THE UNIVERSITY
of EDINBURGH

College of Medicine and Veterinary Medicine
The Roslin Institute and Royal (Dick) School of Veterinary Studies
University of Edinburgh

This thesis is presented for the degree of

Doctor of Philosophy

Tools for monitoring and managing sustainable improvement in honeybee populations

submitted by

Laura Strachan
BSc.Hons Biology (Aberystywth University)

Supervisor: **Prof. Gregor Gorjanc**
Co-Supervisor: **Dr. Jana Obšteter**

September 2024

Declaration

I, **Laura Strachan**, declare that this thesis entitled *Tools for monitoring and managing sustainable improvement in honeybee populations* and the data presented in it are original and my own work, unless otherwise specified. This work has not been submitted for any other degree or professional qualification except as specified. I have read and understood The University of Edinburgh guidelines on Plagiarism and declare that this written dissertation is all my own work except where I indicate otherwise by proper use of quotes and references. This thesis is an account of work conducted by me whilst studying for the degree of Doctor of Philosophy at the University of Edinburgh.

Edinburgh, 04.09.2024 Laura Strachan

List of Publications

Publications included in this thesis

Obšteter, J., Strachan, L. K., Bubnič, J., Prešern, J., Gorjanc, G. (2023). SIMplyBee: an R package to simulate honeybee populations and breeding programs. Genetics Selection Evolution, 55, 31. https://doi.org/10.1186/s12711-023-00798-y

Submitted manuscripts included in this thesis

Strachan, L., Bubnič, J., Petersen, G., Gorjanc, G., Obšteter, J.(2024). The principles of expected and realised genetic relatedness among individual honeybees. Submitted June 17th 2024 to the Heredity journal. In Review. BioRxiv preprint: https://doi.org/10.1101/2024.05.26.595938

Conference Attendance

Strachan, L., Bubnič, J., Obšteter, J., Gorjanc, G. SIMplyBee:stochastic simulator of honeybee populations and breeding programmes. In: *World Congress on Genetics Applied to Livestock Production (WCGLAP)*, July 3rd - 9th, 2022. Presentation.

Strachan, L., Bubnič, J., Petersen, G., Gorjanc, G., Obšteter, J. Demonstrating the principles of genetic inheritance in honeybees using SIMplyBee. In: *British Society of Animal Science. (BSAS)*, March 28th-30th, 2023. Presentation.

Strachan, L., Bubnič, J., Petersen, G., Gorjanc, G., Obšteter, J. Demonstrating the principles of genetic inheritance in honeybees using SIMplyBee. In: *The European Federation of Animal Science. (EAAP)*, August 26th - September 1st, 2023. Presentation.

Strachan, L., Bubnič, J., Prešern, J., Gorjanc, G., Obšteter, J. Comparing expected and observed relationships among a sample of Slovenian honey bees. In: *The European Federation of Animal Science. (EAAP)*, September 1st - 5th, 2024. Poster.

Abstract

The Western honeybee is a species of economic importance globally, yet in recent decades it has been experiencing substantial colony losses that result in economic damage and possibly decreased genetic diversity. This situation is highlighting the need for honeybee breeding and conservation programmes. Monitoring genetic variability is an essential component of breeding programmes to ensure genetic gain and managing both global and local genetic diversity. One way to study breeding and conservation programmes is via stochastic simulation. Stochastic simulators are essential for rapid and low-cost testing of breeding decisions and methods, aiding in the optimization of current programmes or the establishment of new ones. These simulations provide valuable insights when they accurately model realistic populations and parameters, allowing for a deeper understanding of factors such as population genetic variability, responses to selection and levels of inbreeding. There was, however, no existing genetics simulator that allows for a detailed simulation of individual honeybee. Therefore, the aim of this thesis was to develop a simulation tool and demonstrate concepts of honeybee relatedness to aid in the sustainable improvement of managed honeybee populations.

Chapter 1 introduces key concepts essential to the understanding of the thesis. It's first focus is on honeybees; describing some of their basic biology, their economic importance to humans, addressing the stressors affecting honeybee populations, and why maintaining genetic diversity within these population is so critical. The chapter then delves into quantitative genetic methods for calculating relatedness, outlining methods such as relatedness coefficients derived from both pedigree and genome-wide data, as well as their pedigree decomposition of genetic values into parent average and Mendelian sampling deviations. The penultimate section describes stochastic simulators and their application in quantitative genetics and selective breeding. Finally, the chapter concludes by outlining the thesis objectives, providing the necessary understanding and context as to why this thesis is relevant.

Chapter 2 of this thesis describes the implementation of SIMplyBee, a holistic simulator of honeybee populations and breeding programs that a small team and myself have developed as an R package. SIMplyBee builds upon the stochastic simulator AlphaSimR that simulates individuals with their corresponding genomes and quantitative genetic values. To enable honeybee-specific simulations, AlphaSimR was extended by developing classes for global simulation parameters, for a honeybee single colony, and multiple colonies. Functions to address major honeybee specificities were

also developed: honeybee genome, haplodiploid inheritance, social organisation, complementary sex determination, polyandry, colony events, and quantitative genetics at the individual- and colony-levels. SIMplyBee provides a research platform for testing breeding and conservation strategies and their effect on future genetic gain and genetic variability.

Chapter 3 demonstrates principles of genetic relatedness in honeybees by analysing the genetic and pedigree information from a SIMplyBee simulation, comparing closed and hybrid populations. Coefficients of relatedness are regularly used to measure genetic similarity within and between populations and their individuals. Although the haplodiploid inheritance of honeybees is well understood, interpreting the various types of relatedness coefficients based on pedigree and genotype data is a challenge for researchers and practitioners in honeybee breeding. I evaluated the relatedness between individuals within a colony, between queens of the same population, and between queens of different populations. The results demonstrated an alignment of mean relatedness using different sources of information when calculated using the same founder population. Identity-by-state (IBS) relatedness varied significantly when calculated relative to different founder populations. While this result is anticipated, it highlights the need for caution when comparing values across studies that use different founder populations with varying allele frequencies. Misestimation of relatedness if interpreted incorrectly can lead to inappropriate breeding and conservation decisions. These decisions may exacerbate inbreeding, reduce a population's genetic diversity, or compromise genetic gain. This emphasises the need for better understanding and standardising methodologies for computing relatedness coefficients, to ensure accurate comparability in relatedness studies.

In **Chapter 4**, I evaluated pedigree reconstruction and patriline determination in honeybees using both real and simulated data. Comparison of simulated data and real data allows researchers to identify weaknesses in established models of genetic inheritance associated methods, or errors in real data, and analyse the accuracy of outputs. In this chapter, real genotype data collected from the mating experiment "BeeConSel" was replicated using SIMplyBee-generated genotypes. Four simulated SNP array sizes were examined to assess the limitations of each software and used the actual pedigree information of the simulation to measure the precision of sire assignments. While gametic information is an important aspect of genetic analysis, gap in the literature was identified for a method to assign parent-of-origin to haplotypes derived from phased genotypes. To address this gap, I developed an R function for this task. Additionally, the information gathered from these tasks was used to develop and evaluate a method for determining the number of patrilines in a honeybee colony. My analysis demonstrated that the effectiveness of different software tools and SNP array sizes in determining paternal assignments and reconstructing pedigrees varies significantly. Real-world data showed variations in software performance, revealing that simulated results often failed to capture the full complexity of actual genetic data. Moreover, the observed deviations in haplotype assignments highlight potential issues with phasing accuracy and the need for better methods and higher-quality data in genetic studies.

Overall, my thesis explores the complexities of honeybee biology and breeding,

through the development and application of the honeybee-specific simulation tool, SIMplyBee. Through the analysis of the simulation outputs using quantitative genetic methods and comparison to real data, the thesis provides insight into optimizing breeding programmes and managing the genetic diversity of honeybee populations.

The thesis concludes with **Chapter 5**, which reiterates the objectives and findings of Chapters 2-4 before discussing the relevance of these findings in relation to current research. This chapter also expands upon the limitations of the work and the implications of the thesis' findings on future work.

Lay Summary

The honeybee is important species for the economy worldwide due to their crop pollination and honey production. However, in recent decades, many colonies have been dying for a variety of reasons, causing financial loss and reducing the variety of distinctive characteristics within populations. This reduction in characteristics highlights the need for programmes to breed and protect honeybees. Beyond their economic value, honeybees also play a crucial role in maintaining ecosystems by pollinating wild plants, supporting biodiversity, and contributing to the stability of natural habitats. Wild bee populations, which face similar challenges, further emphasize the need for conservation efforts. Genetic diversity is a concept that describes the genetic variety of different characteristics in a species, which helps them survive changes like disease or climate temperature changes. Monitoring diversity is crucial for breeding programmes to improve honeybee traits and manage their health both globally and locally. A breeding programme is a planned way of choosing which animals or plants reproduce to improve certain traits over time.
One way to study these breeding programmes is with computer simulations, used to predict what might happen in real life. They are a quick and cost effective way to test breeding ideas, see how animals might respond to changes, how different genetics mix together and ultimately come up with the best breeding plan. Unfortunately, bees have a complicated biology and the current honeybee simulators available are too simplistic and cannot simulate individual honeybee genetics that are needed to create complex genetic breeding programmes.
Therefore, the aim of this thesis was to develop a honeybee-specific simulation tool and use it to conduct experiments to understand how different bees are related by looking at their genetic information. By knowing their relatedness, one can measure the amount of genetic diversity within and between populations. This knowledge will help us improve managed honeybee populations by creating the best breeding programmes.

Chapter 1 lays out the foundation needed to understand the later chapters. It begins by explaining essential information about honeybees, how they live, why they are important to people, the challenges that they face and why keeping track of their genetic diversity is so essential. Next, the chapter describes the different ways to mathematically measure how related honeybees are to each other using both family-history and genetic information. It also explains how genetic data can be used to reconstruct family histories and determine the number of males a queen has mated with, which is crucial for beekeepers. The chapter then talks about how computer

simulations can help scientists test many scenarios and predict what might happen with honeybee populations. Finally, the first chapter finishes by outlining the aims of the thesis and why it's relevant.

Chapter 2 of the thesis describes the development of a honey-bee specific computer simulator, we named SIMplyBee. A small team and I created SIMplyBee using a programming language called R, and built it on top of another simulator named AlphaSimR. AlphaSimR is able to simulate complex breeding programmes but is not able to simulate honeybees due to their complicated biology. To make SIMplyBee work for honeybees, features were added to simulate colonies and apiaries and properties of their reproductive biology. Mathematical equations were also incorporated to assess relationships among individual bees and within colonies as a whole. SIMplyBee serves as a tool for scientists to test honeybee breeding ideas and better understand how to keep the populations strong and able to respond well to changes.

Chapter 3 explores how honeybees are related to each other by studying the DNA and family-history created using a SIMplyBee simulation. DNA is the genetic code passed from parents to offspring, shaping traits like size and disease resistance. Two types of populations were compared: one where the bees only mate within the same group (a closed population) and one where they mix with other kinds of bees (a hybrid population). I used maths equations commonly used to measure how similar the bees are within the same populations or between different populations. Even though scientists have known for a long time how honeybee DNA is passed down from parents to their children, showing this using actual DNA and family history is still a challenge. I looked at how related the bees were within a colony, between queens in the same population, and between queens in different populations. When setting up the simulation, I started by generating DNA for two different types of bees: the British black bee and the Slovenian Carniolan bee. The DNA for all individuals within each type was unrelated, with respect to recent pedigree history. I used these two groups as separate founder populations to begin with and mate throughout the simulation. A founder population is a small group that starts a new population, bringing only the genes of its members into the new group. The results showed that if you start with the same founder population, the average relatedness matched whether pedigree data (family history) or genetic data was used. However, I found that if you compared relatedness calculated using different founder population, the results varied a lot. This is what would be expected, since each founder population of bees has its own collection of DNA but it does show how important it is to be careful when comparing these numbers between studies. Results from different studies could be misleading since they usually start with founder populations. Misinterpreting the calculated relatedness value could lead to bad breeding decisions and harm the bee population. It could make them more inbred, reduce the genetic diversity, and possibly make the bees weaker and unable to survive changes. These results show why it is crucial for scientists to agree on the best ways to measure relatedness so that everyone's results can be compared accurately.

In **Chapter 4**, I looked at how well four different computer softwares could identify the parents of bee offspring using both simulated and real DNA. I also checked if the

data followed the basic DNA inheritance principles or if, for example, the mother was giving more of her genetic information than the father. Comparing simulated data to real data helps researchers spot problems in the breeding methods, find errors in the real data and test how well the softwares work. In this chapter, I used real DNA data from a European mating experiment called "BeeConSel" and created similar DNA data using the SIMplyBee simulator. I didn't have information about the fathers in the real data, but did in the simulated data, so the data was ran through the parent-identifying softwares and pretended I not to know the fathers of the simulated data. This let me look at the simulated data's real fathers and see how accurately the software identified the correct fathers. This would help determine how likely the fathers picked by the software for the real data were to be their real fathers. This method can be used to improve the accuracy of phasing. Phasing involves analysing an individual's DNA to determine which segments come from each parent. Unfortunately, most phasing software doesn't specify which parent is which—mother or father. I developed a mathematical formula to solve this issue and verified it by comparing the inheritance patterns to our expectations. With this approach, one can analyse the DNA inherited from the father in all offspring to estimate how many males mated with the queen. This information is crucial for beekeepers to monitor genetic diversity and prevent inbreeding. Some difference were seen between our simulated data and the real data, which highlighted how tricky it still is to replicate real data with simulations.

Overall, my thesis explores the complex and fascinating world of honeybee biology and breeding. My team and I developed a simulation tool, SIMplyBee, that can simulate individual honeybee's DNA. By analysing these simulations and comparing them to real data, my research offers understanding that can help improve breeding programmes and manage the genetic diversity of honeybee populations.

The thesis ends with Chapter 5, which summarises the goals and key findings from Chapters 2-4. It also discusses how these findings relate to current research, acknowledges the limits of my work and considers the impact of my research on future work.

Acknowledgments

I would like to thank my supervisor **Prof. Gregor Gorjanc** for being endlessly patient and supportive throughout my entire PhD journey. Your guidance and encouragement have not only helped me become a better scientist but also made this whole experience much more enjoyable. I really appreciate all the time and effort you've invested in helping me grow. I'm genuinely grateful for everything you've done, and I'll always remember your support with a lot of appreciation. Thanks for everything!

A huge thank you to my co-supervisor **Dr. Jana Obšteter** for being an absolute legend throughout my PhD journey. From stepping up as my co-supervisor to offering unwavering support, you've been an incredible mentor with endless patience. Your guidance and friendship have truly made a world of difference, and I'm so grateful for everything you've done. Thank you for being so amazing!

I would like to thank **Dr. Jernej Bubnič** for being such a great part of the B-Team. We've both muddled through the maths together, and I've learned so much about bees thanks to you. I really appreciate how you've always been such a fantastic host whenever I visited, making me feel right at home. Your hospitality and warmth have made every visit enjoyable. Thanks a ton for everything!

I want to give a huge thanks to **Dr. Gertje Petersen** for being such an amazing host during my trip to New Zealand. You not only provided me with a place to stay and delicious meals but also put up with all my antics! Your kindness and hospitality made my visit so much fun, and I learned a ton from you. I really appreciate everything you did to make my trip enjoyable and memorable. Thanks for everything!

I would also like to thank the members of **HighlanderLab** for their support and encouragement throughout my PhD.

Finally, I want to give a heartfelt thank you to all my friends and family. You've patiently listened to me rant about bees, endured countless complaints about coding issues, and supported me through the stress of writing. Your unwavering support and understanding have been my rock throughout this journey. I truly couldn't have made it to where I am today without each and every one of you. Thank you so much for being there for me every step of the way!

Funding

I gratefully acknowledge the financial support I received during my time as a PhD student. Many thanks to **EastBio** and **AbacusBio** for funding my doctoral studies. And I thank you to **AbacusBio** for inviting me to visit the New Zealand offices for three months to learn about the business, where I made some great friends and learned a lot.

List of Abbreviations

AA	Epistasis Deviations
BLUP	Best Linear Unbiased Prediction
BV	Breeding Value
CSD	Complementary Sex Determination locus
DCA	Drone congregation area
DNA	Deoxyribonucleic Acid
DPQ	Drone Producing Queen
DD	Dominance Deviations
EBV	Estimated Breeding Value
GRM	Genomic Relationship Matrix
GWAS	Genome-Wide Association Study
IBD	Identity-By-Descent
IBS	Identity-By-State
LD	Linkage Disequilibrium
mtDNA	Mitochondrial Deoxyribonucleic Acid
QTL	Quantitative Trait Loci
SNP	Single Nucleotide Polymorphism

To my family and friends who have survived
countless bee rambles

Contents

Declaration iii

List of Publications v

Abstract vii

Lay Summary xi

Acknowledgments xv

List of Abbreviations xvii

Dedication xix

Contents xxi

List of Figures xxv

List of Tables xxix

1 General introduction 1
- 1.1 SECTION 1: An introduction to the honeybee . . . 1
  - 1.1.1 The evolutionary origins of the honeybee . . . 1
  - 1.1.2 Biological characteristic of honeybees that help maintain genetic diversity . . . 2
  - 1.1.3 Social organisation . . . 3
- 1.2 SECTION 2: The importance of monitoring honeybee genetic diversity 5
  - 1.2.1 An introduction to genetic diversity . . . 5
  - 1.2.2 Challenges facing honeybees that impact their genetic diversity 6
  - 1.2.3 Wild honeybee populations . . . 8
  - 1.2.4 Determining the number of patrilines . . . 8

1.3 SECTION 3: Quantitative genetic methods used to monitor genetic diversity . . . 10
1.3.1 An introduction to relatedness . . . 10
1.3.2 Pedigree-based calculations of relatedness . . . 11
1.3.3 Genomic-based calculations of relatedness . . . 14
1.3.4 Phasing and parent-of-origin assignments . . . 18
1.4 SECTION 4: Simulations . . . 19
1.4.1 The advantages of simulators . . . 19
1.4.2 Deterministic vs Stochastic simulators . . . 19
1.5 SECTION 5: An introduction to the thesis . . . 21

2 SIMplyBee: An R package to simulate honeybee populations and breeding programs 23
2.1 Author contributions . . . 23
2.2 Abstract . . . 24
2.2.1 Background . . . 24
2.2.2 Implementation . . . 24
2.2.3 Results . . . 25
2.2.4 Discussion . . . 25
2.3 Introduction . . . 25
2.4 Implementation . . . 27
2.5 Results . . . 27
2.6 Discussion . . . 43
2.7 Conclusions . . . 46
2.8 Chapter Discussion . . . 49
2.9 Additional Files . . . 52

3 The principles of expected and realised genetic relatedness in honeybees 129
3.1 Author contribution statement . . . 129
3.2 Abstract . . . 130
3.3 Introduction . . . 130
3.4 Materials and Methods . . . 134
3.4.1 Simulation of the honeybee population . . . 134
3.4.2 Calculations of relatedness . . . 137
3.5 Results and Discussion . . . 139
3.5.1 Results Summary . . . 139
3.5.2 Relatedness within a colony . . . 139
3.5.3 Relatedness of queens within the same populations . . . 147
3.5.4 Relatedness of queens between different populations . . . 149
3.6 Implications . . . 150
3.6.1 Imports debate within the beekeeping community . . . 150
3.6.2 Impact of the methodology . . . 151
3.6.3 Simulation limitations . . . 152
3.7 Conclusions . . . 153
3.8 Acknowledgements . . . 153

