_v\right)+\frac{1}{2}\left(\boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_v+\boldsymbol{m}_j'\boldsymbol{R}\boldsymbol{m}_u\right).$$

Then, by taking absolute values in a chromosome-wise manner as described above, this average conditional covariance defines the similarity $\bar{s}_{ij,uv}$ between parents ij and uv as

$$\bar{s}_{ij,uv} = \frac{1}{2}\left(s_{ij,uv} + s_{ij,vu}\right).$$

This could also be used as an alternative similarity measure in mammals and dioecious plants if the similarity is to be independent of the order of parents—or, equivalently, independent of the parental origin of the haplotypes in the offspring.

Standardized Similarity

The similarity matrix $\boldsymbol{S}$ of either gametes or zygotes can be standardized by pre- and post-multiplying with a diagonal matrix $\boldsymbol{D}^{-1}$ with inverse Mendelian standard deviations as diagonal elements:

$$\boldsymbol{K} = \boldsymbol{D}^{-1} \cdot \boldsymbol{S} \cdot \boldsymbol{D}^{-1}, \tag{16}$$

where $\boldsymbol{D} = \sqrt{diag\left(\boldsymbol{S}\right)}$. As in correlation matrices, all diagonal elements of $\boldsymbol{K}$ are all equal (1), whereas all off-diagonal elements are strictly non-negative and lie within the range of $0 \le k_{ij} \le 1$.

Potential Application of the Similarity Matrices in Mate Allocation and Optimum Mating Decisions

A collection of specific parent pairs (or parents) with different weights can be referred to as a mating portfolio. A breeder aims for an expected genetic return in the offspring by allocating various weights to these parent pairs (or parents) so that the portfolio reflects the breeder's purpose and risk tolerance. Weights can be allocated by producing more offspring from a parent pair (e.g., through multiple ovulation and embryo transfer) or assigning more mates to a preferred parent while other matings are omitted. Here, we understand the main aspect of "risk" as a lack of haplotype diversity among parents and, consequently, among offspring. As introduced by Markowitz (1952), in the field of finance, portfolio optimization involves balancing a portfolio's expected returns and risks via the diversifying effect of factoring in covariances between asset prices. Analogously, our similarity measure, which indicates the degree to which parents share heterozygous chromosome segments with large effects, can help to retain as much haplotype diversity in a population as possible.

To this end, three quantities are required to optimize a mating portfolio. These quantities are the expected genetic return, MSV, and similarities among candidate parents. We propose estimating the expected genetic return for a potential parent pair ij (or parent i) as follows:

$$p_{ij} = 1 - \phi\left(\frac{t - bv_{ij}}{\sigma_{ij}}\right), \tag{17}$$

where p_{ij} is the probability of breeding top-ranking individuals, t is the threshold, bv_{ij} is the GEBV, σ_{ij} is the Mendelian standard deviation (square root of MSV) of the estimated additive genetic values, and ϕ is the cumulative distribution function of standard normal distribution. The GEBV for parent i can be obtained as $bv_i = \boldsymbol{c}_i'\boldsymbol{m}$, and GEBV for parent pair ij can be obtained as the average of the parental GEBVs $bv_{ij} = \frac{1}{2}\cdot\boldsymbol{c}_i'\boldsymbol{m} + \frac{1}{2}\cdot\boldsymbol{c}_j'\boldsymbol{m}$.

To optimize portfolios, we propose an approach that minimizes risk (MSV) and similarity among parent pairs (or parents) at a given level of expected returns. We define the expected genetic return of a portfolio as

$$\Psi = \boldsymbol{n}'\boldsymbol{p}, \tag{18}$$

where vector $\boldsymbol{p}$ is the probability that potential parent pairs (or parents) breed top-ranking individuals and $\boldsymbol{n}$ is a vector of the relative weights of elements n_{ij}. The optimization problem can be stated as

$$\begin{aligned} \underset{\boldsymbol{n}}{\text{minimize}} \quad & \boldsymbol{n}'\boldsymbol{Q}\boldsymbol{n} \\ \text{subject to} \quad & \Psi = \boldsymbol{n}'\boldsymbol{p} \geq 0, \\ & \boldsymbol{n}'\boldsymbol{1} = q \\ & n_{ij} \in \mathbb{Z},\ ij = 1,\ldots,q_1 \\ & n_{ij} \in \mathbb{R},\ ij = q_1+1,\ldots,q \end{aligned} \tag{19}$$

where $\boldsymbol{1}$ is a vector of ones, q is the number of expected progeny or mates, $q_1 \in \{0,\ldots,q\}$ is the number of integer variables, and $\boldsymbol{Q}$ is a similarity matrix. Matrix $\boldsymbol{Q}$ can be $\boldsymbol{S}$ or $\boldsymbol{K}$ for zygotes or gametes, depending on the breeder's purpose.

This is a mixed-integer quadratic optimization problem that can be solved using several efficient algorithms to obtain the optimal solution (relative weights $\boldsymbol{n}$) for any given data. The formulations of the optimization problem can be easily extended to include additional constraints, such as the minimum-maximum number of proportions or the maximum level of inbreeding allowed.

The efficient frontier (EF) is derived by solving the optimization problem in Eq. (19) at varying levels of expected genetic returns Ψ (within the range of $\boldsymbol{p}$). The EF curve offers a set of optimal allocations of parent pairs (or parents) for each level of expected returns with the lowest possible risk of losing genetic variability.

Empirical Data

We analyzed an empirical dataset of five paternal half-sib families of Holstein-Friesian cows from Hampel et al. (2018) to demonstrate our approach. The dataset contains 265 individuals (32–106 candidates per paternal half-sib family), pedigree information, three phenotypes (milk fat content, protein content, and pH value), genotypic data, and a physical map. We approximated genetic map positions linearly by converting the physical map positions (bp) into Mbp. The genotypic data contains 39,780 markers on the autosomes. The marker genotypes were phased using *hsphase* version 2.0.2 (Ferdosi and Gondro, 2018). The marker effects of this population were estimated in a previous study (Melzer et al., 2013) with a 10-fold cross-validation method. These estimates were averaged and taken as the marker effects (see Additional File 2).

We used all 265 individuals to derive similarity matrices using the gametic approach. To demonstrate their use for visualizing the haplotype diversity of a population, we graphically displayed these matrices using *corrplot* version 0.84 (Wei and Viliam, 2017) and performed family-wise clustering using *pheatmap* version 1.0.12 (Kolde, 2019).

To demonstrate the application of the similarity matrix for allocating mates and the optimization of mating decisions, we estimated the GEBV for all individuals using marker effects for milk pH. We selected five individuals at random and assumed they were males. We selected another 40 individuals at random from the remaining population and assumed they were females. All possible mating combinations among the selected parents amounted to 200 parent pairs. We estimated the expected GEBV for each parent pair and calculated the probability of breeding top-ranking individuals using a threshold ($t = 0.024$) set at the 95% quantile of their expected GEBV. We derived a similarity matrix

for these parent pairs using the zygotic approach and obtained an EF by solving the optimization problem as stated in Eq. (19) using *gurobi* version 9.10 (Gurobi Optimization LLC, 2020).

All statistical analyses were performed using programs written by the authors and packages in R version 4.0.2 (R Foundation for Statistical Computing, 2020).

RESULTSs

We implemented a computationally faster approach for estimating MSV, building on the method described by Bonk et al. (2016). Our approach ($\boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_i$) was 75 times faster than the method of Bonk et al. (2016) when MSVs were computed for 265 parents with 39,780 markers (results not shown). Furthermore, the use of parent-specific $\boldsymbol{m}_i$ vectors was essential for measuring the similarity among the Mendelian sampling values of individuals.

Considering the example data from Hampel et al. (2018), the derived genomic relationship matrix (VanRaden, 2008) using marker genotypes clearly showed the family structure in the population, with the progeny within a family being more similar to each other than to that of other families (Figure 1). However, this observed structure/pattern was not pronounced in the gametic similarity matrices because they mirror the Mendelian segregation patterns of the individuals (Figure 1A: milk fat, protein, and pH) rather than marker genotypes as in the genomic relationship matrix.

The diagonal elements of each matrix were the MSV of each parent, and the off-diagonal elements were the similarities among the Mendelian sampling values of potential parents. Thus, the values of these matrices depended on the linkage, heterozygosity, and genetic architecture of traits. Parents with common heterozygosity of markers with large effects for a trait in coupling appeared to be more similar than those with small effects. The latter appeared to be more similar than those that were homozygous at those loci (see small matrix examples in A.3 Additional File 1). Consequently, individuals with comparable MSV values appeared to be similar regardless of the family they came from.

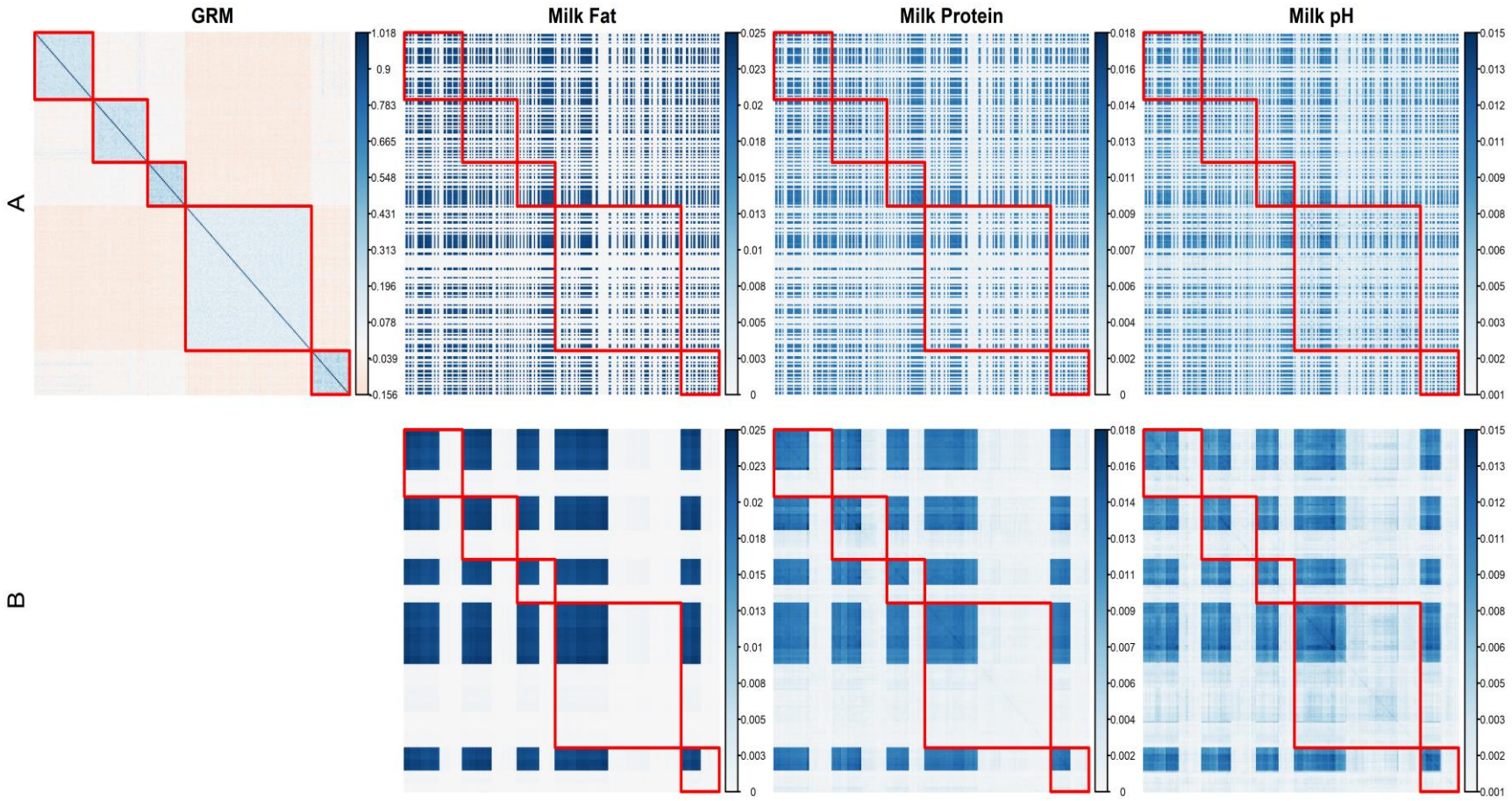


Fig. 1 Genomic relationship matrix (GRM) and similarity matrices for all chromosomes (A) and Euclidean clustering of similarity matrices for all chromosomes (B) for milk fat, protein, and pH.

The red blocks demarcate each paternal half-sib family. Parents are arranged according to their pedigree.

Clustering the individuals within a family revealed pronounced similarity structures across families (Figure 1B). The similarity matrices differed across traits, with pairwise similarities ranging from 0 to 0.025 in milk fat, 0 to 0.018 in milk protein, and 0 to 0.015 in milk pH. Patterns of similarity that encompassed all families were, however, consistently observed for all traits. Standardized similarity matrices (Figure 2) revealed a similar trend to that depicted in Figure 1, with pairwise similarities ranging from 0.037 to 0.995 in milk fat, 0.067 to 0.971 in protein, and 0.100 to 0.959 in pH.

The effects of heterozygosity and trait genetic architecture were observed in chromosome 4, where parents in family 4 appeared to be more variable than those in family 3 regarding milk fat, whereas the opposite was observed for milk protein (Figure 3). Besides the apparent effect of various traits on the structure of the matrices, comparing the matrices from different chromosomes further showed the effect of genetic architecture (Figures 3 and 4). In milk fat, for example, the pairwise similarities ranged from 0 to 0.025 on chromosome 14, whereas the values on chromosome 4 were approximately zero (Figure 3). Ignoring the effect of heterozygosity, such a result is expected if a chromosome (here: 14) has either many markers with small effects or few markers with more substantial effects on milk fat than the other chromosome (here: 4).

Figure 5 shows the EF for matings among parent pairs. The gray circles depict what the breeder should expect if all progenies were produced from a particular parent pair. In other words, the gray circles show the expected return and MSV of each parent pair. The expected genetic return of parent pairs ranged from 0 to 0.93, and MSVs ranged from 0.0045 to 0.027. Hence, a breeder must create a portfolio by allocating weights to these parent pairs such that the average return and variance of the portfolio reflect the breeder's risk tolerance and purpose.

Allocating equal weights to all parent pairs yielded an expected return of 0.14 and a variance of 0.0092 (black triangle). The red dots indicate inefficient portfolios because more could be gained at those levels of risk. As shown in Figure 5, the EF provided a set of optimal allocations of parent pairs for each level of expected return with the lowest possible risk of losing haplotype diversity (blue). The expected returns of the EF ranged from 0.095 to 0.93, and MSVs ranged from 0.0038 to 0.009. The portfolio on the EF curve with the least variance is called the minimum variance portfolio (blue square), and that with the highest expected return per unit risk is called the tangency portfolio (blue diamond).

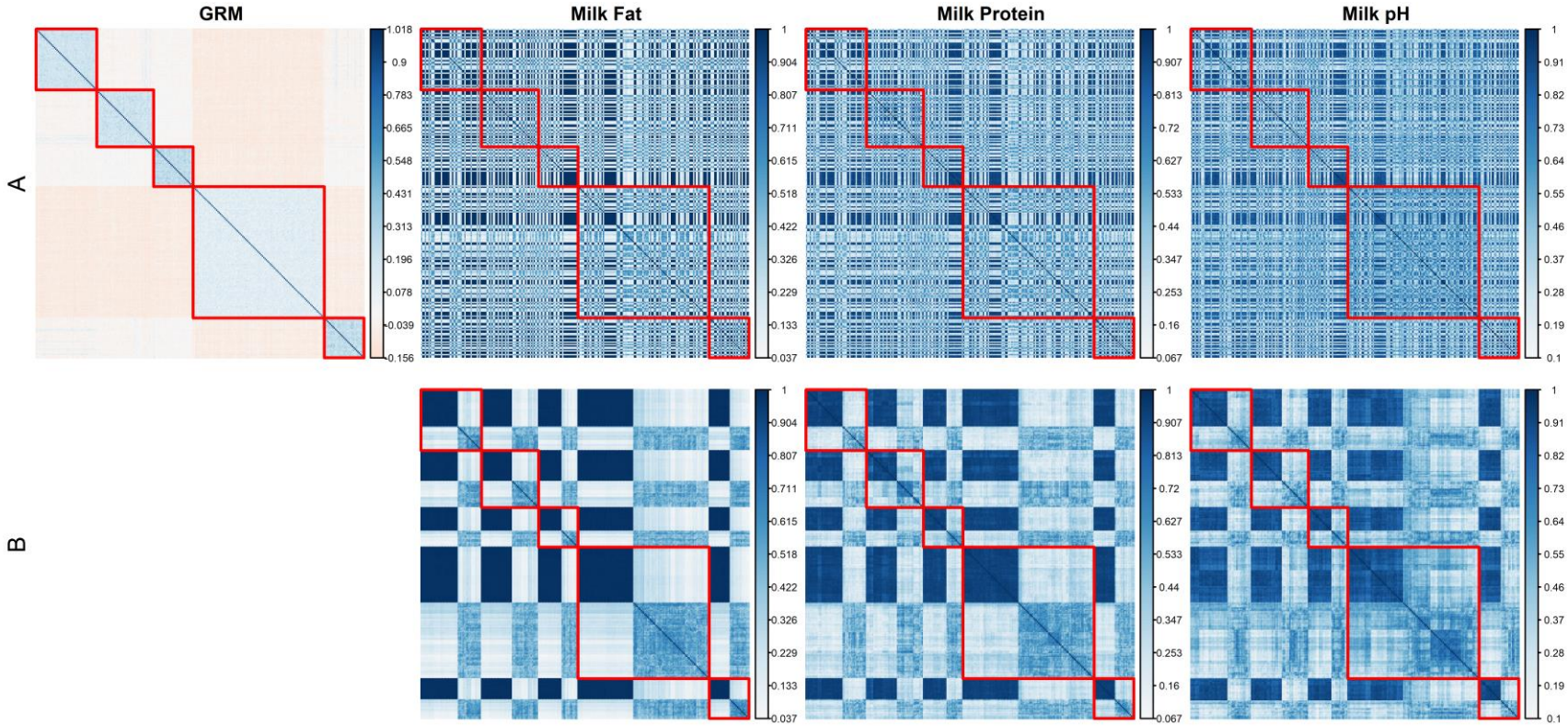


Fig. 2 Genomic relationship matrix (GRM) and standardized similarity matrices (A) and Euclidean clustering of standardized similarity matrices (B) for milk fat, protein, and pH.

The red blocks demarcate each paternal half-sib family. Parents are arranged according to their pedigree.

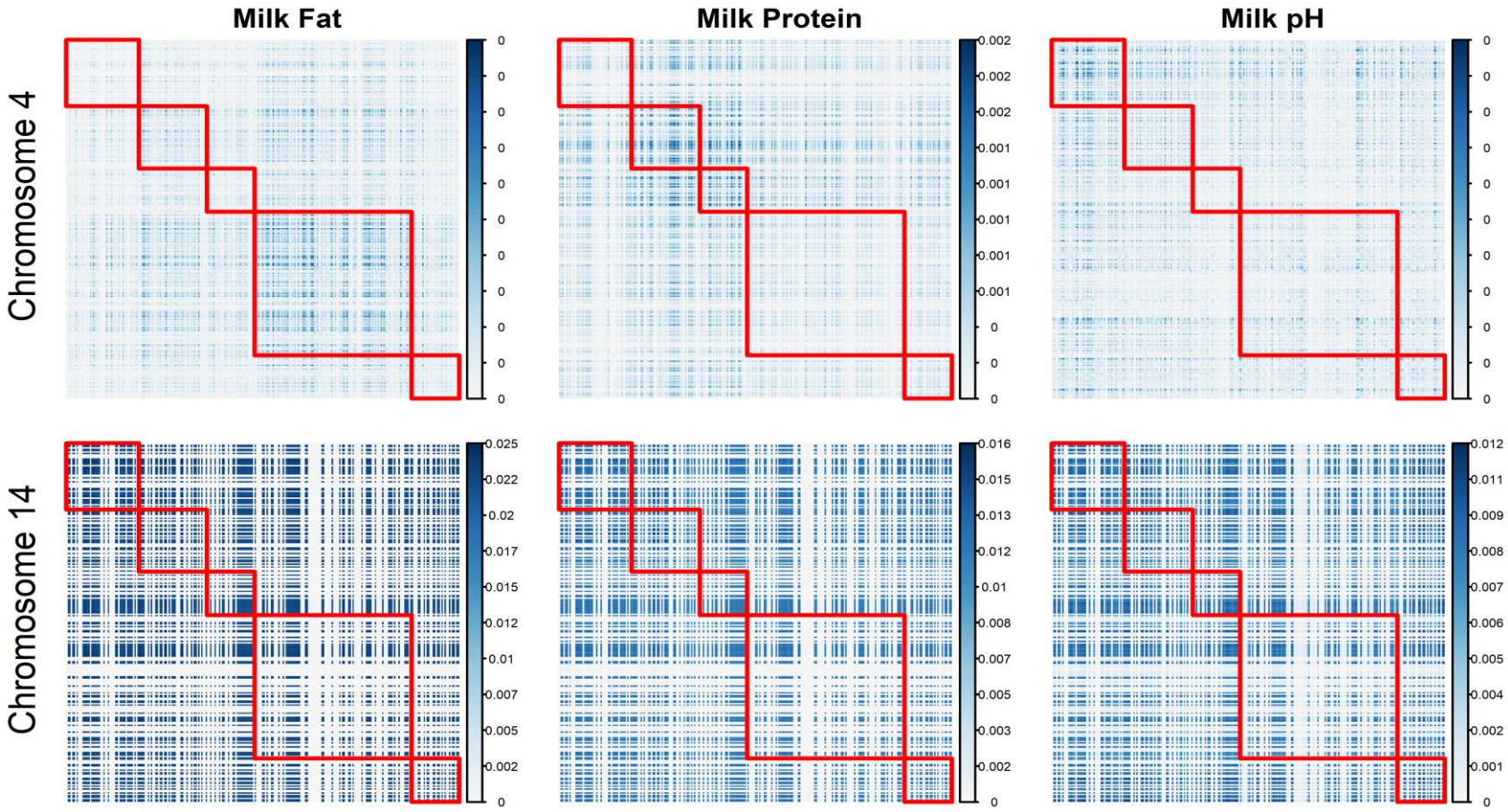


Fig. 3 Similarity matrices showing chromosomes 2 and 14 for three milk traits.

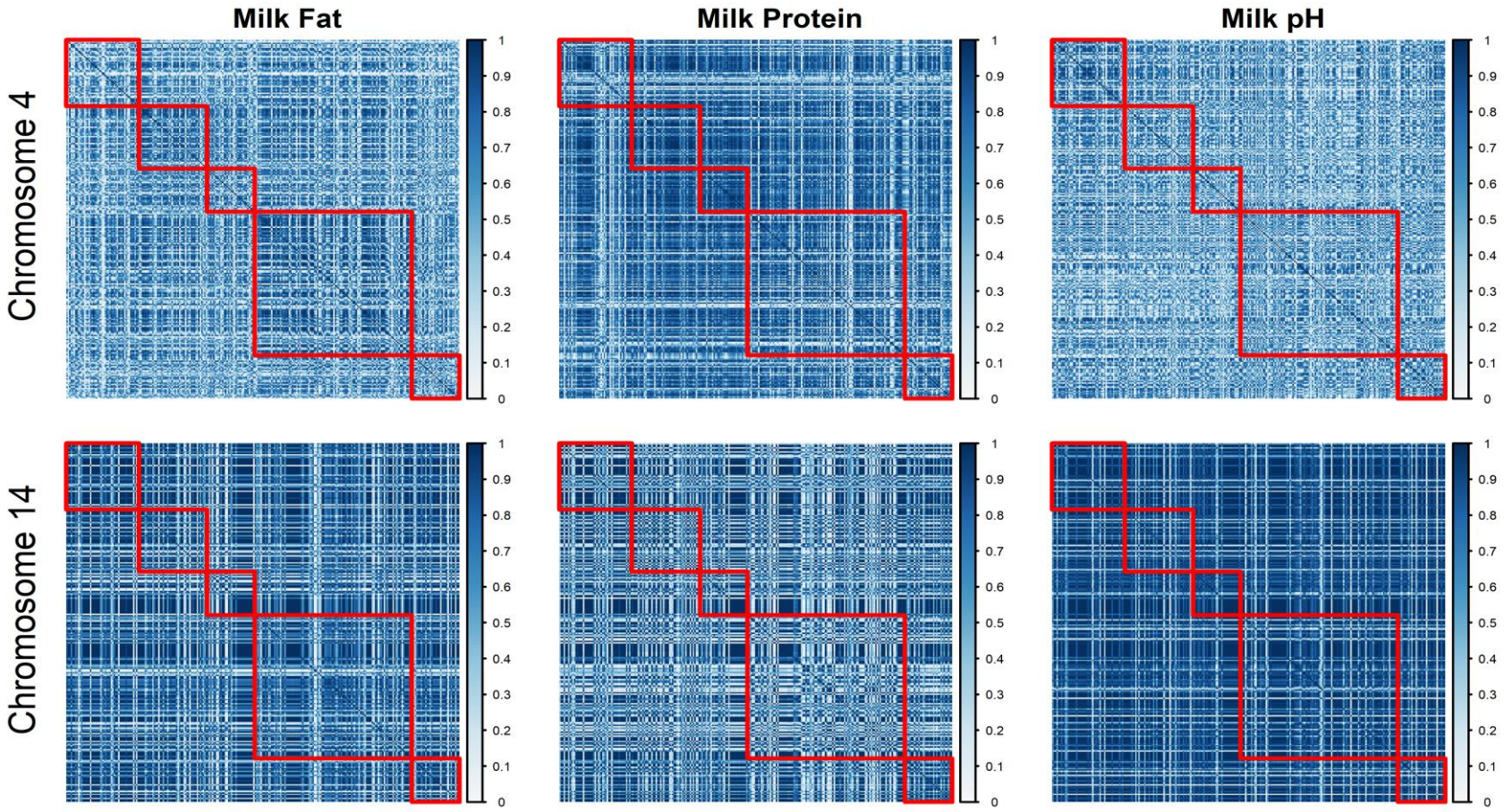


Fig. 4 Standardized similarity matrices showing chromosomes 2 and 14 for three milk traits.

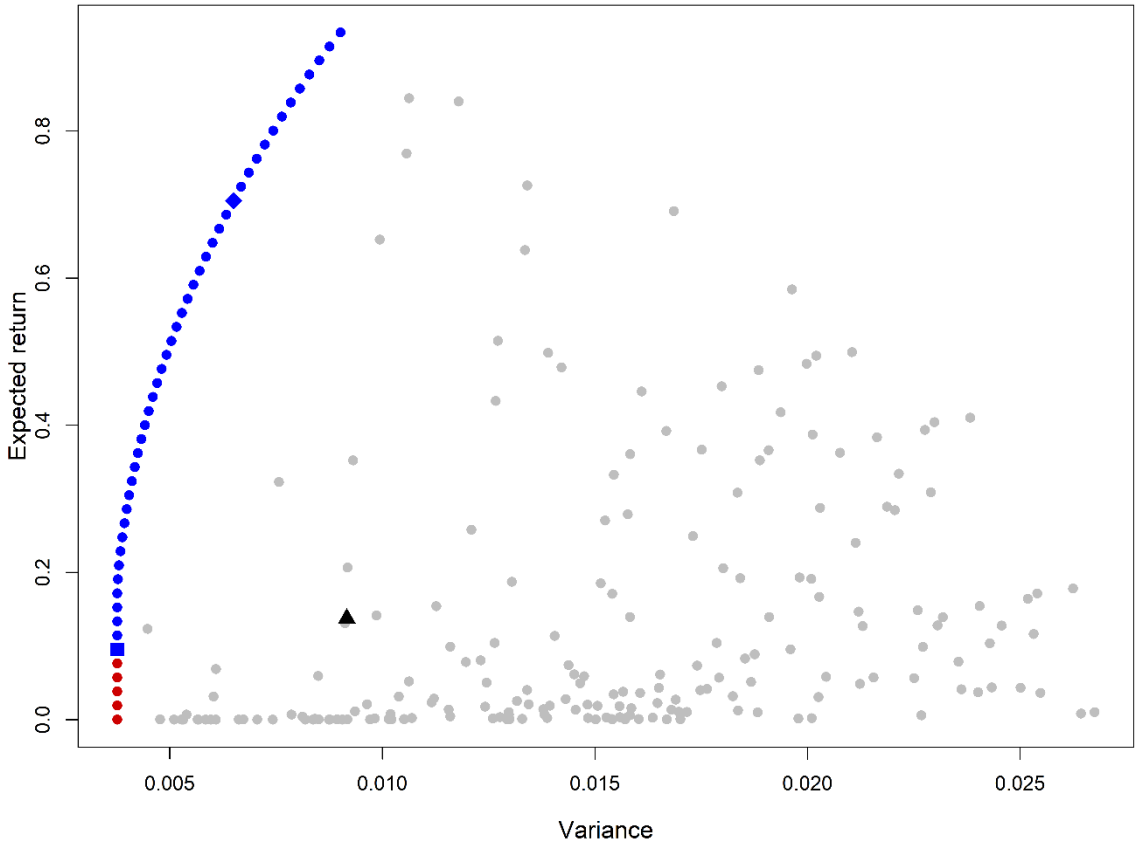


Fig. 5 Optimization of matings (portfolios) based on the efficient frontier.

The gray circles represent 100% investment in each mating. The equal distribution portfolio (black triangle) depicts an equal investment in all matings. The red circles indicate inefficient portfolios. The efficient frontier is depicted in blue, with the square and diamond shapes representing the minimum variance and tangency portfolios, respectively.

DISCUSSION

The novel similarity matrix described in this paper fills a crucial methodological gap by providing a diversifying effect on haplotypes in a group of selected parents in cases where the selection criterion is an index that combines expected GEBVs of offspring with their MSVs. The probability of breeding top-ranking individuals is just one (non-linear) example of such an index (Bijma et al., 2020). Moreover, standardized similarities over the entire range of possible values appeared for several traits in real data, further underpinning the practical relevance of the approach. Notably, the derived

similarity measure has a straightforward genetic interpretation—high similarity between parents arises from many shared chromosome segments with markers of large additive effects in the same linkage phase. When parents have identical haplotypes, the result would be a similarity that translates into a correlation of one. Possible alternatives for comparing Mendelian sampling values could be the Kullback–Leibler (Kullback and Leibler, 1951) and Jensen–Shannon (Nielsen, 2019) divergence methods, as known from information theory. However, their applicability to breeding was not investigated in this study.

Faster MSVs prediction is one of the perks of our approach. Unlike the approach of Bonk et al. (2016) and others (Santos et al., 2019; De Abreu Santos et al., 2020), we used marker effects with parent-specific signs and a population covariance matrix. This required much less time to compute MSV, and we expect the difference in computing time would be more significant when more parents and markers are involved.

The similarity matrices' derivations are based on the availability of phased genotypes, a genetic map, and estimated marker effects. The similarity matrices are trait-specific and require no weighting in a selection involving only a single trait. However, when the selection is made for multiple traits, index weights allow all trait-specific similarity matrices to be summed.

Visualization is one of the many applications of similarity matrices among parents. Using the Holstein-Friesian dataset, we showed how these similarity matrices provide a quick overview of the similarity of Mendelian sampling values within and across families. The size of these similarities depends on the linkage and heterozygosity in parents and the trait genetic architecture (Figures 1–4). Standardized similarities also have an interpretation as the proportion of shared MSV. In the present study, the observed standardized pairwise similarities for all traits ranged almost across the entire possible parameter space (i.e., from 0.037 to 0.995 for fat, 0.067 to 0.971 for protein, and 0.100 to 0.959 for pH) (Figure 2). However, we expect the species and populations of plants and animals to affect these values. Their influence on the similarities of Mendelian sampling values should be investigated in future studies.

Portfolio optimization considers how a particular mating affects the expected return and overall haplotype similarity of parents. Regarding genetic diversity, it seems preferable to aim for a certain expected return by maintaining more diverse haplotypes for generating the next generation—or, equivalently, by mating parents that are less similar to each other. Such efficient portfolios are placed

on the EF of the risk-expected return space. The EF is an established method for minimizing risk at any given level of returns (Markowitz, 1952). It provides a set of optimum mate allocations (portfolios) to yield the optimum tradeoff of genetic return against the risk (Figure 5) of low genetic haplotype variability. This requires knowledge of the expected genetic return together with appropriate similarities between parents or matings. The latter two quantities can be obtained from our similarity matrices.

Our approach regarding mate allocation and optimum mate decision-making enables visualization of the implications of each decision and provides the opportunity to make the best-educated judgments. For instance, a breeder who is more interested in producing progenies with phenotypic uniformity than increasing genetic returns can select the minimum variance portfolio (Figure 5). If the breeder prefers the most risk-efficient portfolio, they could invest in the tangency portfolio. The tangency portfolio has the maximum *Sharpe ratio*, defined as the expected return per unit risk. If a breeder seeks progeny with exceptional breeding values, several opportunities are available to select a portfolio from the EF. In Figure 5, the efficient portfolio with the highest expected return corresponded to the expected return and variance of a particular parent pair. However, even in a species with high reproductive capacity, it is not advisable to invest in this portfolio or a single mating. Instead, there is a need for diversification.

Diversified portfolios contain several parent pairs with high expected genetic returns but low similarity values. The greater the number of parent pairs with different linkage phases of markers with large effects for a trait in coupling, the smaller the similarity values and, consequently, the risk of the portfolio. Thus, a diversified portfolio on the EF provides a set of parent pairs at different weights that offer a minimum risk at a given expected genetic return while maintaining haplotype diversity by keeping different heterozygous chromosome regions in play.

The present study focused on mate allocation and optimal mating decisions regarding the number of progenies produced from each parent pair when genotypic data are present from both male and female parents. However, our approach can also be used in scenarios where genotypic data is available only for a single sex to optimize mating decisions following the EF. In such a scenario, the similarity matrix can be adapted using the gametic approach. Another application of gametic similarities is in plant breeding, where gametes from parents are converted into doubled haploid lines, constituting the next

breeding generation. The long-term consequences of repeated application on genetic gain and genetic variability in breeding programs with recurrent selection remain to be investigated.

CONCLUSIONS

We derived trait-specific similarities between parents that reflect the shared heterozygosity of their markers with large effects and similar linkage phases. The similarity of a parent with itself equals its MSV. If these similarities are combined into a matrix, they provide the opportunity to balance haplotype diversity and genetic gain in selection decisions. However, the effects of recurrent application on genetic gain and variance in breeding programs remain to be investigated.

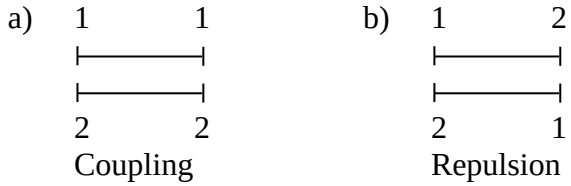
ACKNOWLEDGEMENT

Financial support for AAM from Bundesanstalt für Landwirtschaft und Ernährung (BLE) under Grant 281B101516 is gratefully acknowledged. The publication of this article was funded by the Open Access Fund of the Leibniz Association.

APPENDICES

Appendix 1: Illustration of equivalence

Eqs. (1) and (3) are equivalent. To demonstrate their equivalence, consider the following case of two biallelic marker loci on a single chromosome with haplotypes in coupling and repulsion phases.



According to Bonk et al. (Bonk et al., 2016), for any two-marker interval, we have the following covariance matrices:

$$\boldsymbol{R}_a = \begin{bmatrix} 0.25 & \rho \\ \rho & 0.25 \end{bmatrix} \text{ and } \boldsymbol{R}_b = \begin{bmatrix} 0.25 & -\rho \\ -\rho & 0.25 \end{bmatrix}.$$

Meanwhile, the MSVs are as follows:

a) $$\boldsymbol{m}'\boldsymbol{R}_a\boldsymbol{m} = \begin{bmatrix} m_1 & m_2 \end{bmatrix} \cdot \begin{bmatrix} 0.25 & \rho \\ \rho & 0.25 \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = 0.25 \cdot m_1^2 + 0.25 \cdot m_2^2 + 2\rho m_1 m_2$$

b) $$\boldsymbol{m}'\boldsymbol{R}_b\boldsymbol{m} = \begin{bmatrix} m_1 & m_2 \end{bmatrix} \cdot \begin{bmatrix} 0.25 & -\rho \\ -\rho & 0.25 \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = 0.25 \cdot m_1^2 + 0.25 \cdot m_2^2 - 2\rho m_1 m_2 .$$

After setting up $\boldsymbol{m}_a$ and $\boldsymbol{m}_b$, the same results can be obtained by setting up a population covariance matrix

$$\boldsymbol{R} = \begin{bmatrix} 0.25 & \rho \\ \rho & 0.25 \end{bmatrix},$$

and applying Eq. (3):

a) $$\boldsymbol{m}_a'\boldsymbol{R}\boldsymbol{m}_a = \begin{bmatrix} m_1 & m_2 \end{bmatrix} \cdot \begin{bmatrix} 0.25 & \rho \\ \rho & 0.25 \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = 0.25 \cdot m_1^2 + 0.25 \cdot m_2^2 + 2\rho m_1 m_2$$

b) $$\boldsymbol{m}_b'\boldsymbol{R}\boldsymbol{m}_b = \begin{bmatrix} m_1 & -m_2 \end{bmatrix} \cdot \begin{bmatrix} 0.25 & \rho \\ \rho & 0.25 \end{bmatrix} \cdot \begin{bmatrix} m_1 \\ -m_2 \end{bmatrix} = 0.25 \cdot m_1^2 + 0.25 \cdot m_2^2 - 2\rho m_1 m_2 .$$

The equivalence of the two representations of MSV generalizes to any number of markers (see Appendix 2 for proof of equivalence).

Appendix 2: Proof of equivalence

In Eq. (1) $\boldsymbol{m}'\boldsymbol{R}_i\boldsymbol{m} = \sum_k m_k^2 + \sum_k \sum_l m_k m_l \rho_{kl}$ each ρ_{kl} is positive if the marker alleles to which the effects m_k and m_l have been assigned are on the same chromosome (in coupling). Otherwise, ρ_{kl} is negative. As the marker effects do not depend on the linkage phase, each product $m_k m_l \rho_{kl}$ changes its sign according to the linkage phase of the respective markers.

In Eq. (3) $\boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_i = \sum_k \left(m_k^i\right)^2 + \sum_k \sum_l m_k^i m_l^i \rho_{kl}$ the covariance ρ_{kl} is always positive, but the signs of the marker effects depend on the linkage phase. In coupling, they are as displayed in Eq. (1), yet in the linkage phase, they are either $\left(m_k, -m_l\right)$ or $\left(-m_k, m_l\right)$ depending on the alleles on the first haplotype. In effect, this causes the sign of each product $m_k^i m_l^i \rho_{kl}$ to change according to the linkage

phase as before and $\boldsymbol{m}'\boldsymbol{R}_i\boldsymbol{m} = \boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_i$. Thus, it is quicker to set up $\boldsymbol{m}_i$ with M elements on a chromosome than $\frac{1}{2}M\left(M\text{-}1\right)$ off-diagonal elements of a half-stored $\boldsymbol{R}$-matrix.

SUPPLEMENTARY MATERIALS

Additional file 1: Derivation of Mendelian Sampling Variance, Covariance, and Conditional Covariance from a Probability Distribution

Mendelian sampling is the independent segregation of alleles of various genes into gametes, provided that the genes belong to different linkage groups. Genes on the same chromosome belong to the same linkage group. The tendency of the alleles of genes on the same chromosome to recombine into gametes can be measured from a mapping function (e.g., Haldane's mapping function (Haldane, 1919)). The segregation patterns of alleles into gametes can be obtained straightforwardly using a probability tree representing a sequence of events.

A. Gametes

To demonstrate, we assume three biallelic markers on a single chromosome and two vectors of population additive marker effects for trait 1 $\left(\boldsymbol{m}_{t1}\right)' = \left[1 \quad 0.2 \quad 0.02\right]$ and trait 2 $\left(\boldsymbol{m}_{t2}\right)' = \left[0.2 \quad -0.8 \quad 0.4\right]$. Five potential parents with phased haplotypes—coded as $1-1-1/2-2-2$ for the first, $1-2-1/2-1-2$ for the second, $2-1-2/1-2-1$ for the third, $1-2-1/1-2-2$ for the fourth, and $1-2-2/1-2-1$ for the fifth parents—are assumed. Finally, we assume a vector of genetic distances (Haldane Morgan) between markers $\boldsymbol{d}' = \left[0 \quad 0.1116 \quad 0.3670\right]$, which translates to recombination rates of $\theta_{1,2} = 0.1$ between marker loci 1 and 2 and $\theta_{2,3} = 0.2$ between loci 2 and 3.

Supplementary Figure 1 represents all segregation patterns of alleles into gametes, probabilities of occurrence, and the genetic values for trait 1 of the first and second parents. From the probability tree, it is clear that gametes from two parents have a similar probability of occurrence, but they usually have different genetic values, depending on the parents' genetic makeup. The genetic values obtained

from different segregation patterns of gametes for trait 1 of the first and second parents ($g_{1_{t1}}$ and $g_{2_{t1}}$) are derived from parent-specific vectors (explained in Eq. (3)), and they represent the value transmitted to the offspring (also see Table 1).

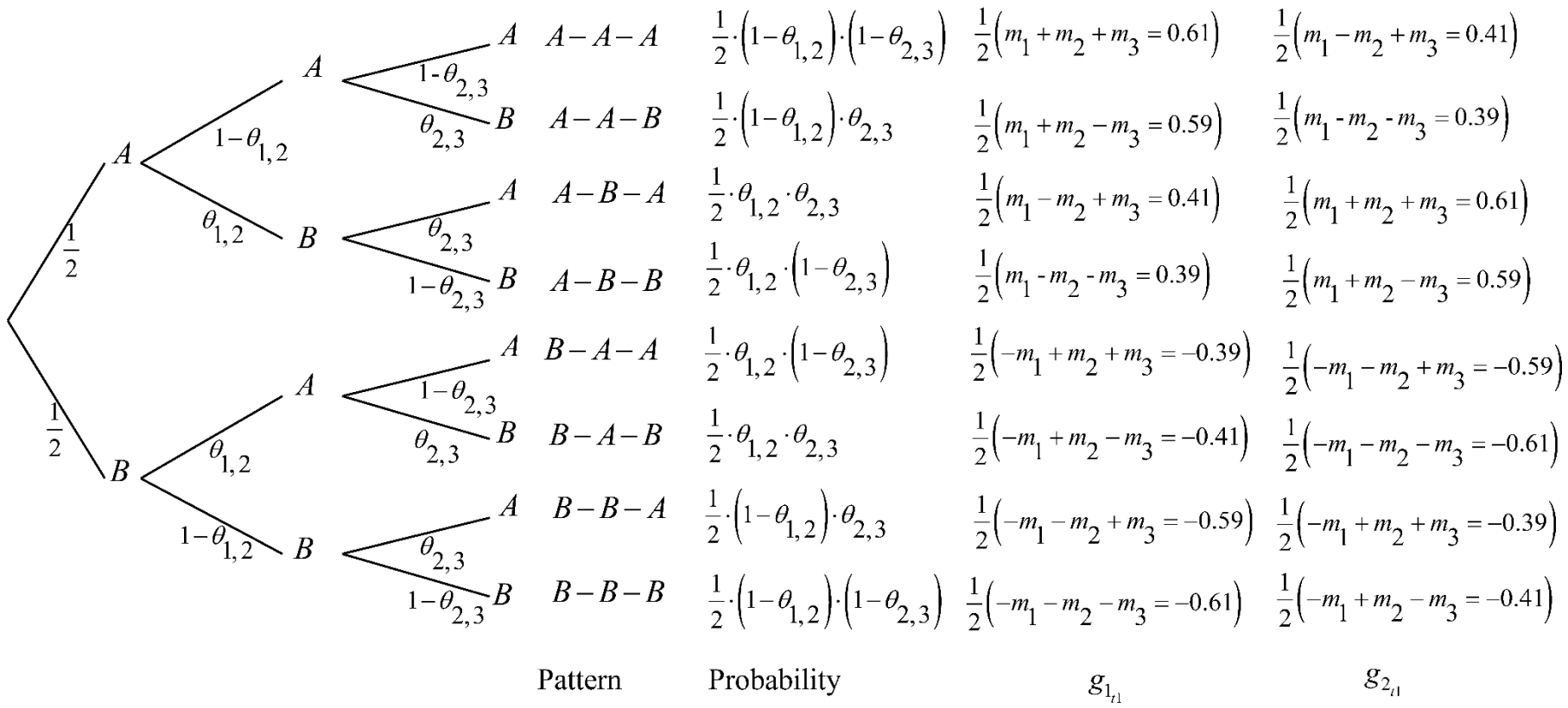
A.1 Mendelian sampling variance of gametes

From the probability distribution of segregation patterns, the Mendelian sampling variances (MSVs) of genetic values for the first ($\sigma^2_{g_{1_{t1}}}$) and second ($\sigma^2_{g_{2_{t1}}}$) parents for trait 1 ($t1$) are 0.3461 and 0.1837 , respectively (see Table 1). Alternatively, and more conveniently, the MSV for parent i can be obtained using the equivalent matrix formula $\boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_i$ (Eq. 3)

$$\sigma^2_{g_{1_{t1}}} = \boldsymbol{m}'_{1_{t1}}\boldsymbol{R}\boldsymbol{m}_{1_{t1}} = \begin{bmatrix}1 & 0.2 & 0.02\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2 & 0.12\\ 0.2 & 0.25 & 0.15\\ 0.12 & 0.15 & 0.25\end{bmatrix}\cdot\begin{bmatrix}1\\ 0.2\\ 0.02\end{bmatrix} = 0.3461$$

$$\sigma^2_{g_{2_{t1}}} = \boldsymbol{m}'_{2_{t1}}\boldsymbol{R}\boldsymbol{m}_{2_{t1}} = \begin{bmatrix}1 & -0.2 & 0.02\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2 & 0.12\\ 0.2 & 0.25 & 0.15\\ 0.12 & 0.15 & 0.25\end{bmatrix}\cdot\begin{bmatrix}1\\ -0.2\\ 0.02\end{bmatrix} = 0.1837\,,$$

where $\boldsymbol{m}'_{1_{t1}}$ and $\boldsymbol{m}'_{2_{t1}}$ are row vectors of parent-specific additive marker effects for trait $t1$ of the first and second parents, respectively. MSV does not depend on the haplotype order chosen when setting up the parent-specific vector $\boldsymbol{m}_i$.



Supplementary Figure 1: A probability tree showing the segregation patterns (sequence of alleles from the first (A) and second (B) parental haplotypes), probabilities of occurrence, and the genetic values for trait $t1$ of the first (g_1) and second (g_2) parents.

A.2 Mendelian sampling covariance for multiple traits of the same parent

Supplementary Figure 2 shows the segregation patterns, probabilities of occurrence, and genetic values of gametes produced by the first parent for multiple traits. The Mendelian sampling covariance (MSC) between gametes for multiple traits $t1$ and $t2$ of the first parent can be derived from the following probability distribution:

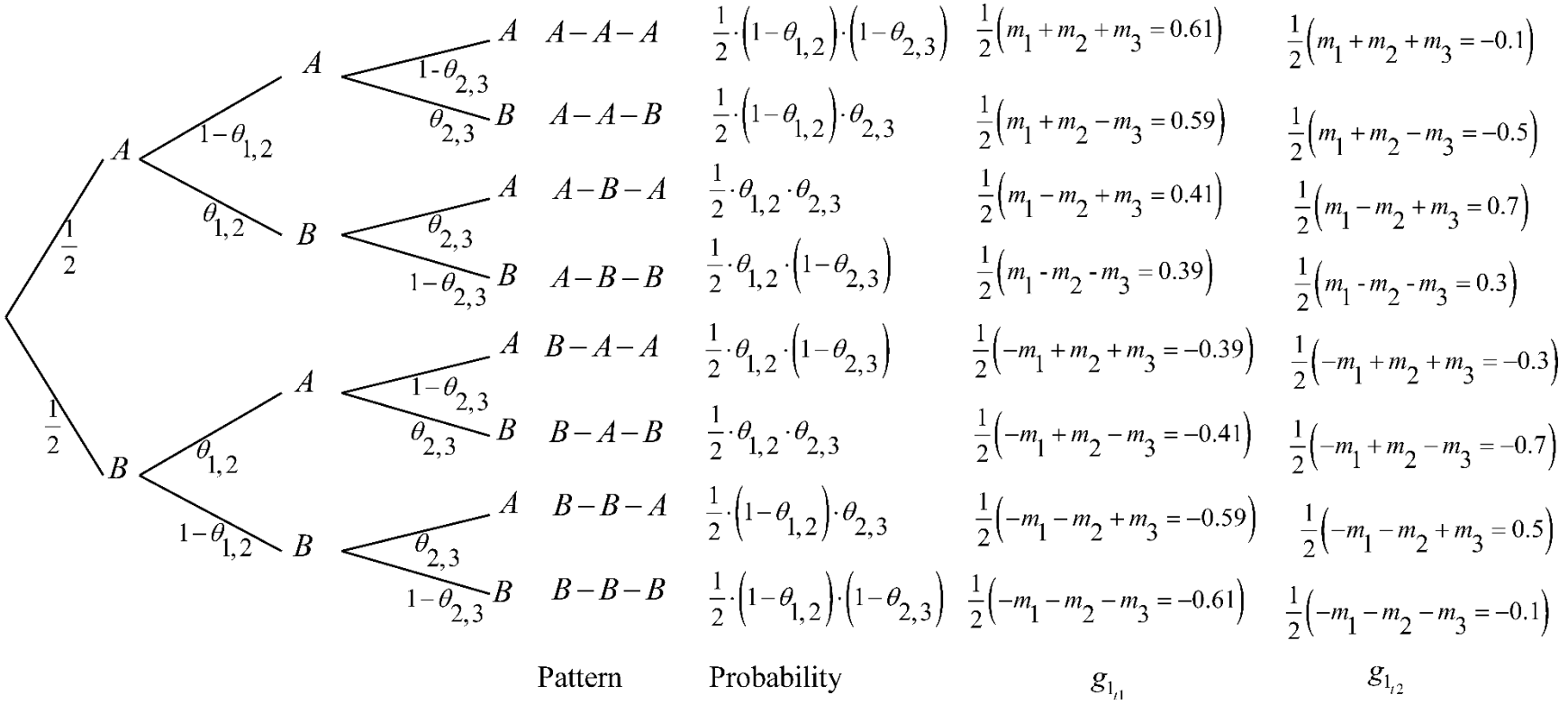
$$E\left(g_{1_{t1}}, g_{1_{t2}}\right) = \frac{1}{2}\sum_{S} p_s g^s_{1_{t1}} g^s_{1_{t2}} = \frac{1}{2}\cdot 0.9\cdot 0.8\cdot 0.61\cdot(-0.1) + \ldots + \frac{1}{2}\cdot 0.9\cdot 0.8\cdot(-0.61)\cdot 0.1 = -0.0819$$

$$cov\left(g_{1_{t1}}, g_{1_{t2}}\right) = E\left(g_{1_{t1}}, g_{1_{t2}}\right) - E\left(g_{1_{t1}}\right)E\left(g_{1_{t2}}\right) = -0.0819\,.$$

Alternatively, the same results can be obtained using $cov\left(g_{1_{t1}}, g_{1_{t2}}\right) = \boldsymbol{m}'_{1_{t1}}\boldsymbol{R}\boldsymbol{m}_{1_{t2}}$, where $\boldsymbol{m}'_{1_{t2}}$ is the parent-specific vector of marker effects for trait $t2$ of the first parent. Applying the formula $var\left(\boldsymbol{g}_i\right) = \boldsymbol{M}_i\boldsymbol{R}\boldsymbol{M}'_i$ (Eq. 5), we obtain

$$var\left(\boldsymbol{g}_1\right) = \boldsymbol{M}_1\boldsymbol{R}\boldsymbol{M}'_1 = \begin{bmatrix} 1 & 0.2 & 0.02 \\ 0.2 & -0.8 & 0.4 \end{bmatrix}\cdot\begin{bmatrix} 0.25 & 0.2 & 0.12 \\ 0.2 & 0.25 & 0.15 \\ 0.12 & 0.15 & 0.25 \end{bmatrix}\cdot\begin{bmatrix} 1 & 0.2 \\ 0.2 & -0.8 \\ 0.02 & 0.4 \end{bmatrix} = \begin{bmatrix} 0.3461 & -0.0819 \\ -0.0819 & 0.0692 \end{bmatrix},$$

where the rows of matrix $\boldsymbol{M}_1$ are parent-specific vectors for each trait ($\boldsymbol{m}'_{1_{t1}}$ and $\boldsymbol{m}'_{1_{t2}}$) of the first parent. The diagonal elements of $var\left(\boldsymbol{g}_1\right)$ are the MSVs of each trait, whereas the off-diagonal elements are the MSCs. Like MSV, the MSC between multiple traits does not depend on the haplotype order chosen when setting up the parent-specific vector $\boldsymbol{m}_i$.



Supplementary Figure 2: A probability tree showing the segregation patterns (sequence of alleles from first (A) and second (B) parental haplotypes), probabilities of occurrence, and the genetic values for multiple traits ($t1$ and $t2$) of the first parent (g_1).

A.3 Conditional covariance of Mendelian sampling values of gametes from different parents

The Mendelian sampling processes in two parents are random and independent. Enumerating the covariance between two parents from a bivariate distribution of gametes with probabilities of all combinations for possible gametes would yield a covariance of zero (Supplementary Table 1). As shown in Table 1 and Supplementary Figure 1, the probability distribution gives the frequencies of all possible segregation patterns of gametes ($A-A-A,\ldots,B-B-B$) within a family. Assuming identical probabilities of the within-family distribution of segregation patterns of gametes from two parents—which usually have different genetic values depending on their genetic makeup—we can obtain a non-zero covariance between two parents. However, this is conditional on the chosen order of haplotypes in the parent-specific vector $\boldsymbol{m}_i$.

The derivation of conditional covariance between Mendelian sampling values for trait $t1$ of the first and second parents, from a probability distribution, is 0.2449 (Table 1). The same result can be obtained using $ccov(g_i, g_j) = \boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_j$ (Eq. 7):

$$ccov\left(g_{1_{t1}}, g_{2_{t1}}\right) = \boldsymbol{m}_{1_{t1}}'\boldsymbol{R}\boldsymbol{m}_{2_{t1}} = \begin{bmatrix}1 & 0.2 & 0.02\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2 & 0.12\\ 0.2 & 0.25 & 0.15\\ 0.12 & 0.15 & 0.25\end{bmatrix}\cdot\begin{bmatrix}1\\ -0.2\\ 0.02\end{bmatrix} = 0.2449.$$

The vectors of parent-specific additive marker effects for trait $t1$ of the first and second parents have been given above. For the third, fourth, and fifth parents, the vectors are $\boldsymbol{m}_{3_{t1}}' = \begin{bmatrix}-1 & 0.2 & -0.02\end{bmatrix}$, $\boldsymbol{m}_{4_{t1}}' = \begin{bmatrix}0 & 0 & 0.02\end{bmatrix}$, and $\boldsymbol{m}_{5_{t1}}' = \begin{bmatrix}0 & 0 & -0.02\end{bmatrix}$, respectively. From this, a comprehensive conditional covariance matrix for all five parents for a single trait $t1$ can be produced:

$$\begin{bmatrix}0.3461 & 0.2449 & -0.2449 & 0.0031 & -0.0031\\ 0.2449 & 0.1837 & -0.1837 & 0.0019 & -0.0019\\ -0.2449 & -0.1837 & 0.1837 & -0.0019 & 0.0019\\ 0.0031 & 0.0019 & -0.0019 & 0.0001 & -0.0001\\ -0.0031 & -0.0019 & 0.0019 & -0.0001 & 0.0001\end{bmatrix}.$$

The MSVs of all parents constitute the diagonal elements of the matrix above, whereas the off-diagonal elements represent the conditional covariance between the Mendelian sampling values of the

parents. As seen in the matrix, the covariance between parents depends on the order of haplotypes. The conditional covariance between the first and second parents is 0.2449, while it is -0.2449 between the first and third parents, indicating that the second and third parents share similar heterozygosity but entirely swapped haplotype order.

Therefore, the similarity between parents i and j should be calculated using the formula $s_{ij} = \left| \boldsymbol{m}_i' \boldsymbol{R} \boldsymbol{m}_j \right|$ (Eq. 8) to obtain an unequivocal value of similarity independent of the order of haplotypes. Applying Eq. (8), we obtain the following similarity matrix:

$$\boldsymbol{S} = \begin{bmatrix} 0.3461 & 0.2449 & 0.2449 & 0.0031 & 0.0031 \\ 0.2449 & 0.1837 & 0.1837 & 0.0019 & 0.0019 \\ 0.2449 & 0.1837 & 0.1837 & 0.0019 & 0.0019 \\ 0.0031 & 0.0019 & 0.0019 & 0.0001 & 0.0001 \\ 0.0031 & 0.0019 & 0.0019 & 0.0001 & 0.0001 \end{bmatrix}.$$

Standardizing matrix $\boldsymbol{S}$, according to Eq. (16), yields:

$$\boldsymbol{K} = \begin{bmatrix} 1 & 0.9713 & 0.9713 & 0.5270 & 0.5270 \\ 0.9713 & 1 & 1 & 0.4433 & 0.4433 \\ 0.9713 & 1 & 1 & 0.4433 & 0.4433 \\ 0.5270 & 0.4433 & 0.4433 & 1 & 1 \\ 0.5270 & 0.4433 & 0.4433 & 1 & 1 \end{bmatrix}.$$

Matrices $\boldsymbol{S}$ and $\boldsymbol{K}$ show that parents with a common heterozygosity of markers with large effects (first and second parents) appear to be more similar than those with small effects (first and fifth parents).

Kullback-Leibler (Kullback and Leibler, 1951) and Jensen-Shannon (Nielsen, 2019) divergence are other options for deriving similarities among probability distributions. However, they compare the probabilities of segregation patterns of gametes produced by two parents without reference to the trait-specific additive genetic values of those haplotypes. In contrast, our approach for deriving the similarity between two parents considers both the probabilities of segregation patterns and the trait-specific additive genetic values of the gametes produced.

Supplementary Table 1: Bivariate probability distribution of segregation patterns of gametes in parents i and j

Parent i \ Parent j	$A-A-A$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\right)$	$A-A-B$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\right)$	$A-B-A$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\right)$	$A-B-B$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\right)$	$B-A-A$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\right)$	$B-A-B$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\right)$	$B-B-A$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\right)$	$B-B-B$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\right)$
$A-A-A$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\right)$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$							
$A-A-B$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\right)$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$						
$A-B-A$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\right)$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$					
$A-B-B$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\right)$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$				
$B-A-A$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\right)$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$			
$B-A-B$ $\left(\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\right)$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$		
$B-B-A$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\right)$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	
$B-B-B$ $\left(\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\right)$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot(1-\theta_{2,3})$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot\theta_{1,2}\cdot\theta_{2,3}$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot\theta_{2,3}$	$\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})\times$ $\frac{1}{2}\cdot(1-\theta_{1,2})\cdot(1-\theta_{2,3})$

A represents alleles on the first haplotype; B represents alleles on the second haplotype. Only the lower triangle is shown.

B. Zygotes

B.1 Similarity between Mendelian sampling values of zygotes

We consider the same parents and assumptions as depicted in Table 2. The haplotypes a zygote inherits from both parents can be characterized by the segregation patterns of parental haplotypes. For two biallelic linked marker loci, the probability of segregation patterns of alleles into gametes is $\frac{1}{2}$ × the probability of recombination of the first allele with the second. Thus, the probability of a zygote's segregation pattern is the product of the probabilities of segregation patterns of gametes from both parents (Supplementary Figure 3). As depicted in Supplementary Figure 3, parent-pairs *ij* and *uv* produce zygotes that share similar segregation patterns, but the genetic values of the zygotes differ due to the genetic makeup of their parents. Considering this fact, we can compute a conditional covariance between the genetic values of zygotes produced by different families.

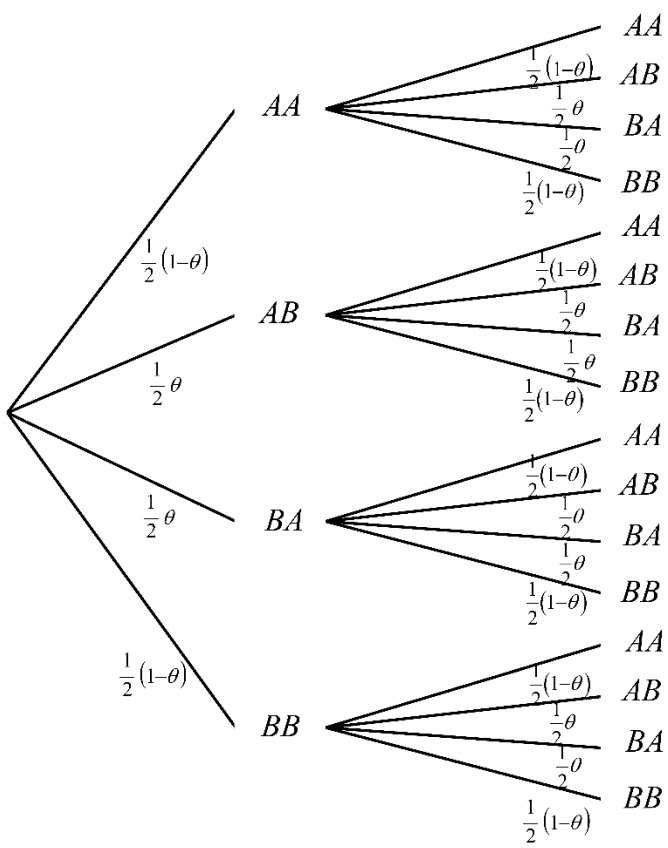
The derivation of MSV for parent-pairs *ij* (0.68) and *uv* (0.26), as well as their conditional covariance (0.26) have been derived using the probability distribution depicted in Table 2. The same results for MSV can be obtained using matrix expression $var\left(g_{ij}\right) = \boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_i + \boldsymbol{m}_j'\boldsymbol{R}\boldsymbol{m}_j$ (Eq. 10),

$$var\left(g_{ij}\right) = \begin{bmatrix}1 & 0.2\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2\\ 0.2 & 0.25\end{bmatrix}\cdot\begin{bmatrix}1\\ 0.2\end{bmatrix} + \begin{bmatrix}1 & 0.2\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2\\ 0.2 & 0.25\end{bmatrix}\cdot\begin{bmatrix}1\\ 0.2\end{bmatrix} = 0.68$$

$$var\left(g_{uv}\right) = \begin{bmatrix}0 & 0.2\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2\\ 0.2 & 0.25\end{bmatrix}\cdot\begin{bmatrix}0\\ 0.2\end{bmatrix} + \begin{bmatrix}1 & 0\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2\\ 0.2 & 0.25\end{bmatrix}\cdot\begin{bmatrix}1\\ 0\end{bmatrix} = 0.26,$$

and for conditional covariance using $ccov\left(g_{ij}, g_{uv}\right) = \boldsymbol{m}_i'\boldsymbol{R}\boldsymbol{m}_u + \boldsymbol{m}_j'\boldsymbol{R}\boldsymbol{m}_v$ (Eq. 12),

$$ccov\left(g_{ij}, g_{uv}\right) = \begin{bmatrix}1 & 0.2\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2\\ 0.2 & 0.25\end{bmatrix}\cdot\begin{bmatrix}0\\ 0.2\end{bmatrix} + \begin{bmatrix}1 & 0.2\end{bmatrix}\cdot\begin{bmatrix}0.25 & 0.2\\ 0.2 & 0.25\end{bmatrix}\cdot\begin{bmatrix}1\\ 0\end{bmatrix} = 0.34\,.$$



Pattern	Probability	g_{ij}	g_{uv}
AA / AA	$\frac{1}{4}(1-\theta)^2$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(m_1+m_2)=1.2$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(m_1-m_2)=0.6$
AA / AB	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(m_1-m_2)=1$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(m_1-m_2)=0.6$
AA / BA	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(-m_1+m_2)=0.2$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(-m_1-m_2)=-0.4$
AA / BB	$\frac{1}{4}(1-\theta)^2$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(-m_1-m_2)=0$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(-m_1-m_2)=-0.4$
AB / AA	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(m_1+m_2)=1$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(m_1-m_2)=0.4$
AB / AB	$\frac{1}{4}\cdot\theta^2$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(m_1-m_2)=0.8$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(m_1-m_2)=0.4$
AB / BA	$\frac{1}{4}\cdot\theta^2$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(-m_1+m_2)=0$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(-m_1-m_2)=-0.6$
AB / BB	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(m_1-m_2)=0.8$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(-m_1-m_2)=-0.6$
BA / AA	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$\frac{1}{2}(-m_1+m_2)+\frac{1}{2}(m_1+m_2)=0.2$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(m_1-m_2)=0.6$
BA / AB	$\frac{1}{4}\cdot\theta^2$	$\frac{1}{2}(-m_1+m_2)+\frac{1}{2}(m_1-m_2)=0$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(m_1-m_2)=0.6$
BA / BA	$\frac{1}{4}\cdot\theta^2$	$\frac{1}{2}(-m_1+m_2)+\frac{1}{2}(-m_1+m_2)=-0.8$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(-m_1-m_2)=-0.4$
BA / BB	$\frac{1}{4}\cdot\theta\cdot(1-\theta)$	$\frac{1}{2}(-m_1+m_2)+\frac{1}{2}(-m_1-m_2)=-1$	$\frac{1}{2}(m_1+m_2)+\frac{1}{2}(-m_1-m_2)=-0.4$
BB / AA	$\frac{1}{4}(1-\theta)^2$	$\frac{1}{2}(-m_1-m_2)+\frac{1}{2}(m_1+m_2)=0$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(m_1-m_2)=0.4$
BB / AB	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$\frac{1}{2}(-m_1-m_2)+\frac{1}{2}(m_1-m_2)=-0.2$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(m_1-m_2)=0.4$
BB / BA	$\frac{1}{4}\cdot(1-\theta)\cdot\theta$	$\frac{1}{2}(-m_1-m_2)+\frac{1}{2}(-m_1+m_2)=-1$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(-m_1-m_2)=-0.6$
BB / BB	$\frac{1}{4}(1-\theta)^2$	$\frac{1}{2}(-m_1-m_2)+\frac{1}{2}(-m_1-m_2)=-1.2$	$\frac{1}{2}(m_1-m_2)+\frac{1}{2}(-m_1-m_2)=-0.6$

Supplementary Figure 3: A probability tree showing the segregation patterns (sequence of alleles from first (A) and second (B) parental haplotypes), probabilities of occurrence, and the genetic values of zygotes from parent-pairs ij and uv.

Additional file 2: Estimated marker effects for milk fat, protein, and pH

The file contains the estimated marker effects for approximately 40 000 markers. As a result of its length, the file was excluded.

LITERATURE CITED

De Abreu Santos, D. J., J. B. Cole, G. E. Liu, P. M. Vanraden, and L. Ma. 2020. Gamevar.f90: A software package for calculating individual gametic diversity. *BMC Bioinformatics*. 21:3–7.

Bernardo, R. 2014. Genomewide selection of parental inbreds: Classes of loci and virtual biparental populations. *Crop Sci.* 54:2586–2595.

Bijma, P., Y. C. J. Wientjes, and M. P. L. Calus. 2020. Breeding Top Genotypes and Accelerating Response to Gametic Variance. *Genetics*. 214:91–107.

Bonk, S., M. Reichelt, F. Teuscher, D. Segelke, and N. Reinsch. 2016. Mendelian sampling covariability of marker effects and genetic values. *Genet. Sel. Evol.* 48:1–11.

Cole, J. B., and P. M. VanRaden. 2011. Use of haplotypes to estimate Mendelian sampling effects and selection limits. *J. Anim. Breed. Genet.* 128:446–455.

Daetwyler, H. D., M. J. Hayden, G. C. Spangenberg, and B. J. Hayes. 2015. Selection on optimal haploid value increases genetic gain and preserves more genetic diversity relative to genomic selection. *Genetics*. 200:1341–1348.

Ferdosi, M. H., and C. Gondro. R package "hsphase": an R package for identification of recombination events, pedigree and haplotype reconstruction and imputation using SNP data from half-sib families. R package version 2.0.2; 2018. https://cran.r-project.org/web/packages/hsphase/vignet.

Goiffon, M., A. Kusmec, L. Wang, G. Hu, and P. S. Schnable. 2017. Improving response in genomic selection with a population-based selection strategy: Optimal population value selection. *Genetics*. 206:1675–1682.

Gurobi Optimization LLC. gurobi: Gurobi Optimizer 9.1 interface. R package version 9.1-0; 2020. https://www.gurobi.com.

Haldane, J. B. S. 1919. The combination of linkage values and the calculation of distances between the loci of linked factors. *J. Genet.* 8:299–309.

Hampel, A., F. Teuscher, L. Gomez-Raya, M. Doschoris, and D. Wittenburg. 2018. Estimation of Recombination Rate and Maternal Linkage Disequilibrium in Half-Sibs. *Front. Genet.* 9:186.

Heffner, E. L., M. E. Sorrells, and J. L. Jannink. 2009. Genomic selection for crop improvement. *Crop*

Sci. 49:1–12.

Hickey, J. M., T. Chiurugwi, I. Mackay, and W. Powell. 2017. Genomic prediction unifies animal and plant breeding programs to form platforms for biological discovery. *Nat. Genet.* 49:1297–1303.

Kemper, K. E., P. J. Bowman, J. E. Pryce, B. J. Hayes, and M. E. Goddard. 2012. Long-term selection strategies for complex traits using high-density genetic markers. *J. Dairy Sci.* 95:4646–4656.

Kolde, R. 2019. pheatmap: Pretty Heatmaps. R package version 1.0.12; 2019. https://CRAN.R-project.org/package=pheatmap.

Kullback, S., and R. A. Leibler. 1951. On Information and Sufficiency. *Ann. Math. Stat.* 22:79–86.

Lehermeier, C., S. Teyssèdre, and C. C. Schön. 2017. Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. *Genetics*. 207:1651–1661.

Lin, Z., N. O. I. Cogan, L. W. Pembleton, G. C. Spangenberg, J. W. Forster, B. J. Hayes, and H. D. Daetwyler. 2016. Genetic Gain and Inbreeding from Genomic Selection in a Simulated Commercial Breeding Program for Perennial Ryegrass. *Plant Genome*. 9:1–12.

Liu, H., T. H. Meuwissen, A. C. Sørensen, and P. Berg. 2015. Upweighting rare favourable alleles increases long-term genetic gain in genomic selection programs. *Genet. Sel. Evol.* 47:19.

Markowitz, H. 1952. Portfolio selection. *J. Finance*. 7:77–91.

Melzer, N., D. Wittenburg, and D. Repsilber. 2013. Integrating Milk Metabolite Profile Information for the Prediction of Traditional Milk Traits Based on SNP Information for Holstein Cows. *PLoS One*. 8:e70256.

Meuwissen, T. H. E. 1997. Maximizing the Response of Selection with a Predefined Rate of Inbreeding. *J. Anim. Sci.* 75:934–940.

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard. 2001. Prediction of total genetic value using genome-wide dense marker maps. *Genetics*. 157:1819–1829.

Meuwissen, T. H. E., and A. K. Sonesson. 1998. Maximizing the Response of Selection with a Predefined Rate of Inbreeding: Overlapping Generations. *J. Anim. Sci.* 76:2575–2583.

Moeinizade, S., G. Hu, L. Wang, and P. S. Schnable. 2019. Optimizing selection and mating in genomic selection with a look-ahead approach: An operations research framework. *G3 Genes, Genomes, Genet.* 9:2123–2133.

Mohammadi, M., T. Tiede, and K. P. Smith. 2015. Popvar: A genome-wide procedure for predicting genetic variance and correlated response in biparental breeding populations. *Crop Sci.* 55:2068–2077.

Müller, D., P. Schopp, and A. E. Melchinger. 2018. Selection on expected maximum haploid breeding

values can increase genetic gain in recurrent genomic selection. *G3 Genes, Genomes, Genet.* 8:1173–1181.

Nielsen, F. 2019. On the Jensen-Shannon symmetrization of distances relying on abstract means. *Entropy*. 21:485.

R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna; 2020. OpenURL.

Rutkoski, J., R. P. Singh, J. Huerta-Espino, S. Bhavani, J. Poland, J. L. Jannink, and M. E. Sorrells. 2015. Genetic gain from phenotypic and genomic selection for quantitative resistance to stem rust of wheat. *Plant Genome*. 8:1–10.

Santos, D. J. A., J. B. Cole, T. J. Lawlor, P. M. VanRaden, H. Tonhati, and L. Ma. 2019. Variance of gametic diversity and its application in selection programs. *J. Dairy Sci.* 102:5279–5294.

Schaeffer, L. R. 2006. Strategy for applying genome-wide selection in dairy cattle. *J. Anim. Breed. Genet.* 123:218–223.

Schnell, F. W., and H. F. Utz. 1976. F1-Leistung und Elternwahl in der Züchtung von Selbstbefruchtern. *Bericht über die Arbeitstagung der Vereinigung Österreichischer Pflanzenzüchter, pp. 234–258, Gumpenstein, Österreich*.

Segelke, D., F. Reinhardt, Z. Liu, and G. Thaller. 2014. Prediction of expected genetic variation within groups of offspring for innovative mating schemes. *Genet. Sel. Evol.* 46:42.

Sonesson, A. K., J. A. Woolliams, and T. H. E. Meuwissen. 2012. Genomic selection requires genomic control of inbreeding. *Genet. Sel. Evol.* 44.

VanRaden, P. M. 2008. Efficient methods to compute genomic predictions. *J. Dairy Sci.* 91:4414–4423.

Wei, T., and S. Viliam. 2017. R package "corrplot": Visualization of a Correlation Matrix. R package version 0.84; 2017. https://github.com/taiyun/corrplot.

Woolliams, J. A., P. Berg, B. S. Dagnachew, and T. H. E. Meuwissen. 2015. Genetic contributions and their optimization. *J. Anim. Breed. Genet.* 132:89–99.

CHAPTER 3

PyMSQ: A Python package for estimating Mendelian sampling-related quantities

A. A. Musa and N Reinsch

Research Institute for Farm Animal Biology (FBN), Institute of Genetics and Biometry, Wilhelm-Stahl-Allee 2, 18196 Dummerstorf, Germany.

PyMSQ: A Python package for estimating Mendelian sampling-related quantities

ABSTRACT

Background: Mendelian sampling variance (MSV), which quantifies the chance of producing offspring with extreme breeding values, has a wide range of breeding applications, including mating decisions. A fast approach for estimating MSV and a similarity matrix that can be combined with a selection criterion to select an optimal set of parents to hedge haplotype diversity have recently been developed. However, they have not yet been implemented in any software, thus limiting their use in plant and animal breeding programs.

Results: Here, we present *PyMSQ*, a Python package for efficiently estimating MSVs and similarity matrices of gametes produced by individuals using marker (or allele substitution) effects, genetic maps, and phased genotypes. In addition, *PyMSQ* computes numerous selection criteria, supports group-specific genetic maps, and extends all calculations to zygotes. We estimated MSVs and derived similarity matrices using a publicly available Holstein-Friesian cattle dataset to assess our package. Finally, we performed a simulation study for benchmarking the time needed for *PyMSQ* and *gamevar* (a recently published Fortran program) to estimate MSVs. We found that *PyMSQ* was faster in all scenarios (up to 240 times faster depending on the number of markers and individuals).

Conclusion: *PyMSQ* is fast and easy to use, and it will increase the widespread use of Mendelian sampling-related quantities in breeding programs for genetic improvement, optimal mating decision-making, and genetic diversity management. It can be installed manually or with pip, the Python package installer. In addition, its source code, documentation, illustration, and empirical data are available to download for free at https://github.com/aromemusa/PyMSQ.

Keywords: Mendelian sampling variance, gametic variance, haplotype similarity matrices, optimal mating decisions, genetic diversity management

BACKGROUND

Long-term genetic improvements in plants and animals necessitate the use of selection criteria that maximize short-term gains while minimizing inbreeding and genetic diversity loss (Jannink, 2010; Woolliams et al., 2015). Conventional genomic breeding programs select parents solely based on the expected genomic estimated breeding value (GEBV) of their future offspring, defined as an average of its parental GEBVs. GEBV is the sum of all estimated marker effects of an individual (Meuwissen et al., 2001). Selection based on GEBV has been widely used in breeding programs due to its success in delivering short-term gains. However, its continuous use promotes inbreeding (Schaeffer, 2006; Liu et al., 2015; Lin et al., 2016), genetic diversity loss, and decreased genetic progress (Estaghvirou et al., 2015). Therefore, several selection criteria have been proposed to counteract these limitations (e.g., Daetwyler et al., 2015; Goiffon et al., 2017; Moeinizade et al., 2019).

Recently proposed selection criteria to achieve the optimal tradeoff between short- and long-term gains are based on an index (Schnell and Utz, 1976; Lehermeier et al., 2017; Müller et al., 2018; Bijma et al., 2020). These indices combine the expected GEBV and within-family standard deviation of the future offspring's GEBVs weighted by a constant. While the GEBV term maximizes short-term gains, the within-family standard deviation (square root of the Mendelian sampling variance, MSV) term maximizes genetic variability or the chance of breeding top-ranking individuals. MSV is predicted from marker effects, genetic maps, and phased genotypes using either a simulation method (Cole and VanRaden, 2011; Bernardo, 2014; Segelke et al., 2014; Mohammadi et al., 2015) or an analytical method (Bonk et al., 2016; Santos et al., 2019). Selections based on these indices yield a more substantial long-term genetic gain than selection based solely on GEBVs (Lehermeier et al., 2017; Müller et al., 2018; Bijma et al., 2020). There are, however, two limitations. First, computing MSV takes a long time. Second, similar parents with high genetic variability potential tend to be selected, potentially leading to unintended losses of favorable haplotypes.

As a solution, Musa and Reinsch (2020) proposed a computationally faster approach for computing MSV that took only a fraction of the time required by the method outlined by Bonk et al. (2016). To hedge against haplotype losses, they also proposed a measure of haplotype similarity between parents (Musa and Reinsch, 2021), which can be combined into a similarity matrix using the parts of this approach. The off-diagonal elements of this matrix reflect pairwise haplotype similarities, while diagonal elements reflect a parent's similarity with itself, which equals the MSV of its gametes (Musa

and Reinsch, 2021). High similarity indicates that the parents share many heterozygous markers with large effects on a trait in the same linkage phase. Following the efficient frontier (Markowitz, 1952), selection criteria can be used with a similarity matrix to select the optimal set of parents while balancing parental haplotype diversity with expected genetic gain in the next generation (Musa and Reinsch, 2021). This approach also permits extensions to include additional constraints, such as the maximum amount of inbreeding permitted and the minimum/maximum number of allocations to parents. The similarity matrix has numerous potential applications in plant and animal breeding, particularly in optimal mating decisions and management of genetic diversity. However, neither the similarity matrix nor the rapid approach for estimating MSVs has been implemented in any software.

Therefore, this study implemented our recently introduced methods into the *PyMSQ* software, which rapidly predicts Mendelian sampling (co-)variances of gametes and haplotype similarity matrices for complex traits for many markers and individuals, thus making it suitable for large-scale breeding programs. *PyMSQ* also calculates various selection criteria for individuals and extends all calculations to zygotes (pairs of parents). To evaluate our package, we analyzed a publicly available Holstein-Friesian cattle dataset. Finally, we compared the times required by *PyMSQ* and *gamevar* (a recently published Fortran program; De Abreu Santos et al., 2020) to estimate the Mendelian sampling (co-)variances of gametes using various simulated data.

IMPLEMENTATION

PyMSQ is written in Python and relies on several Python libraries for data management, scientific computations, and compilation (specifically, pandas, NumPy, SciPy, and Numba). For simplicity, *PyMSQ* comprises one module, named *msq*. Besides implementing a fast approach for estimating Mendelian sampling (co-)variance (Musa and Reinsch, 2021), most *PyMSQ* functions use the vectorization and matrix multiplication capabilities of NumPy and SciPy for fast and efficient computations. Three main types of data—namely phased genotypes, genetic maps, and estimated marker effects—are required to use most *PyMSQ* functions. Detailed documentation is available at https://github.com/aromemusa/PyMSQ/blob/main/docs/docs_msq.md. The implementations of specific functions of this package are described below.