3.9 Conflict of Interest . . . 154
3.10 Data Archiving . . . 154
3.11 Supplementary Figures . . . 154

4 Optimizing Genotype Phasing and Patriline Determination in Honeybees Through Pedigree Reconstruction 157
4.1 Author contributions . . . 158
4.2 Abstract . . . 158
4.3 Introduction . . . 158
4.4 Materials and Methods . . . 161
4.4.1 Real data . . . 162
4.4.2 Simulated data . . . 162
4.4.3 Paternity assignment . . . 164
4.4.4 Phasing . . . 167
4.4.5 Assigning haplotype origin . . . 167
4.4.6 Evaluating the accuracy of the haplotype assignment function . 168
4.4.7 Evaluating the accuracy of the phasing . . . 169
4.4.8 Determining the number of fathers . . . 169
4.5 Results . . . 171
4.5.1 Results Summary . . . 171
4.5.2 Accuracy of parentage assignment using pedigree reconstruction software . . . 171
4.5.3 Phasing and Haplotype parent-of-origin assignment . . . 174
4.5.4 Determining the number of patrilines . . . 175
4.6 Discussion . . . 178
4.6.1 Pedigree reconstruction . . . 178
4.6.2 Parent-of-origin haplotype assignments . . . 180
4.6.3 Number of patriline determination . . . 183
4.6.4 Implementations . . . 184
4.6.5 Limitations . . . 185
4.7 Future work . . . 186
4.8 Conclusions . . . 187
4.8.1 Supplementary Figures . . . 187

5 General Discussion 189
5.1 Thesis motivation and objectives . . . 189
5.2 Summary of Chapter findings . . . 190
5.2.1 Relevance of the thesis findings . . . 192
5.3 Limitations and Future work . . . 195
5.3.1 Simulation limitation and future plans for SIMplyBee . . . 195
5.4 Conclusion . . . 199

References 201

List of Figures

2.1 A flow chart of initialising a honeybee simulation in SIMplyBee. The first step is simulating a desired number of founder genomes and specifying the global simulation parameters in a new `SimParamBee` object. Next, we create the base virgin queens from the founder genomes. We can simulate any number (`nInd`) of virgin queens with the maximum being the number of simulated founder genomes. We choose one virgin queen as the future queen of the colony (left). On the other side (right), we select one virgin queen to provide drones for the DCA. We could select more virgin queens as future queens to create more colonies, and more virgin queen to contribute to the DCA. We next cross the virgin queen to a sample of drones from the DCA and use it to create a colony. We next build-up a colony, which adds in a desired number of workers and drones. The build-up also results in a productive colony. 28

2.2 A flow chart of the colony swarming event in SIMplyBee. The `swarm()` function returns an R list with two colonies, `swarm` and `remnant`, both of which are non-productive. Parameter `p` represent the proportion of workers that leave with the swarm. After the swarm, the user can `cross()` the virgin queen of the `remnant` colony, or use an already mated queen from another source using `reQueen()`, which mimics the beekeepers' options. Refer to the key in Figure 2.1. 35

2.3 A flow chart of the colony splitting event in SIMplyBee. The `split()` function returns an R list with two colonies, `split` and `remnant`, where the *split* is non-productive and the `remnant` is productive. Parameter `p` represents the proportion of workers that are removed in a split. After the split, the user can `cross()` the virgin queen of the `remnant` colony, or use an already mated queen from another source using `reQueen()`, which mimics the beekeepers' options. Refer to the key in Figure 2.1. . 37

2.4 A flow chart of the colony supersedure event in SIMplyBee. The `supersede()` function returns a queen-less colony with a virgin queen. After a supersedure, a colony remains productive since the colony is still at its full size but a `cross()` is required for a new queen. Refer to the key in Figure 2.1. 38

2.5 A flow chart of the colony collapse event in SIMplyBee. The `collapse` function keeps all the individuals in a colony, but turns on the `collapse` parameter, hence marking the colony as collapsed and all the individual within it as dead. Further simulation with a collapsed colony is not allowed in SIMplyBee. Refer to the key in Figure 2.1 39

3.1 A flow diagram of the simulation, showing the founder population, base population and descending generations with the pathway of matings highlighted with arrows. 1600 *A.m.mellifera* and 800*A.m.carnica* founder haplotypes are used to create three founder populations of unrelated virgin queens, 400 *A.m.carnica* and 400 *A.m.mellifera* and a hybrid MelCross population contain 400 *A.m.mellifera* virgin queens. The *A.m.mellifera* and *A.m.carnica* populations had a closed population breeding system, while the hybrid MelCross population virgin queens were mated with *A.m.carnica* drones. 136

3.2 Expected identity-by-descent (eIBD), realised identity-by-descent (rIBD), and identity-by-state (IBS) relatedness within an A.m. carnica colony in year 1 (left) and year 10 (right) of the simulation. Expected IBD (eIBD) represents pedigree-based relatedness, estimating the probability that two individuals share alleles inherited from a common ancestor. Realised IBD (rIBD) accounts for actual genetic sharing due to recombination and segregation, which may differ from expected values. IBS measures genetic similarity based on observed genotype data, without considering inheritance, and was calculated here using allele frequencies of the founder A.m. carnica queens (SingleAF). The figure shows relatedness between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD). Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons. 140

3.3 Expected and realised identity-by-descent (IBD) and identity-by-state (IBS) relatedness for the complementary sex determination (*CSD*) locus within a *A.m.carnica* colony in year 1 (left) and 10 (right) of the simulation. IBS was calculated using allele frequencies of the founder *A.m.carnica* queens (SingleAF). The figure shows relatedness between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD). Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons. 142

3.4 Comparison of identity-by-state (IBS) relatedness within an *A.m.carnica* colony in year 1 (left) and 10 (right) of the simulation. IBS relatedness was calculated using four different allele frequencies: the colony allele frequency (ColonyAF), the founder population allele frequency of *A.m.carnica* queens (SingleAF), the founder population allele frequency of *A.m.carnica* and *A.m.mellifera* queens (MultiBF), and allele frequency of 0.5 (0.5AF). The figure shows relationships between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD). Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons. . . 144
3.5 Identity-by-state (IBS) relatedness in year 10: a) between queens within two closed populations *A.m.mellifera* and *A.m.carnica* and one hybridising population *MelCross* (top-row); b) the self-relatedness of these queens (bottom-row). IBS relatedness was calculated using allele frequencies of the *A.m.carnica* and *A.m.mellifera* founder queens (MultiAF).148
3.6 Identity-by-state (IBS) relatedness of queens between three different populations in year 10 of the simulation. Comparisons were made between the closed populations (Mel_Car) and between each closed population with the hybrid population (Mel_MelCross and MelCross_Car). IBS relatedness was calculated using allele frequencies of the *A.m.carnica* and *A.m.mellifera* founder queens (MultiAF). 149
3.7 Expected and realised identity-by-descent (IBD) and identity-by-state (IBS) relatedness within an *A.m.carnica* colony in year 1 (left) and 10 (right) of the simulation. IBS was calculated using allele frequencies of the founder *A.m.carnica* queens (SingleAF). The figure shows relatedness between queen-to-workers (QW) and queen-to-drones (QD). Vertical lines at every 0.05 between relatedness values 0.5 and 0.75 were added to facilitate comparisons. 154
3.8 Comparison of identity-by-state (IBS) relatedness within an *A.m.carnica* colony calculated using two different allele frequencies - colony allele frequency (ColonyAF) including drones and colony allele frequency without drones (ColonyNM). The figure shows relatedness between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD) in year 10 of the simulation. Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons. 155

4.1 Number of sires assigned (top) and accuracy (bottom) by four pedigree reconstruction software using simulated data, with and without genotyping error (top row). SNP arrays had: 16 SNPs (1), 160 SNPs (2), 800 SNPs (3) and 1700 SNPs (4). 172
4.2 Number of sires assigned by four pedigree reconstruction software using real data with 1700 SNPs. 173

4.3 Summarised Mendelian sampling deviations of the maternal and paternal haplotypes for each offspring, representing the overall Mendelian sampling values across all SNPs. Four haplotype scenarios are shown: True haplotypes taken directly from the simulation (*True*), estimated haplotypes of the simulated data without genotypic errors (*Est_nGE*), and with genotyping errors (*Est_GE*), and estimated haplotypes of the real data (*Est_Real*). 175
4.4 The determined number of patrilines per colony (Colony Id) in simulated data. We used the estimated haplotypes of the simulated data without genotyping errors (Est_nGE), and the estimated haplotypes of the simulated data with genotyping errors (Est_GE). We also show the actual number of fathers in the simulated data (Known nPatrilines) and number of workers (nWorkers) available per queen used to determine the estimated number of patrilines. Similarity father thresholds of 1.0 and 0.95 were used in all calculations, where haplotypes had to be 100% and 95% identical, respectively, to be categorised as a match. Similarity sister thresholds of 1.0, 0.95, 0.85 and 0.75 were used to determine which worker haplotypes were sisters. 177
4.5 The determined number of patrilines per colony (Colony Id) using the estimated haplotypes of the real dataset. We also show the number of workers (nWorkers) available per queen used to determine the estimated number of patrilines. Similarity father thresholds of 1.0 and 0.95 were used in all calculations, where haplotypes had to be 100% and 95% identical, respectively, to be categorised as a match. Similarity sister thresholds of 1.0, 0.95, 0.85 and 0.75 were used to determine which worker haplotypes were sisters. 178
4.6 The proportion of known patrilines was estimated (top row) and the accuracy of these estimated patrilines (bottom row). We used the estimated haplotypes of the simulated data without genotyping errors (Est_nGE) (left), and the estimated haplotypes of the simulated data with genotyping errors (Est_GE) (right). Similarity father thresholds of 1.0 and 0.95 were used in all calculations, where haplotypes had to be 100% and 95% identical, respectively, to be categorised as a match. Similarity sister thresholds of 1.0, 0.95, 0.85 and 0.75 were used to determine which worker haplotypes were sisters. 187
4.7 Summed Mendelian sampling deviations of the maternal and paternal haplotypes for each offspring, representing the overall Mendelian sampling values across all SNPs. Three haplotype scenarios are shown: True haplotypes taken directly from the simulation (*Real*), phased haplotypes without genotypic errors (*Est_nGE*) and phased haplotypes with genotyping errors (*Est_GE*). *Est_nGE* and *Est_GE* VCF files were converted to PLINK PED format and run through the the haplotype assignment function. 188

List of Tables

3.1 Probability of events during the yearly cycle 137

3.2 Mean and standard deviation of expected identity-by-descent (eIBD) relatedness, realised identity-by-descent (rIBD) relatedness, and identity-by-state (IBS) relatedness between workers by sister type within an A.m. carnica colony in year 1 and year 10 of the simulation. Expected IBD (eIBD) represents the pedigree-based relatedness, reflecting the probability of inheriting alleles from a common ancestor, while realised IBD (rIBD) accounts for actual genetic sharing due to recombination and segregation. IBS measures genetic similarity based on observed genotype data, regardless of ancestry, and was calculated here using allele frequencies of the founder A.m. carnica queens (SingleAF). The mean and standard deviation were computed in R across all worker pairs within the colony to capture the distribution of relatedness values. 141

3.3 Mean and standard deviation of identity-by-state (IBS) relatedness between workers by sister type within a *A.m.carnica* colony in year 1 and year 10 of the simulation. IBS relatedness were calculated using four different allele frequencies: the colony allele frequency (ColonyAF), the founder population allele frequency of *A.m.carnica* queens (SingleAF), the founder population allele frequency of *A.m.carnica* and *A.m.mellifera* queens (MultiAF), and allele frequency of 0.5 (0.5AF). . 145

4.1 Genotyped honeybee samples from the BeeConSel project 162

4.2 SNP array sizes modeled in the simulated data 163

4.3 A table containing the information of four pedigree reconstruction software: Sequoia, AlphaAssign, Colony, and KING 166

4.4 Comparison of offspring haplotypes with parent genotypes. 167

4.5 Run-time, in minutes, of the pedigree reconstruction software COLONY, Sequoia, AlphaAssign, and KING in real data and simulated data, with and without genotyping errors. 174

1 General introduction

1.1 SECTION 1: An introduction to the honeybee

1.1.1 The evolutionary origins of the honeybee

Within the taxonomy family *Apidae*, there are estimated to be up to 20,000 species of bees globally, each with their own diverse morphological and behavioural traits (Michener, 2000). Among these thousands of species, only 10 honeybee species are classified within the genus *Apis*. From these 10, *Apis mellifera* stands out as the most globally recognised and economically significant species, due to its substantial domestication (Engel, 1999; Arias and Sheppard, 2005; Cunha et al., 2021). The Western honeybee, *Apis mellifera*, has a native geographical range spreading throughout Europe, Africa, and the Middle East, with at least 33 identified subspecies classified according to historical patterns of geographical isolation and subsequent adaptions (Ruttner, 1988a; Engel, 1999; Arias and Sheppard, 2005; Ilyasov et al., 2020). *Apis mellifera* is estimated to have genetically split from its closest ancestral relative, *Apis cerana*, between 6-25 million years ago, as it expanded its range to the West (Sheppard and Meixner, 2003). The 33 subspecies, identified using morphometric and genetic analyses, exhibit complex evolutionary histories and are estimated to have diversified around 600 thousand to one million years ago (Ruttner, 1988a; Engel, 1999; Sheppard and Meixner, 2003; Cridland et al., 2017). They have been grouped into seven lineage categories using mitochondrial DNA analysis (mtDNA): A (Tropical Africa), M (Western and Northern Europe), C (Eastern Europe) and O (Turkey and the Middle East), Y (Northeast Africa), L (Egypt) and U (Madagascar) (Ruttner, 1988b; Garnery et al., 1992; Arias and Sheppard, 1996; Miguel et al., 2011; Alburaki et al., 2011; Dogantzis et al., 2021). Of the 33 subspecies, there are a few are particularly important in beekeeping and ecology. The Italian honeybee (*A. m. ligustica*) is the most popular, known for its gentle nature and strong honey production, while the Carniolan honeybee (A. m. carnica) thrives in colder climates with rapid spring buildup (Dall'Olio et al., 2007; Sušnik et al., 2004). The African honeybee (*A. m. scutellata*) is highly defensive but exceptionally resilient, and the Cape honeybee (*A. m. capensis*) has a unique ability for worker bees to reproduce without mating (Anderson, 1963; Cheruiyot et al., 2018). Hybridisation events between these lineages have been documented along natural borders and through human-influenced interactions (Franck et al., 1998; Jensen et al., 2005). Colonisation by Europeans is a good demonstration of human-mediated lineage

hybridisations, as they played a significant role in distributing *Apis mellifera* colonies throughout the globe, thereby facilitating interactions with local populations (Sheppard et al., 1991). In modern times, commercial beekeeping, which includes the import and export of queens and migratory beekeeping, continues to promote hybridisation amongst different lineages.

1.1.2 Biological characteristic of honeybees that help maintain genetic diversity

Despite numerous stressors threatening the genetic diversity of honeybee populations, characteristics of their reproductive biology aid in the maintenance and improvement of their diversity. Key characteristics include their polyandrous mating behaviour and sex determination system.

Polyandry

To manage diversity, honeybees evolved to have polyandrous mating, wherein the queen mates with multiple males to collect a large supply of genetically diverse spermatozoa (Remolina and Hughes, 2008; Woyke, 1963). This mating strategy is considered to be one of the most extreme forms of panmixia in the animal kingdom (De la Rúa et al., 2009). Sperm is stored within the queen's spermatheca, an organ within the abdomen responsible for holding and releasing sperm for egg fertilisation (Poole, 1970; Ruttner and Koeniger, 1971). It is estimated that the queen receives around 6 million spermatozoa into the oviducts from each male during mating (Kerr et al., 1962). However, Oldroyd et al. (1998) suggests that sperm ejection can lead to less than 10% of sperm remaining within the queen after mating, and only approximately 3% of the sperm volume per male actively migrates to the spermatheca for long-term storage and use.

Sex determination

The honeybee's multiple sex-determination systems, encompassing haplodiploidy and the complementary sex determination locus (CSD), provide significant evolutionary advantages by encouraging genetic diversity and in turn improving colony fitness.

Haplodiploidy: Haplodiploidy is a reproductive system in which unfertilized eggs develop into haploid males, while fertilized eggs develop into diploid females. A defining characteristic of the taxonomic order Hymenoptera, to which the honeybee belongs, haplodiploidy was first described in the honeybee by Dzierzon (1845). Males carry only one set of chromosomes inherited from the mother and are thereby only able to transmit non-recombined genetic information from the maternal lineage. In contrast, females inherit both maternal and paternal chromosomes, promoting opportunity for heterozygosity and genetic variation in non-inbred populations (Borgia, 1980; Bull, 1981; Berkelhamer, 1983). In honeybees, the queen can control egg fertilisation using

her spermathecal gland, allowing her to control the sex ratio of caste members, producing drones only when necessary, and ensuring optimal worker production to maintain the hive (Flanders, 1950; Ratnieks and Keller, 1998).

Complementary sex determination locus (CSD): Honeybee sex is also established through the actions of a single CSD. Diploids that are heterozygous at the locus (i.e, A1/A2) develop into females, while haploids are hemizygous (i.e, A1 or A2) and develop into males (Beye et al., 2003). However, diploid individuals homozygous at the CSD locus (i.e, A1/A1) also develop into diploid males, but are killed by colony workers upon detection around the first larval instar (Woyke, 1963). The production of diploid males has been reported to increase colony mortality rates due to the reduced number of viable worker brood, leading to lower colony numbers and strength (Cook and Crozier, 1995). The probability of diploid male occurrence increases in scenarios with high inbreeding, where the number of CSD alleles available reduces significantly and the genetic diversity is low. This may occur naturally during population bottleneck events or artificially within intense selective breeding programmes. The actual number of csd alleles globally is still up for debate, with studies suggesting that the number of circulating alleles in honeybee populations is significantly underestimated. Wang et al. (2012)identified 79 haplotypes across six subspecies, Lechner et al. (2014) estimated between 53 and 87 alleles worldwide, and Zareba et al. (2017) found 121 alleles in two Polish populations. A recent analysis of 652 sequences from GenBank, covering eight subspecies across a wide geographic range, identified 225 distinct csd alleles, emphasizing the high diversity of this locus (Bilodeau and Elsik, 2021).

1.1.3 Social organisation

Honeybee colonies live in large, highly organised groups that are often referred to as super-organisms due to the complex eusocial structure happening between their caste members (Rueppell et al., 2009; Kane and Faux, 2021). A honeybee caste refers to a distinct class within this structure, each characterised by their specific roles and responsibilities. The hive contains three separate castes: the matriarchal queen and her offspring: female workers and male drones. While each of these castes plays a different role within the colony, together they function as a cohesive unit and are a prime example of social evolution (Wilson, 1979).

The queen

The honeybee queen has the pivotal role within the hive, serving as the sole reproductive individual responsible for laying up to 2000 eggs per day and maintaining social order through the release of pheromones. The development of a queen begins when specific genes are triggered upon the consumption of royal jelly, prior to which there are no genetic differences between queen eggs and worker eggs (Patel et al., 2007; Kamakura, 2011). The queen's pheromones, secreted from a variety of exocrine glands, produce both physical and behavioural effects in the workers and drones (Kocher et al., 2009). Their influences range from swarming prevention and defence responses to reproduction inhibition in workers. Additionally, pheromone production will guide workers to

complete numerous hive tasks, such as nest building, brood rearing, thermal regulation and foraging activities (Jaycox, 1970; Mumoki and Crewe, 2021). The lifespan of a queen varies, typically ranging from 1 to 4 years, depending on the environmental conditions and colony health (Bodenheimer, 1937; Seeley, 1978; Remolina and Hughes, 2008). Queen succession in a colony can be triggered by colony events such as swarming, splitting, and supersedures, though does not always require the current queen to die. Swarming is a means of colony reproduction, whereby, under favourable conditions, a colony will outgrow its hive and segregate into two distinct groups. The queen and approximately half of her worker offspring will leave the hive to establish a new colony elsewhere, while the remaining workers rear a new queen and maintain the original colony (Seeley and Buhrman, 1999; Kane and Faux, 2021). Beekeepers can artificially mimic a swarm in a process called splitting, to prevent losing the hive in a natural swarm. It involves splitting the colony in two and allowing either a new queen to emerge naturally in the queen-less remaining group or introducing a foreign queen who will, with proper management, integrate into the colony (Kane and Faux, 2021; Oliver, 2021). In contrast, supersedure is a means of preserving the colony, rather than expanding it, by replacing a queen deemed too old or unfit to control the colony with a new one. Workers begin rearing new virgin queens whilst the old queen is still in the hive, eventually committing matricide to let the new queen take over. In unexpected situations where the queen is killed during her mating flights or by accident, workers rear emergency queens. These emergency queens are often weaker and will often be replaced later by a properly reared queen (Allen, 1965b; Allsopp and Hepburn, 1997).