Mendelian sampling (co-)variance for gametes

MSV quantifies the deviation of the realized offspring GEBVs from the expected parental average. The GEBV bv_i of parent i is calculated as follows:

$$bv_i = \boldsymbol{c}_i' \boldsymbol{m} \,, \tag{20}$$

where $\boldsymbol{m}$ is a vector of additive marker (or allele substitution) effects, and $\boldsymbol{c}_i'$ is a row vector describing the genotype of parent i, with elements 1, 0, and -1 if the genotype at each locus is AA, AB or BA, and BB, respectively (Meuwissen et al., 2001). Then, *PyMSQ* calculates the MSV of the additive genetic values g_i of the gametes produced by parent i using a computationally faster representation of the method by Bonk et al. (2016) as follows:

$$\sigma^2_{g_i} = \sum_c \left(\boldsymbol{m}_i^c\right)' \boldsymbol{R}^c \boldsymbol{m}_i^c \;, \tag{21}$$

where $\boldsymbol{m}_i^c$ is the parent-specific vector of additive marker (or allele substitution) effects for parent i on chromosome c and $\boldsymbol{R}^c$ is the population covariance matrix reflecting expected within-family linkage disequilibrium of markers on the same chromosome (Musa and Reinsch, 2021). The element m_{ik}^c of the parent-specific vector $\boldsymbol{m}_i^c$ is given by the formula $m_{ik}^c = \delta_{ik} m_k^c$, where δ_{ik} and m_k^c are the phase indicator and marker (or allele substitution) effect, respectively, at locus k. If the standard allele is observed on the first haplotype, the phase indicator δ_{ik} is 1; otherwise, it is -1. For each homozygous genotype, the indicator is set to zero $\delta_{ik} = 0$. The element ρ_{kl}^c of $\boldsymbol{R}^c$ is calculated as $\rho_{kl}^c = \left(1 - 2r_{kl}^c\right)/4$ or $\rho_{kl}^c = \exp\left(-2d_{kl}^c\right)/4$ (Haldane, 1919), where r_{kl}^c and d_{kl}^c are the recombination rate and genetic distance (Morgan), respectively, between markers k and l on chromosome c. $\boldsymbol{R}^c$ has an autoregressive variance structure with variable marker distances (or recombination rates) and 0.25 along its diagonal.

The method by Bonk et al. (2016) is similar to that of Santos et al. (2019), though the underlying assumptions are different. Bonk et al. (2016) assumed Haldane's mapping function (Haldane, 1919), while Santos et al. (2019) assumed a linear relationship between recombination rate and genetic distance. Therefore, an equivalent but faster representation of the method by Santos et al. (2019) for estimating the MSV of the gametes produced by parent i is similar to formula (21). However, the

element ρ_{kl}^{c} of $\boldsymbol{R}^{c}$ is calculated as $\rho_{kl}^{c}=\left(-\frac{r_{kl}^{c}}{2}+0.25\right)$ or $\rho_{kl}^{c}=\left(-\frac{d_{kl}^{c}}{2}+0.25\right)$. Moreover, loci with r_{kl}^{c} or $d_{kl}^{c}\geq 0.5$ are assumed to be independent. Hence, $\rho_{kl}^{c}=0$.

The MSV of gametes produced by a potential parent i for a single trait can be extended to multiple traits as follows:

$$var\left(\boldsymbol{g}_{i}\right)=\sum_{c}\boldsymbol{M}_{i}^{c}\boldsymbol{R}^{c}\left(\boldsymbol{M}_{i}^{c}\right)', \tag{22}$$

where each row of matrix $\boldsymbol{M}_{i}^{c}$ is a vector $\left(\boldsymbol{m}_{it}^{c}\right)'$ of parent-specific additive marker effects on chromosome c for each trait t of parent i. The diagonal elements of formula (22) are the MSVs of all traits, whereas the off-diagonal elements are the Mendelian sampling covariances (MSCs) between traits. Therefore, the MSV of an aggregate genotype is calculated as $var\left(\boldsymbol{a}'\boldsymbol{g}_{i}\right)=\boldsymbol{a}'\cdot\left(\sum_{c}\boldsymbol{M}_{i}^{c}\boldsymbol{R}^{c}\left(\boldsymbol{M}_{i}^{c}\right)'\right)\cdot\boldsymbol{a}$, where $\boldsymbol{a}$ is a vector of index weights.

Similarities between gametes from two parents

The similarities between Mendelian sampling values of gametes produced by parents i and j are derived as follows:

$$s_{i,j}=\sum_{c}\left|\left(\boldsymbol{m}_{i}^{c}\right)'\boldsymbol{R}^{c}\boldsymbol{m}_{j}^{c}\right|, \tag{23}$$

where $\boldsymbol{m}_{i}^{c}\left(\boldsymbol{m}_{j}^{c}\right)$ is a vector of parent-specific additive marker effects for parent $i\left(j\right)$ on chromosome c (Musa and Reinsch, 2021). A trait-specific similarity matrix $\boldsymbol{S}$ can be assembled for a set of several individuals, with the MSVs of each parent serving as diagonal elements and similarities s_{ij} serving as off-diagonal elements. The similarity between potential parents i and j for a single trait can be extended to multiple traits with respect to aggregate genotype as $s_{ij}=\sum_{c}\left|\boldsymbol{a}'\cdot\left(\boldsymbol{M}_{i}^{c}\boldsymbol{R}^{c}\left(\boldsymbol{M}_{j}^{c}\right)'\right)\cdot\boldsymbol{a}\right|$, where each row of matrix $\boldsymbol{M}_{j}^{c}$ is a vector $\left(\boldsymbol{m}_{jt}^{c}\right)'$ of additive marker effects on chromosome c for each trait t of parent j.

Mendelian sampling (co-)variance and aggregate genotype for zygotes

The MSV of the additive genetic values g_{ij} of zygotes from parents i and j is the sum of two independent contributions from paternal and maternal gametes (Bonk et al., 2016; Musa and Reinsch, 2021):

$$var\left(g_{ij}\right) = \sum_c \left(\boldsymbol{m}_i^c\right)' \boldsymbol{R}^c \boldsymbol{m}_i^c + \sum_c \left(\boldsymbol{m}_j^c\right)' \boldsymbol{R}^c \boldsymbol{m}_j^c \,. \tag{24}$$

The extension to additive values for multiple traits is the Mendelian (co-)variance matrix, computed as follows:

$$var\left(\boldsymbol{g}_{ij}\right) = \sum_c \boldsymbol{M}_i^c \boldsymbol{R}^c \left(\boldsymbol{M}_i^c\right)' + \sum_c \boldsymbol{M}_j^c \boldsymbol{R}^c \left(\boldsymbol{M}_j^c\right)' = \boldsymbol{V}_{ij} \,. \tag{25}$$

Matrix $\boldsymbol{V}_{ij}$ has zygotic MSVs for all traits as diagonal and MSCs between traits as off-diagonal elements. Then, the MSV of the aggregate genotype for the parent pair ij is given by the quadratic form $var\left(\boldsymbol{a}'\boldsymbol{g}_{ij}\right) = \boldsymbol{a}'\boldsymbol{V}_{ij}\boldsymbol{a}$.

Similarities between zygotes from two parents

If $\boldsymbol{m}_i^c$, $\boldsymbol{m}_j^c$, $\boldsymbol{m}_u^c$, and $\boldsymbol{m}_v^c$ are additive marker effects for parents i , j , u , and v , then the similarity $s_{ij,uv}$ between the Mendelian sampling values of zygotes produced by parent pairs ij and uv is given for a single trait as:

$$s_{ij,uv} = \sum_c \left(\left| \left(\boldsymbol{m}_i^c\right)' \boldsymbol{R}^c \boldsymbol{m}_u^c \right| + \left| \left(\boldsymbol{m}_j^c\right)' \boldsymbol{R}^c \boldsymbol{m}_v^c \right| \right). \tag{26}$$

For multiple traits, the similarity is given as

$$s_{ij,uv} = \sum_c \left[\left| \boldsymbol{a}' \cdot \left(\boldsymbol{M}_i^c \boldsymbol{R}^c \left(\boldsymbol{M}_u^c\right)' \right) \cdot \boldsymbol{a} \right| + \left| \boldsymbol{a}' \cdot \boldsymbol{M}_j^c \boldsymbol{R}^c \left(\boldsymbol{M}_v^c\right)' \cdot \boldsymbol{a} \right| \right], \tag{27}$$

where each row of matrices $\boldsymbol{M}_u^c$ and $\boldsymbol{M}_v^c$ are vectors $\left(\boldsymbol{m}_{ut}^c\right)'$ and $\left(\boldsymbol{m}_{vt}^c\right)'$ of additive marker effects on chromosome c for each trait t of parents u and v , respectively (Musa and Reinsch, 2021). Again, MSVs and pairwise similarities can be combined in a similarity matrix $\boldsymbol{S}$, with the diagonal and off-

diagonal elements representing MSVs and similarities between Mendelian sampling values of zygotes, respectively.

Standardized similarity

A gamete's (or zygote's) similarity matrix $\boldsymbol{S}$ can be standardized by pre- and post-multiplying it by a diagonal matrix $\boldsymbol{D}^{-1}$ containing inverse Mendelian sampling standard deviations:

$$\boldsymbol{K} = \boldsymbol{D}^{-1} \cdot \boldsymbol{S} \cdot \boldsymbol{D}^{-1}, \tag{28}$$

where $\boldsymbol{D} = \sqrt{diag(\boldsymbol{S})}$ (Musa and Reinsch, 2021). As in correlation matrices, $\boldsymbol{K}$ has ones along its diagonal, and all off-diagonal entries are strictly non-negative within the range of $0 \leq k_{ij} \leq 1$.

Selection criteria

PyMSQ calculates GEBV bv for parent i as already described in formula (20) and for parent pair ij as $bv_{ij} = \left(\boldsymbol{c}_i'\boldsymbol{m} + \boldsymbol{c}_j'\boldsymbol{m}\right)/2$. Then, the probability p that a parent (or parent pair) will breed top-ranking individuals is calculated as $p = 1 - \phi\left(\frac{t - bv}{\sigma}\right)$, where t is the threshold, bv is the GEBV for parent i (or parent pair ij), σ is the Mendelian sampling standard deviation (square root of the MSV) for parent i (or parent pair ij), and ϕ is the cumulative distribution function of a standard normal distribution (Musa and Reinsch, 2021). Finally, *PyMSQ* calculates a linear approximation of p using an index $I = bv + k\sigma$, where k is a constant (Schnell and Utz, 1976; Müller et al., 2018; Bijma et al., 2020), and users can provide the value of k. If $k = x$, where x is the selection intensity (Falconer and Mackay, 1996), then the index is equivalent to the usefulness criterion (Schnell and Utz, 1976; Lehermeier et al., 2017). If $k = \sqrt{2}x$, where x is the standardized normal truncation point of the selected proportion, the index I is equivalent to that proposed by Bijma et al. (2020).

Data and analysis

Using *PyMSQ* and a publicly available empirical Holstein-Friesian cattle dataset, we estimated Mendelian sampling (co-)variances and derived haplotype similarity matrices for the aggregate genotype of gametes (assuming an equal index weight of 1). There are 265 cows (32–106 cows per paternal half-sib family) in the dataset, a physical map, genotypic data, pedigree information, and phenotypes for three milk traits: pH, fat (FY), and protein (PY) yields (Hampel et al., 2018). In

previous studies, the marker effects of this population were estimated (Melzer et al., 2013), genotypes were phased into haplotypes, and genetic map positions were approximated (Musa and Reinsch, 2021). The genetic map, phased genotypes, marker effects, and pedigree information are all available as package data in *PyMSQ*.

We visualized the Mendelian sampling (co-)variances by deriving density plots using *ggplot2* version 3.3.3 (Wickham, 2016) and visualized the similarity matrices using *corrplot* version 0.84 (Wei and Viliam, 2017). Then, we performed family-wise clustering of the similarities using *pheatmap* version 1.0.12 (Kolde, 2019). For analysis and visualization of results, see Additional files 1 and 2 in the supplementary materials.

To benchmark the performance of *PyMSQ* and *gamevar* (De Abreu Santos et al., 2020), we simulated two data groups. Both groups had a single chromosome and ten traits. However, the first group of data contained 20 datasets, each of which contained 1,000 markers but differing numbers of individuals. The second group was similar to the first but had 500 individuals and differing numbers of markers. We repeated the analysis ten times with each software package and compared their performances using the mean of the time required to estimate Mendelian (co-)variances on a multi-user Linux server (Intel Xeon Gold 6130, 128 CPU @ 2.10GHz, 1536 GB RAM).

RESULTS AND DISCUSSION

We found a high level of variation between individuals. Specifically, we observed coefficients of variation for MSVs of 94, 72, and 59% in FY, PY, and pH, respectively, as well as a coefficient of variation of 89% for the MSV of the aggregate genotype. In addition, the distributions of the traits were bimodal, with the FY peaks being further apart than the PY and pH peaks (Figure 1A) due to the stronger impact of *DGAT1* on FY than PY (Thaller et al., 2003; Santos et al., 2019). MSCs between traits were converted into correlations to facilitate comparisons of trait combinations (Figure 1B). Correlations between traits among individuals also showed large variations with bimodal distributions. For example, the correlations between FY and PY were mainly positive, which is in agreement with a previous study (Bonk et al., 2016). However, correlations between pH and other traits presented a substantial number of individuals with negative values. These results show that MSVs and MSCs differ between individuals. Therefore, their knowledge can be used by breeders to manage resources and optimize mating decisions.

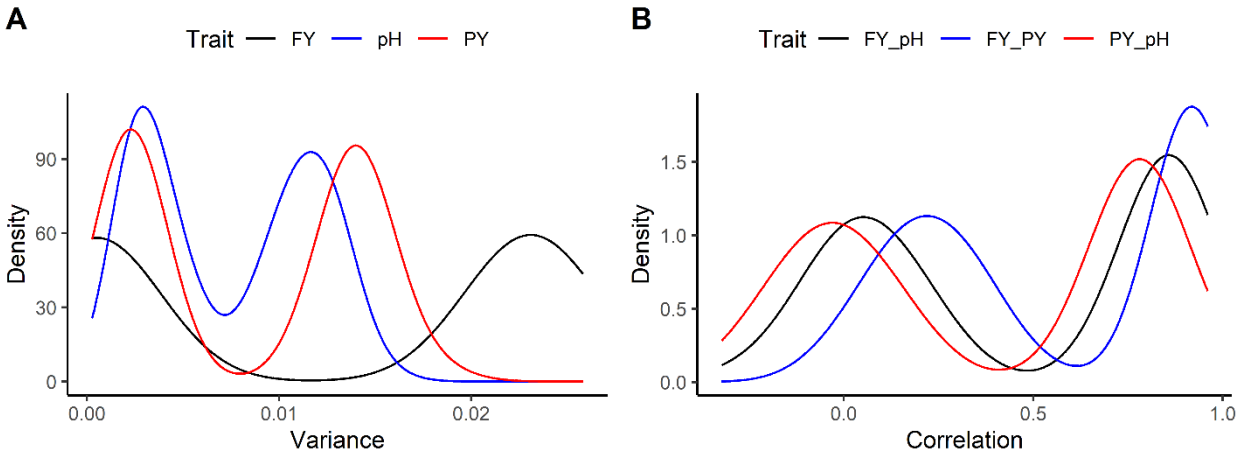


Fig. 1 Density plots of Mendelian sampling variances (A) and Mendelian correlations between milk traits (B). *FY* fat yield (kg), *PY* protein yield (kg), and pH (mol/L).

A similarity matrix of an aggregate genotype (AG) for multiple traits is depicted in Figure 2 (top left). Cows' MSVs for the AG are represented by the diagonal elements of this matrix, while the off-diagonal elements represent haplotype similarities between cows. The elements of this matrix depend on linkage, heterozygosity, and trait genetic architecture. Therefore, a high similarity between individuals can be interpreted as many shared chromosomal segments in the same linkage phase with large marker effects (Musa and Reinsch, 2021). The standardized similarity matrix (Figure 2, top right) shows a similar pattern to the similarity matrix (top left), but it highlighted small similarities more effectively. The standardized pairwise similarities varied greatly, with parents having nearly no similarities to nearly perfect similarities (0.04 to 0.99). Within-family clustering of the individuals revealed pronounced similarity patterns across families (bottom left and right). The (standardized) similarity matrices contribute to a breeder's toolbox for making optimal mating decisions and, consequently, increasing genetic gain and maintaining genetic diversity.

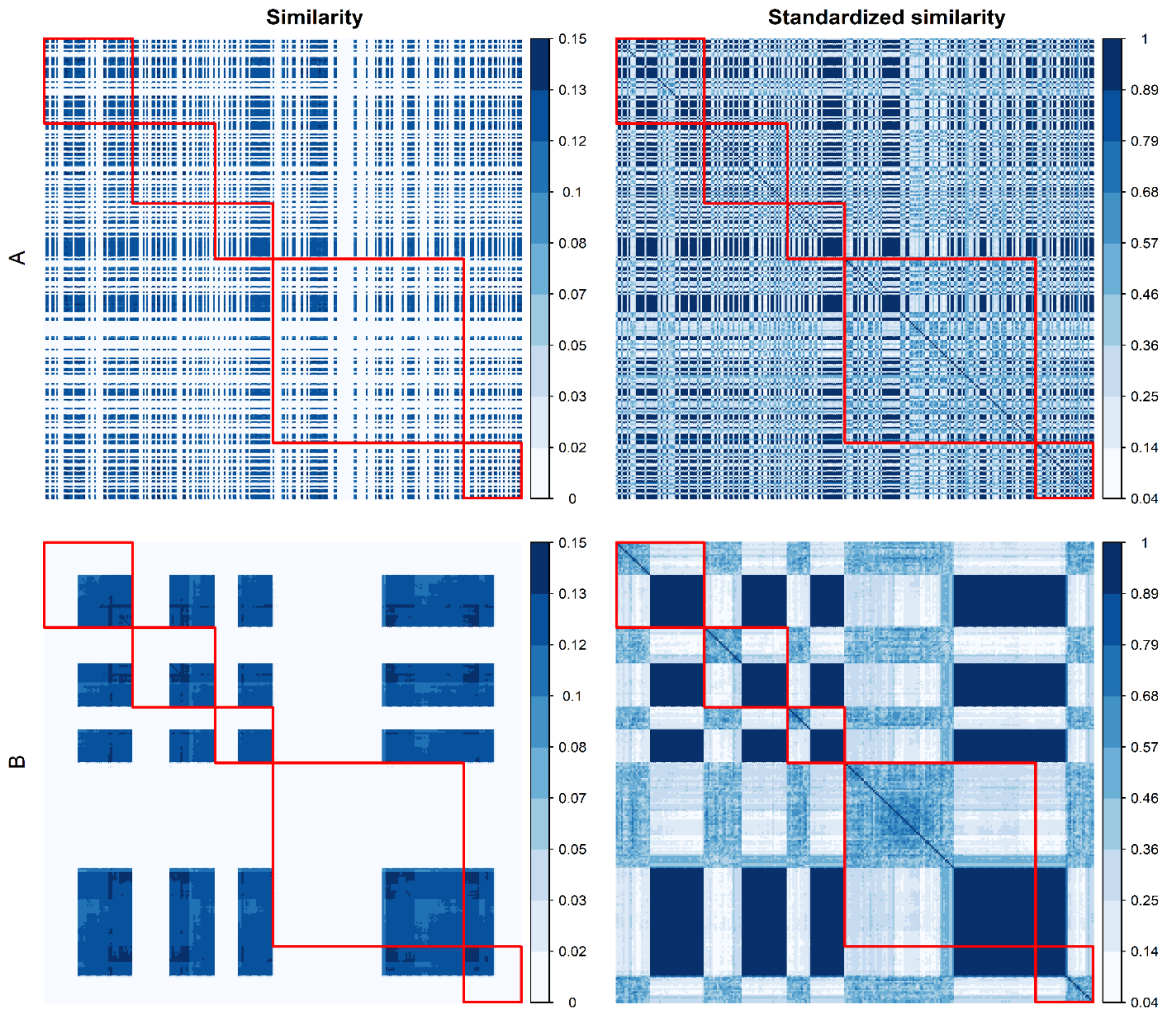


Fig. 2 Similarity matrices and Euclidean clustering of the matrices for the aggregate genotype of milk traits (assuming equal index weights of 1) for 265 cows from five half-sib families (separated by red lines). The Mendelian sampling variance of each cow and haplotype similarities between cows are located on the diagonal and off-diagonals, respectively (upper left). The standardized similarity matrix (upper right) makes pairwise haplotype similarities more visible, with cows presenting nearly no similarity to nearly perfect similarities. The Euclidean clustering of the similarity matrices within half-sib families reveals strong correlations across families (lower left and right).

PyMSQ computed Mendelian sampling (co-)variance faster than *gamevar* in all scenarios (Figure 3). Even though *PyMSQ* computes MSV for AG and covariances between AG and all traits (a feature not present in *gamevar*), it was still 16 to 46 times faster than *gamevar*, depending on the number of individuals (Figure 3A). Furthermore, in the simulation with varying markers, *PyMSQ* was 73 to 240 times faster than *gamevar* (Figure 3B).

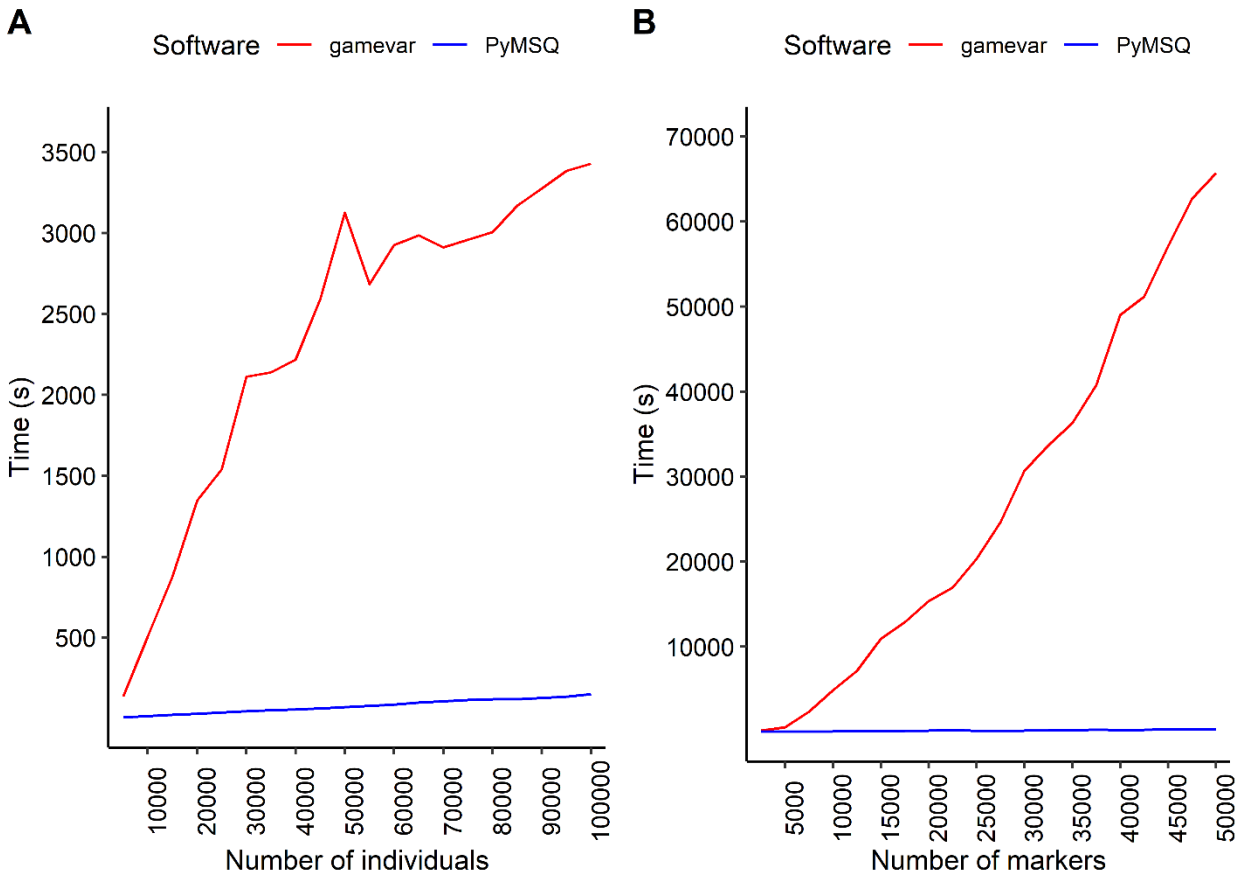


Fig. 3 The average time taken by *PyMSQ* and *gamevar* to compute Mendelian sampling (co-)variability on data with varied numbers of individuals (A) and markers (B).

In another simulation study with three traits and 39,780 markers for 29 chromosomes, it took *PyMSQ* an average of ~13s to 8.4 h to derive similarity matrices for aggregate genotypes for 1,000 to 100,000 individuals of the same sex (Table 1 in Additional file 3 of the supplementary material). The computation time shows that similarity matrices can be computed within a reasonable time, even when many markers and individuals are involved. Therefore, the benefits of applying such matrices can be exploited even in a large-scale breeding program.

CONCLUSIONS

Here, we present *PyMSQ*, a Python package for quickly estimating Mendelian sampling (co-)variances and haplotype similarity matrices for gametes (or zygotes) produced by individuals (or pairs of parents). These quantities have numerous plant and animal breeding applications, including genetic improvement, optimal decision-making, and population genetic diversity management. Additionally, *PyMSQ* supports the use of group-specific genomic maps and calculates a variety of selection criteria. A comparison between *PyMSQ* and *gamevar* showed that *PyMSQ* could be up to 240 times faster at estimating Mendelian sampling (co-)variances. *PyMSQ* is fast and easy to use, requiring only a basic understanding of statistics to enable applications in large-scale plant and animal breeding programs.

ACKNOWLEDGMENTS

Financial support for AAM from Bundesanstalt für Landwirtschaft und Ernährung (BLE) under Grant 281B101516 is gratefully acknowledged. The publication of this article was funded by the Open Access Fund of the Leibniz Association.

Availability and requirements

Project name: *PyMSQ*
Project home page: https://github.com/aromemusa/PyMSQ
Operating systems: Platform independent
Programming language: Python
Other requirements: pandas, NumPy, SciPy, and Numba Python libraries
License: MIT license
Any restrictions to use by non-academics: None

SUPPLEMENTARY MATERIAL

Additional file 1: Analysis of Holstein-Friesian cattle data.

```
#Import PyMSQ module and relevant libraries
from PyMSQ import msq
import numpy as np
import pandas as pd
import time
# Import example data
gmap, meff, gmat, group, ped = msq.example_data()
```

One of our main research interests is estimating Mendelian sampling variances for milk traits (fat, protein, and pH), as well as covariances and correlations between traits. We are also interested in creating a similarity matrix based on Mendelian sampling values for aggregate genotype assuming index weights of one.

```
# check main information for errors and create class object to store data
index_wt = [1, 1, 1]
start = time.time()
data = msq.Datacheck(gmap=gmap, meff=meff, gmat=gmat, group=group,
                     indwt=index_wt, progress=True)
print('Time taken: ', round(time.time() - start, 2), 'secs')

del gmap, meff, gmat, group, index_wt

Converting phased haplotypes to genotypes
Data passed the test!
Number of individuals:   265
Number of groups:        1 :  ['F']
Number of specific maps: 1
Number of chromosomes:   29
Total no. markers:       39780
Number of trait(s):      3
Trait name(s) and Index weight(s)
fat :  1
pH :  1
protein :  1
Time taken:  2.86 secs
# create population covariance matrices
start = time.time()
covmatrices = msq.popcovmat(info=data, mposunit="cM", method=1)
print('Time taken: ', round(time.time() - start, 2), 'secs')
Time taken:  1.97 secs
```

```
# Estimate Mendelian sampling (co)variance
start = time.time()
msvmsc_g = msq.msvarcov_g(info=data, covmat=covmatrices,
                          sub_id=None, progress=True)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(msvmsc_g)

Progress: |████████████████████████████████████████| 100% Complete
Time taken:  27.81 secs
        ID Group       fat    pH_fat        pH  protein_fat  protein_pH  \
0    10001     F  0.000466  0.000154  0.001805     0.000136   -0.000157
1    10002     F  0.023262  0.014908  0.013225     0.017306    0.011082
2    10003     F  0.021665  0.013760  0.010475     0.015662    0.009849
3    10004     F  0.000949  0.000317  0.004836    -0.000044   -0.000158
4    10005     F  0.021892  0.014447  0.012676     0.015576    0.010120
..     ...   ...       ...       ...       ...          ...         ...
260  10261     F  0.022112  0.012953  0.010050     0.017090    0.009524
261  10262     F  0.023853  0.016183  0.012896     0.017691    0.012189
262  10263     F  0.000604  0.000096  0.002694     0.000025    0.000126
263  10264     F  0.022809  0.015301  0.012748     0.017123    0.011174
264  10265     F  0.022491  0.016226  0.013892     0.016460    0.011973

      protein    AG_fat     AG_pH  AG_protein        AG
0    0.002321  0.000756  0.001802    0.002300  0.004858
1    0.015098  0.055475  0.039215    0.043486  0.138176
2    0.013710  0.051087  0.034084    0.039221  0.124392
3    0.001352  0.001223  0.004995    0.001149  0.007367
4    0.013318  0.051915  0.037243    0.039013  0.128172
..        ...       ...       ...         ...       ...
260  0.015024  0.052155  0.032527    0.041637  0.126319
261  0.015089  0.057727  0.041268    0.044968  0.143963
262  0.001841  0.000725  0.002916    0.001992  0.005633
263  0.014931  0.055232  0.039223    0.043227  0.137683
264  0.013582  0.055177  0.042091    0.042014  0.139282

[265 rows x 12 columns]


# Estimate Mendelian correlation
start = time.time()
msvmsc_gcorr = msq.msvarcov_gcorr(msvmsc=msvmsc_g)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(msvmsc_gcorr)
Time taken:  0.05 secs
        ID Group    pH_fat  protein_fat  protein_pH    AG_fat     AG_pH  \
0    10001     F  0.168243     0.130355   -0.076619  0.502305  0.608696
1    10002     F  0.849938     0.923450    0.784271  0.978501  0.917349
2    10003     F  0.913412     0.908779    0.821836  0.984097  0.944224
3    10004     F  0.148098    -0.038853   -0.061912  0.462340  0.836841
4    10005     F  0.867249     0.912187    0.778874  0.980058  0.923970
```

```
..     ...   ...       ...          ...         ...       ...       ...
260  10261     F  0.868944     0.937647    0.775095  0.986844  0.912919
261  10262     F  0.922706     0.932494    0.873801  0.985100  0.957769
262  10263     F  0.075172     0.023946    0.056342  0.393020  0.748465
263  10264     F  0.897337     0.927839    0.809875  0.985611  0.936218
264  10265     F  0.918000     0.941739    0.871634  0.985839  0.956892

     AG_protein
0      0.684896
1      0.952081
2      0.949738
3      0.364236
4      0.944280
..          ...
260    0.955789
261    0.964839
262    0.618574
263    0.953390
264    0.965978

[265 rows x 8 columns]
# Derive similarity matrix based on Mendelian sampling values for aggregate geno
type
start = time.time()
sim_g = msq.simmat_g(info=data, covmat=covmatrices, sub_id=None,
                     chrinterest="none", save=False, stdsim=False,
                     progress=True)
print('Time taken: ', round(time.time() - start, 2), 'secs')
Progress: |████████████████████████████████████████████████| 100% Complete
Creating similarity matrix based on aggregate genotype
Progress: |████████████████████████████████████████████████| 100% Complete
Time taken:  6.26 secs
sim_g = np.array(sim_g[0][0])
sim_g.shape
(265, 265)
# Write data to file for visualization in R
msvmsc_g.to_csv("msvmsc_g.csv", header=True, index=False)
msvmsc_gcorr.to_csv("msvmsc_gcorr.csv", header=True, index=False)
ped.to_csv("pedigree.csv", header=True, index=False)
np.save("sim_g.npy", sim_g)
```

Additional file 2: An R script to generate Figs. 1 and 2.

import relevant packages

```
library(ggplot2)
library(reshape2)
library(corrplot)
require(RColorBrewer)
library(pheatmap)
library(ggpubr)
library(RcppCNPy)
```

Read saved data

```
msvmsc_g <- read.csv("msvmsc_g.csv", header = T)         # Mendelian covariance
msvmsc_gcorr <- read.csv("msvmsc_gcorr.csv", header = T) # Mendelian correlation
```

1. DENSITY PLOTS

Extract relevant columns from data frame

```
df_msv <- data.frame(msvmsc_g[, c(3, 5, 8)])             # MS Variance
colnames(df_msv) <- c("FY", "pH", "PY")                  # rename columns
head(df_msv)
##              FY          pH          PY
## 1 0.0004657436 0.001804908 0.002320987
## 2 0.0232617786 0.013224900 0.015098141
## 3 0.0216646755 0.010474963 0.013710154
## 4 0.0009494060 0.004835635 0.001351647
## 5 0.0218923149 0.012676235 0.013317761
## 6 0.0231965577 0.012756366 0.015687612

df_mscorr <- data.frame(msvmsc_gcorr[, 3:5])              # MS Correlation
colnames(df_mscorr) <- c("FY_pH", "FY_PY", "PY_pH")# rename columns
head(df_mscorr)
##       FY_pH       FY_PY       PY_pH
## 1 0.1682430  0.13035463 -0.07661897
## 2 0.8499381  0.92344974  0.78427147
## 3 0.9134121  0.90877876  0.82183644
## 4 0.1480978 -0.03885255 -0.06191213
## 5 0.8672485  0.91218654  0.77887446
## 6 0.8622727  0.90880696  0.77531606
```

Molt data frame for density plot

```
df_msv <- melt(df_msv)                                   # molted df for MSV
```

```
colnames(df_msv) <- c("Trait", "Variance")                # rename columns
df_mscorr <- melt(df_mscorr)                              # molted df for Corr
colnames(df_mscorr) <- c("Trait", "Correlation")          # rename columns
```

Create density plot for Mendelian sampling variance using ggplot

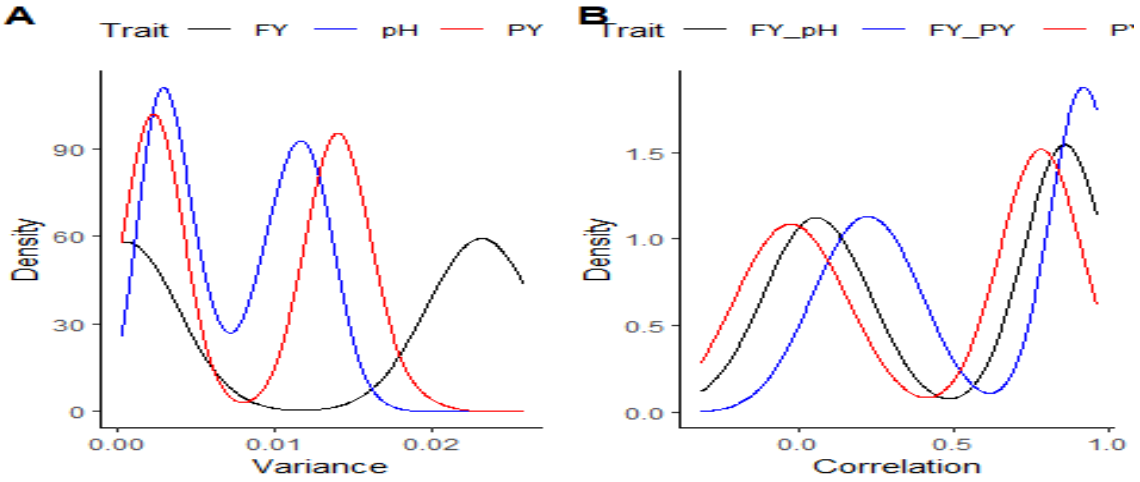
```
plot1 <- ggplot(df_msv, aes(Variance, color = Trait)) +
  stat_density(aes(x = Variance, colour = Trait),
               geom = "line",
               position = "identity") + theme_classic() +
  theme(legend.position = "top") + scale_color_manual(values = c("black", "blue"
, "red")) +
  ylab("Density")
```

Create density plot for Mendelian sampling correlation using ggplot

```
plot2 <- ggplot(df_mscorr, aes(Correlation, color = Trait)) +
  stat_density(aes(x = Correlation, colour = Trait),
               geom = "line",
               position = "identity") + theme_classic() +
  theme(legend.position = "top") + scale_color_manual(values = c("black", "blue"
, "red")) +
  ylab("Density")
```

Combine plots and save to file

```
# tiff("Density plot.tif", width = 8, height = 3, units = 'in', compression = 'l
zw', res = 700)
ggarrange(plot1, plot2, labels = c("A", "B"), ncol = 2, nrow = 1)
```




```
# dev.off()
```

2. SIMILARITY MATRICES

Read saved data

```
ped <- read.csv("pedigree.csv", header = F)              # Pedigree
mat <- npyLoad("sim_g.npy")                              # Similarity matrix
# standardize similarity matrix
matstd <- cov2cor(mat)                                   # Std similarity matrix
# rename rows and columns of the matrices
row.names(mat) <- colnames(mat) <- 1:nrow(mat)
row.names(matstd) <- colnames(matstd) <- 1:nrow(matstd)
```

Within-family clustering of similarities

```
ind <- no <- NULL
for (i in 1:5) {
  first <- which(ped[, 1] == i)[1]                           # 1st in family
  last <- which(ped[, 1] == i)[length(which(ped[, 1] == i))] # last in family
  no <- c(no, length(which(ped[, 1] == i)))                  # no. of inds
  m <- mat[first:last, first:last]                           # subset of matrix
  out <- pheatmap(m, show_rownames = T, cluster_cols = T,
    cluster_rows = T, cex = 1,
    clustering_distance_rows = "euclidean",
    clustering_distance_cols = "euclidean",
    clustering_method = "complete", border_color = F, silent = T)
  ind <- c(ind, (rownames(m[out$tree_row[["order"]], ])))
}
ind <- as.numeric(ind)
sim_clus <- mat[ind, ind]                                  # cluster family
```

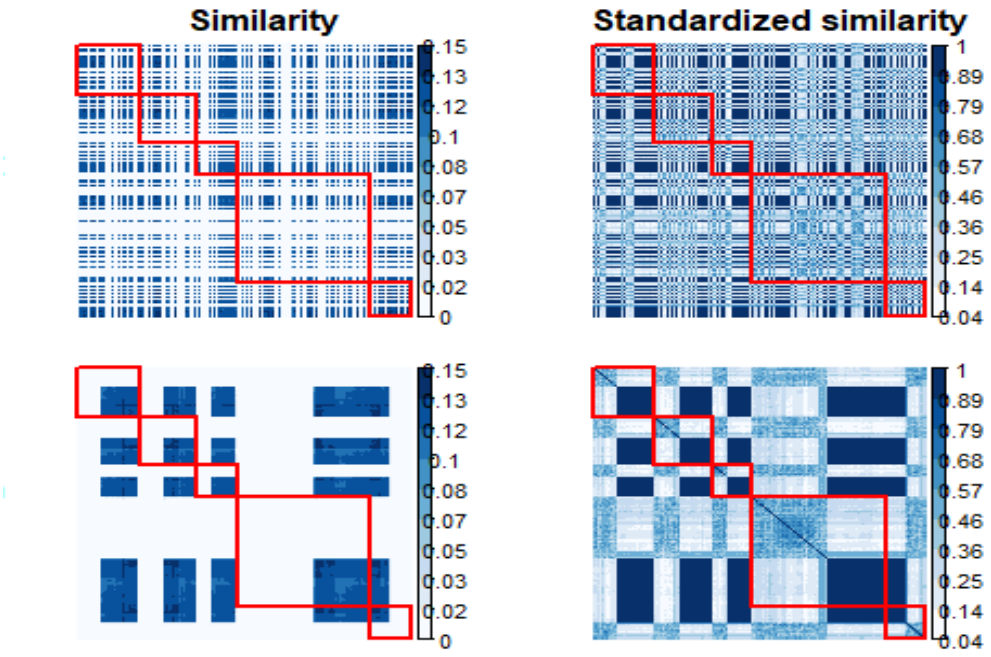
Within-family clustering of standardized similarities

```
ind <- no <- NULL
for (i in 1:5) { #for first two half-sib families only
  first <- which(ped[, 1] == i)[1]                           # 1st in family
  last <- which(ped[, 1] == i)[length(which(ped[, 1] == i))] # last ind family
  no <- c(no, length(which(ped[, 1] == i)))                  # no. of inds  m <-
matstd[first:last, first:last]                          # subset of matrix
  out <- pheatmap(m, show_rownames = T, cluster_cols = T,
    cluster_rows = T, cex = 1,
    clustering_distance_rows = "euclidean",
    clustering_distance_cols = "euclidean",
    clustering_method = "complete", border_color = F, silent = T)
  ind <- c(ind, rev(rownames(m[out$tree_row[["order"]], ]))) # reverse inds
}
ind <- as.numeric(ind)
stdsim_clus <- matstd[ind, ind]                            # cluster family
```

Plot similarity matrices

```
# tiff("sim mats.tif", width = 9, height = 8, units = 'in', compression = 'lzw',
res = 700)
par(mfrow = c(2, 2), oma = c(0, 0, 1, 0.1) + 0.1,
    mar = c(0, 0, 0, 0.5) + 0.1)
cols <- brewer.pal(9,"Blues")
corrplot(mat, is.corr = F, method = "color", cl.lim=range(mat),
         cl.cex = 0.80, tl.col = "black", tl.pos = "n", col = cols,
         cl.align.text = "c", mar = c(0, 0, 1, 0), title = "Similarity")
corrRect(no, col = "red") # draw diagonal boxes to demarcate families
corrplot(matstd, is.corr = F, method = "color", cl.lim=range(matstd),
         cl.cex = 0.80,tl.col = "black",tl.pos = "n", col = cols,
         cl.align.text = "c", mar = c(0, 0, 1, 0),
         title = "Standardized similarity")
corrRect(no, col = "red")
corrplot(sim_clus, is.corr = F, method = "color", cl.lim=range(sim_clus),
         cl.cex = 0.80,tl.col = "black",tl.pos = "n", col = cols,
         cl.align.text = "c", mar = c(0, 0, 1, 0))
corrRect(no, col = "red")
corrplot(stdsim_clus, is.corr = F, method = "color", cl.lim=range(stdsim_clus),
         cl.cex = 0.80,tl.col = "black",tl.pos = "n",
         col = cols, cl.align.text = "c", mar = c(0, 0, 1, 0))
corrRect(no, col = "red")

mtext('A', at=.75, side=2, outer=T, cex=1, las=0, line=0)
mtext('B', at=.25, side=2, outer=T, cex=1, las=0, line=0)
```




```
# dev.off()
```

Additional file 3: Benchmarking of similarity matrices.

Table 1: Time required to derive similarity matrices for different numbers of individuals with a total of 39780 markers on 29 chromosomes

No. of individuals	[1]Time (seconds)	Standard deviation (seconds)
100	4.1	0.6
1,000	12.9	1.0
5,000	86.3	1.4
10,000	255.2	3.2
20,000	912.3	17.3
40,000	3866.7	105.9
60,000	8494.3	408.6
80,000	20824.8	1959.4
100,000	30136.7	1196.8

[1]values represent an average of 5 repetitions

LITERATURE CITED

De Abreu Santos, D. J., J. B. Cole, G. E. Liu, P. M. Vanraden, and L. Ma. 2020. Gamevar.f90: A software package for calculating individual gametic diversity. *BMC Bioinformatics*. 21:3–7.

Bernardo, R. 2014. Genomewide selection of parental inbreds: Classes of loci and virtual biparental populations. *Crop Sci.* 54:2586–2595.

Bijma, P., Y. C. J. Wientjes, and M. P. L. Calus. 2020. Breeding Top Genotypes and Accelerating Response to Gametic Variance. *Genetics*. 214:91–107.

Bonk, S., M. Reichelt, F. Teuscher, D. Segelke, and N. Reinsch. 2016. Mendelian sampling covariability of marker effects and genetic values. *Genet. Sel. Evol.* 48:1–11.

Cole, J. B., and P. M. VanRaden. 2011. Use of haplotypes to estimate Mendelian sampling effects and selection limits. *J. Anim. Breed. Genet.* 128:446–455.

Daetwyler, H. D., M. J. Hayden, G. C. Spangenberg, and B. J. Hayes. 2015. Selection on optimal haploid value increases genetic gain and preserves more genetic diversity relative to genomic selection. *Genetics*. 200:1341–1348.

Estaghvirou, S. B. O., J. O. Ogutu, and H. P. Piepho. 2015. How genetic variance and number of genotypes and markers influence estimates of genomic prediction accuracy in plant breeding. *Crop Sci.* 55:1911–1924.

Falconer, D. S., and T. F. C. Mackay. 1996. Introduction to Quantitative Genetics. Longman, Essex, England.

Goiffon, M., A. Kusmec, L. Wang, G. Hu, and P. S. Schnable. 2017. Improving response in genomic selection with a population-based selection strategy: Optimal population value selection. *Genetics*. 206:1675–1682.

Haldane, J. B. S. 1919. The combination of linkage values and the calculation of distances between the loci of linked factors. *J. Genet.* 8:299–309.

Hampel, A., F. Teuscher, L. Gomez-Raya, M. Doschoris, and D. Wittenburg. 2018. Estimation of Recombination Rate and Maternal Linkage Disequilibrium in Half-Sibs. *Front. Genet.* 9:186.

Jannink, J. L. 2010. Dynamics of long-term genomic selection. *Genet. Sel. Evol.* 42.

Kolde, R. 2019. pheatmap: Pretty Heatmaps. R package version 1.0.12; 2019. https://CRAN.R-project.org/package=pheatmap.

Lehermeier, C., S. Teyssèdre, and C. C. Schön. 2017. Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. *Genetics*. 207:1651–1661.

Lin, Z., N. O. I. Cogan, L. W. Pembleton, G. C. Spangenberg, J. W. Forster, B. J. Hayes, and H. D. Daetwyler. 2016. Genetic Gain and Inbreeding from Genomic Selection in a Simulated Commercial Breeding Program for Perennial Ryegrass. *Plant Genome*. 9:1–12.

Liu, H., T. H. Meuwissen, A. C. Sørensen, and P. Berg. 2015. Upweighting rare favourable alleles increases long-term genetic gain in genomic selection programs. *Genet. Sel. Evol.* 47:19.

Markowitz, H. 1952. Portfolio selection. *J. Finance*. 7:77–91.

Melzer, N., D. Wittenburg, and D. Repsilber. 2013. Integrating Milk Metabolite Profile Information for the Prediction of Traditional Milk Traits Based on SNP Information for Holstein Cows. *PLoS One*. 8:e70256.

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard. 2001. Prediction of total genetic value using genome-wide dense marker maps. *Genetics*. 157:1819–1829.

Moeinizade, S., G. Hu, L. Wang, and P. S. Schnable. 2019. Optimizing selection and mating in genomic selection with a look-ahead approach: An operations research framework. *G3 Genes, Genomes, Genet.* 9:2123–2133.

Mohammadi, M., T. Tiede, and K. P. Smith. 2015. Popvar: A genome-wide procedure for predicting genetic variance and correlated response in biparental breeding populations. *Crop Sci.* 55:2068–

2077.

Müller, D., P. Schopp, and A. E. Melchinger. 2018. Selection on expected maximum haploid breeding values can increase genetic gain in recurrent genomic selection. *G3 Genes, Genomes, Genet.* 8:1173–1181.

Musa, A. A., and N. Reinsch. 2021. A similarity matrix for hedging haplotype diversity among parents in genomic selection. *Submitted*.

Santos, D. J. A., J. B. Cole, T. J. Lawlor, P. M. VanRaden, H. Tonhati, and L. Ma. 2019. Variance of gametic diversity and its application in selection programs. *J. Dairy Sci.* 102:5279–5294.

Schaeffer, L. R. 2006. Strategy for applying genome-wide selection in dairy cattle. *J. Anim. Breed. Genet.* 123:218–223.

Schnell, F. W., and H. F. Utz. 1976. F1-Leistung und Elternwahl in der Züchtung von Selbstbefruchtern. *Bericht über die Arbeitstagung der Vereinigung Österreichischer Pflanzenzüchter, pp. 234–258, Gumpenstein, Österreich*.

Segelke, D., F. Reinhardt, Z. Liu, and G. Thaller. 2014. Prediction of expected genetic variation within groups of offspring for innovative mating schemes. *Genet. Sel. Evol.* 46:42.

Thaller, G., W. Krämer, A. Winter, B. Kaupe, G. Erhardt, and R. Fries. 2003. Effects of DGAT1 variants on milk production traits in German cattle breeds. *J. Anim. Sci.* 81:1911–1918.

Wei, T., and S. Viliam. 2017. R package "corrplot": Visualization of a Correlation Matrix. R package version 0.84; 2017. https://github.com/taiyun/corrplot.

Wickham, H. 2016. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York.

Woolliams, J. A., P. Berg, B. S. Dagnachew, and T. H. E. Meuwissen. 2015. Genetic contributions and their optimization. *J. Anim. Breed. Genet.* 132:89–99.

CHAPTER 4

Similarities in Mendelian sampling values in a large commercial German Holstein cattle population

A. A. Musa[1], N. Frioni-Garcia[2], D. Segelke[2] and N. Reinsch[1]

[1]Research Institute for Farm Animal Biology (FBN), Institute of Genetics and Biometry, Wilhelm-Stahl-Allee 2, 18196 Dummerstorf, Germany.

[2]Vereinigte Informationssysteme Tierhaltung w.V., Heideweg 1, 27283 Verden, Germany.

Similarities in Mendelian sampling values in a large commercial German Holstein cattle population

ABSTRACT

A similarity matrix based on the Mendelian sampling values has several breeding applications. The off-diagonal elements of this matrix represent similarities between Mendelian sampling values of parents, whereas the diagonal elements represent a parent's similarity to itself, which equals the Mendelian sampling variance (MSV) of its gametes. Unanswered questions, however, remain about its summary statistics for a large population, the factors affecting its values, and the heritability of MSV. The objective of this study was to address these questions. Therefore, we computed similarity matrices and summary statistics for a commercial German Holstein population comprising 71,163 animals and four traits: longevity, milk urea content, fat, and protein yields. We used linear mixed model analyses to evaluate the effects of sex, birth year, and selection on the similarity matrix values and estimate the heritability of MSV. The results indicated a considerable variation in MSV across traits, with coefficients of variation ranging from 19.2 to 26.5%. Sex affected the MSV of all the traits, but the birth year and selection had variable effects. The standardized pairwise similarities between the animals varied greatly across traits, with the exhibition of low to perfect similarities (0.1 to 1). This similarity was heavily influenced by the animals' MSVs but only marginally influenced by year of birth, sex, and selection. As such, animals with similar heterozygosity of markers with large effects for a trait in coupling seemed to be more similar than those with small effects. We discovered that MSV has no heritability, implying that high MSVs in selected parents will not accumulate and increase overall trait heritability over successive generations. The novel information on MSV and the similarities between Mendelian sampling values will facilitate their efficient use in breeding programs for genetic gain and the management of genetic diversity.

Keywords: Mendelian sampling variance, haplotype diversity, gametic variance, heritability

INTRODUCTION

Genetic improvement in an outbred population, such as cattle, relies on the recurrent selection of parents based on the expected EBV of their offspring, defined as an average of its parental breeding values plus a Mendelian sampling term (Kennedy et al., 1988). With the availability of genomic data, we can predict the genomic EBV (GEBV) and Mendelian sampling variance (MSV) on parental gametes. GEBV is derived as the sum of all estimated marker effects of a parent (Meuwissen et al., 2001). MSV can be estimated numerically (Segelke et al., 2014; Mohammadi et al., 2015) or analytically (Bonk et al., 2016; Santos et al., 2019), using marker effects, genetic map, and phased genotypes. The selection of parents solely based on GEBV has become popular due to its success in delivering high genetic gain in the short term. Even though its continued use promotes inbreeding (Schaeffer, 2006; Lin et al., 2016), loss of haplotype diversity, and a decrease in genetic gain (Estaghvirou et al., 2015), the vast majority of breeding programs base their selection solely on GEBV without considering its variability due to Mendelian sampling.

MSV, determined by the linkage phase and heterozygosity of the parental haplotypes, quantifies the deviation from the expected and realized GEBV of offspring. In other words, it quantifies the chances of parents breeding top-ranking offspring. Consequently, it influences mate selection and allocation when low or high MSV is desired, depending on the breeder's goals. While plant breeders have long recognized the importance of considering MSV when making selection and mating decisions (Schnell and Utz, 1976; Lehermeier et al., 2017), animal breeders have only recently begun to do so (Santos et al., 2019; Bijma et al., 2020; Musa and Reinsch, 2021a). Recently proposed selection indices for genetic improvement of an outbred population combine GEBV with Mendelian standard deviation (square root of MSV), weighted by a constant (Santos et al., 2019; Bijma et al., 2020). These indices resulted in a greater genetic gain in simulated diploid animals than conventional genomic selection based solely on GEBV. However, under a high selection intensity, these indices are likely to select similar parents with high MSV potential, resulting in the unintended loss of favorable haplotypes (Musa and Reinsch, 2021a).

As a solution, Musa and Reinsch (2021a) proposed a similarity matrix to hedge haplotype losses. This matrix can be used in a portfolio optimization approach to determine the optimal set of parents that balances parental haplotype diversity and expected genetic gain in the next generation (Musa and Reinsch, 2021a). The off-diagonal elements of this matrix represent pairwise haplotype similarities,

whereas the diagonal elements represent a parent's similarity to itself, which equals the MSV of its gametes. High similarity between parents indicates that they share many heterozygous markers with large effects on a trait in the same linkage phase.

The similarity matrix has various potential applications in breeding, most notably in optimal mating decision-making and managing haplotype diversity. However, no previous study has computed such a matrix for a large sample of animals. The impact of important factors such as sex, selection, and birth year on the matrix's values is also unknown. While numerous studies have established the heritability of heterozygosity (Mitton et al., 1993; Nietlisbach et al., 2016), little is known about the inheritance of the linkage phase and, more importantly, the inheritance of MSV. Therefore, the objectives of this study were to (1) derive similarity matrices and compute summary statistics for several example traits, such as longevity and milk urea content, in a large sample of German Holstein cattle; (2) determine the associations between these factors and MSV; (3) determine the heritability of MSV for the example traits; and (4) determine the associations between these factors and similarities between Mendelian sampling values of animals. This study will help better understand MSV and the similarities between Mendelian sampling values and, thus, their use for genetic gain and managing genetic diversity in breeding programs, which is critical, particularly in light of climate change and rapid population growth.

MATERIAL AND METHODS

Data

The data used for this study were provided by the German Evaluation Center, VIT (https://www.vit.de), and were a subset of animals from the German genomic selection program. The data comprised 71,163 top-ranking German Holstein cattle, including bulls and cows born between 2014 and 2020. (The distribution of animals by sex and birth year is provided in Supplemental Table S1.) Pedigree information on all the animals was provided, along with sex, birth years, and whether or not animals were eventually selected for breeding. The genotype data had been generated from 50K SNP chip (Illumina Inc., San Diego, CA, USA), of which 44,746 autosomal SNPs with minor allele frequencies of greater than 1% and call rates greater than 98% were retained. The missing marker

genotypes were imputed and phased using *Fimpute* version 2.2 (Sargolzaei et al., 2014). The order of SNPs on chromosomes was determined using the UMD 3.1 assembly (available at https://bovinegenome.elsiklab.missouri.edu/downloads/UMD_3.1). The BLUP model was used to estimate the marker effects for longevity (LG, days), milk urea content (MUC, mg/kg), fat (FY, kg), and protein (PY, kg) yields, as detailed in a previous study (Liu et al., 2011). The genetic map positions (sex average) were linearly approximated by converting the physical positions (bp) to Mbp.

Statistical Analyses

Derivation of similarity matrices

We derived similarities between Mendelian sampling values among animals (n = 71,163) using *PyMSQ* (Musa and Reinsch, 2021b). Briefly, the similarities s between the Mendelian sampling values of gametes produced by parents i and j were derived as:

$$s_{ij} = \sum_c \left| \left(\boldsymbol{m}_i^c \right)' \boldsymbol{R}^c \boldsymbol{m}_j^c \right|,$$

where $\boldsymbol{m}_i^c \left(\boldsymbol{m}_j^c \right)$ is a vector of parent-specific additive marker effects for the parent $i\left(j\right)$ on the chromosome c and $\boldsymbol{R}^c$ is the population covariance matrix reflecting expected within-family linkage disequilibrium of markers on the same chromosome. The element of parent-specific additive marker effects vector (e.g., $\boldsymbol{m}_i^c$) is given by $m_{ik}^c = \delta_{ik} m_k^c$, where δ_{ik} and m_k^c represent the phase indicator and marker effect at a locus k, respectively. The phase indicator is $\delta_{ik} = 1$ if the standard allele is observed on the first haplotype; $\delta_{ik} = -1$ if the standard allele is observed on the second haplotype; or $\delta_{ik} = 0$ if alleles are homozygous. Each element ρ_{kl}^c of $\boldsymbol{R}^c$ is derived as $\rho_{kl}^c = \exp\left(-2d_{kl}^c\right)/4$ (Haldane, 1919), where d_{kl}^c is the distance (Morgan) between markers k and l on the chromosome c. The similarity between potential parents i and j concerning aggregate genotype (AG) was calculated as follows:

$$s_{ij} = \sum_c \left| \boldsymbol{a}' \cdot \left(\boldsymbol{M}_i^c \boldsymbol{R}^c \left(\boldsymbol{M}_j^c \right)' \right) \cdot \boldsymbol{a} \right|.$$

Here, each row of matrices $\boldsymbol{M}_i^c$ and $\boldsymbol{M}_j^c$ are vectors $\left(\boldsymbol{m}_{it}^c\right)'$ and $\left(\boldsymbol{m}_{jt}^c\right)'$ of additive marker effects on the chromosome c for each trait t of parents i and j, respectively. $\boldsymbol{a}$ is a vector of index weights, which was assumed to be equal (1) for all the studied traits.

Then, matrix $\boldsymbol{S}$ from these similarities was derived, with off-diagonal elements representing pairwise haplotype similarities and diagonal elements representing an animal's similarity to itself, which equals the MSV of its gametes. The similarity matrix was standardized by pre and post-multiplying $\boldsymbol{S}$ by a diagonal matrix $\boldsymbol{D}^{-1}$, where $\boldsymbol{D} = \sqrt{diag\left(\boldsymbol{S}\right)}$ (Musa and Reinsch, 2021a) as follows:

$$\boldsymbol{K} = \boldsymbol{D}^{-1} \cdot \boldsymbol{S} \cdot \boldsymbol{D}^{-1}.$$

To illustrate the effect of trait genetic architecture, heterozygosity, and linkage on the similarity matrix, we used *corrplot* version 0.84 (Wei and Viliam, 2017) to visualize a subset of the data corresponding to 5 full-sib families.

Factors affecting Mendelian sampling variance and heritability

We used a linear mixed model fitted with REML to assess the association between the various factors related to the animals and MSV. We assumed a Gaussian distribution. Therefore, we log-transformed MSVs to meet normality assumptions (Supplemental Figure S1). The linear mixed model analysis was implemented in ASReml 4.2 (Gilmour et al., 2021) using the following model:

$$\boldsymbol{y} = \boldsymbol{Xb} + \boldsymbol{Zs} + \boldsymbol{e}.$$

Here, $\boldsymbol{y}$ = vector of log-transformed MSVs of each animal; $\boldsymbol{b}$ = vector of fixed effects of sex (comprised of two groups: male and female), birth year (comprised of 7 groups: 2014–2020), selection (comprised of two groups: selected and unselected for breeding), and the interaction between sex and birth year. $\boldsymbol{s}$ = vector of random sire effects, and $\boldsymbol{e}$ = vector of a random residual effect specific to each animal. $\boldsymbol{X}$ and $\boldsymbol{Z}$ are the design matrices for $\boldsymbol{b}$ and $\boldsymbol{s}$, respectively. It was assumed that the random effects followed normal distributions of $\boldsymbol{s} \sim N\left(\boldsymbol{0}, \boldsymbol{A}\sigma_s^2\right)$ and $\boldsymbol{e} \sim N\left(\boldsymbol{0}, \boldsymbol{I}\sigma_e^2\right)$, where σ_s^2 represent additive sire and σ_e^2 are residual variances. $\boldsymbol{A}$ represent the numerator relationship matrix for sires, and $\boldsymbol{I}$ is an identity matrix. We checked the distribution of model residuals to confirm the goodness of the fit and plotted residuals to ensure homoscedasticity before utilizing the model's

results. Heritability was then estimated using variance components for sire and residual effects $h^2 = \frac{\sigma_s^2 \times 4}{\sigma_s^2 + \sigma_e^2}$.

Factors affecting similarities between Mendelian sampling values

To determine the factors affecting similarities between animals, we used a subset of 7,611 animals belonging to 11 half-sib families ranging from 502 to 1,193 animals per family. This subset resulted in a 29 million-row data frame. Utilizing all 71,163 animals would require a data frame with ~2.5 billion rows, exceeding the ~2 billion row limit of many software packages (for example, R) or resulting in memory constraints. Nonetheless, the data subset included a sufficient representation for both sexes born between 2016 and 2020 (Supplemental Table S2).

The association between factors and similarities between animals was also tested using the linear mixed model analysis implemented with the same ASReml software:

$$\boldsymbol{y} = \boldsymbol{Xb} + (\boldsymbol{Z}_1 + \boldsymbol{Z}_2)\boldsymbol{a} + \boldsymbol{e}\ .$$

Here, $\boldsymbol{y}$ = vector of log-transformed similarities between animals; $\boldsymbol{b}$ = vector of fixed effects of time lag (categorized into five groups: 0–4); sex (categorized into three groups: similarities between females [FF], males [MM], and both sexes [MF]), selection (categorized into three groups: similarities between animals eventually selected for breeding [SS], unselected [UU], and both [SU]), log-transformed MSV of animal 1, log-transformed MSV of animal 2, and the log-transformed sum of both animals' MSVs, and the interaction between time lag and sex. $\boldsymbol{a}$ = vector of random animal effects; $\boldsymbol{e}$ = vector of a random residual effect specific to each animal; $\boldsymbol{X}$ = incidence matrix for $\boldsymbol{b}$, and matrix $\boldsymbol{Z}_1 + \boldsymbol{Z}_2$ = design matrix for $\boldsymbol{a}$, overlaid by the design matrices for animal 1 and 2. We assumed that the random effects followed normal distributions of $\boldsymbol{a} \sim N\left(\boldsymbol{0}, \boldsymbol{I}\sigma_{\boldsymbol{a}}^2\right)$ and $\boldsymbol{e} \sim N\left(\boldsymbol{0}, \boldsymbol{I}\sigma_e^2\right)$, where $\sigma_{\boldsymbol{a}}^2$ is the animal variance. σ_e^2 and $\boldsymbol{I}$ have been defined earlier. The time lag between the two animals was calculated as the absolute difference in their birth years. Animals in this data subset were born between 2016 and 2020, resulting in a time lag between 0 and 4. It should be noted that animals 1 and 2 refer to two animals from which similarities were derived.

RESULTS AND DISCUSSION

Similarity matrices for various traits

Figure 1 depicts the similarity matrices for 48 animals from 5 full-sib families (top panel). These animals were chosen from a population of 71,163 animals for demonstration purposes. The diagonal elements of these matrices represent the MSV of each animal, and the off-diagonal elements represent the similarities between Mendelian sampling values of the animals. These values are determined by linkage, heterozygosity, and trait genetic architecture (Musa and Reinsch, 2021a). The matrices, therefore, mirror the Mendelian segregation patterns of animals. As such, animals with similar heterozygosity of markers with large effects for a trait in coupling seemed to be more similar than those with small effects, regardless of family (Figure 1).

As full-sib families demonstrate in Figure 1, some families produced more genetically diverse offspring than others. Using FY as an example, Family 4 produced more variable offspring than Family 2 due to differences in parental heterozygosity and linkage phase (Figure 1, top panel). As a result, the offspring of Family 4 were more similar than Family 2. This result can also be seen in the similarity matrix for chromosome 14 (Supplemental Figure S2). Similar results were obtained in a previous study with a small sample size, which reported that some paternal half-sib families produced more variable offspring than others (Musa and Reinsch, 2021a; Figure 1).

The inclusion of MSV and similarities between individuals in selection and mating decisions directly impacts genetic gain and haplotype diversity. Their manner of application, on the other hand, is determined by the breeder's objectives. For instance, breeding companies prefer breeding bulls with a high MSV to increase the chances of breeding offspring with exceptional GEBV. They are interested in full-sib families of parents with high MSVs, which are frequently the result of multiple ovulation and embryo transfer, to increase the chances of breeding offspring with exceptional breeding values. Farmers selecting bulls for ordinary cows, on the other hand, are more interested in half-sib families of elite bulls with (preferably) low MSVs to promote herd uniformity for managerial and economic reasons.

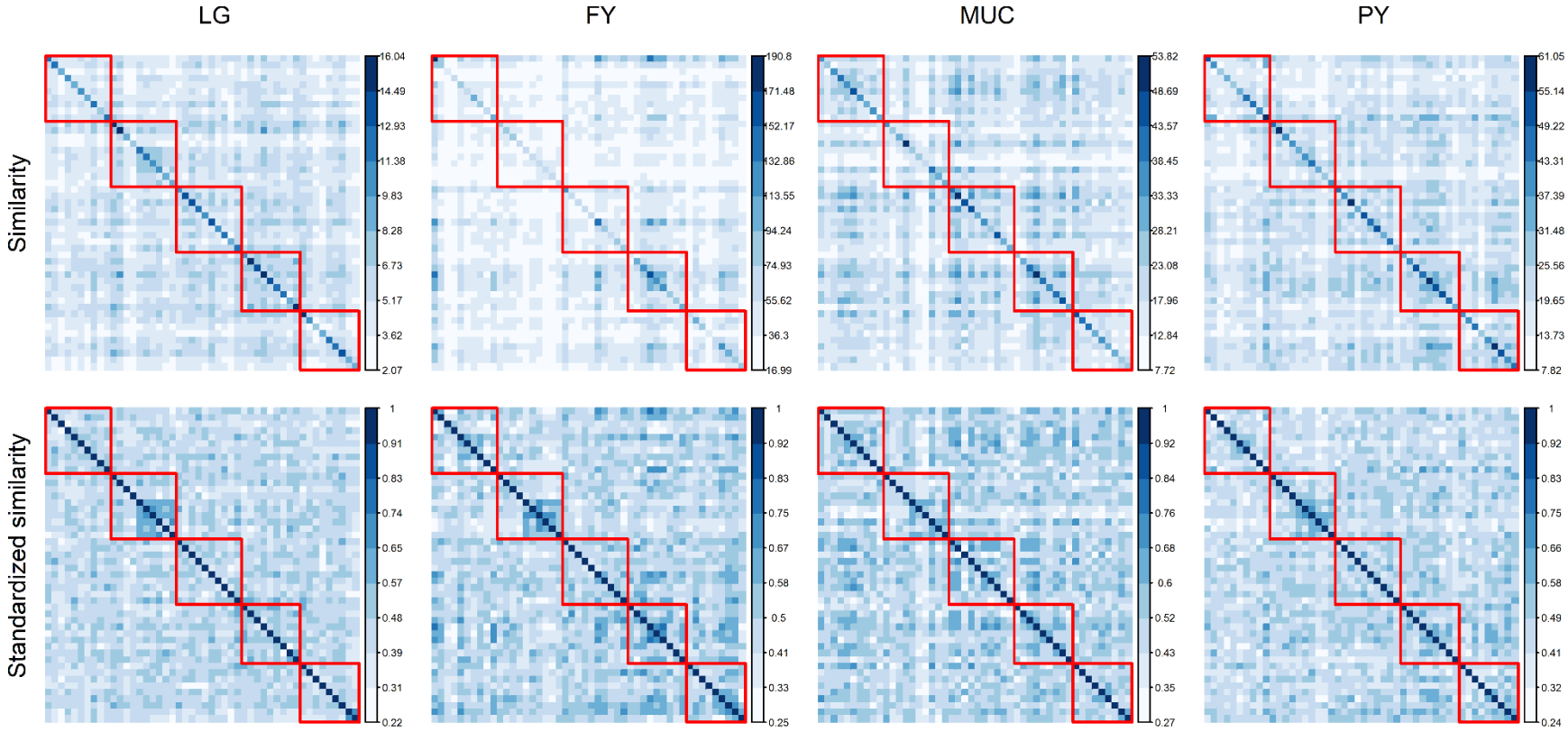


Figure 1. Similarity and standardized similarity matrices of various traits for 48 animals from five full-sib families (separated by red lines). *LG* longevity (days), *FY* fat yield (kg), *MUC* milk urea content (mg/kg), and *PY* protein yield (kg). The Mendelian sampling variance of each individual and haplotype similarities between animals are located on the diagonal and off-diagonals, respectively (top panel). The standardized similarity matrices (bottom panel) make small pairwise haplotype similarities more visible.

Besides the apparent effect of various traits on the structure of the matrices (Figure 1), a comparison of the matrices from different chromosomes further showed the effect of genetic architecture (Supplemental Figures S2 and S3). Using FY as an example, the similarity matrix values on chromosome 14 ranged from 0 to 94.4, whereas they were 0 to 8.2 on chromosome 4 (Supplemental Figures S2). Without considering the impact of linkage and heterozygosity, such a result is expected if a chromosome (here: 14) has either many markers with small effects or few markers with more substantial effects on FY than the other chromosome (here: 4).

The standardized similarity matrices (Figure 1 bottom panel and Supplemental Figure S3) displayed a similar pattern to the similarity matrix (Figure 1 top panel and Supplemental Figure S2), but they more effectively highlight small similarities.

Summary statistics of similarity matrices

Table 1 summarizes the statistics of genome-wide similarity matrices for all animals. MSVs ranged from 5.3 (LG) to 215.1 (FY) across traits, with coefficients of variation (CV) ranging from 19.2 to 26.5%. The MSV for AG for these traits ranged between 89.6 to 462.7, with a CV of 18.8%. These values are comparable to the CV reported for German Holsteins, ranging from 19.0 (PY) to 30.3% (FY) (Bonk et al., 2016). The standardized pairwise similarities between Mendelian sampling values of animals varied greatly across all traits, with animals exhibiting low to perfect similarities (0.1 to 1). Except for same-sex and selection groups, further examinations of animals with perfect similarity revealed no discernible trend in pedigree, birth year, or heterozygosity.

Interestingly, MSVs and similarities, however, differed greatly between chromosomes (Supplemental Table S3). Retaking FY as an example, MSV varied between 0.08 to 25.2 (CV 78%) on chromosome 4, whereas it varied from 0 to 122.4 (CV 112.5%) on chromosome 14. Without considering the role of linkage and heterozygosity, such a result can be attributed to the impact of genetic architecture, indicating that chromosome 14 has a stronger impact on the trait than chromosome 4. However, this result was not surprising because chromosome 14 has the largest variance for milk production (Bouwman et al., 2018; Santos et al., 2019). Additionally, the mean standardized similarities between animals on chromosomes (e.g., 0.66 on chromosome 14) that have a large effect on a trait (e.g., FY) are greater than those on chromosomes with a minor effect (e.g., 0.39 on chromosome 4).

Table 1. Summary statistics of trait-specific similarity matrices (upper triangular)

Traits	LG	FY	MUC	PY	AG
Diagonal elements (Mendelian sampling variances)[1]					
Mean	11.91	85.95	34.78	44.49	231.75
SD	2.28	22.78	7.60	8.93	43.59
Minimum	5.31	30.87	13.84	17.78	89.64
Median	11.70	81.75	33.94	43.61	228.10
Maximum	30.41	215.09	85.90	95.53	462.74
CV (%)	19.17	26.50	21.85	20.07	18.81
Off-diagonal elements (similarity between Mendelian sampling values)[2]					
Mean	5.32	41.63	17.06	20.34	106.13
SD	1.18	13.31	4.39	4.82	23.44
Minimum	1.14	6.98	2.95	3.83	22.50
Median	5.21	39.17	16.50	19.80	104.07
Maximum	21.36	169.23	64.73	70.48	331.28
CV (%)	22.20	31.97	25.75	23.69	22.09
Off-diagonal elements (standardized similarity between Mendelian sampling values)[2]					
Mean	0.449	0.486	0.492	0.459	0.459
SD	0.064	0.087	0.076	0.068	0.064
Minimum	0.118	0.102	0.113	0.120	0.132
Median	0.448	0.483	0.491	0.458	0.459
Maximum	1	1	1	1	1
CV (%)	14.33	17.86	15.40	14.75	13.98

[1] Number of observations = 71,163
[2] Number of observations = 2,532,050,703
LG = longevity (days); FY = fat yield (kg); MUC = milk urea content (mg/kg); PY = protein yield (kg); AG = aggregate genotype; SD = standard deviation; CV = coefficient of variation

The distributions of genome-wide MSVs for all traits in German Holsteins were positively skewed and resembled gamma distributions (Supplemental Figure S4, all chromosomes). This indicates that the MSVs of the majority of animals are less than the mean MSV and that only a few animals have exceptionally high MSVs. A similar distribution for PY was reported in the US Holstein and Jersey cattle population (Santos et al., 2019). However, the distribution observed in this study for FY is different from the bimodal distribution observed for the same trait in German Holsteins (Segelke et al., 2014; Musa and Reinsch, 2021b) and US Holsteins and Jerseys (Santos et al., 2019). The

bimodality has been attributed to the strong impact of the *DGAT1* gene (chromosome 14) on milk production traits (Thaller et al., 2003; Segelke et al., 2014). MSV distributions differed across chromosomes for each trait (Supplemental Figure S3, chromosomes 4 and 14). Except for chromosome 14, where the FY and PY distributions resembled bimodal lognormal distributions, the distributions in this study appeared to be lognormal.

The variability in MSV is unknown for many traits, populations, and species. However, the findings of this study suggest that it may be greater in species with smaller genomes and traits controlled by fewer genes. Therefore, the distribution of MSV is affected by several factors, including trait genetic architecture, animals' heterozygosity and linkage phase, species and population type, and sample size. Furthermore, the method employed to estimate marker (or allele substitution) effects may bias the estimation of MSVs (Lehermeier et al., 2017) and, consequently, their distribution.

Factors affecting Mendelian sampling variance

Table 2 shows the linear mixed model results for MSV of various traits. We found that sex had a significant ($P < 0.001$) effect on all traits, with males having 0.1–4.5% lower MSVs than females across all traits (Table 2). Except for MUC, MSVs for all traits were unaffected ($P > 0.05$) by birth year. This result could be due to the pooling of both sexes. Further examination of the birth year effect on male MSVs revealed that it was significant for all traits but was not significant for females (results not shown). FY and MUC were significantly ($P < 0.05$) impacted by whether or not the animal was selected for breeding, with unselected animals having ~ 2% greater MSVs than selected animals (Table 2). Selection, however, did not affect LG and PY. The interaction between sex and birth year interaction was significant ($P < 0.05$) for LG only.

The observed lower MSVs in males than females could be attributed to the difference in selection intensity. Like most breeding programs, parents were chosen based on their GEBVs, which were calculated as the sum of estimated marker effects. Parents with homozygous dominant alleles for markers with large effects on a trait are more likely to be selected than parents with heterozygous alleles. Also, selection based on GEBV is known to conceal the contributions of small-effect QTL, resulting in their extinction in the population (Schaeffer, 2006; Jannink, 2010; Estaghvirou et al., 2015). As a result, the higher selection intensity in males than females promotes marker homozygosity, resulting in a decrease in MSV.

Table 2. Linear mixed model results (coefficient ± SE) for Mendelian sampling variance of various traits[1,2]

Variable and trait	LG	FY	MUC	PY	AG
Intercept	2.475±0.002***	4.439±0.002***	3.533±0.002***	3.791±0.002***	5.447±0.001***
Sex	***	***	***	***	***
Female (F)	Referent				
Male (M)	-0.043±0.007	-0.046±0.009	-0.001±0.008	-0.043±0.007	-0.058±0.007
Birth year (BY)	NS	NS	*	NS	NS
2019	Referent				
2018	-0.001±0.002	-0.001±0.003	0.002±0.003	-0.003±0.002	-0.001±0.002
2020	-0.002±0.002	-0.001±0.003	-0.007±0.003	-0.004±0.003	-0.004±0.002
2017	0.005±0.003	0.002±0.004	0±0.003	-0.002±0.003	-0.005±0.003
2016	0.006±0.003	0.004±0.005	-0.004±0.004	0.001±0.004	0.001±0.003
2015	-0.004±0.005	0.009±0.007	0.002±0.006	-0.004±0.005	-0.006±0.005
2014	-0.009±0.008	0±0.011	-0.008±0.009	-0.008±0.009	-0.003±0.008
Selection	NS	**	**	NS	NS
Unselected	Referent				
Selected	-0.01±0.006	-0.021±0.008	-0.02±0.007	-0.008±0.007	-0.003±0.006
Sex × BY	**	NS	NS	NS	NS
M × BY 2018	0.006±0.005	0.01±0.006	-0.002±0.005	0.004±0.005	0.005±0.005
M × BY 2020	-0.004±0.005	0.003±0.007	0.005±0.006	0.007±0.006	0.001±0.005
M × BY 2017	-0.003±0.005	0±0.007	0.006±0.006	0.004±0.005	0.006±0.005
M × BY 2016	-0.016±0.006	0.01±0.007	0.002±0.006	-0.004±0.006	-0.001±0.005
M × BY 2015	0.008±0.007	-0.002±0.009	-0.017±0.008	0.009±0.007	0.006±0.007
M × BY 2014	0.02±0.01	0.016±0.014	0.015±0.012	0.018±0.011	0.012±0.01

[1] Mendelian sampling variances of all traits were log-transformed (base 10) to meet linearity assumptions

[2] Number of observations 71,163

LG = longevity; FY = fat yield; MUC = milk urea content; PY = protein yield; AG = aggregate genotype

P-value: * < 0.01; ** < 0.01; *** < 0.001, NS = *P* > 0.05

Selection affected the MSVs for FY and MUC. The selection effect on MUC was intriguing and surprising because the trait had not previously been subjected to direct selection in this population. Recently, interest in MUC has increased as the dairy industry faces increased pressure to reduce

nitrogen emissions (Honerlagen et al., 2021; Jahnel et al., 2021; Müller et al., 2021). However, the potentials for MUC to be used as a correlated trait for improving fertility, health, and milk yield have been investigated (Mitchell et al., 2005; Mucha and Strandberg, 2011; Rzewuska and Strabel, 2013). Weak but positive genetic correlations have been reported between MUC and fertility (Mucha and Strandberg, 2011) as well as milk yield (Rzewuska and Strabel, 2013). Therefore, this result suggests that MUC could be correlated to traits under selection.

Sire variance and heritability of Mendelian sampling variance

We discovered that the heritability estimates for MSV were all zero (Table 3), implying that the MSV is not heritable. This result was surprising as the hypothesis was that the sire effect would be substantial. This is because several studies have demonstrated that heterozygosity is heritable, and its heritability can be pretty high in the presence of unequal allele frequencies and little inbreeding (Mitton et al., 1993; Falconer and Mackay, 1996; Nietlisbach et al., 2016). Aside from the fact that closely linked loci are likely to be passed down to offspring, little was known about the inheritance of linkage. Zero heritability emphasizes that the combination of heterozygosity and linkage is a complex process that cannot be passed down to offspring. As a result, high MSVs in selected parents will not accumulate and increase overall trait heritability over successive generations.

Table 3. Variance components, heritabilities, and their standard errors for Mendelian sampling variance of various traits[1,2]

Trait	LG	FY	MUC	PY	AG
[3]Sire variance	0.00003 ± 0.00002	0.00003 ± 0.00003	0.00004 ± 0.00002	0.000003 ± 0.000013	0.00001 ± 0.00001
Residual variance	0.03537 ± 0.00019	0.06374 ± 0.00034	0.04634 ± 0.00025	0.039214 ± 0.000208	0.03443 ± 0.00018
Heritability	0.0027 ± 0.002	0.0015 ± 0.0018	0.0033 ± 0.0019	0.0003 ± 0.0013	0.0013 ± 0.0016

[1] Mendelian sampling variances were log-transformed (base 10) to meet linearity assumptions
[2] Number of observations 71,163
[3] Number of sires 2,782
LG = longevity (days); FY = fat yield (kg); MUC = milk urea content (mg/kg); PY = protein yield (kg); AG = aggregate genotype

Factors affecting the similarities between Mendelian sampling values

The linear mixed model results revealed that the time lag, sex, and their interaction had a significant ($P < 0.05$) effect on animal similarities, albeit in a minor way (Table 4). The MSVs of the reference animals and the sum of their MSVs had a more significant impact on similarities. This result was anticipated, as animals with similar MSVs appeared to be more similar to one another than those with dissimilar MSVs (Figure 1). Consequently, the variation in the animals explained between 7.2–11% of the variation in similarity estimates (Table 5). The similarity and standardized similarity matrices of the animals selected based on their random effects indicate that some animals shared more similarities with other animals in the population on average (Supplemental Figure S5), with animals with high random effects being more similar to those with low random effects.