The workers

The honeybee worker caste, all of which are female, constitute the majority of the colony population, with numbers fluctuating between 20,000 to 70,000 individuals, contingent upon the seasonal period (Bodenheimer, 1937). Over the course of their lifetimes, worker bees will undertake a myriad of tasks within the hive. After emerging from their brood cell, the worker will, for the next 10 days, predominantly serve as a nurse bee, feeding the queen and the brood cells, in addition to offering honey with higher antimicrobial properties to sick bees within the colony (Israili, 2014). Beyond this period, workers engage in diverse roles such as: comb constructing, food storage, guarding of the hive, thermoregulation, honey processing and undertaking activities. Ultimately, 14-20 days after emergence, workers transition into foragers and venture outside the hive, travelling anywhere up to 21km per day within a 5km radius from the hive in order to find pollen, water, propolis and nectar sources (Neukirch, 1982; Döke et al., 2015; Remolina and Hughes, 2008).

The lifespan of worker bees is closely tied to the season of the queen laying their eggs. Summer workers have the shortest lifespan, typically ranging from 15 to 38 days, while autumn workers live approximately 30 to 60 days as the colony readies itself for winter (Remolina and Hughes, 2008; Page and Peng, 2001). Winter bees have certain adaptation genes activated which change their physiology and extend their lifespan even more, spanning 4 to 8 months, to survive the harsh conditions (Page and Peng, 2001; Döke et al., 2015).

The Drones

Drones are the only male bees present within the colony, whose sole purpose is to mate with the virgin queens of foreign colonies, passing on the genetic information of the queen and contributing to the genetic diversity of colonies within the local radius (Kane and Faux, 2021). Whilst having the longest development time, averaging at 24 days, once emerged from their brood cells drones contribute very little to the hive, instead using up colony resources until they reach sexual maturation (Remolina and Hughes, 2008). Once matured, they leave the hive and travel to a local drone congregation area (DCA), a specific location in the environments were drones various colonies gather in large numbers, with the hopes of meeting and mating a virgin queen. Drones will typically live 20-30 days, if mating doesn't occur, and whose presence is seasonal within the hive, with peak drone production occurring in spring. If matings are unsuccessful, a drone will come back to the hive. However, in autumn and winter they are expelled from the hive by their sister-workers to preserve resources for the winter (Kane and Faux, 2021).

1.2 SECTION 2: The importance of monitoring honeybee genetic diversity

1.2.1 An introduction to genetic diversity

Genetic diversity refers to the variation in DNA amongst species, subspecies, or individuals. This diversity is essential for populations to adapt to changing environmental conditions through the process of natural selection and for improving the performance of desired traits through artificial selection. Generally, populations with high levels of genetic diversity exhibit higher occurrences of heterozygosity due to the lack of inbreeding, reducing the chance of inbreeding depression and maintaining fitness (Woolliams and Toro, 2007; Fernández and Bennewitz, 2017). However, it is important to note that heterozygosity can also be influenced by other factors, such as genetic bottlenecks, founder effects, or strong selection pressures, which may reduce genetic diversity even in populations with limited inbreeding.

The monitoring and management of genetic diversity are essential to ensure that populations remain resilient and capable of adapting to survive in changing environments. By tracking genetic diversity over time, we can assess changes in the risk levels of a population or species and steer breeding efforts by monitoring inbreeding rates, identifying individuals with greater genetic diversity or desirable traits, and by analysing hybridisation levels in the population. To improve genetic diversity in a breeding programme, it is important to incorporate strategies such as introducing individuals from different populations, using genetic tools to select mates that maximise diversity, and avoiding excessive inbreeding by regularly assessing genetic relatedness within the breeding population. In breeding programmes, maintaining or improving genetic diversity guarantees long-term selection success, as it enables adaptation and prevents declines in fitness and other economically important traits due to inbreeding depression (Schwartz et al., 2007; Hoban et al., 2021; Kekkonen, 2016).

Prior to the availability of genomic information, breeding programmes depended solely upon pedigree records to monitor population diversity. Whilst this traditional method provides valuable insights, it often lacks precision which in turn can lead to less effective selective breeding. Limitations in historical records can greatly impact the depth of pedigree information, with some programmes only having data for a few generations, which can lead to underestimations of inbreeding. Given that pedigree information is only ancestral, it does not capture the full range of genetic variation in the population. Additionally, it is unable to capture the individual gene flow, the architecture of polygenic traits or the occurrence of mutations (Lathrop et al., 1983; Fernández and Bennewitz, 2017; Cortellari et al., 2022).

The use of genomic information, and possible integration with pedigree records can significantly improve the selection accuracy and the overall effectiveness of breeding programmes (Meuwissen et al., 2011). This allows for more detailed genetic profiles, precise detection of inbreeding, and more effective tracking of genetic diversity. Genomic information provides deeper insights into the genetic makeup of the population, identifying rare alleles and mutations, and enables the estimation of allele effect and their combinations for a range of traits. Additionally, using genomic information we can improve the quality of pedigrees by verifying known pedigrees or identifying previously unknown ones, which is essential for the effective function of a breeding programme (Velazco et al., 2019; Fernández and Bennewitz, 2017; Meuwissen et al., 2011; Misztal et al., 2009). These genetic tools, used alongside bioinformatic and statistical methods, allow for the efficient analysis of the genetic data (Forcina and Leonard, 2020; Carroll et al., 2018).

When comparing to other well-established domesticated species, breeding programmes for honeybees are scarce. As a result, genetic diversity monitoring in the majority of honeybee breeding programmes is relatively underdeveloped. It still relies on phenotypic and pedigree records, as honeybee genomic information is in its early stages, with the whole genome first sequenced as recently as 2006 (Consortium, 2006). However, some programmes exemplify more advanced approaches. BeeBreed.eu; Viert et al. (2021) is an online database, which offers a platform for European beekeepers to submit and compare breeding data, enhancing selection accuracy and genetic management via breeding value estimations. Additionally, the Buckfast Bee programme incorporates both phenotypic and genetic data in their mating selection for improved fitness and productivity (Buckfast.dk; Olszewski et al., 2012).

1.2.2 Challenges facing honeybees that impact their genetic diversity

Despite the increasing quantity of managed hives, there has been a notable rise in reports of colony losses in the US and European countries in recent decades (Potts et al., 2010). Such declines can rapidly lead to declines in genetic diversity within local populations, potentially impacting their chances of survival if not managed. This, in turn, can pose a threat to entire ecosystems and disrupt the honeybee industry.

Multiple stressors have been correlated to the reported increased colony mortality,

including:

- **Agro-chemical exposure:** chemicals used in the agriculture industry, such as pesticides and insecticides, have been shown to negatively impact the nervous system of bees, adversely affecting multiple behaviours linked to memory, foraging navigation and learning which when impacted can be catastrophic to the colony (Thompson et al., 2003; Johnson et al., 2010).
- **Climate change:** severe weather changes including increased seasonal temperatures, decreased rainfall leading to droughts and heavy flooding, can be devastating to bees and cause them to miss the pollination-window, resulting in reduced crop yields and insufficient hive supplies for colonies to survive the next winter (Deutsch et al., 2008; Morris et al., 2008).
- **Competition with invasive species:** beekeeping has employed the common use of imported queens within hives. However, these foreign subspecies introduce competition for resources with natives in addition to the potential introduction of pathogens (Stout and Morales, 2009). These competitions also occur naturally at hybrid zones at geographical borders.
- **Competition with wild non invasive pollinators:** the presence of wild pollinators competing for limited floral resources can create stress for honeybee colonies, particularly in areas where food availability is seasonal or scarce. This resource competition may lead to nutritional deficiencies, weakening colonies and reducing reproductive success, which in turn limits the genetic diversity of future generations (Nielsen et al., 2017; Wojcik et al., 2018).
- **Beekeeping stresses:** if done incorrectly, modern beekeeping practices can cause unnecessary stress on honeybees due to inadequate colony spacing, poor disease or pest control, poor wintering resource management and also the strain of migratory beekeeping can put stress on the colonies and potentially result in losses (Kane and Faux, 2021; Oldroyd et al., 1998).
- **Losses of floral diversity:** the loss of local floral diversity, influenced by factors such as climate change or human deforestation, can result in sub-optimal pollen sources for bees, in turn leading to nutrition deficiencies and compromising their immune systems (Dolezal et al., 2019; St. Clair et al., 2020)
- **Habitat losses:** whilst rates of world deforestation are declining, human activities such as agriculture, urbanization and timber harvesting still contribute greatly to natural habitat losses or fragmentation, leading to declines in biodiversity and limiting bees' access to diverse nutritional sources, which can ultimately lead to colony deaths (Foley et al., 2005; Brown and Paxton, 2009).
- **Microbial pathogens:** pathogens such as viruses, bacteria and fungi, pose a significant threat to at-risk populations, causing a range of symptoms from physical deformities to neurological symptoms and potentially leading to colony collapse if untreated (Ratnieks and Carreck, 2010; Higes et al., 2009).
- **Parasites and pests:** the most well-know honeybee ectoparasite, *Varroa de-*

structor weakens both brood and adult bees by feeding on them and aiding in the transmission of pathogens (Rosenkranz et al., 2010). In addition to such parasites, pests such a mice can invade hives for honey supplies, causing physical damage and colony number reduction, putting the colony at risk of collapse (O'Brien and Marsh, 1990). Another major threat is the Asian hornet (*Vespa velutina*), an invasive predator that actively hunts honeybees, hovering outside hives to catch foragers and causing colony stress, reducing numbers and thereby weakening the colony (Requier et al., 2019b).

1.2.3 Wild honeybee populations

The increased global efforts to protect and enhance honeybee populations, driven by their economic importance in agriculture and food production, could unintentionally lead to ecological consequences for wild pollinators, including wild honeybees. The stressors faced by wild honeybee populations are often more severe due to the lack of management or support, and in some cases, these pressures are worsened by the presence of managed colonies. Managed honeybee populations, typically larger and more widely distributed, can outcompete wild pollinators for local floral resources (Moritz et al., 2007; Roulston and Goodell, 2011). Additionally, they can displace native species through the introduction of pathogens or pests. Furthermore, the overemphasis on conserving managed honeybee populations may inadvertently divert attention and resources away from the broader conservation needs of wild pollinators. By focusing too narrowly on honeybee management, we risk overlooking the complex interactions between different pollinator species and their unique ecological requirements (Panziera et al., 2022; Requier et al., 2019a). Although claims of wild pollinator declines are becoming more prevalent worldwide, the lack of quantitative data makes it difficult to fully assess these trends, rendering such impacts speculative (Aizen and Harder, 2009). Tools like SIMplyBee, the simulator described in this thesis, can support the study of wild honeybee populations by modelling their genetic diversity and population dynamics, thereby informing strategies that help maintain their resilience and promote comprehensive pollinator conservation.

1.2.4 Determining the number of patrilines

Monitoring inheritance patterns is essential for managing genetic diversity in breeding programmes, to prevent inbreeding and keeping precise pedigree records. However, in an uncontrolled setting, polyandrous mating makes it impossible to visually identify paternity and document a full pedigree (Remolina and Hughes, 2008; Woyke, 1962).

In honeybee breeding, maternal lineage is well-documented and often paternal-granddam information is known through drone-producing-queens (DPQs). However, the specific paternal contributors, or patrilines (distinct paternal lineages from individual drones) are usually unknown. The number of patrilines per colony or population is sought after by both beekeepers and researchers to better assess the genetic diversity. Higher patriline numbers indicate greater diversity, making a population more capable to adapting and maintaining high fitness levels when threatened by extreme changes

(Tarpy et al., 2013). Additionally, the number of patrilines can be a strong indicator of queen quality, as queens that mate with a greater number of drones tend to produce more robust, healthier colonies (Carroll et al., 2023; Wossler et al., 2006).

Honeybee breeders employ several strategies to achieve mating control and enhance the accuracy of pedigree records: artificial insemination, geographical isolation, temporal isolation, and biological isolation. Artificial insemination involves the manual fertilisation of virgin queens, offering full control over mating in factors such as the selection of sperm from favoured paternal lineage and the volume of sperm used (Gillard and Oldroyd, 2020). However, this technique remains uncommon practise in honeybee breeding due to its high costs and specialised training it requires. Geographical isolation involved establishing mating stations in remote areas, such as an island, valleys surrounded by mountains, or tundras, whereupon virgin queens and DPQs are relocated for mating. This technique minimizes the risk of unknown and unwanted drones from local colonies mating with the virgin queens (Uzunov et al., 2017). Temporal isolation techniques, such as the Horner system, manipulate both light and temperature to control when selected virgin queens and drones are active and likely to mate (Oxley et al., 2010). Biological isolation focuses on saturating the local area with selected drones to increase the chances of mating with theses preferred drones. Unfortunately, even with these measures in place to control mating, excluding artificial insemination, it is challenging to determine the exact number of patrilines in a colony.

Several methods for determining honeybee patrilines have been developed over past decades. Initial techniques involved direct mating observations, though these could be influenced by human error, and sperm counts which relied on assumption about the number of spermatozoids per drone received during mating (Roberts, 1944; Woyke, 1962). Other early methods consisted of measuring sperm volume stored in the queen's spermatheca relative to the sperm quantity produced by a drone, which assumed all drones produced an equal sperm volume and that sperm from each drone was stored uniformly in the spermatheca (Woyke, 1962). Additionally, methods involved counting genotypic variations in the offspring of queen homozygous at specific genetic markers being studied (Comparini and Biasiolo, 1991). More recent approaches, such as those by Tarpy et al. (2013) and Mroczek et al. (2024), use microsatellite genotypes to estimate patrilines and paternity frequencies. While microsatellites are highly variable due to the presence of many alleles, such studies typically include only 5-10 loci, which limits the overall genetic resolution. This small number of loci can reduce the accuracy of patriline estimates and make them prone to sampling and sequencing errors (Balloux et al., 2000). Newer methods, such as SNP arrays, as used in this thesis, assess thousands of less-variable loci to provide comprehensive and accurate genetic profile (Flanagan and Jones, 2019).

1.3 SECTION 3: Quantitative genetic methods used to monitor genetic diversity

1.3.1 An introduction to relatedness

Genetic information allows breeders to make informed selection decisions based on breeding goals, whether aiming to preserve genetic variability or achieving genetic gain, or both, while identifying high-risk alleles that may threaten a small population's chance of survival.

Breeding programmes often use a range quantitative genetic methods to analyse genetic information, guiding the selection of animals and design mating strategies. Some of these quantitative methods include: Best Linear Unbiased Prediction (BLUP) of Estimated Breeding Values (EBVs), Genome-Wide Association Studies (GWAS), Quantitative Trait Loci (QTL) mapping, and Genotype-by-Environment Interaction Analysis (Muir, 2007; Calus, 2010; Kang et al., 2004; Korte and Farlow, 2013).

One of the cornerstone concepts used in quantitative genetic analysis is the calculation of relatedness. Relatedness stands as a fundamental concept in the field of genetics, encompassing the degree of similarity between the genomes of individuals. This comparison typically entails cross-examining the genomes of one individual with the other, or even across populations, in a pairwise manner (Weir et al., 2006). Measures of relatedness are built upon two notions: Identity-By-Descent (IBD) and Identity-By-State (IBS).
IBD indicates that a pair of individuals inherited DNA segments from a shared ancestor. Depending on the number of generations between the individuals and shared ancestor, the DNA segments can be broken by recombination. Over many generations, recombination can shorten IBD segments such that they might be challenging to identify from observed data.
In comparison, IBS refers to the occurrence of similar or identical DNA sequences between a pair of individuals at specific genomic locations. Unlike IBD, this similarity can happen due to chance mutations or through shared ancestry since IBS focuses on the matching-ness of the DNA segments without considering their ancestral origins. When comparing a locus of a pair of non-inbred individuals, three outcomes of observed IBS can occur: they share two alleles (IBS2), share one allele (IBS1), or share no alleles (IBS0). To demonstrate more clearly, picture an instance where the alleles available at a locus are A and B in diploid individuals at the same locus:

i) If individual i has genotype AA and individual j has genotype BB then the observation is IBS0

ii) If individual i has genotype AA and individual j has genotype AB, the observation is IBS1

iii) If individual i has genotype AA and individual j has genotype AA, the observation is IBS2

If IBS1 or IBS2 occur at a specific locus, there is a chance the DNA segments are also

IBD. It is important to note that IBD cannot occur without IBS (except mutations since the common ancestor), but IBS can occur independently of IBD due to mutation. Over time, genetic relatedness has been defined according to different types of data. The following section will describe the methodology involved in this analysis using either pedigree and genomic data.

1.3.2 Pedigree-based calculations of relatedness

To better understand population genetic diversity, Fisher (1918) recognised that many desirable phenotypic traits do not follow simple Mendelian inheritance, instead being influenced by multiple genes and environmental factors. From this, he introduced a statistical method for analysing the inheritance of traits with continuous variation. Complementary to this work, Wright (1921) formalised the notion of genetic relatedness by developing a method for calculating the coefficients of relationship and inbreeding from a known pedigree.

Wright's work focused only on diploid individuals and additive gene action, where alleles across multiple loci contribute equally to a phenotypic trait in an additive manner. From these assumptions, the additive genetic value of an individual i (a_i) is the sum is the sum of values of its two genomes: $a_i = a_{i,1} + a_{i,2}$.

Wright's coefficient of relationship:

From this, Wright (1921) defined the coefficient of relationship as a measure of the genetic relatedness between two individuals, based on the correlation between their additive genetic values. This formula expresses the degree to which the additive genetic values of individuals i and j are correlated, taking into account how much of their genetic variance is shared:

$$R_{i,j} = \mathrm{Cor}(a_i, a_j) = \frac{\mathrm{Cov}(a_i, a_j)}{\sqrt{\mathrm{Var}(a_i) \cdot \mathrm{Var}(a_j)}}.$$

The coefficient of relationship, $R_{i,j}$, has a range of -1 to +1, where a value of 1 indicates that individuals i and j are genetically identical (such as in identical twins or clones), 0 indicates absolutely no genetic correlation (such as when comparing unrelated individuals according to pedigree), and negative values do not arise in pedigree analysis. However, negative values of the coefficient of relationship can arise when using genomic data, particularly in the context of kinship estimation. This can occur due to the complexities of allele sharing and genetic variance across populations. For example, in some cases of distant or outbred populations, individuals may show negative kinship values when their alleles are not co-inherited in a manner expected by traditional pedigree-based methods (Goudet et al., 2018).

In the context of additive genetic relationships between individuals i and j ($R_{i,j}$), the formula represents the correlation between their additive genetic values. This formula is valid assuming no inbreeding. In the presence of inbreeding, the genetic correlation

between individuals can increase due to shared ancestry, and this needs to be accounted for separately, as the formula does not capture the additional effects of inbreeding.

Wright's Inbreeding coefficient:

Wright (1921) defined the coefficient of inbreeding as the correlation between the additive genetic values of the two alleles of an individual, denoted as: $F_i = Cor\,(a_{i,1}, a_{i,2})$.

Coefficient F_i measures the probability that the two possible alleles ($a_{i,1}$ and $a_{i,2}$) at a given locus in individual i are IBD (Malécot, 1948). F_i also has a range of -1 to 1. Negative values indicate outbreeding, occurring when individuals from different populations mate, leading to a level of heterozygosity that is lower than the population average. Such negative values typically arise from genomic-based estimates, as pedigree-based calculations assume a starting point of zero inbreeding and do not account for excess heterozygosity. A coefficient value of 0 indicates no inbreeding. If F_i has a value between 0 and 1 ($0 < F_i < 1$), there is a degree of relatedness between the alleles, suggesting that parents of individual i share a common ancestor. A value of 1 indicates total inbreeding, where both alleles are IBD or both incoming chromosomes have all loci IBD. This situation can occur in individuals with highly related parents, such as repeated sibling matings for example. Additionally, while F_i is generally defined for alleles, it can also be applied to portions of the genome or the entire genome, reflecting the overall relatedness across genomic regions.