Table 4. Linear mixed model results for similarities between Mendelian sampling values of animals for various traits[1,2]

Variable and trait	LG	FY	MUC	PY	AG
Fixed effects (coefficient ± SE.)					
Intercept	-1.108±0.0021***	-1.24±0.002***	-1.018±0.002***	-1.45±0.0022***	-2.104±0.0026***
Time lag (TL)	**	***	***	**	***
0	Referent				
1	-0.0001±0.0001	0±0.0001	-0.0001±0.0001	-0.0002±0.0001	0±0.0001
2	-0.0002±0.0001	-0.0001±0.0002	-0.0003±0.0001	0±0.0001	0±0.0001
3	-0.0009±0.0002	0.0009±0.0003	0±0.0002	-0.0003±0.0002	0.0004±0.0002
4	-0.001±0.0004	0.0019±0.0005	-0.0001±0.0004	0.0001±0.0004	0.0012±0.0004
Sex	***	***	***	***	***
FF	Referent				
MF	-0.0187±0.0007	0.0092±0.0008	-0.0189±0.0007	0.0149±0.0007	0.0163±0.0007
MM	-0.0348±0.0014	0.0211±0.0016	-0.0355±0.0014	0.0319±0.0014	0.0337±0.0013
Selection	***	***	***	***	***
UU	Referent				
SU	0.0132±0.0007	0.0127±0.0008	0.0179±0.0007	-0.016±0.0007	-0.0101±0.0007
SS	0.0262±0.0014	0.0233±0.0016	0.0353±0.0014	-0.0321±0.0014	-0.0199±0.0013
Mendelian sampling variance of corresponding animals					
Individual 1	0.9145±0.0012***	1.662±0.0008***	1.287±0.001***	1.035±0.0011***	0.9363±0.0012***
Individual 2	0.9139±0.0012***	1.665±0.0008***	1.288±0.001***	1.032±0.0011***	0.9354±0.0012***
Individual 1 + Individual 2	-0.5489±0.0024***	-1.911±0.0016***	-1.243±0.0019***	-0.752±0.0022***	-0.5579±0.0024***
Time lag × sex	***	**	NS	***	NS
TL1 × MF	0.0003±0.0001	-0.0001±0.0001	0.0001±0.0001	0.0003±0.0001	-0.0002±0.0001
TL1 × MM	0.0004±0.0002	0.0001±0.0002	-0.0001±0.0002	0.0002±0.0002	-0.0002±0.0002
TL2 × MF	0.0004±0.0002	0±0.0002	0.0002±0.0002	0±0.0002	-0.0001±0.0002
TL2 × MM	0.0005±0.0002	-0.0001±0.0003	-0.0003±0.0002	-0.0004±0.0002	-0.0003±0.0002
TL3 × MF	0.0011±0.0003	-0.0011±0.0003	0.0001±0.0003	0.0003±0.0003	-0.0004±0.0003
TL3 × MM	0.0017±0.0003	-0.0013±0.0004	-0.0006±0.0004	-0.0012±0.0004	-0.0001±0.0003
TL4 × MF	0.0015±0.0005	-0.0017±0.0006	0.0003±0.0005	-0.0004±0.0005	-0.0007±0.0005
TL4 × MM	0.0028±0.0009	-0.0022±0.001	-0.0011±0.001	-0.0011±0.0009	-0.0015±0.0009

[1] All numerical variables were log-transformed (base 10) to meet linearity assumptions

[2] Number of observations 28,959,855

LG = longevity; FY = fat yield; MUC = milk urea content; PY = protein yield; AG = aggregate genotype

FF, MM, MF = similarities between females (FF), males (MM), and both sexes (MF)

SS, UU, SU = Similarities between selected (SS), unselected (UU), and both (SU) animals

P-value: ** < 0.01; *** < 0.001, NS = $P > 0.05$.

Table 5. Variance components and proportion of variance explained by animals, and their standard errors for similarities between Mendelian sampling values[1,2]

Trait	LG	FY	MUC	PY	AG
[3]Animal variance	0.0014±0	0.0026±0	0.0019±0	0.0015±0	0.0013±0
Residual variance	0.0174±0	0.0221±0	0.0184±0	0.018±0	0.0162±0
Variance explained by animals	0.0724±0.0011	0.107±0.0016	0.0913±0.0013	0.0748±0.0011	0.0763±0.0011

[1] All numerical variables in the model were log-transformed (base 10) to meet linearity assumptions
[2] Number of observations 28,959,855
[3] Number of animals 7,611
LG = longevity; FY = fat yield; MUC = milk urea content; PY = protein yield; AG = aggregate genotype

CONCLUSIONS

This study contributes to a better understanding of Mendelian sampling values, which are useful in breeding programs. Mendelian sampling variance in German Holsteins is highly variable and is affected by sex, selection, and birth year, according to the findings of this study. The similarities between Mendelian sampling values of animals varied considerably, from almost no similarity to perfect standardized similarity. The similarity between animals was heavily influenced by Mendelian sampling variances but only marginally influenced by sex, selection, and birth year. As a result, animals with similar heterozygosity for markers that have a large effect on a trait in coupling were more similar than those with a small effect. Additionally, our findings indicated that Mendelian sampling variance is not heritable; thus, high MSVs in selected parents will not accumulate and increase overall trait heritability over successive generations. The novel information presented here will help improve the efficiency with which Mendelian sampling values are used, which is critical for increasing and managing genetic diversity in breeding programs.

ACKNOWLEDGEMENTS

The financial support provided by the Bundesanstalt für Landwirtschaft und Ernährung (BLE) for Musa under Grant 281B101516 is gratefully acknowledged. The authors have no conflicts of interest.

SUPPLEMENTARY MATERIAL

Supplemental Table S1. Number of animals by sex and birth year

Sex and Birth year	2014	2015	2016	2017	2018	2019	2020	Total
Female	548	1,525	3,679	6,915	10,686	16,702	9,730	49,793
Male	1,296	2,226	3,321	3,615	3,971	4,635	2,314	21,926
Total	1,844	3,751	7,000	10,530	14,657	21,337	12,044	**71,163**

Supplemental Table S2. Number of animals in data by sex and birth year for exploring the association between similarities in Mendelian sampling values

Sex and Birth year	2016	2017	2018	2019	2020	Total
Female	115	249	819	2,222	1,111	4,516
Male	220	527	828	1,394	126	3,095
Total	335	776	1,647	3,616	1,237	**7,611**

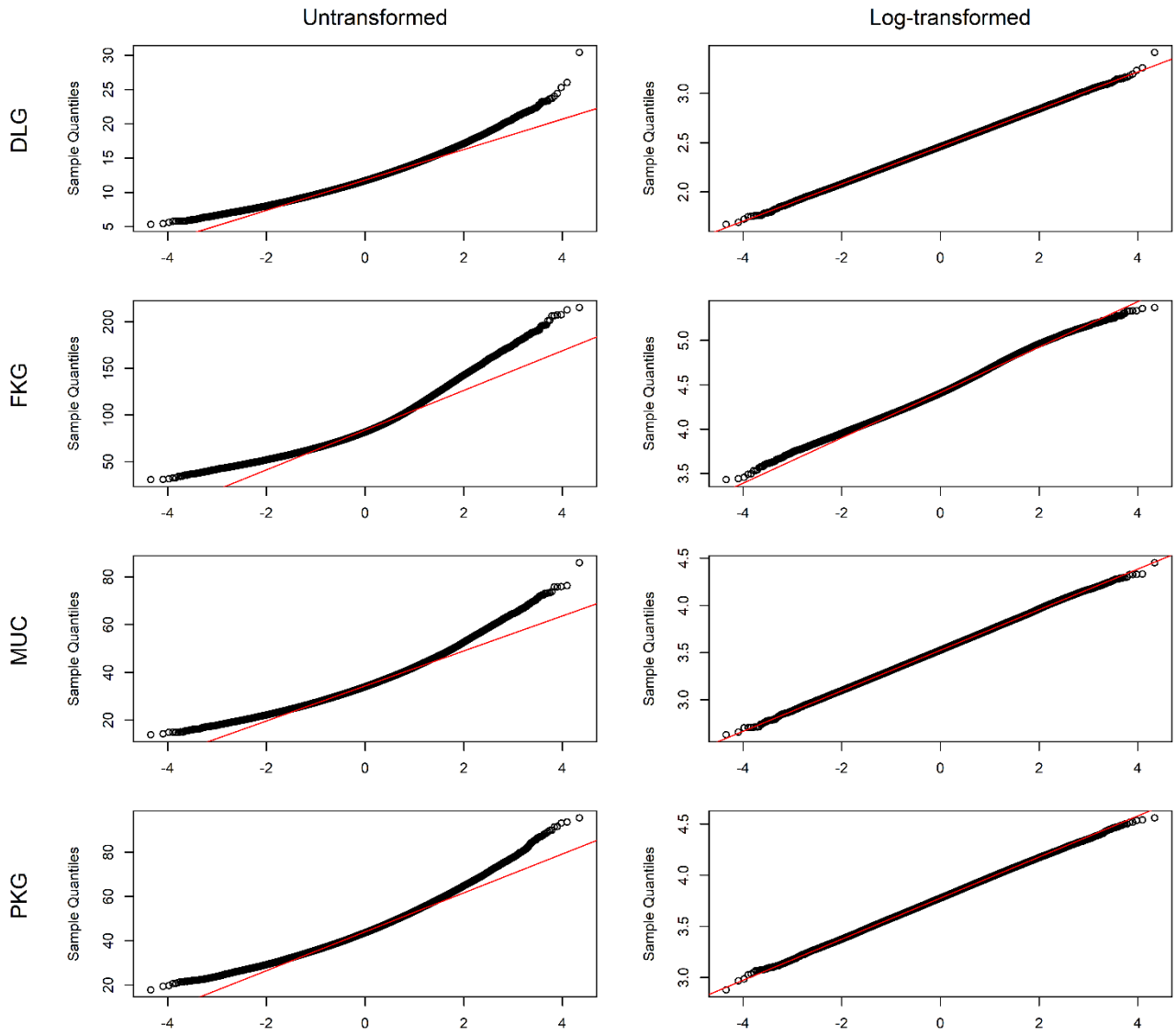
Supplemental Table S3. Summary statistics of trait-specific similarity matrices (upper triangle) for chromosomes 4 and 14

Traits	Chromosome 4				Chromosome 14			
	LG	FY	MUC	PY	LG	FY	MUC	PY
Diagonal elements (Mendelian sampling variance)[1]								
Mean	0.56	2.35	0.72	1.57	0.34	15.48	1.60	3.84
SD	0.46	1.83	0.56	1.27	0.31	17.42	1.52	4.13
Minimum	0.01	0.08	0.02	0.00	0.00	0.00	0.00	0.00
Median	0.43	1.82	0.56	1.20	0.24	9.37	1.10	2.08
Maximum	5.50	25.24	7.07	19.00	4.41	122.36	17.69	34.38
CV (%)	81.60	77.97	78.05	80.89	91.77	112.51	95.51	107.59
Off-diagonal elements (similarity between Mendelian sampling values)[2]								
Mean	0.23	0.90	0.29	0.61	0.15	9.24	0.80	1.89
SD	0.24	0.93	0.29	0.64	0.16	10.69	0.85	2.38
Minimum	0	0	0	0	0	0	0	0
Median	0.15	0.62	0.20	0.42	0.10	5.26	0.52	0.98
Maximum	4.98	22.77	6.14	17.37	4.19	120.23	16.68	33.65
CV (%)	104.30	102.76	101.42	104.00	107.68	115.67	106.73	126.03
Off-diagonal elements (standardized similarity between Mendelian sampling values)[2]								
Mean	0.42	0.39	0.41	0.40	0.46	0.66	0.53	0.53
SD	0.25	0.24	0.25	0.24	0.26	0.29	0.28	0.28
Minimum	0	0	0	0	0	0	0	0
Median	0.41	0.37	0.40	0.39	0.47	0.75	0.56	0.55
Maximum	1	1	1	1	1	1	1	1
CV (%)	59.94	61.36	60.37	60.81	56.66	44.06	52.25	53.40

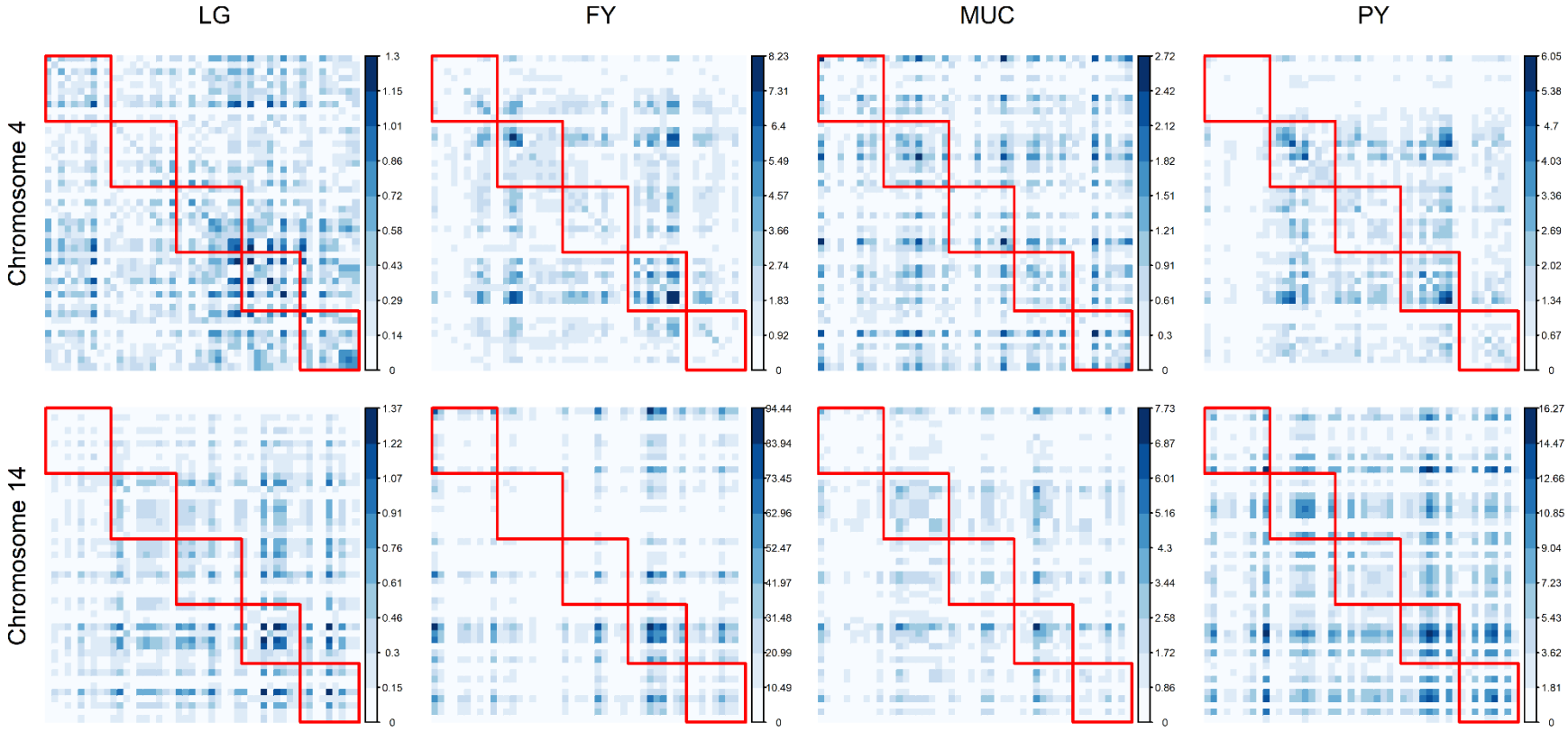
[1] Number of observations = 71,163

[2] Number of observations = 2,532,050,703

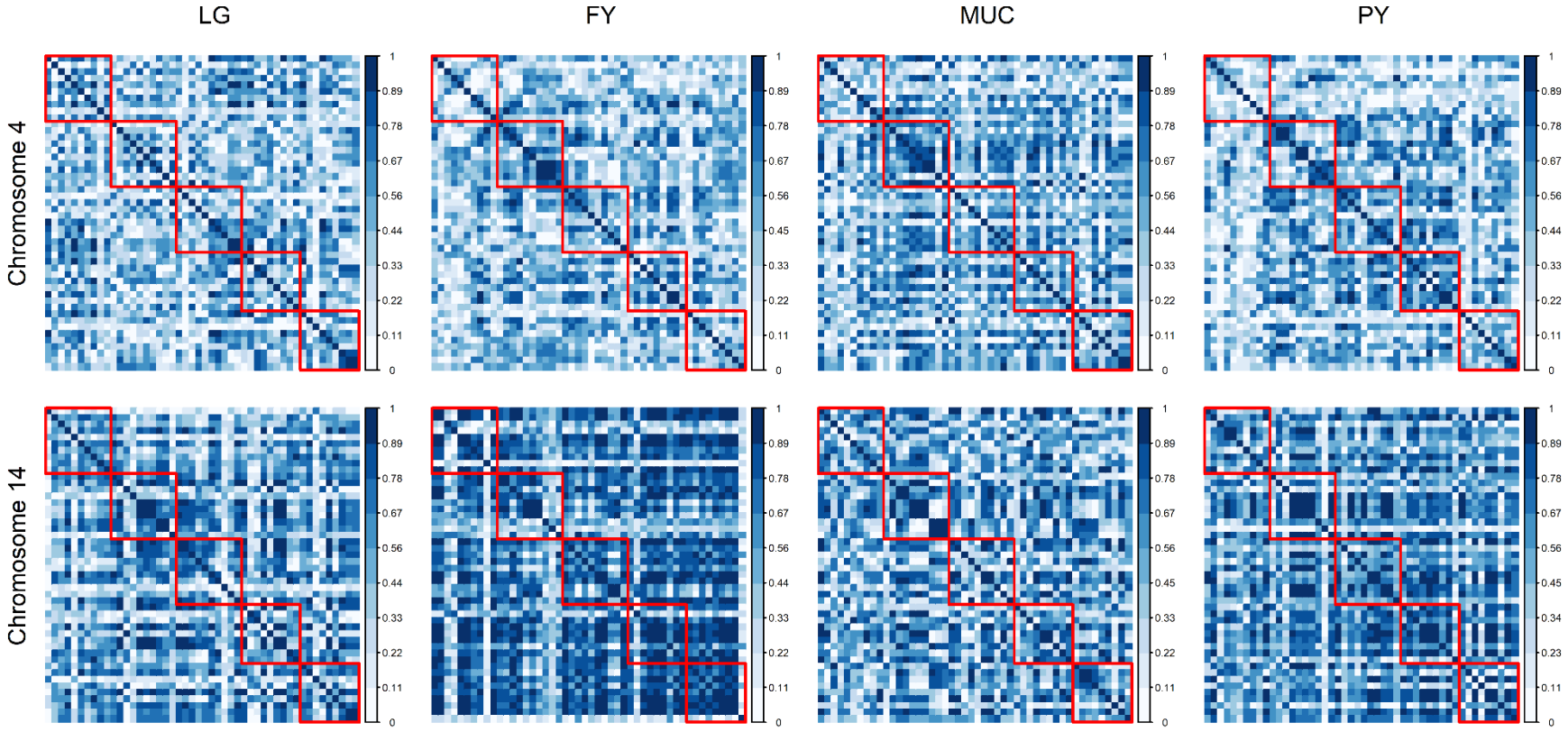
LG = longevity (days); FY = fat yield (kg); MUC = milk urea content (mg/kg); PY = protein yield (kg); AG = aggregate genotype; SD = standard deviation; CV = coefficient of variation



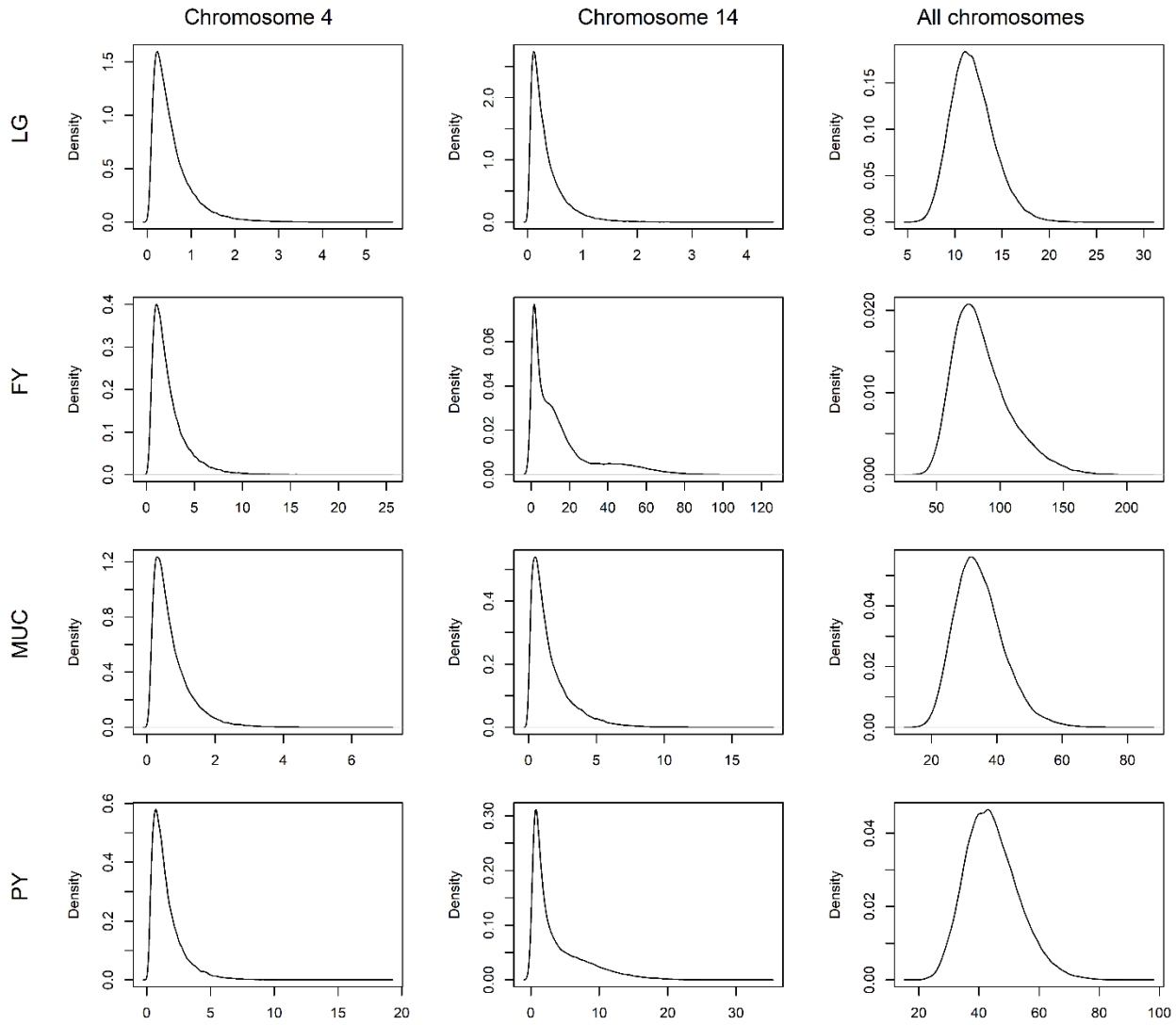
Supplemental Figure S1. QQ-plots of untransformed and log-transformed Mendelian sampling variances. *LG* longevity (days), *FY* fat yield (kg), *MUC* milk urea content (mg/kg), and *PY* protein yield (kg).



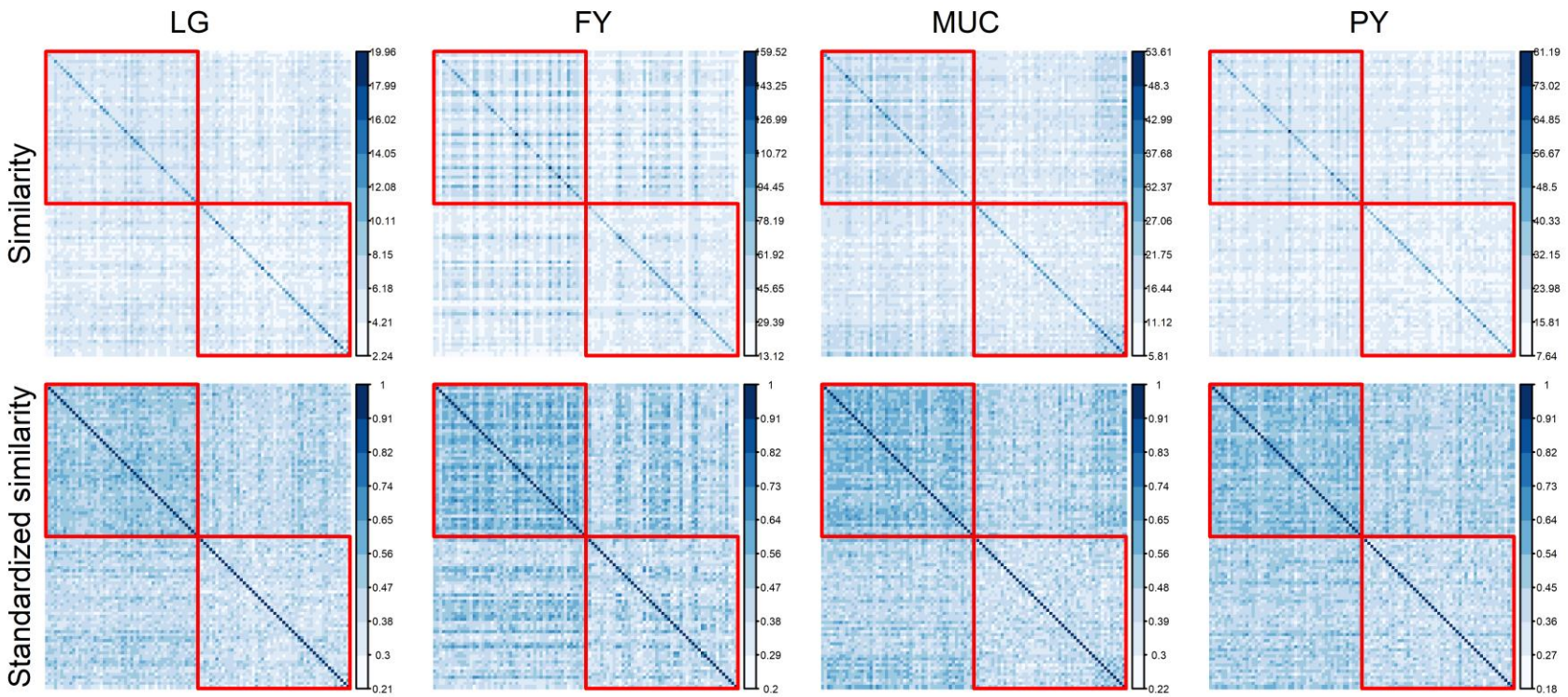
Supplemental Figure S2. Similarity matrices for chromosomes 4 and 14 of various traits for 48 animals from five full-sib families (separated by red lines). *LG* longevity (days), *FY* fat yield (kg), *MUC* milk urea content (mg/kg), and *PY* protein yield (kg). The Mendelian sampling variance of each individual and haplotype similarities between animals are located on the diagonal and off-diagonals, respectively.



Supplemental Figure S3. Standardized similarity matrices for chromosomes 4 and 14 of various traits for 48 animals from five full-sib families (separated by red lines). *LG* longevity (days), *FY* fat yield (kg), *MUC* milk urea content (mg/kg), and *PY* protein yield (kg). Diagonal elements are all one. Off-diagonal elements represent Standardized haplotype similarities between animals.



Supplemental Figure S4. Kernel Density plots of Mendelian sampling variance distributions. LG longevity (days), *FY* fat yield (kg), *MUC* milk urea content (mg/kg), and *PY* protein yield (kg)



Supplemental Figure S5. Similarity and standardized similarity matrices of animals with top and bottom random effects (separated by red lines). *LG* longevity (days), *FY* fat yield (kg), *MUC* milk urea content (mg/kg), and *PY* protein yield (kg). The Mendelian sampling variance of each individual and haplotype similarities between animals are located on the diagonal and off-diagonals, respectively (top panel). The standardized similarity matrices (bottom panel) make small pairwise haplotype similarities more visible.

LITERATURE CITED

Bijma, P., Y. C. J. Wientjes, and M. P. L. Calus. 2020. Breeding Top Genotypes and Accelerating Response to Gametic Variance. *Genetics*. 214:91–107.

Bonk, S., M. Reichelt, F. Teuscher, D. Segelke, and N. Reinsch. 2016. Mendelian sampling covariability of marker effects and genetic values. *Genet. Sel. Evol.* 48:1–11.

Bouwman, A. C., H. D. Daetwyler, A. J. Chamberlain, C. H. Ponce, M. Sargolzaei, F. S. Schenkel, G. Sahana, A. Govignon-Gion, S. Boitard, M. Dolezal, H. Pausch, R. F. Brøndum, P. J. Bowman, B. Thomsen, B. Guldbrandtsen, M. S. Lund, B. Servin, D. J. Garrick, J. Reecy, J. Vilkki, A. Bagnato, M. Wang, J. L. Hoff, R. D. Schnabel, J. F. Taylor, A. A. E. Vinkhuyzen, F. Panitz, C. Bendixen, L. E. Holm, B. Gredler, C. Hozé, M. Boussaha, M. P. Sanchez, D. Rocha, A. Capitan, T. Tribout, A. Barbat, P. Croiseau, C. Drögemüller, V. Jagannathan, C. Vander Jagt, J. J. Crowley, A. Bieber, D. C. Purfield, D. P. Berry, R. Emmerling, K. U. Götz, M. Frischknecht, I. Russ, J. Sölkner, C. P. Van Tassell, R. Fries, P. Stothard, R. F. Veerkamp, D. Boichard, M. E. Goddard, and B. J. Hayes. 2018. Meta-analysis of genome-wide association studies for cattle stature identifies common genes that regulate body size in mammals. *Nat. Genet.* 50:362–367.

Estaghvirou, S. B. O., J. O. Ogutu, and H. P. Piepho. 2015. How genetic variance and number of genotypes and markers influence estimates of genomic prediction accuracy in plant breeding. *Crop Sci.* 55:1911–1924.

Falconer, D. S., and T. F. C. Mackay. 1996. Introduction to Quantitative Genetics. Longman, Essex, England.

Gilmour, A. R., B. J. Gogel, B. R. Cullis, S. J. Welham, and R. Thompson. 2021. ASReml User Guide Release 4.2 Functional Specification. *VSN Int. Ltd, Hemel Hempstead, HP1 1ES, UK www.vsni.co.uk.*

Haldane, J. B. S. 1919. The combination of linkage values and the calculation of distances between the loci of linked factors. *J. Genet.* 8:299–309.

Honerlagen, H., H. Reyer, M. Oster, S. Ponsuksili, N. Trakooljul, B. Kuhla, N. Reinsch, and K. Wimmers. 2021. Identification of genomic regions influencing N-metabolism and N-excretion in lactating Holstein- Friesians. *Front. Genet.*

Jahnel, R. E., D. Wittenburg, M. Mayer, H. Täubert, and N. Reinsch. 2021. Estimation of genetic parameters of milk urea content. 93:157–168.

Jannink, J. L. 2010. Dynamics of long-term genomic selection. *Genet. Sel. Evol.* 42.

Kennedy, B. W., L. R. Schaeffer, and D. A. Sorensen. 1988. Genetic Properties of Animal Models. *J.*

Dairy Sci. 71:17–26.

Lehermeier, C., S. Teyssèdre, and C. C. Schön. 2017. Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. *Genetics*. 207:1651–1661.

Lin, Z., N. O. I. Cogan, L. W. Pembleton, G. C. Spangenberg, J. W. Forster, B. J. Hayes, and H. D. Daetwyler. 2016. Genetic Gain and Inbreeding from Genomic Selection in a Simulated Commercial Breeding Program for Perennial Ryegrass. *Plant Genome*. 9:1–12.

Liu, Z., F. R. Seefried, F. Reinhardt, S. Rensing, G. Thaller, and R. Reents. 2011. Impacts of both reference population size and inclusion of a residual polygenic effect on the accuracy of genomic prediction. *Genet. Sel. Evol.* 43.

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard. 2001. Prediction of total genetic value using genome-wide dense marker maps. *Genetics*. 157:1819–1829.

Mitchell, R. G., G. W. Rogers, C. D. Dechow, J. E. Vallimont, J. B. Cooper, U. Sander-Nielsen, and J. S. Clay. 2005. Milk urea nitrogen concentration: Heritability and genetic correlations with reproductive performance and disease. *J. Dairy Sci.* 88:4434–4440.

Mitton, J. B., W. S. F. Schuster, E. G. Cothran, and J. C. De Fries. 1993. Correlation between the individual heterozygosity of parents and their offspring. *Heredity (Edinb).* 71:59–63.

Mohammadi, M., T. Tiede, and K. P. Smith. 2015. Popvar: A genome-wide procedure for predicting genetic variance and correlated response in biparental breeding populations. *Crop Sci.* 55:2068–2077.

Mucha, S., and E. Strandberg. 2011. Genetic analysis of milk urea nitrogen and relationships with yield and fertility across lactation. *J. Dairy Sci.* 94:5665–5672.

Müller, C. B. M., S. Görs, M. Derno, A. Tuchscherer, K. Wimmers, A. Zeyner, and B. Kuhla. 2021. Differences between Holstein dairy cows in renal clearance rate of urea affect milk urea concentration and the relationship between milk urea and urinary nitrogen excretion. *Sci. Total Environ.* 755:2021.

Musa, A. A., and N. Reinsch. 2021a. A similarity matrix for hedging haplotype diversity among parents in genomic selection. *Submitted*.

Musa, A. A., and N. Reinsch. 2021b. PyMSQ: A Python package for estimating Mendelian sampling-related quantities. *Prep.*

Nietlisbach, P., L. F. Keller, and E. Postma. 2016. Genetic variance components and heritability of

multiallelic heterozygosity under inbreeding. *Heredity (Edinb).* 116:1–11.

Rzewuska, K., and T. Strabel. 2013. Genetic parameters for milk urea concentration and milk traits in Polish Holstein-Friesian cows. *J. Appl. Genet.* 54:473–482.

Santos, D. J. A., J. B. Cole, T. J. Lawlor, P. M. VanRaden, H. Tonhati, and L. Ma. 2019. Variance of gametic diversity and its application in selection programs. *J. Dairy Sci.* 102:5279–5294.

Sargolzaei, M., J. P. Chesnais, and F. S. Schenkel. 2014. A new approach for efficient genotype imputation using information from relatives. *BMC Genomics*. 15.

Schaeffer, L. R. 2006. Strategy for applying genome-wide selection in dairy cattle. *J. Anim. Breed. Genet.* 123:218–223.

Schnell, F. W., and H. F. Utz. 1976. F1-Leistung und Elternwahl in der Züchtung von Selbstbefruchtern. *Bericht über die Arbeitstagung der Vereinigung Österreichischer Pflanzenzüchter, pp. 234–258, Gumpenstein, Österreich*.

Segelke, D., F. Reinhardt, Z. Liu, and G. Thaller. 2014. Prediction of expected genetic variation within groups of offspring for innovative mating schemes. *Genet. Sel. Evol.* 46:42.

Thaller, G., W. Krämer, A. Winter, B. Kaupe, G. Erhardt, and R. Fries. 2003. Effects of DGAT1 variants on milk production traits in German cattle breeds. *J. Anim. Sci.* 81:1911–1918.

Wei, T., and S. Viliam. 2017. R package “corrplot”: Visualization of a Correlation Matrix. R package version 0.84; 2017. https://github.com/taiyun/corrplot.

GENERAL DISCUSSION

The objectives of this thesis were 1) to develop a similarity matrix and demonstrate its application for hedging haplotype diversity; 2) to devise an approach for rapidly estimating MSV; 3) to develop a software package for rapidly estimating MSV and similarity matrices; and 4) to ascertain the heritability of MSV and investigate the summary statistics and factors affecting the similarity matrix values in a large commercial population. A similarity matrix was developed, and its application for hedging haplotype diversity was demonstrated in Chapter 2, which also presented a computationally faster representation of an analytical method (Bonk et al., 2016) for computing MSV. In Chapter 3, a Python package, *PyMSQ*, was developed for quickly estimating MSV and similarity matrix. *PyMSQ*'s performance in computing MSV and similarity matrices were benchmarked using simulated data of varying dimensions. The heritability of MSV was ascertained in Chapter 4. The similarities of parents based on Mendelian sampling values in a large commercial German Holstein population and the factors influencing similarities were also investigated in Chapter 4.

Similarities between Mendelian sampling values

The novel similarity matrix developed in this thesis fills a critical methodological gap by providing a snapshot of population haplotype diversity, which is useful for increasing selection response and managing population diversity. The similarity matrix, in particular, has a straightforward genetic interpretation; its values are dependent on heterozygosity, linkage phase, and trait genetic architecture. The off-diagonal elements of this matrix represent the similarities between the Mendelian sampling values of parents, whereas the diagonal elements represent each parent's MSV, i.e., its similarity to itself. A high degree of similarity between parents indicates that they share many heterozygous markers with large effects on a trait in the same linkage phase. When both parents have the same haplotype, a standardized similarity of one is obtained.

Similarity matrices are trait-specific and do not require weighting in a single-trait selection. When selecting for multiple traits, however, index weights allow all trait-specific similarity matrices to be summed. The Kullback-Leibler (Kullback and Leibler, 1951) and Jensen-Shannon (Nielsen, 2019) divergence methods from information theory are possible alternatives to the similarity metric for

comparing Mendelian sampling values between parents. Their applicability to breeding, however, was not investigated in this study.

When used in conjunction with a selection criterion in a portfolio optimization approach, the similarity matrix provides breeders with a diversified portfolio of parents that balances genetic return and risk in the next generation. In this context, we define "risk" as a lack of haplotype diversity among parents and consequently among offspring. A diversified portfolio includes many parents (or parent pairs) with high expected genetic returns but low similarity values. The greater the number of parent pairs with distinct linkage phases for markers with large effects on a trait in coupling phase, the lower the similarity values. Consequently, the risk of losing haplotype diversity is lower. The portfolio optimization approach also enables breeders to visualize each decision's consequences and make the most informed judgments. In other words, a breeder can choose any diversified portfolio from the efficient frontier based on the breeder's objectives and risk tolerance. Thus, a diversified portfolio on the efficient frontier maximizes genetic return while minimizing the risk of losing haplotype diversity achieved by incorporating many parents with different heterozygous chromosomal regions.

The similarities between Mendelian sampling values of parents in a large commercial German Holstein cattle population revealed considerable variation, with coefficients of variation ranging from 22% to 32% across studied traits. This indicates that genetic variation exists for substantial genetic improvement. The standardized pairwise similarities also varied greatly and ranged almost over the entire parameter space, with parents exhibiting low to perfect similarities (0.1 to 1). A standardized similarity of one (perfect similarity) indicates that the parents share the same heterozygous markers with large effects on a trait in the same linkage phase. However, except for same-sex and selection groups, further examinations of parents with a perfect similarity revealed no discernible trend in pedigree, birth year, or heterozygosity.

The similarity of Mendelian sampling values is influenced, albeit in a minor way, by time lag (the absolute difference between parents' birth years), sex, and selection; a greater impact is caused by the MSVs of the reference parents. The profound influence of reference parents' MSV values on similarity is evident in the similarity matrices of full- or half-sib families, where parents with comparable MSV values appeared to be more similar regardless of which family they came from. Further, some parents, on average, shared more similarities with other parents. Consequently, the variance due to parents explained between 7.2–11% of the variation in similarity estimates in German Holsteins.

Understanding the factors that contribute to the similarity of Mendelian sampling values between parents may help breeders make better use of the quantity.

While the similarity matrix has numerous breeding applications, computing may take some time, depending on the number of markers and parents. *PyMSQ* took between 13 seconds and 8.4 hours to generate similarity matrices for the aggregate genotype for 1,000 to 100,000 parents for different data sets containing the same sex, three traits, and approximately 40K markers on 29 chromosomes. The time needed to derive similarity matrices increases with the increase in the number of traits. Nonetheless, similarity matrices can be computed in a reasonable amount of time, allowing breeding programs to exploit the benefits of the similarity matrix in breeding decisions.

Mendelian sampling variance

MSV is of practical relevance in breeding decisions (Segelke et al., 2014; Bonk et al., 2016; Bijma et al., 2020). However, due to the time-consuming estimation process, its use in breeding programs has been limited. The faster representation of analytical methods (Bonk et al., 2016; Santos et al., 2019) for predicting MSV presented in this thesis and implemented in the developed *PyMSQ* software is critical because it aims to overcome this limitation. Compared to a Fortran program *gamevar* (De Abreu Santos et al., 2020), *PyMSQ* was up to 240 times faster in estimating Mendelian sampling covariability in various simulated data sets. Fast computation of MSV is critical, especially as next-generation sequencing technologies improve and become more affordable, increasing the number of markers and genotyped parents.

Analysis of a large commercial German Holstein cattle population revealed that large variations exist in MSV, with coefficients of variation ranging from 19.2 to 26.5% across the studied traits. MSV depends on heterozygosity, linkage, and trait genetic architecture. Consequently, some families produce more genetically diverse offspring for a trait than others due to variations in heterozygosity and linkage phase of their markers (Segelke et al., 2014; Bonk et al., 2016; Bijma et al., 2020). This result is evident in the similarity matrices derived in this thesis for full-sib and half-sib families. In addition, an examination of the factors influencing MSV revealed that MSV differed between sexes, selection groups, and birth years. Except for sex, this result varied across traits.

The variability in MSV is unknown for many traits, populations, and species. However, the findings of this thesis suggest that variability may be greater in species with smaller genomes and traits controlled by fewer genes. The distribution of the MSV of a population depends on the trait, chromosome, population, and sample of parents. Using the MSV of fat yield as an example, the distribution of the German Holstein data set of 265 cows revealed a bimodal distribution while the data set with 71,163 cattle resembled gamma distributions. This difference could be attributed to the sample size and population type, where the latter comprised top parents and the former were unselected. The distribution of fat yield MSV has been attributed to the strong impact of the *DGAT1* gene (chromosome 14) on milk production traits (Thaller et al., 2003; Segelke et al., 2014; Santos et al., 2019).