Development of the coefficients:

Cotterman (1940) and Malécot (1948) built upon Wright's work, demonstrating that the relatedness and inbreeding coefficients could be used to measure the expected IBD of individuals' genomes within a pedigree, relative to the pedigree founders. The pedigree founders are assumed to be not inbred and unrelated to each other, which forms the basis for these calculations. In addition, observing individuals' genomes directly enables measuring of IBS, providing a deeper understanding of the genetic similarities and differences. IBD is calculated relative to a reference population where all alleles are considered distinct unless they share identity by descent (Wang, 2016). Moreover, negative estimates for kinship and inbreeding coefficients can arise under the IBD framework. These values are relative to a reference population, which may differ from the theoretical pedigree founders. Consequently, these values no longer represent absolute IBD probabilities but instead reflect differences in IBD probabilities, which can result in negative values (Goudet et al., 2018).

In his book, Malécot (1948) also introduced a kinship coefficient, $K_{i,j}$, that measures the similarity between genomes in individuals i and j. In two diploid individuals $K_{i,j}$ can be calculated as:

$$K_{i,j} = \frac{1}{4}\left(K_{(i,1)(j,1)} + K_{(i,1)(j,2)} + K_{(i,2)(j,1)} + K_{(i,2)(j,2)},\right).$$

This kinship coefficient links back to Wright's inbreeding coefficient F_i, where F_i of an offspring corresponds to the kinship coefficient between its parents:

$F_i = K_{(i,1)(i,2)}$ and $R_{i,j}$ can be denoted as:

$$R_{i,j} = \frac{2K_{i,j}}{\sqrt{(1+F_i)(1+F_j)}}.$$

The numerator relationship matrix:

Emik and Terrill (1949) developed a simple algorithm to calculate relatedness coefficients between all pairs of individuals within a pedigree. This algorithm takes a covariance coefficient matrix $\mathbf{A}$ - an $n \times n$ matrix, where n is the number of individuals - and sets it to an identity matrix between the pedigree founders. The rest of the diagonal elements in the matrix, where individuals are compared to themselves, are calculated as:

$$\mathbf{A}_{i,i} = \frac{Var\left(a_i\right)}{\sigma_a^2} = 1 + F_i,$$

with values in the range of 1 to 2 since F_i is the correlation coefficient.

Off-diagonal elements of the matrix, where individuals i and j are compared, are defined as:

$$\begin{aligned}\mathbf{A}_{i,j} &= \frac{\mathrm{Cov}(a_i, a_j)}{\sigma_a^2} \\ &= \frac{1}{2}\left(\mathrm{Cor}(a_{i,1}, a_{j,1}) + \mathrm{Cor}(a_{i,1}, a_{j,2}) + \mathrm{Cor}(a_{i,2}, a_{j,1}) + \mathrm{Cor}(a_{i,2}, a_{j,2})\right).\end{aligned}$$

In the first part of this equation $Cov\left(a_i, a_j\right)$ measures the covariance between additive genetic values a_i and a_j, and is normalised by the variance of additive genetic values in the founder population σ_a^2. Next, we have the decomposition of the covariance (Cov) using correlations (Cor) between the specific alleles. Correlation coefficients range between -1 and 1, so the sum of these coefficients can range from -4 to 4, leading to a theoretical range of -2 to 2 when multiplied by $\frac{1}{2}$. However, in practical pedigree analysis, we typically observe values between 0 and 2, as the correlations between alleles, generally positive due to shared ancestry, result in a sum that does not fall below zero. This is due to the derivation of the $\mathbf{A}$ matrix assuming that alleles are inherited through common ancestry, leading to positive correlations, and no negative relatedness is accounted for in this framework.

Off-diagonal elements are calculated as:

$$\mathbf{A}_{i,j} = \frac{1}{2}\left(\mathbf{A}_{i,m(j)} + \mathbf{A}_{i,f(j)}\right).$$

Here we calculate the average of the genetic covariance coefficients $\mathbf{A}_{i,m(j)}$ and $\mathbf{A}_{i,f(j)}$ for individual i with their maternal $m(j)$ and paternal $f(j)$ ancestors of individual j. This calculation is quantifying if there are contributions of genetic material from the ancestors of j to i. By completing the matrix $\mathbf{A}$ one can calculate Wright's coefficients F_i and $R_{i,j}$. F_i is estimated as half the off-diagonal element between the parents of

i: $\frac{1}{2}\mathbf{A}_{m(i),f(i)}$. Henderson (1976) emphasized the practical application of matrix **A** in animal breeding and naming it the numerator relationship matrix. Nowadays, it's more common to use just the elements of the numerator relationship matrix rather than Wright's coefficients of relatedness.

The S matrix:

Grossman and Fernando (1989) and Fernando and Grossman (1990) developed a pedigree-based relationship matrix, denoted as **S**, tailored for the X chromosome. This matrix addresses haplodiploid inheritance, making it well-suited for honeybees. The **S** matrix represents the relatedness between individuals where males are hemizygous (X) and females are diploid (XX). Elements of the matrix reflect the covariance structure of X-linked genetic traits, based on the relationship between the individuals. For instance, in a diploid-diploid system, the covariance between a diploid mother to her diploid offspring, where she passes on an X chromosome would be:

$$Cov(a_{X,mother}, a_{X,offspring}) = \frac{1}{2}\sigma^2_X.$$

In contrast, for a haploid father and his daughter, the covariance is

$Cov(a_{X,father}, a_{X,daughter}) = \sigma^2_X$, whereas for a father and his son, who only inherits the X chromosome from his mother, $Cov(a_{X,father}, a_{X,son}) = 0$.

Limitations of using only pedigree information:

Pedigree information is invaluable, however, pedigree-based relatedness coefficients has its limitations:

- Pedigree-based coefficients measure expected relatedness and do not take into account recombination and segregation, both of which occur during meiosis and zygote formation. Recombination refers to the physical exchange of genomic information between homologous chromosomes. Segregation describes the separation of allele pairs into different gametes (Hill and Weir, 2011).
- Pedigree-based coefficients assume that all pedigree founders are unrelated, which could lead to an underestimation of relatedness within the population (Wang, 2016).
- The reliability of pedigree-based coefficients is dependent on the accuracy of the pedigree information. Incomplete or incorrect pedigree could significantly alter the coefficients and misrepresent true relatedness within the population (Wang, 2016).

1.3.3 Genomic-based calculations of relatedness

As genomic data has become more readily available, calculations for measuring relatedness using genotypic data have also been developed. Compared to pedigree-based expected coefficients, values measured using genomic data are more accurate as they

take into account the genome reshuffling from segregation and recombination, giving more accurate measurements of relatedness.

In diploid individuals with alleles A or a, a genotype can be expressed as either homozygous (AA or aa) or heterozygous (Aa or aA). We can also demonstrate genotypes within a statistical framework by defining them as random variables. Consider $w_{i,l}$ to represent the genotype of individual i at the specific locus l. In a diploid organism, if we designate the first allele (A) as 0 and second allele (a) as 1, the genotypes can be values 0, 1 or 2, depending on the number of alternative alleles present:

$$w_{i,l} = w_{i,l,1} + w_{i,l,2}.$$

Moments-based methods can also be employed to estimate kinship and inbreeding coefficients, expanding on traditional pedigree-based methods such as Wright's coefficient of relationship and inbreeding coefficients. These moments-based methods, calculate genetic relatedness using the statistical moments (e.g., mean, variance) of allele frequencies in populations (Goudet et al., 2018). These methods allow for the computation of kinship and inbreeding coefficients based on genomic data, providing a more accurate representation of genetic relatedness in populations with available genomic markers. The moments-based methods are particularly valuable when pedigree information is sparse or unavailable, offering a flexible approach to calculating genetic relationships and inbreeding across various types of datasets.

Additionally, the genomic relationship matrix (GRM) estimates the genetic relatedness between individuals using their genomic information, reflecting the proportion of DNA shared across the genome (VanRaden, 2008). Unlike the pedigree-based numerator relationship, which uses only IBD information since pedigree founders, the genomic relationship matrix can utilize either IBD or IBS information. In the following section, I will describe methods for both IBD and IBS in the construction of the genomic relationship matrix.

IBD genomic relationship matrix:

The IBD genomic relationship matrix (GRM) contains the probabilities of genetic segments being IBD between the inherited alleles within an individual and those of other individuals. Smith and Allaire (1985) expanded the $\mathbf{A}$ matrix, proposing selection rules that would increase mating selection precision by considering the genetic contribution from each parent separately and outlining a gametic relationship matrix. The gametic relationship matrix $\mathbf{G}$ can be constructed such that $\mathbf{G}_{X_i,Y_j}$ represents probability that gamete X_i and gamete Y_j are IBD, which can be denoted as:

$$\mathbf{G}_{Xi,Y_j} = P(X_i \equiv Y_j),$$

where $\equiv$ indicates that the gametes are IBD. The diagonal of this matrix is all equal to 1 since a gamete is always IBD to itself. Jamrozik and Schaeffer (1991) expanded the used of the gametic matrix, translating a genetic model that includes both additive and dominance effects into a gametic model. The additive genetic relationship

between pairs of individuals is calculated by summing the contributions of all gametic relationships of the individuals. In a diploid species with individuals X and Y, let X_1 and X_2 represent the gametes of individual X and Y_1 and Y_2 represent the gametes of individual Y. The additive genetic relationship between gametes between individuals X and Y is then expressed as:

$$\mathbf{A}_{X,Y} = 0.5[\mathbf{G}_{X_1Y_1} + \mathbf{G}_{X_1Y_2} + \mathbf{G}_{X_2Y_1} + \mathbf{G}_{X_2Y_2}].$$

This relationship calculation can be represented in a matrix, allowing the numerator relationship matrix $\mathbf{A}$, which capture the overall additive genetic relationship among individuals, to be derived from the gametic relationship matrix $\mathbf{G}$ as:

$$\mathbf{A} = 0.5\mathbf{KGK}',$$

where $\mathbf{K} = I_N \otimes [1\ 1]$. Here, I_N is the identity matrix the size of N individuals and $\otimes$ denotes the Kronecker product.

Tier and Sölkner (1993) applied this gametic relationship matrix to a traditional animal model. This matrix contains the IBD probabilities of both gametes of an individual independently, separating the genetic relationships of paternal and maternal gametes and thus allowing for a more detailed modelling of genetic inheritance.Van Arendonk et al. (1994) further extended this to incorporate marker data, such as SNPs, though not genome-wide. All of these developments have been done with pedigrees and consequently both $\mathbf{G}$ and $\mathbf{A}$ matrices contained expected probabilities. However, the same matrices and expression can also be used when we have IBD information at individual alleles. In that case, we can then empirically calculate a proportion of IBD alleles between gametes/genomes and individuals.

IBS genomic relationship matrix

VanRaden developed two methods for constructing an IBS genotype-based relationship matrix (VanRaden, 2007, 2008). The first method (VR1) computes:

$$\mathbf{G} = \frac{\mathbf{ZZ}'}{\sum 2p_j(1-p_j)}.$$

The elements involved in the IBS GRM calculations include:

- Genotypic matrix $\mathbf{M}$ with dimensions equal to the number of individuals n by the number of loci m with elements of the matrix being genotype values 0, 1 and 2 (such as $w_{i,l}$).
- Matrix $\mathbf{P}$ contains allele frequencies at each locus j, where each column j contains twice the frequency of the second allele. In genotypic models column j is $2p_j$ or p_j in gametic models.
- $\mathbf{Z} = \mathbf{M} - \mathbf{P}$ is the matrix of centered genotype scores and $\mathbf{ZZ}'$ is the cross-product of matrix $\mathbf{Z}$ with itself.

In comparison, the second method (VR2) adjusts the influence of each markers by computing $\mathbf{G} = \mathbf{ZDZ}'$, where $\mathbf{D}$ is a diagonal matrix with reciprocals of $2p_j(1-p_j)$, thus emphasising relatedness at loci with lower minor allele frequencies relative to VR1. VR1 is generally recommended for broad population studies where equal weighting across all loci is desired, while VR2 is more appropriate for studies focusing on loci with lower minor allele frequencies, such as fine-mapping or when rare variants are of interest.

Building upon the theory of the $\mathbf{S}$ matrix (Fernando and Grossman, 1990), Druet and Legarra (2020) recently presented a method for calculating a genotype-based relatedness matrix for the X chromosome, which exhibits haplodiploid inheritance. As such, a bi-allelic locus on the chromosome will have genotype values of 0 or 1 in hemizygous males, and 0, 1 or 2 in diploid females. Consequently, this approach for calculating GRM is well-suited for considering the haplodiploidy characteristic of honeybees.

The Druet and Legarra (2020) genomic relationship matrix for the X chromosome $\mathbf{G}^X$ is expressed similarly to VanRaden's first method (VR1), denoted as:

$$\mathbf{G}^X = \frac{\mathbf{Z}^X\mathbf{Z}^{X\prime}}{\sum 2p_j(1-p_j)}.$$

Here, $\mathbf{Z^X}$ is derived from the genotypic matrices $\mathbf{M}^X_{[females]}$ and $\mathbf{M}^X_{[males]}$, which have been centred by allele frequency $2p_j$ or p_j, respectively, reflecting the haploid nature of males on the X chromosome compared to females.

Calculations of Mendelian sampling

Mendelian inheritance defines how phenotypic traits are passed from parent to their offspring in predictable patterns, first described by its namesake Gregor Mendel. Mendelian sampling introduces a random aspect to this process, as genes an offspring receives from its parents are subject to randomness due to recombination, segregation, or even mutation during meiosis (Germain and Jurca-Simina, 2018; Hill and Weir, 2011; Bateson, 1902). Wright's pedigree model expresses the genotype of an individual g_i as a combination of parental genotypes $g_{m(i)}$ and $g_{f(i)}$ in addition to Mendelian sampling deviation r_i (Wright, 1921). In diploid organism, where parents are expected to contribute half of their genetic material to the offspring, an offspring's genotype can be expressed as:

$$g_i = \frac{1}{2}g_{m(i)} + \frac{1}{2}g_{f(i)} + r_i.$$

This Mendelian sampling deviation can also be expressed at a gametic level:

$$g_{i,1} = \frac{1}{2}g_{m(i),1} + \frac{1}{2}g_{f(i),1} + r_{i,1},$$
$$g_{i,2} = \frac{1}{2}g_{m(i),2} + \frac{1}{2}g_{f(i),2} + r_{i,2}.$$

Large deviations in Mendelian sampling indicate a non-Mendelian inheritance pattern between parents and offspring, specifically deviations from the expected ratios of allele

transmission as predicted by Mendelian inheritance (Hill and Weir, 2011).

Pedigree reconstruction through genetic analysis

Due to aspects of honeybee reproductive biology such as polyandry, mating-on-the-wing and multiple mating flights, obtaining pedigree information through direct observation is challenging. A lack of confidence in the pedigree even occurs when controlled mating is attempted using methods such as isolated mating stations, as some local drones may be present for mating.

In situations where a pedigree is unknown due to unobserved mating, incomplete sampling of individuals, pedigree reconstruction through genetic analysis provides a means for researchers to evaluate and manage genetic diversity by estimating genetic relationships and ancestry (Pemberton, 2008). Parentage assignment is an integral component of pedigree reconstruction, elucidating genetic relationships between individuals which is essential for managing genetic resources effectively (Wang, 2004b). Additionally, parentage assignments help in determining levels of inbreeding within populations, allowing for the monitoring and maintenance of genetic diversity. It also aids in observing the inheritance of desirable traits, thus supporting short- and long-term breeding goals of breeding programmes. Furthermore, accurate parentage assignment testing can uncover hidden genetic relationships that might not be obvious based solely on phenotypic traits. By providing a clearer understanding of relationship between individuals, parentage assignments contribute to more informed decision-making in breeding programmes (Hamilton, 2021; Christie et al., 2017).

In honeybees, SNPs have demonstrated greater accuracy than microsatellite markers in determining kinship and reconstructing pedigrees due to the difference in numbers of readily available markers, particularly in the context of genomic selection, as highlighted by Bernstein et al. (2023) and Bernstein (2022). However, poor quality SNPs as a result of issues such a genotyping errors, mutations, and missing data can significantly affect the accuracy of family relationship inference (Wang, 2019). Parentage assignments using SNP genotypic data are susceptible to errors when these SNPs are of poor quality or if the number of SNPs in the dataset is low, resulting in an increased likelihood of assigning false positives (Anderson and Garza, 2006).

1.3.4 Phasing and parent-of-origin assignments

Whole genome sequence or SNP array data typically contains raw, un-phased genotypes, where it is not clear which of the two parental haplotypes each allele is associated with in a diploid organism. This is relevant for queen and worker honeybees as they are diploid, while drones as haploid (Browning and Browning, 2016; Manichaikul et al., 2010). Phasing is used to analyse the genotypes of individuals in a dataset, pooling their information to distinguish maternally and paternally inherited alleles into distinct haplotypes. Unfortunately, phasing errors have been shown to occur in datasets with poor quality, particularly when they involve small samples sizes, short sequences length, or genotyping errors. Errors are also more likely when data includes sequences with few regions of heterozygosity, unusual recombination events, or populations with

highly complex relationships, such as those resulting from inbreeding or admixture events. Additionally, phasing inaccuracies can occur if pedigree information provided to the software in incorrect or incomplete, leading to mismatches between the offspring haplotypes and the parental references (Browning and Browning, 2007; Wang, 2004a; Huisman, 2017). Although some phasing software provide the parent-of-origin haplotype information in their outputs such as AlphaImpute (Hickey, 2016), many commonly used tools, such as Beagle (Browning and Browning, 2007) or SHAPEIT (Delaneau et al., 2008), do not. This limitation necessitates the development of manual methods for determining the haplotype parent-of-origin.

1.4 SECTION 4: Simulations

Mathematical models are essentially quantitative descriptions of natural phenomena and tend to be deterministic or stochastic in nature. Whilst deterministic models predict the same outcome when given the same inputs, stochastic models predict a range of outcomes, hence the word stochastic derived from Greek meaning "to chance" or "random" (Ripley, 2009; Pinsky and Karlin, 2010). It is also worth defining the term "simulation" in order to avoid misinterpretations. The Oxford Dictionary defines "simulation" as: "a situation in which a particular set of conditions is created artificially in order to study or experience something that could exist in reality" (Dictionary).

1.4.1 The advantages of simulators

Simulators of breeding scenarios offer cost-effective, time-effective and low-risk alternatives to real test studies. While conducting real experimental breeding trials often requires extensive preparation, execution time, staff training, and financial resources, simulations offer total control over variables to explore extreme scenarios whilst negating ethical concerns associated with live trials (Ripley, 2009; Nelson, 2010).

Simulation-generated data has diverse applications, such as mimicking populations to test different hypotheses, improving breeding models, or replicating real data sets to predict outcomes of different breeding scenarios. Simulated data typically holds more genetic information, representing an ideal scenario, whereas real data may be lacking in quantity or quality. However, to make simulations relevant, they must be calibrated with real data, allowing for the testing of population management decisions on genetic gain, variability and selection accuracy. Additionally, simulations enable the evaluation of different statistical methods to optimize breeding outcomes. By comparatively analysing real and generated data, we can improve the accuracy of interpreting results in real data sets (Hickey and Gorjanc, 2012; Ceron-Rojas et al., 2015).

1.4.2 Deterministic vs Stochastic simulators

While both deterministic models and stochastic models offer valuable insights, they are suited to different contexts and have distinct strengths and limitations.

Deterministic models, such as the model proposed by Du et al. (2021a), assume no randomness, producing the same outcome for a given set of inputs. They are computationally efficient, provide clear, repeatable results, and are useful for assessing fixed conditions like population size or selection intensity. However, they fail to capture real-world variability, such as genetic mutations, environmental influences, or complex individual behaviours. Their rigidity makes them less suited for exploring uncertainty, limiting their effectiveness in breeding programs focused on genetic diversity or inbreeding control (Pryce et al., 2010; Silva et al., 2016; Atlin and Econopouly, 2022).