Several studies have demonstrated that heterozygosity is heritable, and its heritability can be high in the presence of unequal allele frequencies and little inbreeding (Mitton et al., 1993; Falconer and Mackay, 1996; Nietlisbach et al., 2016). Aside from the fact that closely linked loci are likely to be passed down to offspring, little is known about the inheritance of linkage. As a result of its dependence on the heterozygosity and linkage phase, the initial hypothesis was that MSV would be somewhat heritable. However, the observed heritability estimates of zero for MSV across studied traits imply that MSV is not heritable. The heritability of zero emphasizes that the combination of heterozygosity and linkage is a complex process that cannot be passed down to offspring. As a result, high MSVs in selected parents will not accumulate and increase overall trait heritability over successive generations.

Previous research has shown that selecting parents based on the selection index, $I = bv + k \cdot \sigma$, increases genetic gains and minimizes genetic variance loss over successive generations with recurrent selection than selection based solely on GEBVs (Lehermeier et al., 2017; Müller et al., 2018; Santos et al., 2019). Again, bv is the GEBV, σ is the Mendelian sampling standard deviation (square root of MSV), and k is a constant. Two hypotheses have been proposed to explain this result (Bijma et al., 2020). First, assuming MSV is heritable, selecting parents with high MSVs will result in offspring with high MSV in the next generation. Second, if parents with high MSVs are selected in each generation, selection benefits will be reflected in subsequent generations. The discovery that MSV is not heritable supports the second hypothesis. However, unlike GEBV, MSV cannot be accumulated over generations, making directional selection of MSV impossible.

An index $I = c \cdot bv + k \cdot \sigma$ could be defined where c could have zero weight, such that selection of parents (for a single or multiple traits) is based on their MSVs only. If the maximum variance in a population is desired (e.g., trait-based conservation genetics or the creation of a new base population for subsequent selection), such an index could be used over generations of recurrent selection. Such an index could stabilize selection by keeping all quantitative trait loci at medium frequencies and the average MSV at a maximum. The similarity matrix would still be required to hedge haplotype diversity and prevent loss of favorable QTL, particularly under high selection intensity (k). However, the performance of such an index for conservation remains to be investigated in simulated and empirical data sets.

Limitations

The similarity matrix and MSV are derived using available information on marker effects, a genetic map, and phased genotypes. These quantities are derived assuming that the genetic map is accurate, based on known recombination rates and the absence of interference (Haldane, 1919). Furthermore, this thesis assumed common sex or sex average genetic map. However, in practice, recombination rates may vary even between parents and families (Bauer et al., 2013), which could bias the MSV and similarity matrix estimates. Furthermore, the methods used in obtaining marker effects and phased genotypes could also bias their estimates (Meuwissen et al., 2001; Druet et al., 2014; Lehermeier et al., 2017).

Future studies

The similarity matrix is derived using available information on estimated marker effects, phased genotypes, and a genetic map. The method used to obtain this information may affect similarity matrix estimates. Apart from being influenced by sex, selection group, and birth year, it is expected that the values of this matrix will vary between species and populations. Future research can focus on the bias in the similarity matrix caused by how this information is derived. The effective use of the factors influencing the values of the similarity matrix in breeding remains to be investigated. Finally, the long-term effects of repeated application of the similarity matrix combined with various selection

criteria on genetic gain and genetic variability in breeding programs with recurrent selection remain to be investigated.

Conclusions

In conclusion, the novel similarity matrix has numerous potential breeding applications—including hedging haplotype diversity—as shown in chapter two. The matrix has a straightforward genetic interpretation, and its values are dependent on heterozygosity, linkage phase, and trait genetic architecture. A high degree of similarity between parents indicates that they share many heterozygous markers with large effects on a trait in the same linkage phase. The fast approach for estimating MSV proposed in chapter two and the software developed in chapter three aim to make Mendelian sampling values more widely used in breeding programs. As shown in chapter four, a large German Holstein population analysis revealed great variation in MSV and similarities between animals. The values of the similarity matrix vary with population and sample size, and they are influenced by sex, selection, and year of birth, as shown in chapter four. Finally, because MSV cannot be inherited, high MSV in selected parents will not accumulate and increase the heritability of the overall trait in subsequent generations. The approaches and information presented in this thesis introduce novel criteria for genomic selection, allowing for increased selection response while hedging haplotype diversity in breeding programs for genetic improvement.

LITERATURE CITED

De Abreu Santos, D. J., J. B. Cole, G. E. Liu, P. M. Vanraden, and L. Ma. 2020. Gamevar.f90: A software package for calculating individual gametic diversity. *BMC Bioinformatics*. 21:3–7.

Bauer, E., M. Falque, H. Walter, C. Bauland, C. Camisan, L. Campo, N. Meyer, N. Ranc, R. Rincent, W. Schipprack, T. Altmann, P. Flament, A. E. Melchinger, M. Menz, J. Moreno-González, M. Ouzunova, P. Revilla, A. Charcosset, O. C. Martin, and C. C. Schön. 2013. Intraspecific variation of recombination rate in maize. *Genome Biol.* 14.

Bijma, P., Y. C. J. Wientjes, and M. P. L. Calus. 2020. Breeding Top Genotypes and Accelerating Response to Gametic Variance. *Genetics*. 214:91–107.

Bonk, S., M. Reichelt, F. Teuscher, D. Segelke, and N. Reinsch. 2016. Mendelian sampling covariability of marker effects and genetic values. *Genet. Sel. Evol.* 48:1–11.

Druet, T., I. M. Macleod, and B. J. Hayes. 2014. Toward genomic prediction from whole-genome sequence data: Impact of sequencing design on genotype imputation and accuracy of predictions. *Heredity (Edinb).* 112:39–47.

Falconer, D. S., and T. F. C. Mackay. 1996. Introduction to Quantitative Genetics. Longman, Essex, England.

Haldane, J. B. S. 1919. The combination of linkage values and the calculation of distances between the loci of linked factors. *J. Genet.* 8:299–309.

Kullback, S., and R. A. Leibler. 1951. On Information and Sufficiency. *Ann. Math. Stat.* 22:79–86.

Lehermeier, C., S. Teyssèdre, and C. C. Schön. 2017. Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. *Genetics*. 207:1651–1661.

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard. 2001. Prediction of total genetic value using genome-wide dense marker maps. *Genetics*. 157:1819–1829.

Mitton, J. B., W. S. F. Schuster, E. G. Cothran, and J. C. De Fries. 1993. Correlation between the individual heterozygosity of parents and their offspring. *Heredity (Edinb).* 71:59–63.

Müller, D., P. Schopp, and A. E. Melchinger. 2018. Selection on expected maximum haploid breeding values can increase genetic gain in recurrent genomic selection. *G3 Genes, Genomes, Genet.* 8:1173–1181.

Nielsen, F. 2019. On the Jensen-Shannon symmetrization of distances relying on abstract means. *Entropy*. 21:485.

Nietlisbach, P., L. F. Keller, and E. Postma. 2016. Genetic variance components and heritability of multiallelic heterozygosity under inbreeding. *Heredity (Edinb).* 116:1–11.

Santos, D. J. A., J. B. Cole, T. J. Lawlor, P. M. VanRaden, H. Tonhati, and L. Ma. 2019. Variance of gametic diversity and its application in selection programs. *J. Dairy Sci.* 102:5279–5294.

Segelke, D., F. Reinhardt, Z. Liu, and G. Thaller. 2014. Prediction of expected genetic variation within groups of offspring for innovative mating schemes. *Genet. Sel. Evol.* 46:42.

Thaller, G., W. Krämer, A. Winter, B. Kaupe, G. Erhardt, and R. Fries. 2003. Effects of DGAT1 variants on milk production traits in German cattle breeds. *J. Anim. Sci.* 81:1911–1918.

SUMMARY

Mendelian sampling variance (MSV) is the genetic variance among full-sibs due to the inheritance of random samples of alleles from both parents. MSV has many breeding applications; however, its use in large-scale breeding programs is limited due to its time-consuming computation. Recently proposed selection indices for achieving long-term genetic gain combine genomic estimated breeding value and MSV. However, these indices tend to select similar parents with high variability potential under high selection intensity, possibly leading to the loss of favorable haplotypes. Therefore, this thesis aimed to develop a faster approach for computing MSV and derive a similarity matrix for hedging haplotype diversity.

The thesis first develops a faster representation of analytical methods to compute MSV using marker effects, a genetic map, and phased genotypes. Then, using the same elements as MSV, it derives a similarity matrix. The off-diagonal elements of this matrix represent the similarities between Mendelian sampling values of parents, and diagonal elements represent the similarity of a parent to itself, which equals its MSV. A high similarity indicates that the parents share many heterozygous markers, with large effects on a trait in the same linkage phase. Similar to how covariance matrices of asset prices are used in finance to create diversified portfolios, the similarity matrix can be used to avoid repeated matings of similar parents and achieve expected genetic gain while hedging haplotype diversity in the next generation.

The thesis then develops the Python package *PyMSQ* for computing Mendelian sampling-related quantities to facilitate their use in breeding programs. Compared to *gamevar* (a recently published Fortran program), *PyMSQ* was up to 240 times faster at computing MSV in the analyzed data sets. Finally, similarity matrices for milk production and longevity traits were calculated using *PyMSQ* for a large German Holstein population to assess their applicability and relevance. The results revealed great variation in MSV and similarities between animals. The values of the similarity matrix vary with population and sample size, and they are influenced by sex, selection, and year of birth. Due to the non-heritability of MSV, it was concluded that high MSVs in selected parents will not accumulate and increase overall trait heritability over successive generations. The similarity matrix presented in this thesis introduces new criteria for genomic selection, allowing for increased genetic gain while hedging haplotype diversity in breeding programs.

ZUSAMMENFASSUNG

Die Mendelsche Zufallsvarianz beschreibt die Variation der Zuchtwerte unter Vollgeschwistern. In züchterischen Anwendungen ist die Schätzung dieser Varianz aus Markerinformationen mit erheblichem Rechenaufwand verbunden. Dies gewinnt auch deshalb an Bedeutung, da in jüngerer Zeit werden verstärkt Kombinationen aus genomischem Zuchtwert und der Mendelschen Zufallsvarianz als Selektionskriterium diskutiert werden, mit dem Ziel einen größeren langfristigen Zuchtfortschritt zu erzielen. Eine Nebenwirkung könnte jedoch ein ungewollter Verlust von wertvollen Haplotypen und damit genetischer Variation sein. Daher verfolgte diese Arbeit zwei Hauptziele. Einerseits, sollte der Rechenaufwand für die Schätzung der Mendelschen Zufallsvarianz gesenkt werden. Andererseits, sollte eine Ähnlichkeitsmatrix abgeleitet werden, die es ermöglicht Anpaarungen so auszuwählen, dass in jeder Selektionsrunde eine hohe Wahrscheinlichkeit für Nachkommen mit hohen Zuchtwerten der Nachkommen besteht bei gleichzeitig niedriger Ähnlichkeit der Haplotypen aller ausgewählten Eltern.

Zunächst wurde eine rechentechnisch effizientere Darstellungsweise für eine bekannte analytische Methode zur Berechnung der Mendelschen Zufallsvarianz eingeführt. Diese Varianzen bilden gleichzeitig die Diagonalelemente der neu entwickelten Ähnlichkeitsmatrix für alle Anpaarungen. Die neu abgeleiteten Nebendiagonalelemente lassen sich dann auf analoge Weise berechnen. Wie in der Portfoliotheorie aus der Finanzmathematik kann man mit Hilfe der Ähnlichkeitsmatrix wiederholte Anpaarung ähnlicher Tiere vermeiden und einen „diversifizierenden“ Effekt auf die merkmalsrelevanten Haplotypenabschnitte der ausgewählten Eltern erzielen.

Für das Aufstellen der Ähnlichkeitsmatrix wurde ein Python- Paket *PyMSQ* entwickelt für eine einfache Anwendung in Zuchtprogrammen, das im Vergleich zu einem kürzlich veröffentlichten Programm 240 mal schneller die Mendelsche Zufallsvarianz berechnen konnte. Für Milchleistungs- und Langlebigkeitsmerkmale in Deutschen Holsteins wurden große Streuungen in den paarweisen Ähnlichkeiten und Mendelschen Zufallsvarianzen gefunden. Signifikante Einflüsse von Geschlecht, Selektion und Geburtsjahr auf die Elemente der Ähnlichkeitsmatrix konnte gezeigt werden, während Heritabilitäten von nahezu Null geschätzt wurden. Mit der Einführung der Ähnlichkeitsmatrix können neue Kriterien, die zu einem schnelleren Selektionserfolg führen, in der genomischen Selektion angewandt und gleichzeitig die Diversität der Haplotypen in der Zuchtpopulation absichert werden.

APPENDIX

Documentation of PyMSQ functions

The main functions of `msq` module.

msq.example_data()

Imports example data (genetic map, allele substitution effects for three milk traits, phased genotypes, and phenotypic information).

msq.Datacheck(gmap, meff, gmat, group, indwt, progress=False)

Checks the data for errors, converts haplotypes to genotypes and stores relevant info as an object.

Parameters

- gmap : pandas DataFrame providing genetic map(s)
  Index: RangeIndex
  Columns:
  Name: CHR, dtype: int64; chromosome number
  Name: SNPName, dtype: object; marker name
  Name: Position: dtype: int64; marker position in bp
  Name: group: dtype: float64; marker distance (cM) or reco rates
- meff : pandas DataFrame providing allelle substitution or marker effects
  Index: RangeIndex
  Columns:
  Name: trait names: float64; no. of columns = no of traits
- gmat : pandas DataFrame providing phased genotypes
  Index: RangeIndex
  Columns:
  Name: ID, dtype: int64 or str; identification of individuals
  Name: haplotypes, dtype: object; must be biallelic
- group : pandas DataFrame providing phenotypic information
  Index: RangeIndex
  Columns:
  Name: group, dtype: object; group code of individuals, e.g., M, F
  Name: ID, dtype: int64 or str; identification of individuals
- indwt : list of index weights for each trait
- progress : bool, optional; print progress of the function if True

Returns

- class object : input data as a class object

msq.popcovmat(info, mposunit, method)

Derives population-specific covariance matrices.

Parameters

- info : A class object created using the function "msq.Datacheck"
- mposunit : string containing "cM" or "reco".
- method : An integer with a value of 1 for Bonk et al.'s approach or 2 for Santos et al's approach

Returns

- list : A list containing group-specific pop covariance matrices for each chr.

msq.msvarcov_g(info, covmat, sub_id, progress=False)

Derives Mendelian sampling co(variance) and aggregate genotype.

Parameters

- info : A class object created using the function "msq.Datacheck"
- covmat : A list of pop cov matrices created using "msq.popcovmat" function
- sub_id : pandas dataframe of one column containing ID numbers of specific individuals to be evaluated
  Index: RangeIndex (minimum of 2 rows)
  Columns:
  Name: ID, dtype: int64 or str; identification of individuals
- progress : bool, optional; print progress of the function if True

Returns

- df : pandas dataframe containing the Mendelian sampling (co)variance and aggregate genotype

`Note`: If sub_id is None, Mendelian (co-)variance will be estimated for all individuals. Otherwise, Mendelian (co-)variance will be estimated for the individuals in sub_id

msq.msvarcov_gcorr(msvmsc)

Standardizes Mendelian sampling covariances.

Parameters

- msvmsc: pandas dataframe containing the Mendelian sampling (co)variance derived using function "msq.msvarcov_g"

Returns

- df : pandas dataframe containing the Mendelian sampling correlations

msq.selstrat_g(selstrat, info, sub_id, msvmsc, throrconst)

Calculates selection criteria (GEBV, PBTI, or bvindex using gametic approach.

Parameters

- selstrat : str A str containing any of GEBV, PBTI or bvindex
- info : A class object created using the function "msq.Datacheck"

- sub_id : pandas dataframe of one column containing ID numbers of specific individuals to be evaluated
  Index: RangeIndex (minimum of 2 rows)
  Columns:
  Name: ID, dtype: int64 or str; identification of individuals
- msvmsc : pandas dataframe created using the function "msq.msvarcov_g"
- throrconst : float
  If selstrat is PBTI, a throrconst of value 0.05 sets threshold at top 5% of GEBV.
  If selstrat is bvindex, throrconst is a constant.

Returns

- df : pandas dataframe with estimated selection criteria for each trait and aggregate
  Index: RangeIndex
  Columns: ID, Group, trait names and Overall

`Note` : If selstrat is GEBV, None may be used for throrconst and msvmsc. If sub_id is None and selstrat is GEBV, GEBVs will be estimated for all individuals. However, if selstrat is not GEBV, the chosen selection criterion will be estimated for all individuals in msvmsc data frame.

msq.simmat_g(info, covmat, sub_id, chrinterest, save=False, stdsim=False, progress=False)

Compute similarity matrices using gametic approach.

Parameters

- info : A class object created using the function "msq.Datacheck"
- covmat : A list of pop cov matrices created using "msq.popcovmat" function
- sub_id : pandas dataframe of one column containing ID numbers of specific individuals to be evaluated
  Index: RangeIndex (minimum of 2 rows)
  Columns:
  Name: ID, dtype: int64 or str; identification of individuals
- chrinterest : str or list of int
  list of chromosome numbers of interest or str with "all" or "none"
- save : bool, optional; write trait-specific sim mats to file if true
- stdsim : bool, optional; print write std sim mats to file if true
- progress : bool, optional; print progress of the task if true

Returns

- list : list containing simimlarity matrices for each group

msq.selstrat_z(selstrat, info, sub_idz, msvmsc, throrconst, selmale, selfm, maxmale)

Calculate selection criteria (GEBV, PBTI, or bvindex) for zygotes.

Parameters

- selstrat : str
  A str containing any of GEBV, PBTI or bvindex
- info : A class object created using the function "msq.Datacheck"
- sub_idz : pandas dataframe containing ID numbers of specific mate pairs to be evaluated

Index: RangeIndex (minimum of 2 rows)
Columns:
Name: Male, dtype: int64 or str; identification of males
Name: Female, dtype: int64 or str; identification of females

- msvmsc : pandas dataframe created using the function "msq.msvarcov_g"
- throrconst : float
  If selstrat is PBTI, a throrconst of value 0.05 sets threshold at top 5% of GEBV of the population.
  If selstrat is bvindex, throrconst is a constant.
- selmale : list
  list of two items. 1st item is the str coding for males as in phenotypic information. The 2nd item is a float representing x% of males to be used
- selfm : list
  list of two items. 1st item is the str coding for females as in phenotypic information. The 2nd item is a float representing x% of females to be used
- maxmale : integer of maximum number of allocations for males

Returns

- df : pandas dataframe
  Index: RangeIndex
  Columns:
  MaleID, FemaleID, MaleIndex, FemaleIndex, trait names and Overall

`Note` : If selstrat is GEBV, None may be used for throrconst and msvmsc. If sub_idz is None and selstrat is GEBV, GEBVs will be estimated for all matepairs. However, if selstrat is not GEBV, the chosen selection criterion will be estimated for all individuals in msvmsc data frame.

msq.simmat_z(info, sub_idz, covmat, chrinterest, save=False, stdsim=False, progress=False)

Compute similarity matrices using zygotic approach for specific matepairs.

Parameters

- info : A class object created using the function "msq.Datacheck"
- sub_idz : pandas dataframe containing ID numbers of specific mate pairs to be evaluated
  Index: RangeIndex (minimum of 2 rows)
  Columns:
  Name: Male, dtype: int64 or str; identification of males
  Name: Female, dtype: int64 or str; identification of females
- covmat : A list of pop cov matrices created using "msq.popcovmat" function
- chrinterest : str or list of int
  list of chromosome numbers of interest or str with "all" or "none"
- save : bool, optional; write trait-specific sim mats to file if True
- stdsim : bool, optional; print write std sim mats to file if True
- progress : bool, optional; print progress of the task if True

Returns

- list : containing simimlarity matrices for each group

Illustration: Using functions of *PyMSQ*

Import relevant packages and modules

```
from PyMSQ import msq
import numpy as np
import time
```

Importation of example data and expected data formats

In the following, we import a Holstein-Friesian cattle dataset with 265 cows from five half-sib families from Musa and Reinsch (2021), a subset of data from previous studies (Melzer et al., 2013; Hampel et al., 2018). The dataset contains 39780 markers for 29 (autosomal) chromosomes and marker effects for three example milk traits.
The genetic map, marker effects, phased genotypes, and phenotypic information can be imported using the function `msq.example_data()`:

```
gmap, meff, gmat, group, _ = msq.example_data()
```

The main information required by the functions in the `msq` module and how they should be provided are explained below:

1. Phenotypic information: This information should be provided as a pandas data frame with two columns. As indicated below, the first column should provide group classification (e.g., sex, breed, or strain), whereas the second column should provide identification numbers (ID) of individuals:

```
print(group)
    group     ID
0       F  10001
1       F  10002
2       F  10003
3       F  10004
4       F  10005
..    ...    ...
260     F  10261
261     F  10262
262     F  10263
263     F  10264
264     F  10265

[265 rows x 2 columns]
```

All individuals in this data are females (F). In order to demonstrate the use of some functions for zygotes, let us assume the first 130 individuals are males (M) like:

```
group.iloc[0:130, 0] = "M"
print(group)
```

```
    group     ID
0       M  10001
1       M  10002
2       M  10003
3       M  10004
4       M  10005
..    ...    ...
260     F  10261
261     F  10262
262     F  10263
263     F  10264
264     F  10265

[265 rows x 2 columns]
```

2. Genetic map: The genetic map should be provided as a pandas data frame. The first three columns must be the chromosome number, marker name, and marker position in base pairs. The remaining column(s) should be maker distance/position (cM) or recombination rates in the order listed in the group column of the phenotypic information (1). Let us assume the group classification is sex. If males are listed first, then the fourth column should be the marker distance/position (or recombination rates) for males. The fifth column should be the marker distance/position (or recombination rates) for females. If an average map for all groups is provided, the data frame will contain four columns only, with the average marker distance/position as the fourth column (or recombination rates). Below is an example of how the genetic map should be provided:

```
print(gmap)
       CHR   SNPName  Position     group1
0        1      SNP1    113641   0.113641
1        1      SNP2    244698   0.244698
2        1      SNP3    369418   0.369418
3        1      SNP4    447277   0.447277
4        1      SNP5    487653   0.487653
...    ...       ...       ...        ...
39775   29  SNP39776  51899151  51.899151
39776   29  SNP39777  51920849  51.920849
39777   29  SNP39778  51986600  51.986600
39778   29  SNP39779  52030414  52.030414
39779   29  SNP39780  52112161  52.112161

[39780 rows x 4 columns]
```

Let us add another column to the data frame to simulate a data frame with group-specific maps.

```
gmap.insert(4, "group2", gmap.iloc[:, 3], True)
print(gmap)
```

```
       CHR   SNPName  Position     group1     group2
0        1      SNP1    113641   0.113641   0.113641
1        1      SNP2    244698   0.244698   0.244698
2        1      SNP3    369418   0.369418   0.369418
3        1      SNP4    447277   0.447277   0.447277
4        1      SNP5    487653   0.487653   0.487653
...    ...       ...       ...        ...        ...
39775   29  SNP39776  51899151  51.899151  51.899151
39776   29  SNP39777  51920849  51.920849  51.920849
39777   29  SNP39778  51986600  51.986600  51.986600
39778   29  SNP39779  52030414  52.030414  52.030414
39779   29  SNP39780  52112161  52.112161  52.112161

[39780 rows x 5 columns]
```

PyMSQ will use `group1` as the marker distance/position (or recombination rates) for males and `group2` for females because males are first in the data frame for phenotypic information.
Note: When using recombination rates, the first marker on a chromosome must be zero, followed by recombination rate between markers m and m+1. However, the first element on each chromosome does not have to be zero when using genetic position/distance (specified in cM) because the function utilizes the differences between given values. This allows any subset of markers from a genetic map to be conveniently used by copying their physical (base-pairs) and genetic map (cM) positions.

3. Allele substitution effects: The allele substitution or marker effects should be provided as a pandas data frame with named columns. This data frame provides the trait names used in *PyMSQ* functions. As a result, users should name these columns according to how they want them to appear in the results. The number of rows of this data frame should be equal to the number of markers, and the number of columns should be equal to the number of traits plus one. The first column should be the marker name (same as in the genetic map), and the remaining column(s) should be allele substitution or marker effects for each trait. Here is an example:

```
print(meff)
        SNPName       fat        pH   protein
0          SNP1  0.000059 -0.000163 -0.000211
1          SNP2 -0.000051 -0.000773 -0.000006
2          SNP3  0.000034 -0.000047 -0.000075
3          SNP4  0.000026  0.000101 -0.000118
4          SNP5  0.000021  0.000066  0.000075
...         ...       ...       ...       ...
39775  SNP39776 -0.000104 -0.000078 -0.000017
39776  SNP39777  0.000205 -0.000145 -0.000053
39777  SNP39778  0.000034  0.000025  0.000043
39778  SNP39779 -0.000148 -0.000033 -0.000003
39779  SNP39780 -0.000059 -0.000004  0.000013

[39780 rows x 4 columns]
```

4. Phased genotypes: Phased genotypes: Individuals' paternal and maternal haplotypes should be arranged in rows with their IDs. The phased genotypes should also be provided as a pandas data frame in one of the following ways:

- i: as strings array – The data frame should have two columns (ID and haplotypes as strings without spaces), and the number of rows should be 2 * the number of individuals, as follows:

```
print(gmat)
         0                                                    1
0    10001  022222202222222222220022222022200000200022222...
1    10001  222202020222222022222222220222222222222220022...
2    10002  220022222222002222222202222222222222220002222...
3    10002  222222220222022222222022222222222222222000222...
4    10003  220022222222002222222202222222222222220002222...
..     ...                                                ...
525  10263  200022222222222200002202222222222222220002222...
526  10264  022222202222222222220022222022200002222222022...
527  10264  222222220222022222222022202022222222222222222...
528  10265  222222222222222222222022222222222222222222222...
529  10265  220022222222002222222202222222222222220002222...

[530 rows x 2 columns]
```

- ii: as an integer array. Here the number of rows is still the same as above, but the number of columns is 1 (ID number) + the number of markers.

Note: Since genotypes are already phased into haplotypes, *PyMSQ* does not expect missing values and, therefore, does not accept data with missing values. It does, however, accept any coding for phased haplotypes as long as they are biallelic (i.e., two integers must be used for coding) and taken from integers ranging from 0 to 9. As a result, outputs from several genotype phasing software can be used with ease.

Checking the input information for errors

PyMSQ requires that the main information, including index weights, be checked for errors using the function `msq.Datacheck` to enhance the smooth running of other functions. Briefly, it performs checks to ensure the following:

- The number of traits matches the length of index weights.
- The ID of individuals in phenotypic information and phased genotypes are ordered, and the same.
- There are no missing allele substitution or marker effects, and all entries are numeric.
- Marker names in the genetic map and marker effects are ordered, and the same.
- The number of maps is the same as the number of groups if the map number is more than one.
- The number of markers in phased genotypes, genetic map, and marker effects is the same.
- The marker distance/position or recombination rates on each chromosome are ordered.

A corresponding error message is printed to screen if any of the above checks fail.

In addition to checking for errors, the function transforms phased haplotypes to genotypes and saves all relevant information as a class object, subsequently used by other functions.

```
index_wt = [1, 1, 1]
start = time.time()
data = msq.Datacheck(gmap=gmap, meff=meff, gmat=gmat, group=group,
                     indwt=index_wt, progress=True)
print('Time taken: ', round(time.time() - start, 2), 'secs')
Converting phased haplotypes to genotypes
Data passed the test!
Number of individuals:   265
Number of groups:        2 :  ['M' 'F']
Number of specific maps: 2
Number of chromosomes:   29
Total no. markers:       39780
Number of trait(s):      3
Trait name(s) and Index weight(s)
fat :  1
pH :  1
protein :  1
Time taken:  2.9 secs
```

The main information may be deleted because they are already stored as a class object.

```
del gmap, meff, gmat, group, index_wt
```

Setting up the population covariance matrix

The population covariance matrix is derived using function `msq.popcovmat` according to Bonk et al. (2016) if parameter `method=1` or Santos et al. (2019) if `method=2`. The map type must be specified using `mposunit= "cM"` for genetic position/distance (centimorgan), or `mposunit= "reco"` for recombination rates. If `mposunit='reco'`, an additional check is run to ensure that the first marker on each chromosome has a value of zero. An error message is displayed on the screen if the value is not zero. When group-specific maps are available, this function returns a named list of lists containing population covariance matrices (for each chromosome) for each group.

```
start = time.time()
covmatrices = msq.popcovmat(info=data, mposunit="cM", method=1)
print('Time taken: ', round(time.time() - start, 2), 'secs')
Time taken:  3.94 secs
```

Estimation of Mendelian sampling (co)variance for gametes produced by an individual

The function `msq.msvarcov_g` can be used to estimate the Mendelian sampling variance (MSV) for each trait, MSV of aggregate genotype (AG) for multiple traits, as well as their covariances. As a result, a data frame with trait names indicating Mendelian sampling variance for those traits, as well as a combination of trait names separated by "_" indicating Mendelian covariances between those traits, is created.

```
start = time.time()
msvmsc_g = msq.msvarcov_g(info=data, covmat=covmatrices,
                          sub_id=None, progress=True)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(msvmsc_g)
Progress: |████████████████████████████████████████| 100% Complete
Time taken:  25.21 secs
        ID Group       fat    pH_fat        pH  protein_fat  protein_pH  \
0    10001     M  0.000466  0.000154  0.001805     0.000136   -0.000157
1    10002     M  0.023262  0.014908  0.013225     0.017306    0.011082
2    10003     M  0.021665  0.013760  0.010475     0.015662    0.009849
3    10004     M  0.000949  0.000317  0.004836    -0.000044   -0.000158
4    10005     M  0.021892  0.014447  0.012676     0.015576    0.010120
..     ...   ...       ...       ...       ...          ...         ...
260  10261     F  0.022112  0.012953  0.010050     0.017090    0.009524
261  10262     F  0.023853  0.016183  0.012896     0.017691    0.012189
262  10263     F  0.000604  0.000096  0.002694     0.000025    0.000126
263  10264     F  0.022809  0.015301  0.012748     0.017123    0.011174
264  10265     F  0.022491  0.016226  0.013892     0.016460    0.011973

      protein    AG_fat     AG_pH  AG_protein        AG
0    0.002321  0.000756  0.001802    0.002300  0.004858
1    0.015098  0.055475  0.039215    0.043486  0.138176
2    0.013710  0.051087  0.034084    0.039221  0.124392
3    0.001352  0.001223  0.004995    0.001149  0.007367
4    0.013318  0.051915  0.037243    0.039013  0.128172
..        ...       ...       ...         ...       ...
260  0.015024  0.052155  0.032527    0.041637  0.126319
261  0.015089  0.057727  0.041268    0.044968  0.143963
262  0.001841  0.000725  0.002916    0.001992  0.005633
263  0.014931  0.055232  0.039223    0.043227  0.137683
264  0.013582  0.055177  0.042091    0.042014  0.139282

[265 rows x 12 columns]
```

As seen above, Mendelian sampling (co-)variance will be estimated for all individuals if parameter `sub_id` is None. There may be cases where a user might be interested in a subset of individuals. In this case, the user can provide individuals of interest as a pandas data frame with one column. Assuming the candidates of interest are

```
aaa = np.arange(0, 265, 40)
df_gam = data.group.iloc[aaa, 1]
print(df_gam)
0      10001
40     10041
80     10081
120    10121
160    10161
200    10201
240    10241
Name: ID, dtype: int64
```

Then the Mendelian sampling (co-)variances of traits for these individuals are given below:

```
start = time.time()
msvmsc_gsub = msq.msvarcov_g(info=data, covmat=covmatrices,
                             sub_id=df_gam, progress=False)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(msvmsc_gsub)
Time taken:  0.66 secs
      ID Group       fat    pH_fat        pH  protein_fat  protein_pH  \
0  10001     M  0.000466  0.000154  0.001805     0.000136   -0.000157
1  10041     M  0.000421  0.000161  0.003291     0.000095   -0.000169
2  10081     M  0.000629  0.000242  0.003329     0.000291   -0.000025
3  10121     M  0.022551  0.014175  0.011532     0.016422    0.010774
4  10161     F  0.000469 -0.000055  0.002331     0.000368   -0.000489
5  10201     F  0.000466 -0.000022  0.002190     0.000254   -0.000319
6  10241     F  0.000566  0.000213  0.001609     0.000415    0.000152

    protein    AG_fat     AG_pH  AG_protein        AG
0  0.002321  0.000756  0.001802    0.002300  0.004858
1  0.001163  0.000676  0.003282    0.001089  0.005048
2  0.003116  0.001162  0.003546    0.003382  0.008090
3  0.014838  0.053148  0.036481    0.042034  0.131663
4  0.002253  0.000782  0.001786    0.002132  0.004701
5  0.002331  0.000698  0.001850    0.002266  0.004815
6  0.002229  0.001194  0.001974    0.002796  0.005965
```

Mendelian correlations

Mendelian sampling covariance can be converted to correlations using the function `msq.msvarcov_gcorr` to better compare trait combinations. The function returns a data frame containing correlations between traits. Note that the output does not contain a trait's correlation with itself (equals one).

```
start = time.time()
msvmsc_gsubcorr = msq.msvarcov_gcorr(msvmsc_gsub)
```

```
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(msvmsc_gsubcorr)
Time taken:  0.01 secs
      ID Group    pH_fat  protein_fat  protein_pH    AG_fat     AG_pH  \
0  10001     M  0.168243     0.130355   -0.076619  0.502305  0.608696
1  10041     M  0.136506     0.135905   -0.086558  0.464180  0.805301
2  10081     M  0.167167     0.208082   -0.007698  0.515131  0.683314
3  10121     M  0.879009     0.897727    0.823645  0.975373  0.936237
4  10161     F -0.052918     0.358214   -0.213430  0.526718  0.539680
5  10201     F -0.021511     0.244284   -0.141048  0.466434  0.569633
6  10241     F  0.223733     0.369907    0.080044  0.650271  0.637240

   AG_protein
0    0.684896
1    0.449414
2    0.673654
3    0.950996
4    0.655163
5    0.676601
6    0.766810
```

Estimation of selection criteria

2. Gametic approach: The function `msq.selstrat_g` can be used to obtain selection criteria for each characteristic as well as an individual's aggregate breeding value (ABV). The parameter `selstrat` can be "`GEBV`," "`PBTI`," or "`INDEX`," repre- senting genomic estimated breeding values, probability to breed top-ranking individuals, or a selection index combining `GEBV` and Mendelian standard deviations, respectively. If `selstrat` is "`PBTI`" or "`INDEX`", parameter sub_id is ignored and estimates are provided for individuals in Mendelian sampling (co-)variance (`msvmsc` parameter) data frame.

```
start = time.time()
sel_g = msq.selstrat_g(selstrat="PBTI", info=data, sub_id=df_gam,
                       msvmsc=msvmsc_g, throrconst=0.05)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(sel_g)
Time taken:  0.49 secs
        ID Group       fat        pH   protein       ABV
0    10001     M  0.996264  0.000000  0.651731  0.033476
1    10002     M  0.084399  0.000003  0.134932  0.047714
2    10003     M  0.052664  0.006790  0.128033  0.141537
3    10004     M  0.999113  0.000138  0.000003  0.183958
4    10005     M  0.192267  0.000004  0.000017  0.007792
..     ...   ...       ...       ...       ...       ...
260  10261     F  0.002609  0.001245  0.000039  0.003536
261  10262     F  0.003807  0.019924  0.000075  0.011257
```

```
262  10263     F  0.623545  0.998058  0.015548  0.999982
263  10264     F  0.002814  0.004171  0.000219  0.007632
264  10265     F  0.000879  0.099573  0.000032  0.012122

[265 rows x 6 columns]
```

If `selstrat` is `GEBV`, estimates are provided for individuals in the `sub_id` data frame. Other parameters may be set to `None` because they are not required for `GEBV`. For example, a user may set the `sub_id` parameter to `None` if interested in all individuals.

```
start = time.time()
sel_g = msq.selstrat_g(selstrat="GEBV", info=data, sub_id=None,
                       msvmsc=None, throrconst=None)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(sel_g)
Time taken:  0.31 secs
        ID Group       fat        pH   protein       ABV
0    10001     M  0.270993 -0.214728  0.291675  0.347941
1    10002     M  0.003386 -0.284890  0.137310 -0.144194
2    10003     M -0.025118 -0.017753  0.139902  0.097031
3    10004     M  0.309569 -0.018075  0.106850  0.398344
4    10005     M  0.084600 -0.269435 -0.205395 -0.390230
..     ...   ...       ...       ...       ...       ...
260  10261     F -0.202100 -0.068357 -0.211195 -0.481652
261  10262     F -0.198903  0.001456 -0.192614 -0.390060
262  10263     F  0.220997  0.384744  0.180376  0.786116
263  10264     F -0.204883 -0.062984 -0.156713 -0.424580
264  10265     F -0.255890  0.083525 -0.192927 -0.365292

[265 rows x 6 columns]
```

If parameter `selstrat="INDEX"`, a constant of `throrconst = 1.5` is used to weight Mendelian sampling standard deviations

```
start = time.time()
sel_gsub = msq.selstrat_g(selstrat="INDEX", info=data, sub_id=None,
                          msvmsc=msvmsc_gsub, throrconst=1.5)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(sel_gsub)
Time taken:  0.02 secs
      ID Group       fat        pH   protein       ABV
0  10001     M  0.167868 -0.043638  0.218103  0.278515
1  10041     M  0.144649  0.132887 -0.023245  0.192882
2  10081     M  0.136536  0.168263  0.244436  0.476266
3  10121     M  0.149041  0.149686  0.147139  0.421094
4  10161     F -0.246005 -0.108670  0.080309 -0.347634
```

```
5  10201     F -0.235311 -0.055924 -0.015538 -0.377672
6  10241     F  0.033224  0.193590  0.103777  0.279773
```

3. Zygotic approach: Selection criteria for zygotes can be obtained using the function `msq.selstrat_z`. Parameter `sub_idz` plays a slightly different role. Here, if `sub_idz` is `None`, the best (based on chosen selection criterion) combinations of top males and females in the Mendelian sampling (co-)variance data frame (`msvmsc` parameter) are outputted. The maximum allowed allocation of males can also be set. However, if `sub_idz` is a data frame with IDs of parent pairs, the parameters `selmale` (selection proportion for males; 0.01 selects 1%, 1 selects all males), `selfm` (selection proportion for females), and `maxmale` (maximum number of allocation allowed per male) can be set to `None`.

```
start = time.time()
sel_z = msq.selstrat_z(selstrat="PBTI", info=data, sub_idz=None,
                       msvmsc=msvmsc_g, throrconst=0.05,
                       selmale=["M", 1], selfm=["F", 1],
                       maxmale=2)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(sel_z)
Time taken:  1.22 secs
    MaleID FemaleID MaleIndex FemaleIndex       fat        pH   protein  \
0    10009    10131         8         130       0.0  0.015273  0.196209
1    10009    10132         8         131       0.0  0.000266  0.009127
2    10046    10133        45         132   0.33206  0.321037  0.168052
3    10046    10134        45         133  0.252743  0.251529  0.151957
4    10031    10135        30         134       0.0  0.000101  0.000019
..     ...      ...       ...         ...       ...       ...       ...
130  10017    10261        16         260  0.116728  0.002163  0.008054
131  10017    10262        16         261  0.127544  0.011316  0.009853
132  10085    10263        84         262  0.136705  0.302792  0.020996
133  10085    10264        84         263  0.037648   0.02904  0.006313
134  10061    10265        60         264   0.02589  0.025851  0.020836

          ABV
0    0.002325
1    0.000001
2    0.516713
3     0.42109
4         0.0
..        ...
130  0.060115
131  0.089903
132  0.254442
133  0.049238
134  0.055669

[135 rows x 8 columns]
```

Same results can be obtained by providing a data frame `sub_idz` with male and female IDs in the first and second columns, provided these individuals are in Mendelian sampling (co-)variance data frame (`msvmsc_g`).

```
df_zyg = sel_z.iloc[0:5, 0:2]        # dataframe containing a subset of mate pairs
start = time.time()
sel_z = msq.selstrat_z(selstrat="PBTI", info=data, sub_idz=df_zyg,
                       msvmsc=msvmsc_g, throrconst=0.05,
                       selmale=None, selfm=None, maxmale=None)
print('Time taken: ', round(time.time() - start, 2), 'secs')
print(sel_z)
Time taken:  0.34 secs
  MaleID FemaleID  MaleIndex  FemaleIndex       fat        pH   protein  \
0  10009    10131          8          130  0.000000  0.015273  0.196209
1  10009    10132          8          131  0.000000  0.000266  0.009127
2  10046    10133         45          132  0.332060  0.321037  0.168052
3  10046    10134         45          133  0.252743  0.251529  0.151957
4  10031    10135         30          134  0.000000  0.000101  0.000019

            ABV
0  2.324804e-03
1  1.146761e-06
2  5.167133e-01
3  4.210896e-01
4  9.598058e-10
```

Derivation of similarity matrices based on Mendelian sampling values

3. Gametic approach: This can be done using the function `msq.simmat_g`. The function returns a named list containing similarity or standardized similarity matrix based on aggregate genotypes for each group. If parameter chrinterest is a string of `'all'` or 'none,' `all` or `none` of the chromosome-wise similarity matrices are saved to file. If `chrinterest` is a list (e.g., [4, 14]), similarity matrices for only chromosomes 4 and 14 will be saved to file. Trait-specific (standardized or) similarity matrices will be saved to file when `save`=True. If `stdsim`=True, a standardized similarity matrix is outputted. If `progress` = True, the progress of the calculations is printed to screen.

```
start = time.time()
sim_g = msq.simmat_g(info=data, covmat=covmatrices, sub_id=df_gam,
                     chrinterest="none", save=False, stdsim=False,
                     progress=True)
print('Time taken: ', round(time.time() - start, 2), 'secs')
sim_g
Processing group  M
Progress: |████████████████████████████████████████████████| 100% Complete
Creating similarity matrix based on aggregate genotype
```

```
Progress: |████████████████████████████████████████████| 100% Complete
Processing group  F
Progress: |████████████████████████████████████████████| 100% Complete
Creating similarity matrix based on aggregate genotype
Progress: |████████████████████████████████████████████| 100% Complete
Time taken:  6.85 secs

{'M': [array([[0.00485757, 0.00267605, 0.00388947, 0.00525877],
         [0.00267605, 0.00504775, 0.00347402, 0.00408954],
         [0.00388947, 0.00347402, 0.00809001, 0.0053334 ],
         [0.00525877, 0.00408954, 0.0053334 , 0.13166252]])],
 'F': [array([[0.00470106, 0.00280465, 0.00311769],
         [0.00280465, 0.00481454, 0.0030983 ],
         [0.00311769, 0.0030983 , 0.00596466]])]}
```

When group-specific maps are used, as seen in the example above, a named list of group-specific similarity matrices is returned (e.g., M and F). Because the individuals are organized into groups, the order of the individuals in each group's similarity matrix is saved in a (csv) file. However, if an average genetic map is used, a similarity matrix for all groups will be returned. In this case, individual orders are not saved to file because they remain unmodified.