In contrast, stochastic models incorporate randomness, capturing the variability and uncertainty of biological systems. They simulate a range of possible outcomes, making them more reflective of real-world breeding dynamics. These models are valuable for assessing breeding programs under uncertainty, considering factors like environmental fluctuations, genetic drift, and random reproduction. Their adaptability allows testing diverse conditions and potential impacts. However, they are computationally intensive, requiring multiple iterations, and their results can be complex to interpret, demanding rigorous statistical analysis. Additionally, their accuracy depends on high-quality input data, as poor calibration can lead to misleading outcomes (Goodman, 2006; Lepadatu, 2009; Nelson, 2010). A key challenge in stochastic modelling is ensuring its variability aligns with real breeding data. Sensitivity analyses can help by systematically altering input parameters like mutation rate or population size and observing changes in outcomes. Comparing this variability to real breeding data assesses how well the model reflects biological randomness. Alternatively, statistical measures like the coefficient of variation (CV) quantify outcome variability in both simulated and real scenarios. A high CV suggests significant stochastic variability, while a low CV may indicate excessive determinism. These comparisons help evaluate the model's effectiveness in capturing real-world uncertainty (Jensen et al., 2015; Lotfi et al., 2010; Wang and Ghanem, 2023).

Due to the complex nature of honeybee biology, it was determined that a stochastic model approach would be taken in this thesis to better mimic real-world populations. An example stochastic simulator of plant and animal breeding programmes is the R package AlphaSimR, designed to model the genome and associated values of all individuals. AlphaSimR has a highly flexible approach that allows the users to simulate many complex breeding scenarios and generates both pedigree and genomic data. This makes it a valuable tool for researchers aiming to optimize or create breeding programmes and predict their outcomes (Gaynor et al., 2021). However, due to the complexities of honeybee biology, such as their high recombination rate, high mutation rate, and unique haplodiploidy system, AlphaSimR is not tailored to simulate honeybee breeding scenarios. Honeybee genetics also involves the csd locus (which determines sex), polyandry (with queens mating with multiple males), and complex colony-level events like swarming that are not easily modelled. These genetic and biological features, along with the intricate interactions between the colony's genetic structure and environmental factors, create significant challenges for simulating honeybee populations accurately. Therefore, as a part of my work we built upon AlphaSimR to develop a simulator of honeybee populations and breeding programmes, called SIMplyBee.

We saw a need to develop a novel honeybee-specific simulator as the existing honeybee simulators either lack the features for simulating complex breeding programmes, are oriented towards population-level dynamics rather than individual genetic-information or they are not publicly available.

1.5 SECTION 5: An introduction to the thesis

This thesis aims to advance our understanding of honeybee genetics and relatedness and develop new tools to contribute to the sustainable improvement of managed honeybee populations

The thesis contains three research chapters, each dealing with one main objective, as follows:

- **Chapter 2: SIMplyBee: An R package to simulate honeybee populations and breeding programs**
  The aim of this Chapter was to develop a simulation tool, named SIMplyBee, to accommodate for the complexities of honeybee biology. An increased interest in honeybee breeding programs in recent decades has in turn driven research into the quantitative genetics of honeybees and the development of honeybee breeding programs. Whilst honeybee simulators are available, they are too simplistic, lack ability to simulate complex genetic breeding programmes at an individual level or are not publicly available. SIMplyBee expands upon the R package AlphaSimR, and is able to simulate i) the genomes, quantitative genetic values, and breeding values on both an individual level, colony level, and multiple colony level ii) major biological characteristics of honeybees, and iii) colony events. Chapter 2 describes the theory and implementation of the SIMplyBee simulator, in addition to providing demonstrations of major functions and simulation set-ups.
- **Chapter 3: The principles of expected and realised genetic relatedness among individual honeybees**
  The aim of this chapter was to demonstrate the principles of genetic relatedness in honeybees using simulated data generated by SIMplyBee. Monitoring and managing genetic diversity is crucial in breeding programmes. Coefficients of relatedness are commonly used to measure genetic similarity between individuals. Challenges arise for researchers when interpreting the various relatedness coefficients calculated using both pedigree and genomic data, especially when dealing with different sources of information from different individuals. Two breeding scenarios, closed mating or hybrid mating, were simulated in three populations over a 10-year period, after which we evaluated the three types of relatedness:
  i) expected IBD from pedigree data, ii) realised IBD from pedigree and genotypic data, and iii) IBS from only genotypic data. These relatedness coefficients were inspected between individuals of the same colony, between queens of the same population, and between queens of different populations.
- **Chapter 4: Optimizing Haplotype Phasing and Patriline Determination in Honeybees Through Pedigree Reconstruction**

The principle aim of this paper was to evaluate the effectiveness of pedigree reconstruction and patriline determination in honeybees through both real and simulated dataset. By comparing simulated and real data, researchers can identify weakness in established genetic inheritance models or errors in real data, and evaluate the accuracy of their results. In this chapter, real genotype data collected from the mating experiment "BeeConSel" was replicated using genotypes generated by SIMplyBee. We analysed four decreasing SNP array sizes to assess the limitations and measure the precision of sire assignments of each software using known pedigree information. While gametic information is an import aspect of genetic analysis, we identified a gap in the literature for a method to assign parent-of-origin to phased haplotypes. To address this gap, we developed an R function for this purpose. Additionally, we used the information gathered from these tasks, to develop and evaluated a method for determining the number of patrilines in a honeybee colony.

The thesis concludes with **Chapter 5**, where I summarise the objectives and findings of the research Chapters before discussing the relevance of these findings in relation to the current research. Additionally, this chapter expands upon the limitations of the work and the implications of the thesis' findings on future work.

2 SIMplyBee: An R package to simulate honeybee populations and breeding programs

This chapter contains the manuscript *SIMplyBee: An R package to simulate honeybee populations and breeding programs* that was published May 2023 in the Genetics Selection Evolution Journal (https://doi.org/10.1186/s12711-023-00798-y) .

This manuscript presents SIMplyBee, a stochastic simulator for honeybee populations and breeding programmes. We describe how SIMplyBee implements honeybee-specific characteristics, such as haplodiploidy, the complementary sex determination, polyandry and social organisation. We demonstrate the use of the simulations, from installation of the programme, the simulation of individuals or colonies, simulating colony events, to the calculation of quantitative values. SIMplyBee provides a research platform for testing breeding and conservation strategies and their effect on future genetic gain and genetic variability.

The manuscript is a joint work of Jana Obšteter[1], Laura Strachan[2], Jernej Bubnič [1], Janez Prešern [1] and Gregor Gorjanc[2]

[2] The Roslin Institute and Royal (Dick) School of Veterinary Studies, University of Edinburgh, Easter Bush Research Centre, Midlothian EH25 9RG, UK; [2] Department of Animal Science, The Agricultural Institute of Slovenia, Ljubljana, Slovenia.

2.1 Author contributions

The SIMplyBee project was initiated by Gregor and Jana, who were responsible for planning its implementation and leading the development of the simulator. At the start of the project, Laura's limited coding experience and unfamiliarity with R package development led her to focus on the biological aspects. Her initial tasks involved identifying the essential characteristics of honeybee biology required for accurate simulations. She evaluated which of these could be directly adopted from AlphaSimR, which required minor coding modifications, and which needed to be developed from scratch.

A significant portion of Laura's early work was dedicated to designing the representation of the colony, including how it would be stored, how individuals would interact, how colony events would be simulated, and how matings could be modelled across

multiple generations. Over the course of two years, approximately 180 unique SIMplyBee functions were created to achieve a cohesive simulation framework. Laura's initial contributions to writing these functions were simple, focusing on basic designs as she learned to code. However, as her skills improved, she progressed to developing more complex and sophisticated functions. Code development was a collaborative effort, with Gregor, Jana, Jernej, and Laura contributing alongside each other. Regular communication ensured that all components worked cohesively. In addition to writing functions, Laura played a key role in testing them to ensure they worked as intended. This was essential for meeting CRAN requirements for package publication. She also contributed to writing detailed documentation with examples for each function, a process carried out in close collaboration with Jana and Jernej.

To provide a platform for users, Laura developed the website (https://simplybee.info), where they can easily access function documentation created during testing, as well as detailed vignettes with practical examples. The website features eight vignettes, of which Laura wrote the first draft of three and served as editor for the remaining five. All authors contributed to these vignettes.

Jana wrote the first draft of this manuscript, after which Laura served as the primary editor, working closely with Jana. All authors subsequently contributed to the final version of the manuscript.

Manuscript Status: Published

Keywords: stochastic simulator, honeybees, breeding programs, conservation programs

2.2 Abstract

2.2.1 Background

The Western honeybee is an economically important species globally, but has been experiencing colony losses that lead to economical damage and decreased genetic variability. This situation is spurring additional interest in honeybee breeding and conservation programs. Stochastic simulators are essential tools for rapid and low-cost testing of breeding programs and methods, yet no existing simulator allows for a detailed simulation of honeybee populations. Here we describe SIMplyBee, a holistic simulator of honeybee populations and breeding programs. SIMplyBee is an R package and hence freely available for installation from CRAN `http://cran.r-project.org/package=SIMplyBee`.

2.2.2 Implementation

SIMplyBee builds upon the stochastic simulator AlphaSimR that simulates individuals with their corresponding genomes and quantitative genetic values. To enable a honeybee-specific simulation, we extended AlphaSimR by developing classes for

global simulation parameters, `SimParamBee`, for a honeybee colony, `Colony`, and multiple colonies, `MultiColony`. We also developed functions to address major honeybee specificities: honeybee genome, haplodiploid inheritance, social organisation, complementary sex determination, polyandry, colony events, and quantitative genetics at individual- and colony-level.

2.2.3 Results

We describe and show implementation regarding simulating a honeybee genome, creating a honeybee colony and its members, haplodiploid inheritance and complementary sex determination, colony events, creating and managing multiple colonies at once, and obtaining genomic data and honeybee quantitative genetics. Further documentation at `http://www.SIMplyBee.info` provides details on these operations and describes additional operations related to genomics, quantitative genetics, and other functionality.

2.2.4 Discussion

SIMplyBee is a holistic simulator of honeybee populations and breeding programs. It simulates individual honeybees with their genomes, colonies with colony events, and individual- and colony-level quantitative values. SIMplyBee takes a user-defined function to combine individual- into colony-level values and hence allows for modeling any type of interaction within a colony. SIMplyBee provides a research platform for testing breeding and conservation strategies and their effect on future genetic gain and variability. Future development of SIMplyBee will focus on improving the simulation of honeybee genomes, optimizing the simulator's performance, and including spatial awareness to crossing functions and phenotype simulation. We welcome the honeybee genetics and breeding community to join us in the future development of SIMplyBee.

2.3 Introduction

The Western honeybee (*Apis mellifera*) is an economically important species globally, playing a major role in pollination and food production. The value of insect pollinators is estimated at 150 billion euros per year worldwide, which is approximately 10 percent of the global agriculture production (Breeze et al., 2017; Nicola Gallai et al.; Strano et al., 2015). In recent decades, wild and managed populations have been experiencing increased colony losses due to numerous biotic and abiotic factors (Steinhauer et al., 2018; Espregueira Themudo et al., 2020; Smith et al., 2013). Besides the economic loss, high colony mortality and human-mediated hybridisation have also driven the loss of within-species diversity during the last century and put native subspecies at risk (Lodesani and Costa, 2003; Espregueira Themudo et al., 2020; Groeneveld et al., 2020; Panziera et al., 2022). Although honeybees are a diverse species that are differentiated into 7 evolutionary lineages and 33 subspecies (Ilyasov et al., 2020; Dogantzis et al., 2021), two subspecies, *A. m. ligustica* and *A. m. carnica*, dominate the vast majority of commercial beekeeping operations (Moritz et al., 2005). The loss of genetic variability can decrease the fitness of the populations and further increases the susceptibility of

populations to ecological and anthropogenic factors (Espregueira Themudo et al., 2020; Panziera et al., 2022).

Due to increased colony losses and a decline in genetic diversity there has been an increased interest in honeybee management programs, either for breeding, conservation, or both. Breeding programs aim to improve honeybee production, behaviour, and resistance to pathogens, and manage genetic diversity that enables long-term response to selection. Conservation programs aim to preserve populations of endangered or native species by managing genetic diversity, reducing inbreeding depression, maintaining locally adaptive traits, and reducing the prevalence of pathogens.

The increased interest in honeybee breeding has spurred additional research in quantitative genetics of honeybees. Stochastic simulators are an essential tool for *in-silico* development and testing of quantitative genetic and statistical methods, and breeding strategies (Sargolzaei and Schenkel, 2009; Plate et al., 2019; Gaynor et al., 2021; Pook et al., 2020). While simulations rest on many assumptions, they enable cost-effective and rapid testing of hypotheses before practical deployment. There are some simulators available for the most commercially interesting mammalian or plant species (Sargolzaei and Schenkel, 2009; Gaynor et al., 2021; Pook et al., 2020). Due to the differences in biology and social organisation, these simulators cannot simulate honeybee populations. Although there are existing honeybee simulators, they are either too simplistic, do not simulate genomes and genetic and phenotypic values of individual honeybees, lack the flexibility to simulate the honeybee colony life cycle or the entire breeding program, or are not available as open source (Plate et al., 2019; Becher et al., 2014). One such honeybee simulator is BeeSim (Plate et al., 2019) that accounts for the quantitative genetics of the honeybees, but simulates quantitative values on the colony level, does not account for the colony events, and is also not publicly available. Another honeybee simulator, BEEHAVE (Becher et al., 2014), simulates colony and population dynamics and environmental variation to explore causes of colony failures and colony performance, but does not include genetics.

The aim of this work was to develop a holistic simulator of honeybee population management programs, SIMplyBee. SIMplyBee simulates i) genomes and quantitative values of individual honeybees as well as whole colonies, ii) major biological, reproductive, and organisational specificities of the honeybees, and iii) colony events. SIMplyBee is freely available for installation from CRAN (`http://cran.r-project.org/package=SIMplyBee`) with extensive help pages, examples, and vignettes. See also `http://www.SIMplyBee.info`. We welcome contributions from the community at `https://www.github.com/HighlanderLab/SIMplyBee`. In the following, we describe the theory and technical implementation in the SIMplyBee, demonstrate the use of SIMplyBee, and discuss the potential use of SIMplyBee and plans for its future development.

2.4 Implementation

SIMplyBee builds upon an established simulator, AlphaSimR (Faux et al., 2016; Gaynor et al., 2021), and shares its core simulation principles and functionality. AlphaSimR is a stochastic simulator that simulates individuals with their corresponding genomes and quantitative genetic and phenotypic values. The most important classes in AlphaSimR are the `SimParam` class for global simulation parameters and the `Pop` class for objects that hold a group of individuals with their individual identification, parent identifications, as well as genomes and trait values.

To enable a honeybee-specific simulation, SIMplyBee expands AlphaSimR with three classes: `SimParamBee` for global simulation parameters, `Colony` for a honeybee colony, and `MultiColony` for multiple honeybee colonies. Associated functions simulate honeybee populations and their events and facilitate an inspection or analysis of the results. The functions address major honeybee specificities: honeybee genome, haplodiploid inheritance, complementary sex determination, social organisation, polyandry, and colony events. There are five function groups related to: genome and genomic information, caste operations, colony and multicolony operations, quantitative genetics, and auxiliary operations. These functions operate at four levels with respect to their simplest return objects: level 0 being auxiliary functions returning standard R class objects such as vectors, matrices, and lists; level 1 returning an AlphaSimR `Pop` class object; level 2 returning a SIMplyBee `Colony` class object; and level 3 returning a SIMplyBee `MultiColony` class object. The codebase spans over 16,000 lines of R code, documentation, and unit tests.

2.5 Results

Here, we present the SIMplyBee functionality. We briefly describe the biological mechanisms behind the SIMplyBee functionality and demonstrate its use. We describe: i) simulating honeybee genomes; ii) creating a honeybee colony and its members; iii) haplodiploid inheritance and the complementary sex determination locus *CSD*; iv) colony events; v) working with multiple colonies; and vi) honeybee genomics and quantitative genetics. Supplementary vignettes give further details for these and additional topics (Additional Files 1-7; `http://SIMplyBee.info`).

Honeybee genome and initiating a honeybee simulation

To initiate the simulation we first need to simulate honeybee genomes and set simulation parameters (Figure 2.1). The honeybee genome is small in its physical length, only 250 million bp, but large in its genetic length, 23 Morgans, due to a very high recombination rate of 2.3 x 10^{-7} per bp (Beye et al., 2006). SIMplyBee generates honeybee genome sequences with the approximate (Markovian) coalescent simulator MaCS (Chen et al., 2009). It implements state-of-the-art honeybee demographic model (Wallberg et al., 2014) allowing for the simulation of three subspecies: *A. m. ligustica*, *A. m. carnica*, and *A. m. mellifera*.

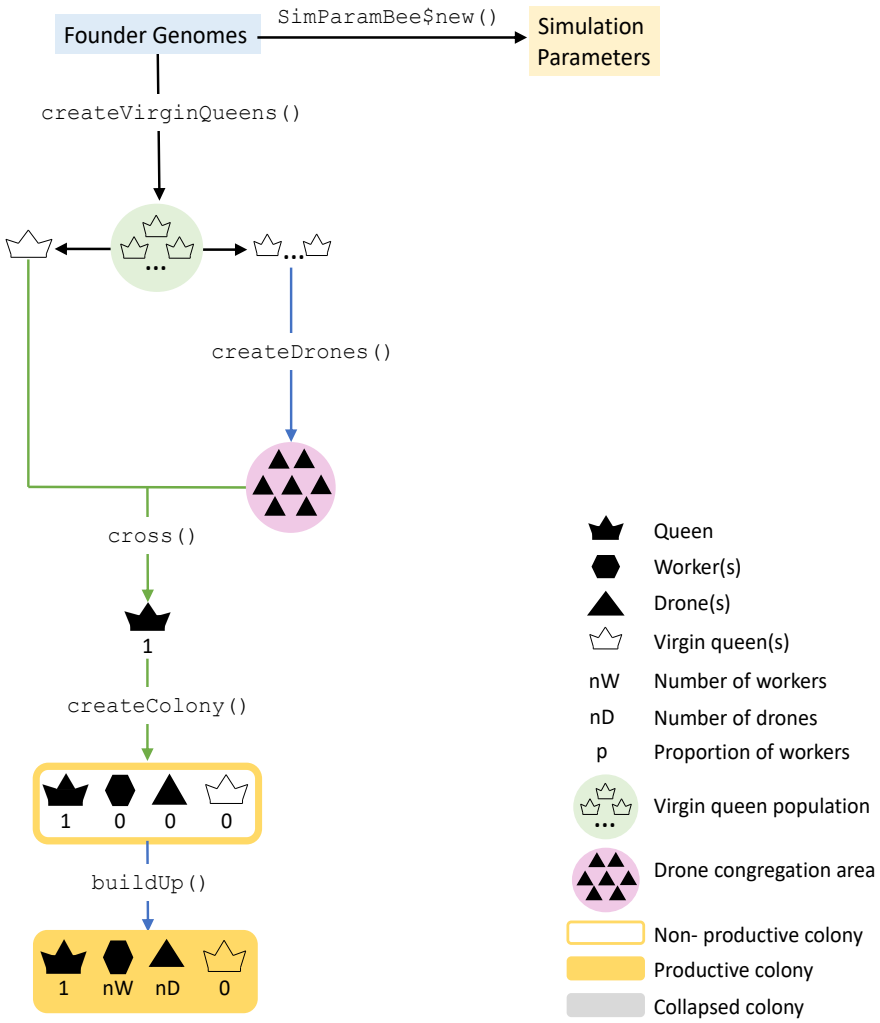


Figure 2.1: A flow chart of initialising a honeybee simulation in SIMplyBee. The first step is simulating a desired number of founder genomes and specifying the global simulation parameters in a new `SimParamBee` object. Next, we create the base virgin queens from the founder genomes. We can simulate any number (`nInd`) of virgin queens with the maximum being the number of simulated founder genomes. We choose one virgin queen as the future queen of the colony (left). On the other side (right), we select one virgin queen to provide drones for the DCA. We could select more virgin queens as future queens to create more colonies, and more virgin queen to contribute to the DCA. We next cross the virgin queen to a sample of drones from the DCA and use it to create a colony. We next build-up a colony, which adds in a desired number of workers and drones. The build-up also results in a productive colony.

To start the simulation we initially install the SIMplyBee package and load it.

```
> install.packages(pkg = "SIMplyBee")
> library(package = "SIMplyBee")
```

The first simulation step is generating founder honeybee genomes using `simulateHoneybeeGenome()`. Here, we simulate 10 *A. m. carnica* honeybees with

only three chromosomes, each with only 100 segregating sites. These numbers are not realistic, but enable a fast demonstration. Alternatively, we can import chromosome haplotypes, say from drones, or from phased queen or worker genotypes. You can read more about initiating a simulation in the Additional File 1.

```
> founderGenomes <-
    simulateHoneyBeeGenomes(nCar = 10,
                            nChr = 3,
                            nSegSites = 100)
```

The second step is setting the global simulation parameters with `SimParamBee`. `SimParamBee` builds upon the AlphaSimR class `SimParam`, which contains global user-defined simulation parameters that apply to all individuals and populations, including genome and trait parameters, but also global pedigree and recombination events. `SimParamBee` additionally holds honeybee-specific information: default numbers of workers (`nWorkers`), drones (`nDrones`), and virgin queens (`nVirginQueens`) in a full-size colony, default number of drones that a queen mates with (`nFathers`), default proportions of workers that leave in a colony swarm (`swarmP`) or are removed in a colony split (`splitP`), and a default percentage of workers removed during colony downsize (`downsizeP`). These default numbers can be changed according to the needs of a simulation. The default numbers can even be replaced by providing functions to sample numbers. For example, to sample variable number of fathers, workers, drones, and virgin queens from Poisson or truncated Poisson distributions (Additional File 7). Most SIMplyBee functions that take the number of individuals as an argument can accept these sampling functions as input, meaning that the output of such function calls will be stochastic. `SimParamBee` also holds information about the *CSD* locus: the chromosome (`csdChr`), the physical position on the chromosome (`csdPos`), and the number of alleles (`nCsdAlleles`). `SimParamBee` also holds the caste of each individual in the simulation. The caste can be a queen, father (drones that successfully mated and died), worker, drone, or virgin queen. The caste can change during the life of a honeybee. For example, after successful mating, a virgin queen becomes a queen and drones become fathers.

Here, we set the `SimParamBee` with the default number of workers in a colony being 100, default number of drones in a colony being 10, and the *CSD* locus to have 32 alleles. We save the output to the `SP` object, which enables its direct use for other SIMplyBee functions without explicitly passing it as an argument.

```
> SP <- SimParamBee$new(founderGenomes,
                        nWorkers = 100,
                        nDrones = 10,
                        csdChr = 3,
                        nCsdAlleles = 32)
```

Colony as an operational unit

A honeybee colony consists of individuals of two sexes, males being the drones and females being differentiated into two castes, the queen and the workers. The queen is

a single reproductive diploid female, the workers are non-reproductive diploid females and perform various colony maintenance tasks (collect food, nurse larvae, clean cells, etc.), and the drones are reproductive haploid males. A single colony can contain up to 65,000 workers (Farrar, 1937). Drones can represent up to 20% of a honeybee colony (Allen, 1965a). In SIMplyBee, we accounted for this social organisation by creating a class `Colony` that holds all the above-mentioned individuals as AlphaSimR populations. For ease of use, all these groups are referred to as "castes", including the drones that the queen mated with (fathers) and the virgin queens (queen-cells or emerged virgin queens). The `Colony` further contains technical information about the colony, its identification (`id`) and location (`location`) coded as `(latitude, longitude)` coordinates. Further, it contains logical information about the past colony events: `split`, `swarm`, `supersedure`, or `collapse`. It also contains `production` status, which indicates whether we can collect a production phenotype from the colony. The latter is possible when the colony is built-up to its full size and has not swarmed. The production is turned off when a colony swarms, collapses, is downsized, or is split from a colony.

Here we show how to create a colony in SIMplyBee (Figure 2.1).
The `createVirginQueens()` function creates a base population of virgin queens (an AlphaSimR's `Pop` class object) by recombining founder genomes. The `isVirginQueen()` function checks whether individuals are virgin queens. A similar `is*()` function checks the caste of each honeybee, where `*` is the inquired caste. These functions return `TRUE` or `FALSE` when an individual belongs or does not belong to the caste.

```
> baseQueens <-
    createVirginQueens(founderGenomes)
> baseQueens
An object of class "Pop"
Ploidy: 2
Individuals: 10
Chromosomes: 3
Loci: 300
Traits: 0
> all(isVirginQueen(baseQueens))
[1] TRUE
```

The `createColony()` function creates a `Colony` object from the first virgin queen. The functions can also use `n` simulated queens to simulate `n` colonies. Printout of a `Colony` object returns its basic information: the `id` (1), location (not set, hence `NA`), queen (not yet available, hence `NA`), the number of fathers, workers, drones, and virgin queens, as well as colony event statuses.

```
> colony <- createColony(x = baseQueens[1])
> colony
An object of class "Colony"
Id: 1
Location:
Queen: NA
Number of fathers: 0
Number of workers: 0
```

```
Number of drones: 0
Number of virgin queens: 1
Has split: FALSE
Has swarmed: FALSE
Has superseded: FALSE
Has collapsed: FALSE
Is productive: FALSE
```

In honeybees, the virgin queens mate with multiple drones, a phenomena termed polyandry. A honeybee virgin queen will undergo several mating flights to a drone congregation area (DCA) that consists of thousands of drones from up to 240 colonies (Koeniger et al., 2005). There, she will mate with 6 to 24 drones (Neumann et al., 1999) and store the sperm in her spermatheca for life.

SIMplyBee contains a `create*()` function for each of the castes. For example, the `createDrones()` function creates drones from either a virgin queen, to kick-start the simulation in the absence of mated queens, or a mated queen in a colony. This function can use more than one virgin queen to create a DCA. The simulation of genomes for these individuals is described in the next section (haplodiploid inheritance).

```
> baseDrones <- createDrones(x = baseQueens[2],
                             nInd = 15)
```

Here, the function `cross` mates our virgin queen in the colony to the created drones. This promotes her to a queen so she can lay eggs for workers and drones. After the mating, the colony printout shows that the identification of the queen is `"2"`, that there are 15 fathers, and that there are no more virgin queens in the colony.

```
> colony <- cross(colony,
                  drones = baseDrones)
> colony
An object of class "Colony"
Id: 1
Location:
Queen: 2
Number of fathers: 15
...
Number of virgin queens: 0
...
```

SIMplyBee includes additional functionality regarding open or controlled mating that are described in detail in the Crossing vignette in Additional File 4. There is a function to i) create a DCA for open mating, `createDCA()`, or a DCA with drones from sister queens, as commonly found on honeybee mating stations, `createMatingStationDCA()`; ii) sample a desired number of drones from a DCA, `pullDroneGroupsFromDCA()`; iii) create a cross plan, which includes information about which drones will mate with each virgin queen, `createRandomCrossPlan()`; iv) cross a virgin queen to a selected population of drones or according to a user-defined cross plan, `cross()`.

Next, the `buildUpColony()` function builds-up the colony with a specified number of workers and drones. Without specifying these numbers, the function uses default numbers in the `SimParamBee` object. Building up the colony always switches the production status to `TRUE`.

```
> colony <- buildUp(colony)
> colony
Id: 1
Queen: 2
Number of fathers: 15
Number of workers: 100
Number of drones: 10
...
Is productive: TRUE
```

SIMplyBee also contains `n*()` functions to count individuals in each caste.

```
> nQueens(colony)
[1] 1
> nFathers(colony)
[1] 15
> nWorkers(colony)
[1] 100
> nDrones(colony)
[1] 10
> nVirginQueens(colony)
[1] 0
```

The `get*()` functions access individuals. Note that these functions copy individuals and hence leave individuals in the colony - check `pull*()` functions for "pulling" individuals out of the colony.

```
> queen <- getQueen(colony)
> fathers <- getFathers(colony)
> workers <- getWorkers(colony)
> drones <- getDrones(colony)
> virginQueens <- getVirginQueens(colony)
```

The *getCaste()* function accesses the caste information of every individual.

```
> getCaste(queen)
[1] "queen"
> getCaste(drones[1:2])
[1] "drones" "drones"
```

The `getCaste()` can be very useful when you have a group of honeybees and you do not know their source.

```
> bees <- c(queen, fathers[1:2],
            workers[1:2], drones[1])
>  getCaste(bees)
```

```
[1] "queen"   "fathers" "fathers"
    "workers" "workers" "drones"
```

Additional functions for caste operations include obtaining the identifications of caste members, and set or get the year of birth and age of the queen. The `addCastePop()`, `replaceCastePop()`, or `removeCastePop()` function work with castes within a `Colony` object, and they all return a modified `Colony` object.

Haplodiploidy and *CSD*

Honeybees belong to the insect order Hymenoptera. The order is characterised by haplodiploid inheritance (Goulet et al., 1993; New, 2012; Peters et al., 2017). In SIMplyBee, we accounted for the haplodiploidy by simulating queens and workers as proper diploids and males as doubled haploids. Doubled haploids are fully homozygous individuals. However, we only use one (haploid) genome in all drone operations inside the functions. Drones' genomes are generated by recombining and segregating the queen's genome. Workers' and virgin queens' genomes are generated by recombining and segregating the queen's genome and segregating the fathers' genomes. Hence, every simulated individual has a complete genome generated following the haplodiploid inheritance.

Second, besides haplodiploidy, sex in honeybees is determined by the complementary sex determination (*CSD*) locus. Fertilised eggs that are heterozygous at the *CSD* locus develop into diploid females, while homozygotes develop into diploid drones that are killed by workers (Beye et al., 2003). In SIMplyBee, we assign a specific genomic region to represent the *CSD* gene. This region corresponds to the position of the *CSD* on chromosome three (Woyke, 1963). SIMplyBee simulates the *CSD* region as a sequence of non-recombining biallelic SNPs that determine a *CSD* allele. To account for balancing selection (Cho et al., 2006) at the *CSD* locus, we edit the initial founder genomes to achieve the desired number and frequency of *CSD* alleles in a population. The user can control the number of possible *CSD* alleles (2^{length}) by controlling the length of the locus (in number of SNPs) in the simulated population.

The `getCsdAlleles()` function retrieves *CSD* alleles and for diploids reports two non-recombining haplotypes as strings of 0s and 1s representing respectively ancestral and mutation alleles along the *CSD* locus. Below are the queen's *CSD* alleles. The first row of the output shows locus identifications (`chromosome_locus`) and the first column shows haplotype identifications (`individual_haplotype`). We see that the two sequences are different, meaning that she is heterozygous, as expected - otherwise her egg would have developed into a diploid drone that would have been killed by workers.

```
> getCsdAlleles(queen)
    3_86 3_87 3_88 3_89 3_90
2_1    1    1    0    1    0
2_2    0    1    1    0    1
>  isCsdHeterozygous(queen)
   2
TRUE
```

The *CSD* heterozygosity of honeybees is critical. Comparing *CSD* alleles of this queen and the drones she mated with (compare `getCsdAlleles(queen)` and `getCsdAlleles(fathers)` - not shown) shows no allele matches, which means we do not expect any homozygous brood in this colony. The `pHomBrood()` function calculates the theoretical brood homozygosity of a queen and the `nHomBrood()` function returns the realized number of homozygous offspring.

```
> pHomBrood(colony)
[1] 0
```

Here we create an inbred colony, by mating a virgin queen from our colony with her brothers, and inspect the expected brood homozygosity.

```
> newVirginQueen <-
    createVirginQueens(x = colony,
                       nInd = 1)
> fathers <- createDrones(colony,
                          nInd = 10)
> newQueen <- cross(newVirginQueen,
                    drones = fathers)
> inbredColony <- createColony(newQueen)
> pHomBrood(inbredColony)
[1] 0.25
```

We expect that about 25% of diploid brood will be homozygous. We now add workers to the colony to observe how many of them are homozygous. Inheritance is a random process, so a realised number of homozygotes will deviate from the expected proportion.

```
> inbredColony <- addWorkers(inbredColony,
                             nInd = 100)
> nWorkers(inbredColony)
[1] 71
> nHomBrood(inbredColony)
[1] 29
```

We tried adding 100 workers, but we only got 71. The difference of 29 is due to `CSD` homozygous brood. The information about the number of homozygous brood is stored in the queen's miscellaneous slot and is updated every time we create offspring from her.

Colony events

A honeybee colony can experience a series of events during its life: swarming, superseding, splitting, and collapsing. We present the details of colony events and their simulation in the Additional File 3.

In swarming a proportion of workers leave the hive with the queen, while the rest of the workers and drones stay in the hive. New virgin queens emerge and compete in the colony. The winner undergoes mating flights as described above.

The `swarm()` function swarms a colony (Figure 2.2). The function takes the percentage of workers that leaves with the swarm, `p`. This can be either a fixed number or a function that samples the `p` from either a uniform distribution or in some cases also from a beta distribution that accounts for the number of individuals in a colony (colony strength). You can read more about the sampling functions in the Additional File 7. The `swarm` function returns an R list with two colonies: `swarm` that contains the old queen and a proportion `p` of workers, and `remnant` that contains the rest of workers, all the drones, and virgin queens that are daughters of the queen that swarmed. The function also changes the `swarm` status to `TRUE` and `production` status to `FALSE`.

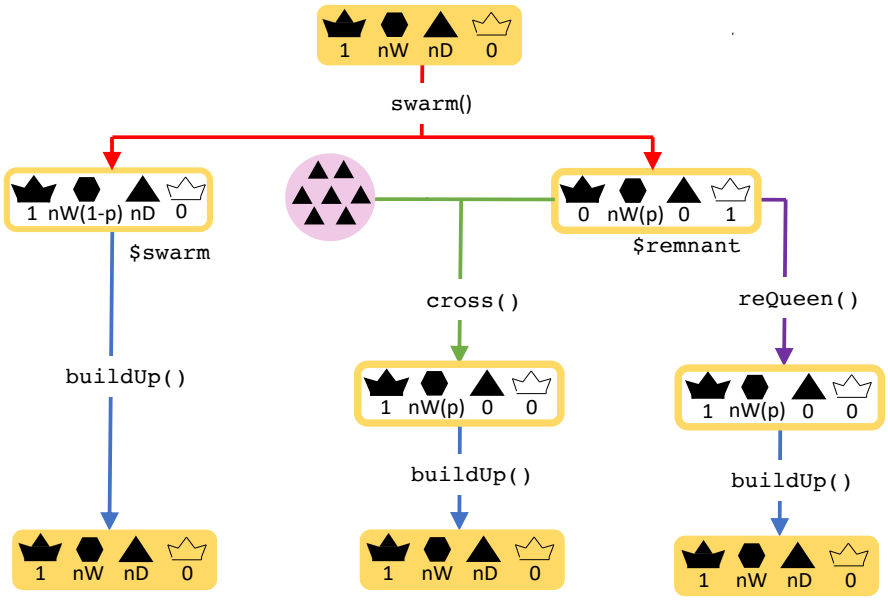


Figure 2.2: A flow chart of the colony swarming event in SIMplyBee. The `swarm()` function returns an R list with two colonies, `swarm` and `remnant`, both of which are non-productive. Parameter `p` represent the proportion of workers that leave with the swarm. After the swarm, the user can `cross()` the virgin queen of the `remnant` colony, or use an already mated queen from another source using `reQueen()`, which mimics the beekeepers' options. Refer to the key in Figure 2.1.

```
> tmp <- swarm(colony, p = 0.5)
> tmp$swarm
Id: 3
Queen: 2
Number of fathers: 15
Number of workers: 50
Number of drones: 0
Number of virgin queens: 0
...
Has swarmed: TRUE
```

```
...
Is productive: FALSE

> tmp$remnant
Id: 4
Queen: NA
Number of fathers: 0
Number of workers: 50
Number of drones: 10
Number of virgin queens: 1
...
Has swarmed: TRUE
...
Is productive: FALSE
```

To build-up the population or prevent swarming, beekeepers often split strong colonies by taking away a proportion of workers and starting a new colony with a new queen. The rest of the workers stay in the hive with the old queen. The `split()` function takes a colony and proportion of workers that we remove with the split, `p`. The `split` function returns an R list with two colonies: `split` that contains proportion `p` of workers taken from the main hive and virgin queens, and `remnant` that contains the queen, the remaining workers and drones (Figure 2.3). After the split, the remnant colony is still productive, while the split is not.

```
> tmp <- split(colony, p = 0.3)
> tmp$split
Id: 8
Queen: NA
Number of fathers: 0
Number of workers: 30
Number of drones: 0
Number of virgin queens: 1
Has split: TRUE
...
Is productive: FALSE

> tmp$remnant
Id: 1
Queen: 2
Number of fathers: 15
Number of workers: 70
Number of drones: 10
Number of virgin queens: 0
Has split: TRUE
...
Is productive: TRUE
```

In supersedure, the queen dies or is killed and its workers raise new virgin queens. The `supersede()` function removes the queen and produces new virgin queens from

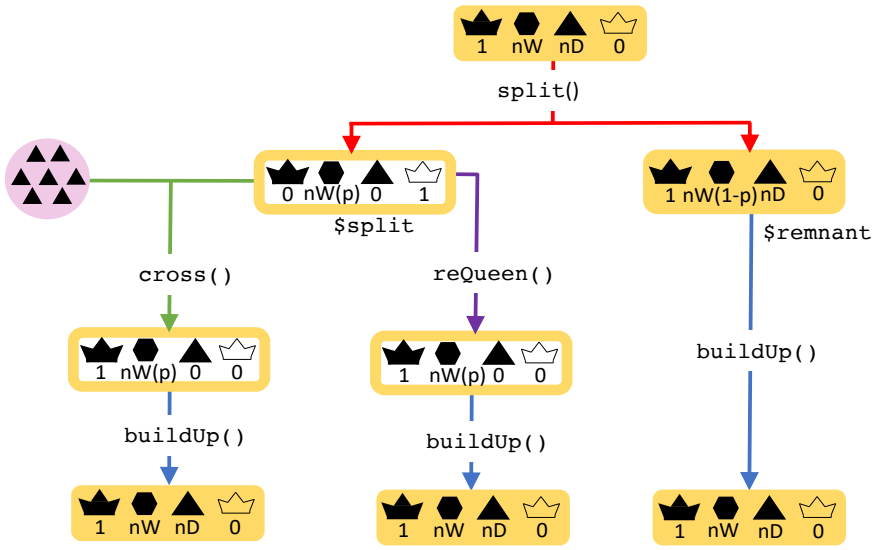


Figure 2.3: A flow chart of the colony splitting event in SIMplyBee. The `split()` function returns an R list with two colonies, `split` and `remnant`, where the *split* is non-productive and the `remnant` is productive. Parameter `p` represents the proportion of workers that are removed in a split. After the split, the user can `cross()` the virgin queen of the `remnant` colony, or use an already mated queen from another source using `reQueen()`, which mimics the beekeepers' options. Refer to the key in Figure 2.1.

the brood (Figure 2.4). After a supersedure, the colony is still productive because the workers are still present and working within the colony.