4. Zygotic approach: Similarity matrices between Mendelian sampling values of zygotes produced by mate pairs can be derived using the function `msq.simmat_z`. A data frame with IDs of parent pairs must be provided to use this function. For example, if the selection criterion is estimated using the function `msq.selstrat_z`, the IDs in the first and second columns of the data frame may be provided as `sub_idz` parameter.

```
start = time.time()
sim_z = msq.simmat_z(info=data, sub_idz=df_zyg, covmat=covmatrices,
                     chrinterest="none", stdsim=True, save=False,
                     progress=False)
print('Time taken: ', round(time.time() - start, 2), 'secs')
sim_z
Time taken:  6.52 secs

array([[1.        , 0.76503754, 0.2302074 , 0.19034405, 0.48636912],
       [0.76503754, 1.        , 0.31258909, 0.30383955, 0.60487113],
       [0.2302074 , 0.31258909, 1.        , 0.97591852, 0.30035285],
       [0.19034405, 0.30383955, 0.97591852, 1.        , 0.30850437],
       [0.48636912, 0.60487113, 0.30035285, 0.30850437, 1.        ]])
```

Source code for PyMSQ package

```
"""Genetic evaluation of individuals."""
# Import relavent libraries and modules
import os
import sys
# import time
from collections import Counter
from itertools import compress
from numba import njit
import pkg_resources
import numpy as np
import pandas as pd
import scipy.linalg
import scipy.stats


def example_data():
    """Provide data to the package."""
    cwd = os.getcwd()
    stream = pkg_resources.resource_stream(__name__, 'data/chr.txt')
    chrmosomedata = pd.read_table(stream, sep=" ")
    stream = pkg_resources.resource_stream(__name__, 'data/group.txt')
    groupdata = pd.read_table(stream, sep=" ")
    stream = pkg_resources.resource_stream(__name__, 'data/effects.txt')
    markereffdata = pd.read_table(stream, sep=" ")
    stream = pkg_resources.resource_stream(__name__, 'data/phase.txt')
    genodata = pd.read_table(stream, header=None, sep=" ")
    stream = pkg_resources.resource_stream(__name__, 'data/ped.txt')
    ped = pd.read_table(stream, header=None, sep=" ")
    os.chdir(cwd)
    return chrmosomedata, markereffdata, genodata, groupdata, ped

if __name__ == "__main__":
    example_data()


@njit
def fnrep2(gen, aaxx, aaxx1):
    """Code phased genotypes into 1, 2, 3 and 4."""
    qqq = np.empty((int(gen.shape[0]/2), gen.shape[1]), np.int_)
    for i in range(qqq.shape[0]):
        for j in range(qqq.shape[1]):
            if gen[2*i, j] == aaxx and gen[2*i+1, j] == aaxx:
                qqq[i, j] = 1
            elif gen[2*i, j] == aaxx1 and gen[2*i+1, j] == aaxx1:
                qqq[i, j] = 2
            elif gen[2*i, j] == aaxx and gen[2*i+1, j] == aaxx1:
                qqq[i, j] = 3
            else:
                qqq[i, j] = 4
    return qqq


def haptogen(gen, progress=False):
    """Convert haplotypes to coded genotypes."""
    if progress:
        print("Converting phased haplotypes to genotypes")
    if gen.shape[1] == 2:
        gen = np.array(gen.iloc[:, 1])  # del col containing ID
```

```
        # convert string to 2D array of integers
        gen = [list(gen[i].rstrip()) for i in range(gen.shape[0])]
        gen = np.array(gen, int)
        # derives the frequency of alleles to determine the major allele
        allele = np.asarray(np.unique(gen, return_counts=True)).T.astype(int)
        if len(allele[:, 0]) != 2:
            sys.exit("method only supports biallelic markers")
        aaxx = allele[:, 0][np.argmax(allele[:, 1])]  # major allele
        aaasns = np.isin(allele[:, 0], aaxx, invert=True)
        aaxx1 = int(allele[:, 0][aaasns])  # minor allele
        gen = np.array(gen, int)
        gen = fnrep2(gen, aaxx, aaxx1)
    elif gen.shape[1] > 2:
        gen = gen.iloc[:, 1:gen.shape[1]]  # del col containing ID
        # derives the frequency of alleles to determine the major allele
        allele = np.asarray(np.unique(gen, return_counts=True)).T.astype(int)
        if len(allele[:, 0]) != 2:
            sys.exit("method only supports biallelic markers")
        aaxx = allele[:, 0][np.argmax(allele[:, 1])]  # major allele
        aaasns = np.isin(allele[:, 0], aaxx, invert=True)
        aaxx1 = int(allele[:, 0][aaasns])  # minor allele
        gen = np.array(gen, int)
        gen = fnrep2(gen, aaxx, aaxx1)
    return gen


class Datacheck:
    """Check the input data for errors and store relevant info as an object."""
    def __init__(self, gmap, meff, gmat, group, indwt, progress=False):
        """
        Check input data for errors and store relevant info as class object.
        Parameters
        ----------
        gmap : pandas.DataFrame
            Index: RangeIndex
            Columns:
            Name: CHR, dtype: int64; chromosome number
            Name: SNPName, dtype: object; marker name
            Name: Position: dtype: int64; marker position in bp
            Name: group: dtype: float64; marker distance (cM) or reco rates
        meff : pandas.DataFrame
            Index: RangeIndex
            Columns:
            Name: trait names: float64; no. of columns = no of traits
        gmat : pandas.DataFrame
            Index: RangeIndex
            Columns:
            Name: ID, dtype: int64 or str; identification of individuals
            Name: haplotypes, dtype: object; must be biallelic
        group : pandas.DataFrame
            Index: RangeIndex
            Columns:
            Name: group, dtype: object; group code of individuals, e.g., M, F
            Name: ID, dtype: int64 or str; identification of individuals
        indwt : list of index weights for each trait
        progress : bool, optional; print progress of the function if True
        Returns stored input files
        -------
        """
        # check: ensures number of traits match size of index weights
        indwt = np.array(indwt)
```

```
        if (meff.shape[1]-1) != indwt.size:
            sys.exit('no. of index weights do not match marker effects cols')
        # check: ensure individuals' genotypes match group and ID info
        id_indgrp = pd.Series(group.iloc[:, 1]).astype(str)  # no of inds
        if not pd.Series(
                pd.unique(gmat.iloc[:, 0])).astype(str).equals(id_indgrp):
            sys.exit("ID of individuals in group & genotypic data don't match")
        # check: ensure marker names in marker map and effects match
        if not (gmap.iloc[:, 1].astype(str)).equals(meff.iloc[:, 0].astype(str)):
            print("Discrepancies between marker names")
            sys.exit("Check genetic map and marker effects")
        # check: ensure marker or allele sub effect are all numeric
        meff = meff.iloc[:, 1:meff.shape[1]]
        test = meff.apply(
            lambda s: pd.to_numeric(s, errors='coerce').notnull().all())
        if not test.all():
            sys.exit("Marker or allele sub effects contain non-numeric values")
        # check: ensure unique maps match no of groups if map more than 1
        grpg = pd.unique(group.iloc[:, 0])  # groups of individuals
        grp_chrom = gmap.shape[1]-3  # no of unique maps
        gmat = haptogen(gmat, progress)
        if grp_chrom > 1 and grp_chrom != grpg.size:
            sys.exit("no. of unique maps does not match no. of groups")
        # check no of markers in genotype and map and marker effects match
        no_markers = gmap.shape[0]  # no of markers
        if no_markers != gmat.shape[1] or no_markers != meff.shape[0]:
            sys.exit("markers nos in gen, chrm or marker effects don't match")
        # check: ordered marker distance or recombination rates
        for grn in range(grp_chrom):
            for chrm in pd.unique(gmap.iloc[:, 0]):
                mpx = np.array(gmap.iloc[:, 3+grn][gmap.iloc[:, 0] == chrm])
                if not (mpx == np.sort(sorted(mpx))).any():
                    sys.exit(
                        f"Faulty marker map on chr {chrm} for grp {grpg[grn]}")
        if progress:
            print('Data passed the test!')
            print("Number of individuals:  ", len(id_indgrp))
            print("Number of groups:       ", len(grpg), ": ", grpg)
            print("Number of specific maps:", grp_chrom)
            print("Number of chromosomes:  ", len(pd.unique(gmap.iloc[:, 0])))
            print("Total no. markers:      ", no_markers)
            print("Number of trait(s):     ", meff.columns.size)
            print("Trait name(s) and Index weight(s)")
            if meff.columns.size == 1:
                print(meff.columns[0], ": ", indwt[0])
            elif meff.columns.size > 1:
                for i in range(meff.columns.size):
                    print(meff.columns[i], ": ", indwt[i])
        self.gmap = gmap
        self.meff = meff
        self.gmat = gmat
        self.group = group
        self.indwt = indwt


def elem_cor(mylist, mprc, ngp, mposunit, method, chrm):
    """Derive pop cov matrix."""
    if method == 1:  # Bonk et al's approach
        if mposunit in ("cM", "cm", "CM", "Cm"):
            tmp = np.exp(-2*(np.abs(mprc - mprc[:, None])/100))/4
        elif mposunit in ("reco", "RECO"):
```

```
            if mprc[0] != 0:
                sys.exit(f"First value for reco rate on chr {chrm} isn't zero")
            aaa = (1-(2*mprc))/4
            ida = np.arange(aaa.size)
            tmp = aaa[np.abs(ida - ida[:, None])]
    elif method == 2:  # Santos et al's approach
        if mposunit in ("cM", "cm", "CM", "Cm"):
            tmp = (-1*(np.abs(mprc - mprc[:, None])/200))+0.25
            cutoff = (-1*(50/200))+0.25
            tmp = np.where(tmp < cutoff, 0, tmp)
        elif mposunit in ("reco", "RECO"):
            if mprc[0] != 0:
                sys.exit(f"First value for reco rate on chr {chrm} isn't zero")
            aaa = (-1*(mprc/2))+0.25
            ida = np.arange(aaa.size)
            tmp = aaa[np.abs(ida - ida[:, None])]
            cutoff = (-1*(0.5/2))+0.25
            tmp = np.where(tmp < cutoff, 0, tmp)
    # append chromosome-specific covariance matrix to list
    mylist[int(ngp)].append(tmp)
    return mylist


def popcovmat(info, mposunit, method):
    """
    Derive population-specific covariance matrices.
    Parameters
    ----------
    info : class object
        A class object created using the function "datacheck"
    mposunit : string
        A sting with containing "cM" or "reco".
    method : int
        An integer with a value of 1 for Bonk et al.'s approach or
        2 for Santos et al's approach'
    Returns
    -------
    mylist : list
        A list containing group-specific pop covariance matrices for each chr.
    """
    if mposunit not in ("cM", "cm", "CM", "Cm", "reco", "RECO"):
        sys.exit("marker unit should be either cM or reco")
    # unique group name for naming the list if map is more than 1
    probn = pd.unique(info.group.iloc[:, 0].astype(str)).tolist()
    chromos = pd.unique(info.gmap.iloc[:, 0])  # chromosomes
    no_grp = info.gmap.shape[1]-3  # no of maps
    mylist = []  # list stores chromosome-wise covariance matrix
    for ngp in range(no_grp):
        mylist.append([])
        # marker position in cM or recombination rates
        grouprecodist = info.gmap.iloc[:, 3+ngp]
        for chrm in chromos:
            mpo = np.array(grouprecodist[info.gmap.iloc[:, 0] == (chrm)])
            elem_cor(mylist, mpo, ngp, mposunit, method, chrm)
    if no_grp > 1:
        # if map is more than one, name list using group names
        mylist = dict(zip(probn, mylist))
    return mylist


@njit
```

```
def makemems(gmat, meff):
    """Set up family-specific marker effects (Mendelian sampling)."""
    qqq = np.zeros((gmat.shape))
    for i in range(gmat.shape[0]):
        for j in range(gmat.shape[1]):
            if gmat[i, j] == 4:
                qqq[i, j] = meff[j]*-1
            elif gmat[i, j] == 3:
                qqq[i, j] = meff[j]
            else:
                qqq[i, j] = 0
    return qqq


@njit
def makemebv(gmat, meff):
    """Set up family-specific marker effects (GEBV)."""
    qqq = np.zeros((gmat.shape))
    for i in range(gmat.shape[0]):
        for j in range(gmat.shape[1]):
            if gmat[i, j] == 2:
                qqq[i, j] = meff[j]*-1
            elif gmat[i, j] == 1:
                qqq[i, j] = meff[j]
            else:
                qqq[i, j] = 0
    return qqq


def traitspecmatrices(gmat, meff):
    """Store trait-specific matrices in a list."""
    notr = meff.shape[1]  # number of traits
    slist = []  # list stores trait-specific matrices
    slist.append([])
    for i in range(notr):
        # specify data type for numba
        mefff = np.array(meff.iloc[:, i], float)
        matrix_ms = makemems(gmat, mefff)
        slist[0].append(matrix_ms)
    return slist


def namesdf(notr, trait_names):
    """Create names of dataframe columns for Mendelian co(var)."""
    tnn = np.zeros((notr, notr), 'U20')
    tnn = np.chararray(tnn.shape, itemsize=30)
    for i in range(notr):
        for trt in range(notr):
            if i == trt:
                tnn[i, trt] = str(trait_names[i])
            elif i != trt:
                tnn[i, trt] = "{}_{}".format(trait_names[i], trait_names[trt])
    colnam = tnn[np.tril_indices(notr)]
    return colnam


def mrmmult(temp, covmat):
    """Matrix multiplication (MRM' or m'Rm)."""
    return temp @ covmat @ temp.T
```

```
def dgmrm(temp, covmat):
    """Matrix multiplication (MRM') for bigger matrices."""
    temp1111 = scipy.linalg.blas.dgemm(alpha=1.0, a=temp, b=covmat)
    return scipy.linalg.blas.dgemm(alpha=1.0, a=temp1111, b=temp.T)


def progr(itern, total):
    """Print progress of a task."""
    fill, printend, prefix, suffix = '', "\r", 'Progress:', 'Complete'
    deci, length = 0, 50
    percent = ("{0:." + str(deci) + "f}").format(100 * (itern / float(total)))
    filledlen = int(length * itern // total)
    bars = fill * filledlen + '-' * (length - filledlen)
    print(f'\r{prefix} |{bars}| {percent}% {suffix}', end=printend)
    if itern == total:
        print()


def subindcheck(info, sub_id):
    """Check if inds provided in pd.DataFrame (sub_id) are in group data."""
    sub_id = pd.DataFrame(sub_id).reset_index(drop=True)
    if sub_id.shape[1] != 1:
        sys.exit("Individuals' IDs (sub_id) should be provided in one column")
    numbers = info.group.iloc[:, 1].astype(str).tolist()
    sub_id = sub_id.squeeze().astype(str).tolist()
    aaa = [numbers.index(x) if x in numbers else None for x in sub_id]
    aaa = np.array(aaa)
    if len(aaa) != len(sub_id):
        sys.exit("Some individual ID could not be found in group data")
    return aaa


def msvarcov_g_st(info, covmat, sub_id, progress=False):
    """Derive Mendelian sampling co(variance) for single trait."""
    if sub_id is not None:
        aaa = subindcheck(info, sub_id)
        idn = info.group.iloc[aaa, 1].reset_index(drop=True).astype(str)  # ID
        groupsex = info.group.iloc[aaa, 0].reset_index(drop=True).astype(str)
        matsub = info.gmat[aaa, :]
    else:
        idn = info.group.iloc[:, 1].reset_index(drop=True).astype(str)  # ID
        groupsex = info.group.iloc[:, 0].reset_index(drop=True).astype(str)
        matsub = info.gmat
    if (info.gmap.shape[1]-3 == 1 and len(pd.unique(groupsex)) > 1):
        print("The same map will be used for all groups")
    if progress:
        progr(0, matsub.shape[0])  # print progress bar
    snpindexxx = np.arange(start=0, stop=info.gmap.shape[0], step=1)
    notr = info.meff.columns.size
    slist = traitspecmatrices(matsub, info.meff)
    # dataframe to save Mendelian sampling (co)variance and aggregate breeding
    msvmsc = np.empty((matsub.shape[0], 1))
    for i in range(matsub.shape[0]):  # loop over no of individuals
        mscov = np.zeros((notr, notr))  # Mendelian co(var) mat for ind i
        for chrm in pd.unique(info.gmap.iloc[:, 0]):
            # snp index for chromosome chrm
            s_ind = np.array(snpindexxx[info.gmap.iloc[:, 0] == (chrm)])
            # family-specific marker effects for ind i
            temp = np.zeros((notr, len(s_ind)))
            for trt in range(notr):
                temp[trt, :] = slist[0][trt][i, s_ind]
```

```
            if info.gmap.shape[1]-3 == 1:
                mscov = mscov + mrmmult(temp, covmat[0][chrm-1])
            else:
                mscov = mscov + mrmmult(temp, covmat[groupsex[i]][chrm-1])
        msvmsc[i, 0] = mscov
        if progress:
            progr(i + 1, matsub.shape[0])  # print progress bar
    msvmsc = pd.DataFrame(msvmsc)
    msvmsc.columns = info.meff.columns
    msvmsc.insert(0, "ID", idn, True)
    msvmsc.insert(1, "Group", groupsex, True)  # insert group
    return msvmsc


def msvarcov_g_mt(info, covmat, sub_id, progress=False):
    """Derive Mendelian sampling co(variance) for multiple traits."""
    if sub_id is not None:
        aaa = subindcheck(info, sub_id)
        idn = info.group.iloc[aaa, 1].reset_index(drop=True).astype(str)  # ID
        groupsex = info.group.iloc[aaa, 0].reset_index(drop=True).astype(str)
        matsub = info.gmat[aaa, :]
    else:
        idn = info.group.iloc[:, 1].reset_index(drop=True).astype(str)  # ID
        groupsex = info.group.iloc[:, 0].reset_index(drop=True).astype(str)
        matsub = info.gmat
    if (info.gmap.shape[1]-3 == 1 and len(pd.unique(groupsex)) > 1):
        print("The same map will be used for all groups")
    if progress:
        progr(0, matsub.shape[0])  # print progress bar
    snpindexxx = np.arange(start=0, stop=info.gmap.shape[0], step=1)
    notr = info.meff.columns.size
    slist = traitspecmatrices(matsub, info.meff)
    # dataframe to save Mendelian sampling (co)variance and aggregate breeding
    mad = len(np.zeros((notr+1, notr+1))[np.tril_indices(notr+1)])
    msvmsc = np.empty((matsub.shape[0], mad))
    for i in range(matsub.shape[0]):  # loop over no of individuals
        mscov = np.zeros((notr+1, notr+1))  # Mendelian co(var) mat for ind i
        for chrm in pd.unique(info.gmap.iloc[:, 0]):
            # snp index for chromosome chrm
            s_ind = np.array(snpindexxx[info.gmap.iloc[:, 0] == (chrm)])
            # family-specific marker effects for ind i
            temp = np.zeros((notr+1, len(s_ind)))
            for trt in range(notr):
                temp[trt, :] = slist[0][trt][i, s_ind]
                temp[notr, :] = np.matmul(info.indwt.T, temp[0:notr, :])
            if info.gmap.shape[1]-3 == 1:
                mscov = mscov + mrmmult(temp, covmat[0][chrm-1])
            else:
                mscov = mscov + mrmmult(temp, covmat[groupsex[i]][chrm-1])
        msvmsc[i, :] = mscov[np.tril_indices(notr+1)]
        if progress:
            progr(i + 1, matsub.shape[0])  # print progress bar
    msvmsc = pd.DataFrame(msvmsc)
    tnames = np.concatenate((info.meff.columns, "AG"), axis=None)
    colnam = namesdf(notr+1, tnames).decode('utf-8')
    msvmsc.columns = colnam
    msvmsc.insert(0, "ID", idn, True)
    msvmsc.insert(1, "Group", groupsex, True)  # insert group
    return msvmsc
```

```
def msvarcov_g(info, covmat, sub_id, progress=False):
    """
    Derive Mendelian sampling co(variance) and aggregate genotype.

    Parameters
    ----------
    info : class object
        A class object created using the function "datacheck"
    covmat : A list of pop cov matrices created using "popcovmat" function
    sub_id : pandas.DataFrame with one column
        Index: RangeIndex (minimum of 2 rows)
        Containing ID numbers of specific individuals to be evaluated
    progress : bool, optional; print progress of the function if True
    Returns
    -------
    msvmsc : pandas.DataFrame
        containing the Mendelian sampling (co)variance and aggregate genotype
    Note: If sub_id is None, Mendelian (co-)variance will be estimated for
    all individuals. Otherwise, Mendelian (co-)variance will be estimated for
    the individuals in sub_id
    """
    notr = info.meff.columns.size
    if notr == 1:
        msvmsc = msvarcov_g_st(info, covmat, sub_id, progress)
    elif notr > 1:
        msvmsc = msvarcov_g_mt(info, covmat, sub_id, progress)
    return msvmsc


def array2sym(array):
    """Convert array to stdized symm mat, and back to array without diags."""
    dfmsize = array.size
    for notr in range(1, 10000):
        if dfmsize == len(np.zeros((notr, notr))[np.tril_indices(notr)]):
            break
    iii, jjj = np.tril_indices(notr)
    mat = np.empty((notr, notr), float)
    mat[iii, jjj], mat[jjj, iii] = array, array
    mat = np.array(mat)
    mat1 = cov2corr(mat)
    return np.array(mat1[np.tril_indices(notr, k=-1)])


def msvarcov_gcorr(msvmsc):
    """
    Standardize Mendelian sampling co(variance) and aggregate genotype.

    Parameters
    ----------
    msvmsc : pandas.DataFrame
        containing the Mendelian sampling (co)variance and aggregate genotype
        created using msvarcov_g function
    Returns
    -------
    dfcor : pandas.DataFrame
        containing standardized Mendelian sampling (co)variance
    """
    if msvmsc.columns.size == 3:
        sys.exit("Correlation cannot be derived for a single trait")
    dfm = msvmsc.iloc[:, 2:msvmsc.shape[1]]  # exclude ID and group
    dfmsize = dfm.shape[1]
```

```
    # derive number of traits
    for notr in range(1, 10000):
        if dfmsize == len(np.zeros((notr, notr))[np.tril_indices(notr)]):
            break
    # standardize covariance between traits
    dfcor = dfm.apply(array2sym, axis=1)
    # extract column names
    listnames = dfm.columns.tolist()
    cnames = [x for x in listnames if "_" in x]
    # convert pd.series of list to data frame
    dfcor = pd.DataFrame.from_dict(dict(zip(dfcor.index, dfcor.values))).T
    dfcor.columns = cnames
    # insert ID and group info
    dfcor = [pd.DataFrame(msvmsc.iloc[:, 0:2]), dfcor]  # add ID and GRP
    dfcor = pd.concat(dfcor, axis=1)
    return dfcor


def calcgbv(info, sub_id):
    """Calculate breeding values for each trait."""
    if sub_id is not None:
        aaa = subindcheck(info, sub_id)
        idn = info.group.iloc[aaa, 1].reset_index(drop=True).astype(str)  # ID
        groupsex = info.group.iloc[aaa, 0].reset_index(drop=True).astype(str)
        matsub = info.gmat[aaa, :]
    else:
        idn = info.group.iloc[:, 1].reset_index(drop=True).astype(str)  # ID
        groupsex = info.group.iloc[:, 0].reset_index(drop=True).astype(str)
        matsub = info.gmat
    no_individuals = matsub.shape[0]  # Number of individuals
    trait_names = info.meff.columns  # traits names
    notr = trait_names.size  # number of traits
    if notr == 1:
        gbv = np.zeros((no_individuals, notr))
        mefff = np.array(info.meff.iloc[:, 0], float)  # type spec for numba
        matrix_me = makemebv(matsub, mefff)  # fam-spec marker effects BV
        gbv[:, 0] = matrix_me.sum(axis=1)  # sum all effects
        gbv = pd.DataFrame(gbv)
        gbv.columns = trait_names
    elif notr > 1:
        gbv = np.zeros((no_individuals, notr+1))
        for i in range(notr):
            mefff = np.array(info.meff.iloc[:, i], float)  # type spec 4 numba
            matrix_me = makemebv(matsub, mefff)  # fam-spec marker effects BV
            gbv[:, i] = matrix_me.sum(axis=1)  # sum all effects for each trait
            gbv[:, notr] = gbv[:, notr] + info.indwt[i]*gbv[:, i]  # Agg gen
        gbv = pd.DataFrame(gbv)
        colnames = np.concatenate((trait_names, "ABV"), axis=None)
        gbv.columns = colnames
    gbv.insert(0, "ID", idn, True)  # insert ID
    gbv.insert(1, "Group", groupsex, True)  # insert group
    return gbv


def calcprob(info, msvmsc, thresh):
    """Calculate the probability of breeding top individuals."""
    aaa = subindcheck(info, pd.DataFrame(msvmsc.iloc[:, 0]))
    gbvall = calcgbv(info, None)  # calc GEBV for all inds used by thresh
    gbv = gbvall.iloc[aaa, :].reset_index(drop=True)  # GEBV matching msvmsc
    no_individuals = gbv.shape[0]  # Number of individuals
    trait_names = info.meff.columns  # traits names
```

```
    notr = trait_names.size  # number of traits
    if notr == 1:
        probdf = np.zeros((no_individuals, notr))
        ttt = np.quantile(gbvall.iloc[:, (0+2)], q=1-thresh)  # threshold
        probdf[:, 0] = 1 - scipy.stats.norm.cdf(
            ttt, loc=gbv.iloc[:, (0+2)], scale=np.sqrt(msvmsc.iloc[:, 0+2]))
        probdf = pd.DataFrame(probdf)
        probdf.columns = trait_names
    elif notr > 1:
        colnam = np.concatenate((info.meff.columns, "AG"), axis=None)
        colnam = namesdf(notr+1, colnam).decode('utf-8')
        ttt = np.quantile(gbvall.iloc[:, (notr+2)], q=1-thresh)  # threshold
        probdf = np.zeros((no_individuals, notr+1))
        t_ind = np.arange(colnam.shape[0])[np.in1d(colnam, trait_names)]
        for i in range(notr):
            ttt = np.quantile(gbvall.iloc[:, (i+2)], q=1-thresh)  # threshold
            probdf[:, i] = scipy.stats.norm.cdf(
                ttt, loc=gbv.iloc[:, (i+2)], scale=np.sqrt(
                    msvmsc.iloc[:, (t_ind[i])+2]))
            probdf[:, i] = np.nan_to_num(probdf[:, i])  # convert Inf to zero
            probdf[:, i] = 1 - probdf[:, i]
        ttt = np.quantile(gbvall.iloc[:, (notr+2)], q=1-thresh)
        probdf[:, notr] = scipy.stats.norm.cdf(
            ttt, loc=gbv.iloc[:, (notr+2)], scale=np.sqrt(
                msvmsc["AG"]))
        probdf[:, notr] = np.nan_to_num(probdf[:, notr])  # Agg
        probdf[:, notr] = 1 - probdf[:, notr]
        probdf = pd.DataFrame(probdf)  # convert matrix to dataframe
        colnames = np.concatenate((trait_names, "ABV"), axis=None)
        probdf.columns = colnames
    probdf = [pd.DataFrame(gbv.iloc[:, 0:2]), probdf]  # add ID and GRP
    probdf = pd.concat(probdf, axis=1)
    return probdf


def calcindex(info, msvmsc, const):
    """Calculate the index if constant is known."""
    sub_id = pd.DataFrame(msvmsc.iloc[:, 0])
    gbv = calcgbv(info, sub_id)  # calc GEBV
    no_individuals = gbv.shape[0]  # Number of individuals
    trait_names = info.meff.columns  # traits names
    notr = trait_names.size
    if notr == 1:
        indexdf = np.zeros((no_individuals, notr))
        indexdf[:, 0] = (gbv.iloc[:, (0+2)]/2) + np.sqrt(
            msvmsc.iloc[:, 0+2])*const
        indexdf = pd.DataFrame(indexdf)
        indexdf.columns = trait_names
    elif notr > 1:
        colnam = np.concatenate((info.meff.columns, "AG"), axis=None)
        colnam = namesdf(notr+1, colnam).decode('utf-8')
        indexdf = np.zeros((no_individuals, notr+1))
        t_ind = np.arange(colnam.shape[0])[np.in1d(colnam, trait_names)]
        for i in range(notr):
            indexdf[:, i] = (gbv.iloc[:, (i+2)]/2) + np.sqrt(
                msvmsc.iloc[:, (t_ind[i]+2)])*const
        indexdf[:, notr] = (gbv.iloc[:, (notr+2)]/2) + np.sqrt(
            msvmsc["AG"])*const
        indexdf = pd.DataFrame(indexdf)
        colnames = np.concatenate((trait_names, "ABV"), axis=None)
        indexdf.columns = colnames
```

```
    indexdf = [pd.DataFrame(gbv.iloc[:, 0:2]), indexdf]  # add ID and GRP
    indexdf = pd.concat(indexdf, axis=1)
    return indexdf


def selstrat_g(selstrat, info, sub_id, msvmsc, throrconst):
    """
    Calc selection criteria (GEBV, PBTI, or index using gametic approach.

    Parameters
    ----------
    selstrat : str
        A str containing any of GEBV, PBTI or index
    info : class object
        A class object created using the function "datacheck"
    sub_id : pandas.DataFrame with one column
        Index: RangeIndex (minimum of 2 rows)
        Containing ID numbers of specific individuals to be evaluated
    msvmsc : pandas.DataFrame
        DF created using the function "msvarcov_g"
    throrconst : float
        If selstrat is PBTI, a throrconst of value 0.05 sets threshold at
        top 5% of GEBV. If selstrat is index, throrconst is a constant.
        If selstrat is GEBV, throrconst can be any random value.

    Returns
    -------
    data : pandas.DataFrame
        Index: RangeIndex
        Columns:
        ID, Group, trait names and Aggregate Breeding Value (ABV)
    Note: If selstrat is GEBV, None may be used for throrconst and msvmsc.
    If sub_id is None and selstrat is GEBV, GEBVs will be estimated for all
    individuals. However, if selstrat is not GEBV, the chosen selection
    criterion will be estimated for all individuals in msvmsc data frame.
    """
    if selstrat in ("PBTI", "pbti", "index", "INDEX") and msvmsc is None:
        sys.exit("Provide Mendelian (co-)variance dataframe: 'msvmsc'")
    if selstrat in ("PBTI", "pbti", "index", "INDEX") and throrconst is None:
        sys.exit("Provide value for throrconst parameter")
    if selstrat not in ('GEBV', 'gebv', 'PBTI', 'pbti', 'index', 'INDEX'):
        sys.exit("selection strategy should be one of GEBV, PBTI or INDEX")
    if selstrat in ('GEBV', 'gebv'):
        data = calcgbv(info, sub_id)
    elif selstrat in ('PBTI', 'pbti'):
        if throrconst > 1 or throrconst < 0:
            sys.exit("value must be in the range of 0 and 1")
        data = calcprob(info, msvmsc, throrconst)
    elif selstrat in ('index', 'INDEX'):
        data = calcindex(info, msvmsc, throrconst)
    return data


def cov2corr(cov):
    """Convert covariance to correlation matrix."""
    cov = np.asanyarray(cov)
    std_ = np.sqrt(np.diag(cov))
    with np.errstate(invalid='ignore'):
        corr = cov / np.outer(std_, std_)
    return corr
```

```
def aggen(us_ind, no_markers, slst, indwt):
    """Set up additive effects matrix of aggregate genotype."""
    mmfinal = np.empty((len(us_ind), no_markers))
    xxx = 0
    for iii in us_ind:
        tmpmt1 = np.array([slst[0][trt][iii, :] for trt in range(indwt.size)])
        mmfinal[xxx, :] = np.matmul(indwt.transpose(), tmpmt1)
        xxx = xxx + 1
    return mmfinal


def chr_int(xxxxx):
    """Format chromomosome of interest parameter."""
    if 'all' in xxxxx:
        xxxxx = 'all'
    elif 'none' in xxxxx:
        xxxxx = 'none'
    else:
        xxxxx = np.array([int(i) for i in xxxxx])
    return xxxxx


def writechr(covtmpx, chrinterest, chrm, trtnam, probx, stdsim):
    """Write matrices to file."""
    if isinstance(chrinterest, str):
        if chrinterest == 'all':
            chrfile1 = "{}/Sim mat for {} chrm {} grp {}.npy".format(
                os.getcwd(), trtnam, chrm, probx)
            np.save(chrfile1, covtmpx)
    elif chrm in chrinterest:
        chrfile1 = "{}/Sim mat for {} chrm {} grp {}.npy".format(
            os.getcwd(), trtnam, chrm, probx)  # output file
        np.save(chrfile1, covtmpx)
    if stdsim:
        if isinstance(chrinterest, str):
            if chrinterest == 'all':
                chrfilec = "{}/Stdsim mat for {} chrm {} grp {}.npy".format(
                    os.getcwd(), trtnam, chrm, probx)  # output file
                np.save(chrfilec, cov2corr(covtmpx))
        elif chrm in chrinterest:
            chrfilec = "{}/Stdsim mat for {} chrm {} grp {}.npy".format(
                os.getcwd(), trtnam, chrm, probx)  # output file
            np.save(chrfilec, cov2corr(covtmpx))


def writechrunspec(covtmpx, chrinterest, chrm, trtnam, stdsim):
    """Write matrices to file."""
    if isinstance(chrinterest, str):
        if chrinterest == 'all':
            chrfile1 = "{}/Sim mat for {} chrm {}.npy".format(
                os.getcwd(), trtnam, chrm)
            np.save(chrfile1, covtmpx)
    elif chrm in chrinterest:
        chrfile1 = "{}/Sim mat for {} chrm {}.npy".format(
            os.getcwd(), trtnam, chrm)  # output file
        np.save(chrfile1, covtmpx)
    if stdsim:
        if isinstance(chrinterest, str):
            if chrinterest == 'all':
                chrfilec = "{}/Stdsim mat for {} chrm {}.npy".format(
```

```
                    os.getcwd(), trtnam, chrm)  # output file
                np.save(chrfilec, cov2corr(covtmpx))
        elif chrm in chrinterest:
            chrfilec = "{}/Stdsim mat for {} chrm {}.npy".format(
                os.getcwd(), trtnam, chrm)  # output file
            np.save(chrfilec, cov2corr(covtmpx))


def grtonum(numnx):
    """Map chracters to numeric (0-no of groups)."""
    numnx = numnx.reset_index(drop=True)
    probn = pd.unique(numnx).tolist()
    alt_no = np.arange(0, len(probn), 1)
    noli = numnx.tolist()
    numnx = np.array(list(map(dict(zip(probn, alt_no)).get, noli, noli)))
    return numnx, probn


def datret(info, rw_nms, pfnp, us_ind, slist, covmat, cov_indxx, stdsim,
           progress):
    """Return sim mat based on aggregate genotypes."""
    snpindexxxx = np.arange(start=0, stop=info.gmap.shape[0], step=1)
    if info.meff.shape[1] == 1 and not stdsim:
        mat = cov_indxx
    elif info.meff.shape[1] == 1 and stdsim:
        mat = cov2corr(cov_indxx)
    elif info.meff.shape[1] > 1:
        if info.gmap.shape[1]-3 > 1:
            rw_nms = pd.DataFrame(rw_nms)
            rw_nms.to_csv(f"order of inds in mat grp {pfnp}.csv", index=False)
        if progress:
            print('Creating similarity matrix based on aggregate genotype')
            progr(0, max(pd.unique(info.gmap.iloc[:, 0])))
        tmpmt1 = aggen(us_ind, info.gmap.shape[0], slist, info.indwt)
        # stores ABV covariance btw inds
        mat = np.zeros((len(us_ind), len(us_ind)))
        # loop over chromososomes
        for chrm in pd.unique(info.gmap.iloc[:, 0]):
            s_ind = np.array(snpindexxxx[info.gmap.iloc[:, 0] == (chrm)])
            if info.gmap.shape[1]-3 == 1:
                covtmpx = abs(dgmrm(tmpmt1[:, s_ind], covmat[0][chrm-1]))
            else:
                covtmpx = abs(dgmrm(tmpmt1[:, s_ind], covmat[pfnp][chrm-1]))
            mat = mat + covtmpx
            if progress:
                progr(chrm, max(pd.unique(info.gmap.iloc[:, 0])))
        if stdsim:
            mat = cov2corr(mat)
    return mat


def mrmcals(info, us_ind, stdsim, slist, covmat, probn, chrinterest, save,
            progress):
    """Compute similarity matrix for each chromosome."""
    if progress:
        progr(0, info.meff.columns.size)
    for i in range(info.meff.columns.size):
        cov_indxx = np.zeros((len(us_ind), len(us_ind)))
        for chrm in pd.unique(info.gmap.iloc[:, 0]):
            s_ind = np.array(np.arange(0, info.gmap.shape[0], 1
                                       )[info.gmap.iloc[:, 0] == (chrm)])
```

```
            if info.gmap.shape[1]-3 == 1:  # map is 1
                covtmpx = abs(dgmrm(slist[0][i][:, s_ind], covmat[0][chrm-1]))
            else:  # if map is more than 1
                covtmpx = abs(dgmrm(slist[0][i][us_ind[:, None], s_ind],
                                    covmat[probn][chrm-1]))
            cov_indxx = cov_indxx + covtmpx  # sums up chrm-specific sims
            if len(pd.unique(info.group.iloc[:, 0].astype(str))) == 1:
                writechrunspec(covtmpx, chrinterest, chrm,
                               info.meff.columns[i], stdsim)
            else:
                writechr(covtmpx, chrinterest, chrm, info.meff.columns[i],
                         probn, stdsim)  # write sim to file
        if stdsim:
            if save is True:
                if info.gmap.shape[1]-3 == 1:
                    covxfile = "{}/Stdsim mat for {}.npy".format(
                        os.getcwd(), info.meff.columns[i])
                else:
                    covxfile = "{}/Stdsim mat for {} grp {}.npy".format(
                        os.getcwd(), info.meff.columns[i], probn)
                np.save(covxfile, cov2corr(cov_indxx))  # write std sim mats
        else:
            if save is True:
                if info.gmap.shape[1]-3 == 1:
                    covxfile = "{}/Sim mat for {}.npy".format(
                        os.getcwd(), info.meff.columns[i])
                else:
                    covxfile = "{}/Sim mat for {} grp {}.npy".format(
                        os.getcwd(), info.meff.columns[i], probn)
                np.save(covxfile, cov_indxx)  # write sim matrices
        if progress:
            progr(i + 1, info.meff.columns.size)
    return cov_indxx


def simmat_g(info, covmat, sub_id, chrinterest, save=False, stdsim=False,
             progress=False):
    """
    Compute similarity matrices using gametic approach.
    Parameters
    ----------
    info : class object
        A class object created using the function "datacheck"
    covmat : A list of pop cov matrices created using "popcovmat" function
    sub_id : pandas.DataFrame with one column
        Index: RangeIndex (minimum of 2 rows)
        Containing ID numbers of specific individuals to be evaluated
    chrinterest : str or list of int
        list of chromosome numbers of interest or str with "all" or "none"
    save : bool, optional; write trait-specific sim mats to file if true
    stdsim : bool, optional; print write std sim mats to file if true
    progress : bool, optional; print progress of the task if true
    Returns
    -------
    multgrpcov : list containing simimlarity matrices for each group
    """
    if sub_id is None:
        inda = np.arange(0, info.gmat.shape[0], 1)
        sub_id = pd.DataFrame(info.group.iloc[inda, 1])
        aaa = subindcheck(info, sub_id)
    else:
```

```
        aaa = subindcheck(info, sub_id)
    chrinterest = chr_int(chrinterest)
    slist = traitspecmatrices(info.gmat[aaa, :], info.meff)  # trt-spec mat
    grp = info.gmap.shape[1]-3
    if (grp == 1 and len(pd.unique(info.group.iloc[:, 0].astype(str))) > 1):
        print("The same map will be used for all groups")
    numbers, probn = grtonum(info.group.iloc[aaa, 0].astype(str))
    multgrpcov = []
    for gnp in range(grp):
        multgrpcov.append([])
        if grp == 1:
            us_ind = np.arange(start=0, stop=info.gmat[aaa, :].shape[0],
                               step=1)
        else:
            tng = numbers == gnp
            us_ind = np.array(list(compress(np.arange(0, len(tng), 1),
                                            tng))).T
            print("Processing group ", probn[gnp])
        rw_nms = info.group.iloc[aaa, 1].reset_index(drop=True).astype(
            str)[us_ind]
        cov_indxx = mrmcals(info, us_ind, stdsim, slist, covmat, probn[gnp],
                            chrinterest, save, progress)
        multgrpcov[int(gnp)].append(
            datret(info, rw_nms, probn[gnp], us_ind, slist, covmat,
                   cov_indxx, stdsim, progress))
        if len(probn) == 1:
            break
    if grp > 1 and len(probn):
        multgrpcov = dict(zip(probn, multgrpcov))
    return multgrpcov


def submsvmsc(msvmsc, sub_idz):
    """Extract index in msvmsc data frame."""
    sub_idz = pd.DataFrame(sub_idz)
    numbs = msvmsc.iloc[:, 0].astype(str).tolist()
    sub_idz = sub_idz.reset_index(drop=True).squeeze()
    mal = sub_idz.iloc[:, 0].astype(str).tolist()
    fem = sub_idz.iloc[:, 1].astype(str).tolist()
    if sub_idz is not None:
        for i in mal:
            if i not in numbs:
                sys.exit("Individuals are not in msvmsc parameter")
        for i in fem:
            if i not in numbs:
                sys.exit("Individuals are not in msvmsc parameter")
    mal1 = [numbs.index(x) if x in numbs else None for x in mal]
    fem1 = [numbs.index(x) if x in numbs else None for x in fem]
    return mal1, fem1


def pot_parents(info, data, selmale, selfm):
    """Subset individuals of interest."""
    trait_names = info.meff.columns
    if trait_names.size == 1:
        datamale = data[data.iloc[:, 1] == selmale[0]]
        pos = subindcheck(info, pd.DataFrame(datamale.iloc[:, 0]))
        datamale.insert(0, "pos", pos, True)
        no_sire = int(datamale.shape[0] * selmale[1])
        datamale = datamale.sort_values(
            by=[trait_names[0]], ascending=False).iloc[0:no_sire, :]
```

```
        datafemale = data[data.iloc[:, 1] == selfm[0]]
        pos = subindcheck(info, pd.DataFrame(datafemale.iloc[:, 0]))
        datafemale.insert(0, "pos", pos, True)
        no_dam = int(datafemale.shape[0] * selfm[1])
        datafemale = datafemale.sort_values(
            by=[trait_names[0]], ascending=False).iloc[0:no_dam, :]
    elif trait_names.size > 1:
        datamale = data[data.iloc[:, 1] == selmale[0]]
        pos = subindcheck(info, pd.DataFrame(datamale.iloc[:, 0]))
        datamale.insert(0, "pos", pos, True)
        no_sire = int(datamale.shape[0] * selmale[1])
        datamale = datamale.sort_values(
            by=['ABV'], ascending=False).iloc[0:no_sire, :]
        datafemale = data[data.iloc[:, 1] == selfm[0]]
        pos = subindcheck(info, pd.DataFrame(datafemale.iloc[:, 0]))
        datafemale.insert(0, "pos", pos, True)
        no_dam = int(datafemale.shape[0] * selfm[1])
        datafemale = datafemale.sort_values(
            by=['ABV'], ascending=False).iloc[0:no_dam, :]
    matlist = np.array(np.meshgrid(
        datamale.iloc[:, 0], datafemale.iloc[:, 0])).T.reshape(-1, 2)
    ids = np.array(np.meshgrid(
        datamale.iloc[:, 1], datafemale.iloc[:, 1])).T.reshape(-1, 2)
    if trait_names.size == 1:
        matndat = pd.DataFrame(index=range(matlist.shape[0]), columns=range(
            4+trait_names.size))
    else:
        matndat = pd.DataFrame(
            index=range(matlist.shape[0]), columns=range(5+trait_names.size))
    matndat.iloc[:, [0, 1]] = ids
    matndat.iloc[:, [2, 3]] = matlist
    return matndat


def selsgebv(notr, matndat, gbv, maxmale):
    """Calculate breeding values for each trait (zygote)."""
    mal = matndat.iloc[:, 2].tolist()
    fem = matndat.iloc[:, 3].tolist()
    if notr == 1:
        matndat.iloc[:, 4] = (np.array(gbv.iloc[mal, (0+2)]) + np.array(
            gbv.iloc[fem, (0+2)]))/2
    elif notr > 1:
        matndat.iloc[:, 4:(5+notr)] = (np.array(
            gbv.iloc[mal, 2:(notr+3)]) + np.array(gbv.iloc[fem, 2:(notr+3)]))/2
    idfxxx = np.unique(matndat.iloc[:, 3])
    mmat = pd.DataFrame(index=range(len(idfxxx)),
                        columns=range(matndat.shape[1]))
    for mmm in np.arange(0, len(idfxxx), 1):
        axx = matndat.loc[matndat.iloc[:, 3] == idfxxx[mmm]]
        tsire = np.array(axx.iloc[:, 2])
        mmat.iloc[mmm, :] = axx.iloc[np.argmax(
            axx.iloc[:, axx.columns.size-1]), :]
        norepssire = Counter(mmat.iloc[:, 2])
        lents = len(tsire)
        for nrs in range(lents):
            if norepssire[tsire[nrs]] <= maxmale-1:
                mmat.iloc[mmm, :] = np.array(axx[axx.iloc[:, 2] == tsire[nrs]])
                break
    matndat = mmat
    if notr == 1:
        matndat.columns = np.concatenate((
```

```
            ['MaleID', 'FemaleID', 'MaleIndex', 'FemaleIndex'],
            gbv.columns[gbv.columns.size-1]), axis=None)
    else:
        matndat.columns = np.concatenate((
            ['MaleID', 'FemaleID', 'MaleIndex', 'FemaleIndex'],
            gbv.columns[2:gbv.columns.size].tolist()), axis=None)
    return matndat


def selspbtizyg(notr, gbv, matndat, msvmsc, throrconst, maxmale):
    """Calculate prob of breeding top inds (zygote)."""
    mal1, fem1 = submsvmsc(msvmsc, pd.DataFrame(matndat.iloc[:, 0:2]))
    mal = matndat.iloc[:, 2].tolist()
    fem = matndat.iloc[:, 3].tolist()
    if notr == 1:
        matndat.iloc[:, 4] = (np.array(gbv.iloc[mal, (0+2)]) + np.array(
                gbv.iloc[fem, (0+2)]))/2
        ttt = np.quantile(gbv.iloc[:, 0+2], q=1-throrconst)
        msvtemp = np.array(msvmsc.iloc[mal1, 0+2]) + np.array(
            msvmsc.iloc[fem1, 0+2])
        matndat.iloc[:, 4] = 1 - scipy.stats.norm.cdf(
            ttt, loc=matndat.iloc[:, 4], scale=np.sqrt(
                msvtemp))
    elif notr > 1:
        trait_names = gbv.columns[2:2+notr]
        colnam = np.concatenate((trait_names, "AG"), axis=None)
        colnam = namesdf(notr+1, colnam).decode('utf-8')
        t_ind = np.arange(colnam.shape[0])[np.in1d(colnam, trait_names)]
        for i in range(notr):
            matndat.iloc[:, 4+i] = (
                np.array(gbv.iloc[mal, (i+2)]) + np.array(
                    gbv.iloc[fem, (i+2)]))/2
            ttt = np.quantile(gbv.iloc[:, 2+i], q=1-throrconst)
            msvtemp = np.array(msvmsc.iloc[mal1, t_ind[i]+2]) + np.array(
                msvmsc.iloc[fem1, t_ind[i]+2])
            matndat.iloc[:, 4+i] = 1 - scipy.stats.norm.cdf(
                ttt, loc=matndat.iloc[:, 4+i], scale=np.sqrt(msvtemp))
        matndat.iloc[:, 4+notr] = (
            np.array(gbv.iloc[mal, (notr+2)]) + np.array(
                    gbv.iloc[fem, (notr+2)]))/2
        ttt = np.quantile(gbv.iloc[:, 2+notr], q=1-throrconst)
        msvtemp = np.array(msvmsc.loc[mal1, ["AG"]]) + np.array(
            msvmsc.loc[fem1, ["AG"]])
        matndat.iloc[:, 4+notr] = 1 - scipy.stats.norm.cdf(
            ttt, loc=matndat.iloc[:, 4+notr], scale=np.sqrt(msvtemp.ravel()))
    idfxxx = np.unique(matndat.iloc[:, 3])
    mmat = pd.DataFrame(index=range(len(idfxxx)),
                        columns=range(matndat.shape[1]))
    for mmm in np.arange(0, len(idfxxx), 1):
        axx = matndat.loc[matndat.iloc[:, 3] == idfxxx[mmm]]
        tsire = np.array(axx.iloc[:, 2])
        mmat.iloc[mmm, :] = axx.iloc[np.argmax(
            axx.iloc[:, axx.columns.size-1]), :]
        norepssire = Counter(mmat.iloc[:, 2])
        lents = len(tsire)
        for nrs in range(lents):
            if norepssire[tsire[nrs]] <= maxmale-1:
                mmat.iloc[mmm, :] = np.array(axx[axx.iloc[:, 2] == tsire[nrs]])
                break
    matndat = mmat
    if notr == 1:
```

```
        matndat.columns = np.concatenate((
            ['MaleID', 'FemaleID', 'MaleIndex', 'FemaleIndex'],
            gbv.columns[gbv.columns.size-1]), axis=None)
    else:
        matndat.columns = np.concatenate((
            ['MaleID', 'FemaleID', 'MaleIndex', 'FemaleIndex'],
            gbv.columns[2:gbv.columns.size].tolist()), axis=None)
    return matndat


def selsindex(notr, gbv, matndat, msvmsc, throrconst, maxmale):
    """Calculate the index if constant is known (zygote)."""
    mal1, fem1 = submsvmsc(msvmsc, pd.DataFrame(matndat.iloc[:, 0:2]))
    mal = matndat.iloc[:, 2].tolist()
    fem = matndat.iloc[:, 3].tolist()
    if notr == 1:
        matndat.iloc[:, 4] = (np.array(gbv.iloc[mal, (0+2)]) + np.array(
                gbv.iloc[fem, (0+2)]))/2
        msvtemp = np.array(msvmsc.iloc[mal1, 0+2]) + np.array(
                msvmsc.iloc[fem1, 0+2])
        matndat.iloc[:, 4] = matndat.iloc[:, 4] + np.sqrt(msvtemp)*throrconst
    elif notr > 1:
        trait_names = gbv.columns[2:2+notr]
        colnam = np.concatenate((trait_names, "AG"), axis=None)
        colnam = namesdf(notr+1, colnam).decode('utf-8')
        t_ind = np.arange(colnam.shape[0])[np.in1d(colnam, trait_names)]
        for i in range(notr):
            matndat.iloc[:, 4+i] = (
                np.array(gbv.iloc[mal, (i+2)]) + np.array(
                    gbv.iloc[fem, (i+2)]))/2
            msvtemp = np.array(msvmsc.iloc[mal1, t_ind[i]+2]) + np.array(
                msvmsc.iloc[fem1, t_ind[i]+2])
            matndat.iloc[:, 4+i] = matndat.iloc[:, 4+i] + np.sqrt(
                msvtemp)*throrconst
        matndat.iloc[:, 4+notr] = (
             np.array(gbv.iloc[mal, (notr+2)]) + np.array(
                 gbv.iloc[fem, (notr+2)]))/2
        msvtemp = np.array(msvmsc.loc[mal1, ["AG"]]) + np.array(
            msvmsc.loc[fem1, ["AG"]])
        matndat.iloc[:, 4+notr] = matndat.iloc[:, 4+notr] + (
            np.sqrt(msvtemp)*throrconst).ravel()
    idfxxx = np.unique(matndat.iloc[:, 3])
    mmat = pd.DataFrame(index=range(len(idfxxx)),
                        columns=range(matndat.shape[1]))
    for mmm in np.arange(0, len(idfxxx), 1):
        axx = matndat.loc[matndat.iloc[:, 3] == idfxxx[mmm]]
        tsire = np.array(axx.iloc[:, 2])
        mmat.iloc[mmm, :] = axx.iloc[np.argmax(
            axx.iloc[:, axx.columns.size-1]), :]
        norepssire = Counter(mmat.iloc[:, 2])
        lents = len(tsire)
        for nrs in range(lents):
            if norepssire[tsire[nrs]] <= maxmale-1:
                mmat.iloc[mmm, :] = np.array(axx[axx.iloc[:, 2] == tsire[nrs]])
                break
    matndat = pd.DataFrame(mmat)
    if notr == 1:
        matndat.columns = np.concatenate((
            ['MaleID', 'FemaleID', 'MaleIndex', 'FemaleIndex'],
            gbv.columns[gbv.columns.size-1]), axis=None)
    else:
```

```
        matndat.columns = np.concatenate((
            ['MaleID', 'FemaleID', 'MaleIndex', 'FemaleIndex'],
            gbv.columns[2:gbv.columns.size].tolist()), axis=None)
    return matndat


def subindcheckzyg(info, sub_idz):
    """Check sex and if matepairs provided in sub_idz are in group data."""
    numbs = info.group.iloc[:, 1].astype(str).tolist()
    sub_idz = pd.DataFrame(sub_idz).reset_index(drop=True).squeeze()
    mal = sub_idz.iloc[:, 0].astype(str).tolist()
    fem = sub_idz.iloc[:, 1].astype(str).tolist()
    mal1 = [numbs.index(x) if x in numbs else None for x in mal]
    fem1 = [numbs.index(x) if x in numbs else None for x in fem]
    if len(pd.unique(info.group.iloc[mal1, 0])) != 1:
        sys.exit("Group class in sub_idz is not unique to ID of males")
    if len(pd.unique(info.group.iloc[fem1, 0])) != 1:
        sys.exit("Group class in sub_idz is not unique to ID of females")
    idn = sub_idz.reset_index(drop=True)
    mgp = list(set(info.group.iloc[mal1, 0]))
    fgp = list(set(info.group.iloc[fem1, 0]))
    if len(mgp) > 1 or len(fgp) > 1:
        sys.exit("multiple sexes detected in data")
    probn = [mgp[0], fgp[0]]
    return mal1, fem1, idn, probn


def calcgbvzygsub(info, sub_idz):
    """Calc breeding values for matepairs."""
    mal1, fem1, idn, _ = subindcheckzyg(info, sub_idz)
    no_individuals, trait_names = idn.shape[0], info.meff.columns
    notr = trait_names.size
    if notr == 1:
        gbv = np.zeros((no_individuals, notr))
        mefff = np.array(info.meff.iloc[:, 0], float)
        matrix_me1 = makemebv(info.gmat[mal1, :], mefff)
        matrix_me2 = makemebv(info.gmat[fem1, :], mefff)
        gbv[:, 0] = (matrix_me1.sum(axis=1) + matrix_me2.sum(axis=1))/2
        gbv = pd.DataFrame(gbv)
        gbv.columns = trait_names
    elif notr > 1:
        gbv = np.zeros((no_individuals, notr+1))
        for i in range(notr):
            mefff = np.array(info.meff.iloc[:, i], float)
            matrix_me1 = makemebv(info.gmat[mal1, :], mefff)
            matrix_me2 = makemebv(info.gmat[fem1, :], mefff)
            gbv[:, i] = (matrix_me1.sum(axis=1) + matrix_me2.sum(axis=1))/2
            gbv[:, notr] = gbv[:, notr] + info.indwt[i]*gbv[:, i]
        gbv = pd.DataFrame(gbv)
        colnames = np.concatenate((trait_names, "ABV"), axis=None)
        gbv.columns = colnames
    gbv.insert(0, "FemaleIndex", fem1, True)  # insert ID
    gbv.insert(0, "MaleIndex", mal1, True)  # insert ID
    gbv.insert(0, "FemaleID", idn.iloc[:, 1], True)  # insert ID
    gbv.insert(0, "MaleID", idn.iloc[:, 0], True)  # insert ID
    return gbv


def calcprobzygsub(info, msvmsc, thresh, sub_idz):
    """Calculate the probability of breeding top individuals."""
    subindcheckzyg(info, sub_idz)
```

```
    mal1, fem1 = submsvmsc(msvmsc, sub_idz)
    gbv = calcgbvzygsub(info, sub_idz)
    trait_names = info.meff.columns  # traits names
    notr = trait_names.size
    gbvall = calcgbv(info, None)
    if notr == 1:
        probdf = np.zeros((gbv.shape[0], notr))
        ttt = np.quantile(gbvall.iloc[:, (0+2)], q=1-thresh)
        msvmsc111 = np.array(msvmsc.iloc[mal1, (0+2)]) + np.array(
            msvmsc.iloc[fem1, (0+2)])
        probdf[:, 0] = 1 - scipy.stats.norm.cdf(
            ttt, loc=gbv.iloc[:, (0+4)], scale=np.sqrt(msvmsc111))
        probdf = pd.DataFrame(probdf)
        probdf.columns = trait_names
    elif notr > 1:
        colnam = np.concatenate((trait_names, "AG"), axis=None)
        colnam = namesdf(notr+1, colnam).decode('utf-8')
        probdf = np.zeros((gbv.shape[0], notr+1))
        t_ind = np.arange(colnam.shape[0])[np.in1d(colnam, trait_names)]
        for i in range(notr):
            ttt = np.quantile(gbvall.iloc[:, (i+2)], q=1-thresh)
            msvmsc111 = np.array(msvmsc.iloc[mal1, (t_ind[i])+2]) + np.array(
                msvmsc.iloc[fem1, (t_ind[i])+2])
            probdf[:, i] = scipy.stats.norm.cdf(
                ttt, loc=gbv.iloc[:, (i+4)], scale=np.sqrt(msvmsc111))
            probdf[:, i] = np.nan_to_num(probdf[:, i])
            probdf[:, i] = 1 - probdf[:, i]
        ttt = np.quantile(gbvall.iloc[:, (notr+2)], q=1-thresh)
        msvmsc111 = np.array(msvmsc.loc[mal1, ["AG"]]) + np.array(
            msvmsc.loc[fem1, ["AG"]])
        probdf[:, notr] = scipy.stats.norm.cdf(
            ttt, loc=gbv.iloc[:, (notr+4)], scale=np.sqrt(msvmsc111.ravel()))
        probdf[:, notr] = np.nan_to_num(probdf[:, notr])
        probdf[:, notr] = 1 - probdf[:, notr]
        probdf = pd.DataFrame(probdf)  # convert matrix to dataframe
        colnames = np.concatenate((trait_names, "ABV"), axis=None)
        probdf.columns = colnames
    probdf = pd.concat([gbv.iloc[:, 0:4], probdf], axis=1)
    return probdf


def calcindexzygsub(info, msvmsc, const, sub_idz):
    """Calc index matepairs if constant is known."""
    subindcheckzyg(info, sub_idz)
    mal1, fem1 = submsvmsc(msvmsc, sub_idz)
    gbv = calcgbvzygsub(info, sub_idz)
    trait_names = info.meff.columns  # traits names
    notr = trait_names.size
    if notr == 1:
        indexdf = np.zeros((gbv.shape[0], notr))
        msvmsc111 = np.array(msvmsc.iloc[mal1, (0+2)]) + np.array(
            msvmsc.iloc[fem1, (0+2)])
        indexdf[:, 0] = gbv.iloc[:, (0+4)] + np.sqrt(msvmsc111)*const
        indexdf = pd.DataFrame(indexdf)
        indexdf.columns = trait_names
    elif notr > 1:
        colnam = np.concatenate((trait_names, "AG"), axis=None)
        colnam = namesdf(notr+1, colnam).decode('utf-8')
        indexdf = np.zeros((gbv.shape[0], notr+1))
        t_ind = np.arange(colnam.shape[0])[np.in1d(colnam, trait_names)]
        for i in range(notr):
```

```
            msvmsc111 = np.array(msvmsc.iloc[mal1, (t_ind[i])+2]) + np.array(
                msvmsc.iloc[fem1, (t_ind[i])+2])
            indexdf[:, i] = gbv.iloc[:, (i+4)] + np.sqrt(msvmsc111)*const
        msvmsc111 = np.array(msvmsc.loc[mal1, ["AG"]]) + np.array(
            msvmsc.loc[fem1, ["AG"]])
        indexdf[:, notr] = gbv.iloc[:, (notr+4)] + (
            np.sqrt(msvmsc111)*const).ravel()
        indexdf = pd.DataFrame(indexdf)
        colnames = np.concatenate((trait_names, "ABV"), axis=None)
        indexdf.columns = colnames
    indexdf = pd.concat([gbv.iloc[:, 0:4], indexdf], axis=1)  # grp
    return indexdf


def selstrat_z(selstrat, info, sub_idz, msvmsc, throrconst, selmale, selfm,
               maxmale):
    """
    Calculate selection criteria (GEBV, PBTI, or index) for zygotes.

    Parameters
    ----------
    selstrat : str
        A str containing any of GEBV, PBTI or index
    info : class object
        A class object created using the function "datacheck"
    sub_idz : pandas.DataFrame
        Index: RangeIndex (minimum of 2 rows)
        Containing ID numbers of specific individuals to be evaluated.
        The 1st and 2nd columns must be IDS of males and females, respectively.
    msvmsc : pandas.DataFrame
        DF created using the function "msvarcov_g"
    throrconst : float
        If selstrat is PBTI, a throrconst of value 0.05 sets threshold at
        top 5% of GEBV of the population. If selstrat is index, throrconst
        a constant.
    selmale : list
        list of two items. 1st item is the str coding for males as in group
        dataframe. The 2nd item is a float representing x% of males to be used
    selfm : list
        list of two items. 1st item is the str coding for females as in group
        dataframe.The 2nd item is a float representing x% of females to be used
    maxmale : integer
        maximum number of allocations for males
    Returns
    -------
    matndat : pandas.DataFrame
        Index: RangeIndex
        Columns:
        MaleID, FemaleID, MaleIndex, FemaleIndex, trait names and ABV
    Note: If selstrat is GEBV, None may be used for throrconst and msvmsc.
    If sub_idz is None and selstrat is GEBV, GEBVs will be estimated for all
    individuals. However, if selstrat is not GEBV, the chosen selection
    criterion will be estimated for all individuals in msvmsc data frame.
    """
    if len(pd.unique(info.group.iloc[:, 0])) == 1:
        sys.exit("Inds are the same group. Use 'selstrat_g' function")
    if selstrat not in ('gebv', 'GEBV', 'pbti', 'PBTI', 'index', 'INDEX'):
        sys.exit("Options must be one of 'GEBV', PBTI', or 'INDEX'")
    if selstrat in ("PBTI", "pbti", "index", "INDEX") and msvmsc is None:
        sys.exit("Provide Mendelian (co-)variance dataframe: 'msvmsc'")
    if selstrat in ("PBTI", "pbti", "index", "INDEX") and throrconst is None:
```

```
        sys.exit("Provide value for throrconst parameter")
    if sub_idz is None:
        if maxmale is None:
            sys.exit("Provide maximum allocation for males 'maxmale'")
        elif selmale is None:
            sys.exit("Provide value for propoertio of males to be selected")
        elif selfm is None:
            sys.exit("Provide value for propoertio of females to be selected")
    if sub_idz is not None:
        if selstrat in ('gebv', 'GEBV'):
            matndat = calcgbvzygsub(info, sub_idz)
        elif selstrat in ('pbti', 'PBTI'):
            matndat = calcprobzygsub(info, msvmsc, throrconst, sub_idz)
        elif selstrat in ('index', 'INDEX'):
            matndat = calcindexzygsub(info, msvmsc, throrconst, sub_idz)
    else:
        for item in [selmale[0], selfm[0]]:
            if item not in pd.unique(info.group.iloc[:, 0]).tolist():
                sys.exit("Sex name does not match group names")
        if selstrat in ('gebv', 'GEBV'):
            matndat = pot_parents(info, calcgbv(info, None), selmale, selfm)
            matndat = selsgebv(info.meff.columns.size, matndat,
                               calcgbv(info, None), maxmale)
        elif selstrat in ('pbti', 'PBTI'):
            probdf = calcprob(info, msvmsc, throrconst)
            matndat = pot_parents(info, probdf, selmale, selfm)
            matndat = selspbtizyg(info.meff.columns.size, calcgbv(info, None),
                                  matndat, msvmsc, throrconst, maxmale)
        elif selstrat in ('index', 'INDEX'):
            indexdf = calcindex(info, msvmsc, throrconst)
            matndat = pot_parents(info, indexdf, selmale, selfm)
            matndat = selsindex(info.meff.columns.size, calcgbv(info, None),
                                matndat, msvmsc, throrconst, maxmale)
    return matndat


def agggenmsezygsub(no_markers, no_individuals, slist, slist1, indwt):
    """Set up add effects mat of agg gen (zygote) subset."""
    mmfinal = np.empty((no_individuals, no_markers))
    mmfinal1 = np.empty((no_individuals, no_markers))
    for i in range(no_individuals):
        tmpmt1 = np.zeros((indwt.size, no_markers))
        tmpmt2 = np.zeros((indwt.size, no_markers))
        for trt in range(indwt.size):
            tmpmt1[trt, :] = slist[0][trt][i, :]
            tmpmt2[trt, :] = slist1[0][trt][i, :]
        mmfinal[i, :] = np.matmul(indwt.transpose(), tmpmt1)
        mmfinal1[i, :] = np.matmul(indwt.transpose(), tmpmt2)
    return mmfinal, mmfinal1


def writechrzyg(covtmpx, chrinterest, chrm, trtnam, stdsim):
    """Write matrices to file (zygote)."""
    if isinstance(chrinterest, str):
        if chrinterest == 'all':
            chrfile1 = "{}/Sim mat zygotes {} chrm {}.npy".format(
                os.getcwd(), trtnam, chrm)  # output
            np.save(chrfile1, covtmpx)
    elif chrm in chrinterest:
        chrfile1 = "{}/Sim mat zygotes {} chrm {}.npy".format(
            os.getcwd(), trtnam, chrm)  # output file
```

```
            np.save(chrfile1, covtmpx)
        if stdsim:
            if isinstance(chrinterest, str):
                if chrinterest == 'all':
                    chrfilec = "{}/Stdsim zygotes {} chrm {}.npy".format(
                        os.getcwd(), trtnam, chrm)
                    np.save(chrfilec, cov2corr(covtmpx))
            elif chrm in chrinterest:
                chrfilec = "{}/Stdsim mat zygotes {} chrm {}.npy".format(
                    os.getcwd(), trtnam, chrm)
                np.save(chrfilec, cov2corr(covtmpx))


def mrmcalszygsub(info, sub_idz, stdsim, covmat, chrinterest, save, progress):
    """Compute similarity matrix for each chromosome (zygote) for matepairs."""
    if progress:
        progr(0, info.meff.columns.size)
    mal1, fem1, _, probn = subindcheckzyg(info, sub_idz)
    chrinterest = chr_int(chrinterest)
    slist = traitspecmatrices(info.gmat[mal1, :], info.meff)
    slist1 = traitspecmatrices(info.gmat[fem1, :], info.meff)
    for i in range(info.meff.columns.size):
        mat = np.zeros((len(mal1), len(mal1)))
        for chrm in pd.unique(info.gmap.iloc[:, 0]):
            s_ind = np.array(np.arange(0, info.gmap.shape[0], 1
                                       )[info.gmap.iloc[:, 0] == (chrm)])
            if info.gmap.shape[1]-3 == 1:
                covtmpx = abs(
                    dgmrm(slist[0][i][:, s_ind], covmat[0][chrm-1])) + abs(
                        dgmrm(slist1[0][i][:, s_ind], covmat[0][chrm-1]))
            else:
                covtmpx = abs(
                    dgmrm(slist[0][i][:, s_ind], covmat[probn[0]][chrm-1])
                    ) + abs(dgmrm(slist1[0][i][:, s_ind],
                                  covmat[probn[1]][chrm-1]))
            mat = mat + covtmpx  # sums up chrm specific covariances
            writechrzyg(covtmpx, chrinterest, chrm, info.meff.columns[i],
                        stdsim)
        if stdsim:
            if save is True:
                covxfile = "{}/Sim mat zygote {}.npy".format(
                    os.getcwd(), info.meff.columns[i])  # output file
                np.save(covxfile, cov2corr(mat))
        else:
            if save is True:
                covxfile = "{}/Sim mat zygotes {}.npy".format(
                    os.getcwd(), info.meff.columns[i])
                np.save(covxfile, mat)
        if progress:
            progr(i + 1, info.meff.columns.size)
    return mat, probn, np.arange(0, info.gmap.shape[0], 1), slist, slist1


def simmat_z(info, covmat, sub_idz, chrinterest, save=False, stdsim=False,
             progress=False):
    """
    Compute similarity matrices using zygotic approach for specific matepairs.

    Parameters
    ----------
    info : class object
```

```
    A class object created using the function "datacheck"
sub_idz : pandas.DataFrame
    Index: RangeIndex
    Containing ID numbers of specific individuals to be evaluated.
    The 1st and 2nd columns must be IDS of males and females, respectively.
covmat : A list of pop cov matrices created using "popcovmat" function
chrinterest : str or list of int
    list of chromosome numbers of interest or str with "all" or "none"
save : bool, optional; write trait-specific sim mats to file if True
stdsim : bool, optional; print write std sim mats to file if True
progress : bool, optional; print progress of the task if True
Returns
-------
multgrpcov : list containing simimlarity matrices for each group
"""
if len(pd.unique(info.group.iloc[:, 0])) == 1:
    sys.exit("Inds are the same group. Try 'simmat_gsub' function")
_, _, _, probn = subindcheckzyg(info, sub_idz)
if (info.gmap.shape[1]-3 == 1 and len(probn) > 1):
    print("The same map will be used for both sexes")
mat, probn, snpindexxx, slist, slist1 = mrmcalszygsub(
    info, sub_idz, stdsim, covmat, chr_int(chrinterest), save, progress)
if info.meff.columns.size == 1:
    if stdsim:
        mat = cov2corr(mat)
elif info.meff.columns.size > 1:
    if progress:
        print('Creating similarity matrix based on aggregate genotype')
        progr(0, max(pd.unique(info.gmap.iloc[:, 0])))
    tmpmm, tmpmfm = agggenmsezygsub(
        info.gmap.shape[0], sub_idz.shape[0], slist, slist1, info.indwt)
    mat = np.zeros((sub_idz.shape[0], sub_idz.shape[0]))
    # loop over chromososomes
    for chrm in pd.unique(info.gmap.iloc[:, 0]):
        s_ind = np.array(snpindexxx[info.gmap.iloc[:, 0] == (chrm)])
        if info.gmap.shape[1]-3 == 1:
            covtmpx = abs(dgmrm(tmpmm[:, s_ind], covmat[0][chrm-1])) + abs(
                    dgmrm(tmpmfm[:, s_ind], covmat[0][chrm-1]))
        else:
            covtmpx = abs(dgmrm(
                tmpmm[:, s_ind], covmat[probn[0]][chrm-1])) + abs(
                    dgmrm(tmpmfm[:, s_ind], covmat[probn[1]][chrm-1]))
        mat = mat + covtmpx
        if progress:
            progr(chrm, max(pd.unique(info.gmap.iloc[:, 0])))
    if stdsim:
        mat = cov2corr(mat)
return mat
```