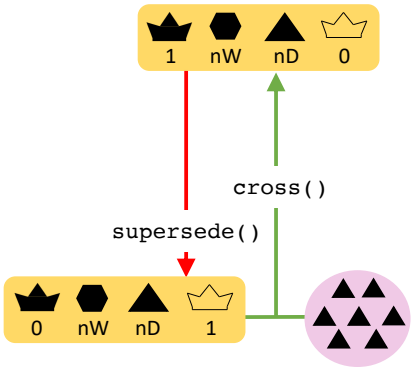


Figure 2.4: A flow chart of the colony supersedure event in SIMplyBee. The `supersede()` function returns a queen-less colony with a virgin queen. After a supersedure, a colony remains productive since the colony is still at its full size but a `cross()` is required for a new queen. Refer to the key in Figure 2.1.

```
> supersede(colony)
Id: 1
Queen: NA
Number of fathers: 0
Number of workers: 100
Number of drones: 10
Number of virgin queens: 1
...
Has superseded: TRUE
...
Is productive: TRUE
```

Finally, some colonies can collapse due to the death of all its members. The `collapse()` function collapses a colony by changing the `collapse` status to `TRUE` (Figure 2.5). The function keeps the individuals in the colony to enable the study of genetic and environmental causes contributing to the collapse. In reality, dead honeybees would also be present in a collapsed colony.

```
> collapse(colony)
Id: 1
Queen: 2
Number of fathers: 15
Number of workers: 100
Number of drones: 10
Number of virgin queens: 0
...
```

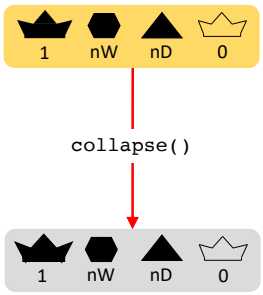


Figure 2.5: A flow chart of the colony collapse event in SIMplyBee. The `collapse` function keeps all the individuals in a colony, but turns on the `collapse` parameter, hence marking the colony as collapsed and all the individual within it as dead. Further simulation with a collapsed colony is not allowed in SIMplyBee. Refer to the key in Figure 2.1

```
Has collapsed: TRUE
...
Is productive: FALSE
```

SIMplyBee also includes functions to build-up a colony, as shown above, a `downsize()` function that removes a proportion of workers, all drones and all virgin queens, and a function `combine()` that combines a strong and a weak colony. You can read more about simulating events in Additional File 3.

Working with multiple colonies

Beekeepers regularly work with a collection of colonies at once. The `MultiColony` class collects a list of colonies to represent an apiary, a region, an age group, etc. You can read more about working with multiple colonies in the Additional File 2.

The `createMultiColony()` function creates a `MultiColony` object. Here, we create an apiary with three virgin colonies. The printout of the object returns basic information including the number of all, empty, and `NULL` colonies, and information about the colony events for the colonies within.

```
> apiary <-
   createMultiColony(x = baseQueens[3:5])
> apiary
An object of class "MultiColony"
Number of colonies: 3
Are empty: 0
Are NULL: 0
Have split: 0
Have swarmed: 0
Have superseded: 0
Have collapsed: 0
Are productive: 0
```

All the functions for managing a `Colony` object can also be applied to a `MultiColony`

object, which streamlines simulation scripts. These functions include functions for crossing, adding or removing individuals, simulating colony events, etc. Here, we demonstrate how to cross the `apiary`, build-up its colonies, and swarm some of them. The `createDrones` function creates a DCA from base population virgin queens and the `pullDroneGroupsFromDCA` function samples three groups of drones to mate the three virgin queens. As already mentioned, functions that sample individuals can use either fixed numbers or sampling functions (Additional File 7).

```
> DCA <- createDrones(baseQueens[5:10],
                          nInd = 100)
> fun <- nFathersPoisson
> droneGroups <-
    pullDroneGroupsFromDCA(DCA, n = 3,
                               nDrones = fun)
```

Next, the `cross` function crosses the virgin queens in the apiary with the provided drone groups. We test for the presence of queens before and after mating to show that mating was successful. For this, we can use `is*Present()` functions, where `*` is the caste, that check the presence of a caste in a colony. The `buildUp` function builds-up all the colonies.

```
> all(isQueenPresent(apiary))
[1] FALSE
> apiary <- cross(apiary, drones = droneGroups)
> all(isQueenPresent(apiary))
[1] TRUE
> apiary <- buildUp(apiary)
```

Next, the `pullColonies()` function samples one colony that will swarm. This returns an R list with two `MultiColony` objects: `pulled` with the sampled colonies, and `remnant` with the remaining ones. We save the latter back in the apiary.

```
> tmp <- pullColonies(apiary, n = 1)
> apiary <- tmp$remnant
[1] TRUE
```

Now, the `swarm` function swarms the pulled colonies and returns an R list with two `MultiColony` objects, `remnant` and `swarm`. When applied to `MultiColony`, all the colonies are swarmed with the same `p`, unless specified otherwise. You can read more about simulating colony events for `MultiColony` in the Additional File 3.

```
> tmp <- swarm(tmp$pulled, p = 0.5)
```

Assuming that we did not catch the swarm(s), we combine the colonies that did not swarm with the swarm remnant(s) into an updated apiary.

```
> apiary <- c(apiary, tmp$remnant)
> swarms <- tmp$swarm
```

Genomics and Quantitative genetics

Similar to extracting the *CSD* genomic sequence, `get*Haplo()` or `get*Geno()` extract whole-genome information for any set of individuals. Here, `*` can be `SegSite` to extract all segregating/polymorphic loci tracked in the simulation, `Snp` to extract marker loci, or `Qtl` to extract Quantitative Trait Loci. There is also `getIbdHaplo()` to extract Identity By Descent information, with IBD alleles defined as those originating from the base population genomes. These functions leverage AlphaSimR functionality, but work with SIMplyBee `Colony` or `MultiColony` objects and in addition take the `caste` argument to extract information only for a specific caste. All this genome information is a result of haplodiploid inheritance. For example, the code below extracts genotypes of the first five workers in the colony at the first five tracked segregating sites. See further details in the Additional File 5.

```
> getSegSiteGeno(colony, caste = "workers")[1:5, 1:5]
   1_1 1_2 1_3 1_4 1_5
26   0   2   1   0   1
27   0   1   2   1   1
28   0   2   2   1   1
29   0   1   1   0   1
30   0   2   2   1   1
```

Honeybee phenotypes are characterized by two important phenomena. First, in honeybee keeping and breeding, phenotypes are mostly collected on a colony level as opposed to on an individual level. Second, phenotypes in honeybees are a complex interaction between queen and worker effects that are often negatively correlated (Bienefeld and Pirchner, 1990; Andonov et al., 2019). For most traits, the queen indirectly contributes to the colony phenotype by laying eggs (Korb et al., 2021; Remolina and Hughes, 2008) and stimulating the workers through pheromones (Hoover et al., 2003; Kocher et al., 2009), while workers contribute directly by doing the actual work.

SIMplyBee simulates genetic and phenotypic values for each individual honeybee, but can also calculate colony-level values from individual-level values. Quantitative genetic simulation is initiated by specifying the assumptions about the genetic architecture of traits in `SimParamBee`, including the number of quantitative trait loci, distribution of their effects, as well as genetic and environmental variances and covariances. Below we initiate another simulation and specify two negatively correlated traits that will represent the queen and worker effect for honey yield. You can read a more extensive explanation of this simulation in the Additional file 6.

```
> SP <- SimParamBee$new(founderGenomes)
> nQtlPerChr <- 100
# The means for both effects
> mean <- c(10, 10 / SP$nWorkers)
# The variances for both effects
> varA <- c(1, 1 / SP$nWorkers)
# The genetic correlation matrix
> corA <- matrix(data = c( 1.0, -0.5,
                           -0.5,      1.0),
```

```
                    nrow = 2, byrow = TRUE)
# Add the traits to SimParamBee
> SP$addTraitA(nQtlPerChr = nQtlPerChr,
    mean = mean, var = varA, corA = corA,
    name = c("queenTrait", "workersTrait"))
# Environmental effect parameters
> varE <- c(3, 3 / SP$nWorkers)
> corE <- matrix(data = c(1.0, 0.3,
                          0.3, 1.0),
                    nrow = 2, byrow = TRUE)
> SP$setVarE(varE = varE, corE = corE)
```

This initiation triggers the calculation of individual-level genetic and phenotypic values. Using the AlphaSimR `Pop` class object, genetic and phenotypic values are stored respectively in `gv` and `pheno` slots and can be accessed with `getGv()` and `getPheno()` functions. Both functions have the `caste` argument and work on `Colony` and `MultiColony`.

```
# Base population virgin queens
> basePop <- createVirginQueens(founderGenomes,
                                n = 20)
# Create an apiary, cross it, and build it up
> DCA <- createDrones(x = basePop[1:5],
                      nInd = 100)
> droneGroups <- pullDroneGroupsFromDCA(DCA,
    n = 3, nDrones = 15)
> apiary <- createMultiColony(x = basePop[6:8])
> apiary <- cross(x = apiary,
    drones = droneGroups)
> apiary <- buildUp(x = apiary)
# Inspect workers values in the first colony
> head(getWorkersGv(apiary[[1]]))
     queenTrait workersTrait
[1,]   9.741739   0.02025635
[2,]   9.918515   0.05704912
> head(getWorkersPheno(apiary[[1]]))
     queenTrait workersTrait
[1,]   9.110582   0.31215045
[2,]  10.851997   0.09501006
```

The `calcColonyValue()` function maps individual values to colony values. SIMplyBee includes an established mapping function from the literature (Bienefeld et al., 2007; Plate et al., 2019; Brascamp and Bijma, 2014), but users can provide their own mapping function. Examples of such quantitative genetic simulations of one or multiple correlated traits are shown in the Additional File 6.

Here, we compute the colony-level genetic and phenotypic values for the colonies in our apiary.

```
> colonyGv <- calcColonyGv(apiary)
> colonyGv
```

```
       [,1]
11 14.69092
12 24.62362
13 18.51376
> colonyPheno <- calcColonyPheno(apiary)
> colonyPheno
       [,1]
11 15.15722
12 20.89392
13 11.90577
```

The best colony according to the genetic as well as the phenotypic value is colony with ID "12", hence we would select it for further reproduction. These values can be passed into the `use` parameter of the `selectColonies()` function.

```
bestColony <- selectColonies(apiary, n = 1, by = colonyGv)
```

Computational time

We tested the computational time needed to perform some basic actions with SIMplyBee: create drones, and create, cross, and build-up colonies (Additional File 8). The results show that most operations take seconds and increase with larger numbers.

2.6 Discussion

SIMplyBee is an R package for holistic simulation of honeybee breeding and conservation programs. In comparison to previously developed general genetics and breeding simulators (Gaynor et al., 2021; Pook et al., 2020), it simulates honeybee-specific genomes, social organisation, and behaviours. SIMplyBee differs from previously developed honeybee-specific simulators (Plate et al., 2019; Becher et al., 2014) by simulating individual honeybees, individual-level and colony-level quantitative values, and colony events that can affect genetic and phenotypic variation in a population.

SIMplyBee provides a valuable research platform for testing different population-management decisions and answering various questions regarding design of breeding schemes. SIMplyBee can be used to test the effect of various decisions in a breeding program on genetic gain, genetic diversity and inbreeding; or to test the accuracy of inferences with competing quantitative genetic models. For example, users can test the effect of different phenotyping schemes by varying the frequency of phenotyping or the measuring scale. Furthermore, SIMplyBee can be used to test different mating control designs and the effect of varying the number of sires or drone-producing queens on a mating station. The users can also test different selection strategies by varying the time of selection, the number of selected queens, or the sources of information (pedigree, genomic, and phenotypic data). The list of such potential studies is long. SIMplyBee is also a valuable platform for answering questions regarding the conservation of honeybees. Users might be interested in the effect of mating and management decisions on the genetic diversity in a population along the whole genome

or only at the *CSD* locus; in comparing how different migration or import practices and associated policies affect genetic diversity; or how to design a conservation program to preserve the existing genetic diversity.

Providing individual-level functionality might seem excessive since colony-level values in honeybees can be seen as equivalent to individual-level values in other species, say, mammals. However, there are at least seven reasons why this supports current and future honeybee research. First, colonies are made up of individual honeybees; therefore, simulating individual honeybees with associated genomes and values is the correct thing to do. Second, having individual workers' values allows us to simulate different interactions between them. For example, some phenotypes might be additive and the combined workers' contribution is just a simple sum of individual values. This additive model has been shown for honey yield once the colony reaches a certain size to have a surplus of honey (Farrar, 1937). However, such a model cannot be assumed for all phenotypes. There is much debate and contradictory results regarding the contribution of individual bees to the colony-level phenotypic value. For example, for defensive behavior, some studies show that a single or a couple of aggressive workers in a colony can stimulate the rest and lead to highly defensive colony behavior (Taber, 1982), while others suggest an additive model (Moritz and Southwick, 1987). Although this knowledge gap is understandable, given the sheer number of honeybees in a colony, future advances in sensor and beekeeping technologies and data science (machine learning) will provide ever more fine-grained data. Such data could further contribute to explaining the relationship between individual-level and colony-level phenotypes. To facilitate different models, SIMplyBee leaves the construction of the colony-level phenotype to the user, but it provides the additive model as the simplest example. As such, it can serve as a research platform for modeling different relationships. Third, simulating individuals within a colony enables the study of genetic relationships within and between colonies. Genetic variability can be driven by genetic processes and colony events. For example, splitting and swarming can substantially affect genetic variability within a colony. Fourth, related to the latter, individual-level simulation allows studying and developing methods to compute genetic relationships in honeybees for systems that deviate from the commonly studied breeding design (Bienefeld et al., 2007; Brascamp and Bijma, 2014). Fifth, having individual genomes allows studying how pooled genotyping samples, commonly used for honeybees, represent the queen's or the colony's genetics (Eynard et al., 2022), either in parentage testing and discovery, as well as for quantitative genetic analyses, say genomic prediction and GWAS. Sixth, simulating individual drones inherently simulates multiple patrilines within a colony, a patriline being all offspring of a single drone. This simulates a genetically diverse colony and allows to model potential differential contributions of partilines to the colony performance (Graham et al., 2006). And seventh, researchers have already begun to collect individual-level drone phenotypes, for example, information on drone sperm quality (Slater and Harpur, 2022).

Less ambitiously, SIMplyBee can serve as a research platform to test assumptions about current quantitative genetic models (Brascamp and Bijma, 2014; Plate et al., 2019; Bienefeld et al., 2007). With quantitative genetic models we can estimate the joint effect of workers and the effect of the queen on the colony-level value. However,

sometimes simpler models are used (Andonov et al., 2019; Basso et al., 2022). Based on the worker and queen effect estimates we can form performance, selection, and inheritance criteria (Du et al., 2021b).

Another challenge that we faced in developing SIMplyBee was in providing functionality to calculate statistical genetic values, that is, breeding values, dominance deviations, and epistasis deviations. Since SIMplyBee leverages AlphaSimR (Gaynor et al., 2021), all these values can be calculated using `bv()`, `dd()`, and `aa()` functions. However, caution is required since these functions assume Hardy-Weinberg equilibrium (Falconer, 1985) and are computed "relative" to the population of individuals at hand. The latter means that for a honeybee simulation we would either have to report these values relative to each colony population, which would make the output useless, or we would have to create a large "meta" population object of all currently living honeybees. Further development is required to address this aspect in SIMplyBee.

Future development of the SIMplyBee package will focus on additional features and improving the functionality and efficiency of existing features. Our immediate focus is on the following three features. First, on developing a new honeybee demographic model to include more subspecies and to improve estimates of the model parameters (Obšteter et al., 2022). While SIMplyBee currently uses MaCS (Chen et al., 2009) to simulate the genome, we will consider novel simulators that may offer more flexibility and computational speed, such as msprime (backward in time) (Kelleher et al., 2016; Baumdicker et al., 2022) and SLiM (forward in time) (Haller and Messer, 2019) simulators. These simulators rely on a public library of species' genome information and demographic models, named stdpopsim, to which we have already added the honeybee (Adrion et al., 2020; Lauterbur et al., 2022). Second, we will further optimize the speed and memory performance of SIMplyBee. A simulation of an entire real-size honeybee colony or a breeding program with such colonies can be computationally demanding because a single colony can hold up to several tens of thousands of workers. By timing some of the basic SIMplyBee functions, we identified that the current crossing implementation is slow (Additional File 8). However, crossing a large number of queens by proving a single drone population and a cross plan halves the time compared to crossing queens to predefined drone "packages". The users can also decrease the computational burden by simulating workers and drones only when needed, for example, drones at the time of mating and workers at the time of generating and analyzing phenotypes. We plan to further optimize SIMplyBee by allowing one to simulate workers' contribution to colony-level phenotype without storing the workers, which saves some time, but mostly memory. Such a solution has already been implemented in AlphaSimR for the simulation of hybrid plant breeding programs. Running time can also be decreased by, for example, working with the expectation and variance of genetic values in progeny (Lehermeier et al., 2017; Werner et al., 2020) instead of simulating tens of thousands of workers. We will also strive to optimize functions by leveraging C++ via the Rcpp package (Eddelbuettel and Balamuta, 2018). Third, we will add a geospatial component to the simulation. Colony location plays a major role in honeybee mating and colony performance. The current implementation enables setting the location of every `Colony` and `MultiColony` object. We will develop functionality to create a DCA or sample the drones for a virgin queen mating according to the location of colonies, for

example, in a certain radius, since virgin queens are more likely to mate with drones from nearby colonies. Furthermore, we will add spatially-aware simulation of environmental effects. Honeybee colony performance depends heavily on the environment due to food provision, weather, pests, etc. Such environmental conditions usually change continuously through space, hence colonies closer together usually experience more similar environmental conditions than colonies further apart. The framework for such spatially aware simulation and modelling has already been developed and tested in a livestock setting (Selle et al., 2020).

We welcome the honeybee genetics and breeding community to join us in the future development of SIMplyBee. The development is hosted on GitHub at `https://github.com/HighlanderLab/SIMplyBee`. We welcome users and developers to fork this git repository and provide "pull request (PR)" contributions. Each pull request is reviewed by one of the developers within the core team. Based on the review, pull requests are updated before being merged into the development branch. The development branch is periodically merged into the main branch for publication on CRAN and user installation. For each function we request documentation with examples and unit tests to ensure future changes will not break the functionality.

This work describes the usage of SIMplyBee for simulating honeybee populations. However, other bee species share a similar organisation and behaviour as the honeybee. Hence, SIMplyBee could also be used to simulate other *Apis* species. For example, *Apis florea*, the dwarf honeybee, and *Apis cerana*. *Apis florea* importantly contributes to pollination in some countries of the Middle East and Asia. Its range is predicted to increase due to climate change (Shabnam Parichehreh et al., 2022) and SIMplyBee could be used to model a breeding program in this bee species as well.

2.7 Conclusions

This paper presents a stochastic simulator, SIMplyBee, for holistic simulation of honeybee populations and population management programs. SIMplyBee builds upon its predecessors by simulating genomes of individual honeybees and the corresponding individual-level quantitative values. SIMplyBee stores individual honeybees as caste population within a colony object, which enables the simulation of colony events and calculation of colony-level values. Colonies can be further organised into multi-colony objects for ease of use. SIMplyBee provides a valuable research platform for honeybee genetics, breeding, and conservation. Possible uses include testing the effects of breeding or conservation decisions on genetic gain and genetic variability in honeybee populations, testing the performance of existing and novel statistical methods, etc. Future directions include improvements to the simulation of honeybee chromosomes through new demographic models, the addition of spatial awareness in mating and phenotype simulation, reducing computational bottlenecks, and encouraging community engagement. We welcome the honeybee genetics and breeding community to collaborate with us in improving SIMplyBee.

Availability and requirements

Project name: SIMplyBee
Home page: `http://www.SIMplyBee.info`
Installation: `http://cran.r-project.org/package=SIMplyBee`
Development: `https://github.com/HighlanderLab/SIMplyBee`
Operating systems: Windows, Linux, and MacOS
Programming language: R
License: MIT + file

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The data and material for this study are available at SIMplyBee website `http://www.SIMplyBee.info`,
CRAN `https://cran.r-project.org/package=SIMplyBee`,
and the SIMplyBee GitHub repository
`https://github.com/HighlanderLab/SIMplyBe`.

Competing interests

Not applicable.

Funding

JO acknowledges support from the ARRS Research program P4-0133. JO, LS, JB, JP, and GG acknowledge support from the ARRS research project L4-2624. JB acknowledges support from the ARRS PhD studentship 1000-20-0401. JB and JP acknowledge the support from the ARRS Research program P4-0431. LS and GG acknowledge support from the BBSRC DTP (EASTBio) CASE PhD studentship with AbacusBio and the BBSRC ISP grant BBS/E/D/30002275 to The Roslin Institute. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising from this submission.

Acknowledgements

The authors would like to thank R. Chris Gaynor for suggestions on how to leverage AlphaSimR functionality to implement honeybee specificities in SIMplyBee, and Philip Greenspoon for suggestions on improving the manuscript.