ASReml (Gilmour et al., 2021) job control files (.as file) and post-analysis procedure file to calculate functions of variance components (.pin file)

ASReml .as file for investigating MSV-related factors using milk fat yield (FY) as an example:

```
!Workspace 3200 !DEBUG !LOG
Analysis of factors affecting MSV for FY
      isn         !A 71163          # Animal ID
      sex         !A                # Sex
      gebdat      !A                # Birth year
      Homo                          # Number of homozygous loci
      SIRE        !A !P 2782        # Sire in pedigree file
      mother_isn  !A 54136          # Dam in pedigree file
      Selection   !A                # Selected or unselected
      LG                            # Longevity
      FY                            # Fat yield
      MUN                           # Milk urea nitrogen
      PY                            # Protein yield
      AG                            # Aggregate genotype

# =============== pedigree ==================
pedigree.dat !SKIP 1 !ALPHA !MAKE !SORT

# ================= data ====================
MixedModel_data_logmsv.csv !SKIP 1 FCON !DDF 1 !MAXIT 100 !NODISPLAY

# ============= Linear mixed model ===========
FY ~ mu sex gebdat Selected sex.gebdat !r SIRE
```

ASReml .pin file to estimate variance components and heritability

```
F phenvar 1 + 2                   # 3 phenotypic variance
F genvar 1 * 4                    # 4 genetic variance
H herit 4 3                       # 5 heritability
```

ASReml .as file for investigating similarity-related factors using milk fat yield (FY) as an example:

```
!Workspace 3200 !DEBUG !LOG
Analysis of factors affecting similarity for FY
      TL          !A 5              # Time lag
      sex         !A 3              # Sex
      sel         !A 3              # Selection
      anim1       !I 7611           # Animal ID of 1st animal
      anim2       !I 7611           # Animal ID of 2nd animal
      msv1                          # MSV of 1st animal
      msv2                          # MSV of 2nd animal
      mscov                         # log-transformed similarity for FY
      msv1msv2                      # log-transformed msv1 * msv2
      msv1pmsv2                     # log-transformed msv1 + msv2

# ================= data ====================
dat.dat !SKIP 1 FCON !DDF 1 !MAXIT 100 !NODISPLAY

# ============= Linear mixed model ===========
mscov ~ mu TL sex sel msv1 msv2 msv1pmsv2 TL.sex !r anim1 and(anim2,1)
```

ASReml .pin file to estimate variance components and variance due to animals

```
F phenvar 1 + 2                     # 3 phenotypic variance
H animvar 1 3                       # 4 Variance explained by animals
```

LITERATURE CITED

Bonk, S., M. Reichelt, F. Teuscher, D. Segelke, and N. Reinsch. 2016. Mendelian sampling covariability of marker effects and genetic values. *Genet. Sel. Evol.* 48:1–11.

Gilmour, A. R., B. J. Gogel, B. R. Cullis, S. J. Welham, and R. Thompson. 2021. ASReml User Guide Release 4.2 Functional Specification. *VSN Int. Ltd, Hemel Hempstead, HP1 1ES, UK www.vsni.co.uk*.

Hampel, A., F. Teuscher, L. Gomez-Raya, M. Doschoris, and D. Wittenburg. 2018. Estimation of Recombination Rate and Maternal Linkage Disequilibrium in Half-Sibs. *Front. Genet.* 9:186.

Melzer, N., D. Wittenburg, and D. Repsilber. 2013. Integrating Milk Metabolite Profile Information for the Prediction of Traditional Milk Traits Based on SNP Information for Holstein Cows. *PLoS One*. 8:e70256.

Musa, A. A., and N. Reinsch. 2021. A similarity matrix for hedging haplotype diversity among parents in genomic selection. *Submitted*.

Santos, D. J. A., J. B. Cole, T. J. Lawlor, P. M. VanRaden, H. Tonhati, and L. Ma. 2019. Variance of gametic diversity and its application in selection programs. *J. Dairy Sci.* 102:5279–5294.

ACKNOWLEDGEMENT

I've learned a lot in the last five years! I learned a lot about people, cultures, patience, resilience, true happiness, love, science, and, of course, genomic selection. Many people helped make this happen.

I would like to express my sincere gratitude to Professor Dr. Norbert Reinsch, my main supervisor, for introducing me to the fascinating field of genomic selection. I gained a broad knowledge and understanding of livestock genetics and breeding due to his patient guidance, which will allow me to work more freely and independently in the future. Special thanks also go to my minor supervisor, the late PD Dr. Manfred Mayer, who was always patient, approachable, and open to new ideas. Your contributions and memories will be remembered. Dr. Dörte Wittenburg provided constructive criticism, advice, and encouragement, for which I am grateful.

I would also like to thank the entire Institute of Genetics and Biometry, particularly Drs. Nina Melzer, Inga Blunk, Jilun Meng, Michael Doschoris, and Saber Qanbari for their contributions and friendship. Special thanks go to Alexander Hampel, Beate Garske, Jackson Mbuthia, Jan Klosa, Lidia de los Ríos Pérez, Manuela Reichelt, and Ricarda Jahnel, who also helped translate my thesis summary into German. My final thanks go to the administration department, specifically Ms. Michaela Zenk, Anja Sasche, and Birgit Mennenga, for ensuring that my residence permits never expired and assisting me in reuniting with my family.

Professors S.O. Aribido, O.J. Saliu, C.I. Oyewole, T. Oluwagbemi, and S.O. Amhakhian of Kogi State University, Anyigba, and Drs. B.O. Oyewole and A.A. Okpe of the same university also deserve special thanks for their encouragement, guidance, and mentorship.

It is a pleasure to thank my friends Olatunde Olagunju, Mustapha Tanko, and Drs. O.O. Akinsola and A.A. Omede for their unwavering friendship over the years. I especially want to thank Drs. I. Ikoyi and A. Ikoyi for being there for me and assisting me with research articles in journals to which we did not have institutional access.

I am grateful for the financial assistance provided by the Bundesanstalt für Landwirtschaft und Ernährung (BLE) through Grant 281B101516 and the Tertiary Education Trust Fund Nigeria (TETFund).

My heartfelt thanks also go to my family for their unending and unrivaled love, assistance, and support. I will be forever grateful to Professors S.D. Musa and M.S. Audu for their mentorship, inspiration, support, and guidance on this career path. I am thankful to my sisters, Aisha and Saudat, for always being there for me. I'm also grateful to my brothers, particularly Yusuf, Ibrahim, and the late Usman, for their love, sacrifice, and support. It also gives me great pleasure to express my gratitude to my wonderful in-laws, Dr. Sayid Adam, Hajiya Hauwakulu Sayid, Jamila Sayid, and Aminu Umar, for their patience and support.

Please accept my heartfelt gratitude, Fatima, my beautiful wife and best friend! Thank you for believing in me and for your unending support, thoughtfulness, and understanding of the difficulties that come with Ph.D. study. My son, Ayaan, who joined us during my Ph.D. studies, has brought me endless joy and pleasure. I love you both.

Last but not least, I praise and thank Allah SWT for His greatness and for providing me with the strength and courage to finish this thesis, as well as the gift of a good family, friends, health, and life. Alhamdulillah.

I will be eternally grateful to my parents, the late Alhaji Musa Usman and Hajiya Salime Usman, for providing me with the opportunities and experiences that have shaped who I am. They selflessly encouraged me to try new things and follow my own path in life. This journey would not have been possible without them, and I, therefore, dedicate this milestone to them.

Selbständigkeitserklärung

M.Sc. Abdulraheem Arome Musa

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Dissertation mit dem Titel „A similarity matrix and its application in genomic selection for hedging haplotype diversity“ selbständig und ohne unerlaubte Hilfe angefertigt habe und dass ich die Arbeit noch keinem anderen Fachbereich bzw. noch keiner anderen Fakultät vorgelegt habe.

Dummerstorf, den 17.08.2021

M.Sc. Abdulraheem Arome MUSA

Hiermit erkläre ich, dass gegen mich kein strafrechtliches Ermittlungsverfahren schwebt.

Dummerstorf, den 17.08.2021

M.Sc. Abdulraheem Arome Musa

Hiermit erkläre ich, dass die Dissertation nach den Regeln guter wissenschaftlicher Praxis (Standard wissenschaftlichen Arbeitens nach den Empfehlungen der DFG) abgefasst wurde.

Dummerstorf, den 17.08.2021