2.8 Chapter Discussion

Since this chapter was published before the thesis submission, any additional reflections that should have been included in the manuscript discussion will be addressed here, as the published manuscript cannot be modified.

Modelling the interactions between individuals, such as a honeybee queen and her workers, is still a challenge for simulations. These interactions are complex and depend on various factors like genetics, environment, and social context, making them difficult to replicate accurately in models. The complexity arises because social behaviours can vary significantly between individuals and can be influenced by both direct and indirect interactions. For example, in the context of honeybees, behaviours such as foraging, hive defence, and brood care can differ depending on individual tendencies and the roles they play within the colony. Wang et al. (2023) highlighted these challenges, emphasizing the difficulty of simulating social interactions in livestock. They note that current agent-based models are limited by the available data and the complexity of capturing how animals influence each other. While advances in technology, like computer vision and sensors, are improving data collection, accurately simulating these interactions remains a significant challenge in computational biology.

In the manuscript introduction, we discussed the honeybee simulators BeeSim (Plate et al., 2019) and BEEHAVE (Becher et al., 2014); however, another noteworthy simulator is the one developed by Kistler et al. (2021). The Kistler et al. (2021) simulator is designed for modelling honeybee breeding programs using an infinitesimal polygenic model, which effectively simulates selection and genetic progress over generations. However, SIMplyBee offers greater flexibility and realism by explicitly tracking individual alleles, making it more suited for studying genetic variation, inbreeding effects, and colony-level traits. Additionally, SIMplyBee models social inheritance and worker effects, capturing the complex colony dynamics that influence overall productivity, whereas the Kistler et al. (2021) simulator focuses primarily on structured breeding scenarios. With its ability to simulate both managed and wild populations, incorporate environmental factors, and explore diverse selection pressures, SIMplyBee provides a more comprehensive tool for honeybee research and conservation.

SIMplyBee models haplodiploidy by treating males (drones) as double haploids, consisting of two identical haploid chromosomes. This approach simplifies the simulation, as the system runs more efficiently when all individuals are treated as diploid. When drones are involved in functions such as crossing, only one of their two identical haploids is selected, effectively making them functionally diploid for the purpose of reproduction. The use of double haploid drones is primarily a practical choice for simplifying computation and data storage, without affecting the biological accuracy of the simulation.

SIMplyBee accounts for balancing selection at the CSD (complementary sex determiner) gene by modelling the genetic dynamics that promote the maintenance of genetic diversity at this locus. In the simulation, SIMplyBee incorporates this mechanism by ensuring that heterozygotes (which are female) are favoured, and homozygotes (which

are male) are selectively removed from the population. This maintains genetic diversity at the CSD locus over generations, as the homozygous males are less likely to reproduce. By accurately simulating this genetic process, SIMplyBee reflects the balancing selection that stabilizes allele frequencies at the CSD locus, helping to ensure the persistence of the gene pool in the population.

SIMplyBee incorporates an established mapping function from the literature (Bienefeld et al., 2007; Plate et al., 2019; Brascamp and Bijma, 2014), while also allowing users to define their own. These functions mathematically convert recombination frequency between genetic markers into genetic distance (centimorgans), accounting for crossover probabilities during meiosis.

To simulate realistic honeybee genomes using MaCS (Chen et al., 2009), we used the demographic parameters outlined in Wallberg et al. (2014), which included the historical population splits and the effective population size (Ne) at the time of the split. This approach enabled the simulation of different lineages and the generation of various subspecies, which could be used to investigate hybridization and assess the impacts of queen importation.

In the manuscript it is mentioned that a queen's phenotypic trait values for a trait such as honey yield are typically higher. This is because, while she does not contribute to honey production directly, her genetics determine the productivity of her worker offspring, who do. Workers, despite not reproducing, still express genetic traits inherited from their queen and father, which influence their individual performance but are not passed to the next generation; however, the collective effect of all workers determines the colony's overall productivity.

In SIMplyBee, the calculation of the colony-level phenotype is left to the user, offering flexibility in defining how individual caste members (like queens, workers, drones) contribute to the overall colony phenotype. The function `calcColonyPheno` simplifies this process by using a default mapping function (`mapCasteToColonyPheno`) to aggregate the phenotype values of individual caste members into a colony-wide phenotype. If needed, users can provide their own mapping function, allowing them to customize how caste-specific traits combine to reflect the colony's phenotype. Essentially, `calcColonyPheno` streamlines the calculation while giving users control over the underlying logic. An example demonstrating how to set up a simulation of bee populations, define traits influencing colony phenotype, and use `calcColonyPheno` and `calcColonyGv` to calculate colony-level phenotype and genetic values for both individual and multiple colonies can be found in the documentation for the `calcColonyValue` function. For more details, visit `simplybee.info`.

To maximize user flexibility, we offer default global simulation parameters, but users can customize sampling methods, including the choice between Poisson or truncated Poisson distributions for varying caste sizes. The Poisson distribution is ideal for modelling unconstrained numbers, while the truncated Poisson is better for restricting values within specified minimum or maximum limits.

Future simulations using SIMplyBee will provide a more precise estimation of computational run times for entire breeding programs, as execution time is highly dependent

on the chosen simulation parameters. Factors such as population size, number of generations, and the complexity of genetic models will significantly influence overall computational efficiency.

Amendments to be acknowledged: Within the manuscript an incorrect citation was used when stating that the CSD locus was located on the third chromosome. It cites Woyke (1963), which is an error, and the correct citation is Beye et al. (2003). Additionally, upon reflection, the use of Andonov et al. (2019) to support the statement that "phenotypes in honeybees are a complex interaction between queen and worker effects that are often negatively correlated" was incorrect, as the paper describes a positive genetic correlation between two traits. However, the statement remains accurate, as it is broad and does not imply that all traits are negatively correlated. More appropriate references to support this claim would be Kistler et al. (2024) or Bienefeld et al. (2007). In the manuscript, it is stated that SIMplyBee implements a state-of-the-art honeybee demographic model ((Wallberg et al., 2014). It is acknowledged that this "state-of-the-art" description is not an accurate depiction of the model, a more appropriate wording would have omitted this phrase.

2.9 Additional Files

Additional file 1 — Honey biology vignette This vignette introduces SIMplyBee package by describing and demonstrating how SIMplyBee implements honeybee biology. Specifically, it describes how to initiate simulation with founder genomes and simulation parameters, how to create and build-up a colony, the colony structure, and complementary sex determining (*CSD*) locus. This vignette can also be found on `https://cran.r-project.org/package=SIMplyBee` and `http://www.SIMplyBee.info`.

Additional file 2 — Multiple colonies vignette This vignette introduces working with multiple colonies by demonstrating how to create and work with `MultiColony` objects in SIMplyBee. This vignette can also be found on `https://cran.r-project.org/package=SIMplyBee` and `http://www.SIMplyBee.info`

Additional file 3 — Colony events vignette This vignette introduces the colony events and how to simulate them in SIMplyBee. It shows how to simulate swarming, splitting, superseding, and collapsing either a `Colony` or `MultiColony` objects. This vignette can also be found on `https://cran.r-project.org/package=SIMplyBee` and `http://www.SIMplyBee.info`.

Additional file 4 — Crossing vignette This vignette demonstrated how to cross virgin queens in SIMplyBee. It demonstrates how to cross a single or multiple virgin queens, cross either with pre-selected population/group of drones or according to a cross plan, and cross queens on an open DCA or mating station. This vignette can also be found on `https://cran.r-project.org/package=SIMplyBee` and `http://www.SIMplyBee.info`.

Additional file 5 — Genomics vignette This vignette demonstrates how to obtain genomic information of simulated honeybees. It also demonstrates, how to compute honeybee genomic relationship matrices in SIMplyBee. This vignette can also be found on `https://cran.r-project.org/package=SIMplyBee` and `http://www.SIMplyBee.info`.

Additional file 6 — Quantitative genetics vignette This vignette describes and demonstrates how SIMplyBee implements quantitative genetics principles for honeybees. Specifically, it describes three different examples where we simulate a single colony trait, two colony traits, and two colony traits where one trait impacts the other one via the number of workers. This vignette can also be found on `https://cran.r-project.org/package=SIMplyBee` and `http://www.SIMplyBee.info`.

Additional file 7 — Sampling functions vignette This vignette introduces sampling functions that sample either the number of caste individuals or the proportion of workers that stay or are removed in colony events. This vignette can also be found on `https://cran.r-project.org/package=SIMplyBee` and `http://www.SIMplyBee.info`.

Additional file 8 — Computational time The table shows the mean computational time for basic SIMplyBee functions of ten replicates. It shows the time to create ten or a million drones; create ten or a thousand empty or virgin colonies; to cross ten or

a thousand colonies by providing `n` drone populations (drone packages), where n is the number of virgin queens, or by providing a single drone population and a cross plan; and to build-up ten or a thousand colonies to a thousand or 60 thousand workers.

Additional file 1 - Honeybee biology vignette

2022-12-15

Introduction

This vignette describes and demonstrates how SIMplyBee implements honeybee biology. Specifically, it describes:

1. initiating simulation with founder genomes and simulation parameters,
2. creating and building up a colony,
3. colony structure, and
4. complementary sex determining (*CSD*) locus.

First, you need to install the package with `install.packages(pkg = "SIMplyBee")`.

Now load the package and dive in! You load the package by running:

```
library(package = "SIMplyBee")
#> Loading required package: AlphaSimR
#> Loading required package: R6
#>
#> Attaching package: 'SIMplyBee'
#> The following object is masked from 'package:base':
#>
#>     split
```

Initiating simulation with founder genomes and global parameters

Figure 1 visualizes the initiation of the simulation. First, we simulate some honeybee genomes that represent the founder population. You can quickly generate random genomes using AlphaSimR's `quickHaplo()`. These founder genomes are rapidly simulated by sampling 0s and 1s, and do not include any species-specific demographic history. This is equivalent to all loci having allele frequency 0.5 and being in linkage equilibrium. We use this approach only for demonstrations and testing.

Alternatively, you can more accurately simulate honeybee genomes with SIMplyBee's `simulateHoneybeeGenomes()`. This function simulates the honeybee genome using MaCS (Chen et al., 2009) for three subspecies: *A. m. ligustica*, *A. m. carnica*, and *A. m. mellifera* according to the demographic model described by Wallberg et al. (2014).

As a first demonstration, we will use `quickHaplo()` and simulate genomes of two founding individuals. In this example, the genomes will be represented by only three chromosomes and 1,000 segregating sites per chromosome. Honeybees have 16 chromosomes and far more segregating sites per chromosome, but we want a quick simulation here.

```
founderGenomes <- quickHaplo(nInd = 2, nChr = 3, segSites = 100)
```

Alternatively, we use `simulateHoneybeeGenomes()` to sample genomes of a founder population including 4 *A. m. mellifera* (North) individuals and 2 *A. m. carnica* individuals. The genomes will be represented by only three chromosomes and 5 segregating sites per chromosome. These numbers are of course extremely

low because we want a quick examample for demonstrative reasons. This chunk of code should take a few minutes to run.

```
founderGenomes2 <- simulateHoneyBeeGenomes(nMelN = 4,
                                           nCar = 2,
                                           nChr = 3,
                                           nSegSites = 5)
```

Unfortunately, due to the complexity of this function, even using such small numbers takes a while to run. Simulating a group of founder genomes with more realistic numbers will therefore require a lot of time to run. We suggest running this part to an external server and save the outcome as an RData file, which we can load in our environment and work with it.

```
save(founderGenomes2, file="FounderGenomes2_3chr.RData")
```

Besides specifying the number of individuals, chromosomes, and segregating sites, `simulateHoneybeeGenomes()`, also takes a number of genomic parameters: effective population size, ploidy, length of chromosomes in base pairs, genetic length of a chromosome in Morgans, mutation rate, recombination rate, and time of population splits. The default values for these numbers follow published references (Wallberg et al., 2014). While you can change these parameters, we don't advise doing this because such demographic models, and their parameters, are estimated jointly, so we should not be changing them independently. You can read more about these parameters in the help page:

```
??simulateHoneybeeGenomes
```

Now we are ready to setup global simulation parameters using `SimParamBee`. `SimParamBee` builds upon AlphaSimR's `SimParam`, which includes genome and trait parameters, but also global pedigree and recombination events. We usually save the output of `SimParamBee` as the `SP` object (we will assume this in all vignettes). Namely, all SIMplyBee functions will use this object if you don't directly specify `simParamBee` argument. `SimParamBee` additionally holds honeybee specific simulation information (Figure 1):

- default number of workers (`SP$nWorkers`) and drones (`SP$nDrones`) in a full-sized colony; these numbers are used by functions such as `createWorkers/Drones()`, `addWorkers/Drones()` and `buildUp()`;
- default number of drones that a virgin queen mates with (`SP$nFathers`)
- the *CSD* information: the chromosome of the *CSD* (`SP$csdChr`), the position (`SP$csdPos`), and the desired number of *CSD* alleles in a population (`SP$nCsdAlleles`). The number of *CSD* alleles determines the length of the *CSD* locus (`SP$nCsdSites`): `nCsdAlleles = nCsdSites**2`. By default, the *CSD* is placed on its real genomic position on chromosome 3. However, if the user simulates less than three chromosomes, the *CSD* is placed on chromosome 1;
- pedigree for each individual created in the simulation (`SP$pedigree`) if requested by `SP$setTrackPed(TRUE)`; and
- caste information for each individual created in the simulation (`SP$caste`).

You can read more about the `SimParam` and `SimParamBee` in their help pages (`help(SimParam)` and `help(SimParamBee)`).

Below we use set the number of *CSD* alleles and default number of workers and drones in a colony:

```
SP <- SimParamBee$new(founderGenomes, nCsdAlleles = 32)
SP$nWorkers <- 100
SP$nDrones <- 10
```

After creating the `SimParamBee` object, you can inspect it! This returns a lot of output and we suggest you return back to this point once you are comfortable with the basic functionality!

```
print(SP)
```

From the simulated founder genomes, we can create virgin queens (Figure 1). These will serve as our our first honeybee individuals (the so called base or founder population). In AlphaSimR and SIMplyBee,

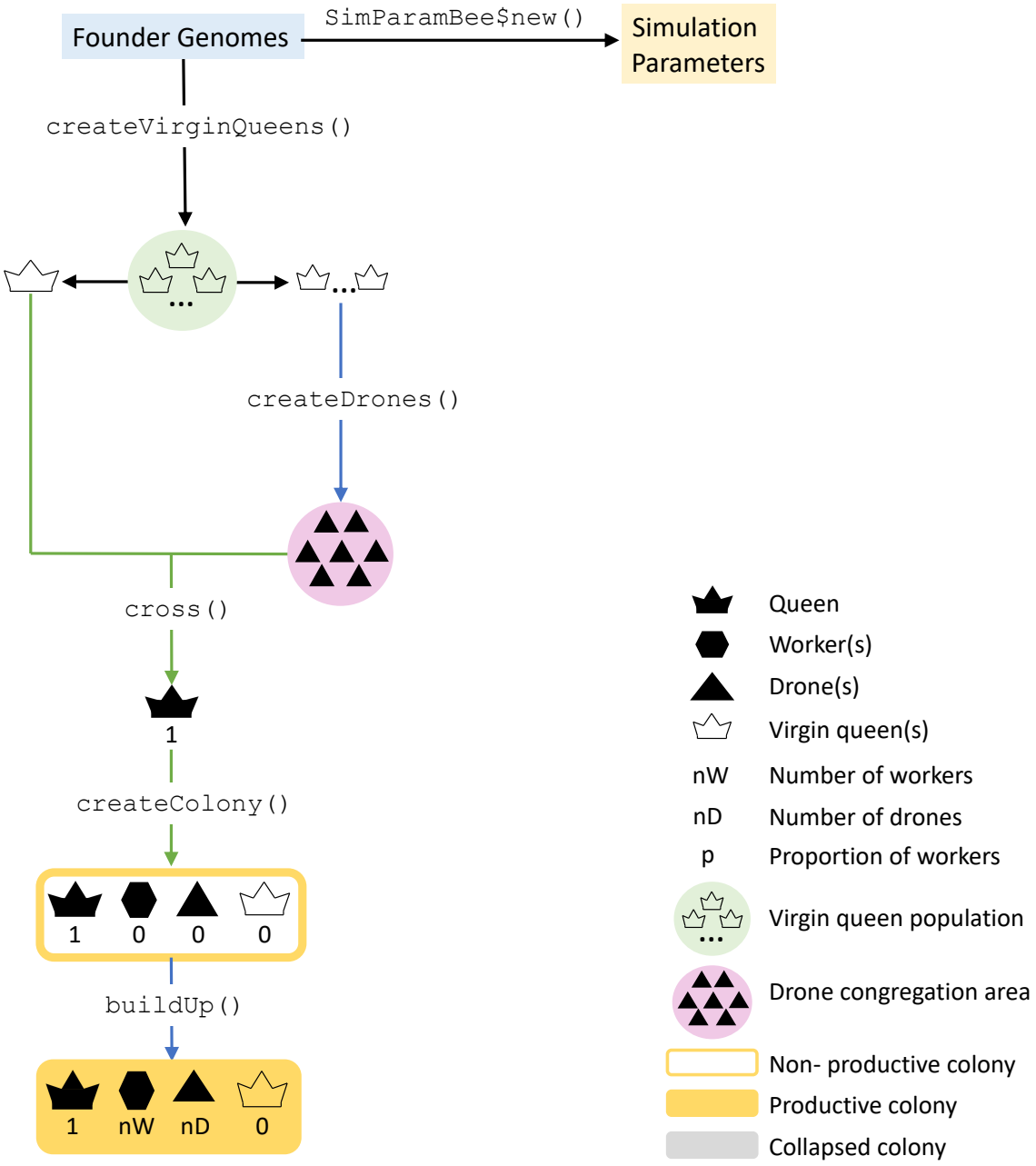


Figure 1: Simulation initiation.

individuals are stored in `Pop` class objects, that hold a group of individuals with their individual identification, parent identifications, as well as genomes and trait values. So, the `basePop` is a population (`Pop` class object) of two individuals, our two virgin queens. If we print out `basePop`, we see some basic information about the population: the ploidy, number of individuals, chromosome, loci, and traits. We next check whether our individuals are of certain caste with `is*()` functions, where `*` can be either `queen`, `worker`, `drone`, `virginQueen`, or `father`. These functions return `TRUE` if the individual is a member of the caste in question and `FALSE` is it is not. These functions check the caste information in the `SP$caste`. Here, we use `isVirginQueen()` to check whether our base population individuals are virgin queens.

```
baseQueens <- createVirginQueens(founderGenomes)
baseQueens
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 2
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
isVirginQueen(baseQueens)
#> [1] TRUE TRUE
```

Similarly, you can use the function `getCaste()` to get the caste of each individual.

We will use the first virgin queen to create five drones for future mating. Note that virgin queens do not create drones. Only queens with colonies create drones. However, to get the simulation up and running, we need drones and the function `createDrones()` can work both with virgin queens or colonies (we will present colonies in the next section). You can use more than one virgin queen to create the drones or even an entire drone congregation area (DCA) with as many drones per virgin queen as you want (`nInd`).

```
baseDrones <- createDrones(x = baseQueens[1], nInd = 15)
baseDrones
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 15
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
```

Creating and building up a colony

We will use the other virgin queen to create a colony. You can use more than one virgin queen to create more than one colony. In SIMplyBee, a honeybee colony is stored in an object of `Colony` class. You can create a new colony with the function `createColony()`. You can create a completely empty colony or a colony with either a virgin or a mated queen. The `Colony` class organises all its members in five castes: `queen`, `fathers`, `workers`, `drones`, and `virginQueens`. We describe the castes in next section. The `Colony` further contains technical information about the colony, its identification `id` and `location` coordinates coded as (`latitude`, `longitude`). Further, it contains logical information about the past colony events: `split`, `swarm`, `supersedure`, or `collapse`. It also contains `production` status, which indicates whether we can collect a production phenotype from the colony. The latter is possible when the colony is built-up to its full size and has not swarmed. The production is turned off when a colony downsizes, collapses, or swarms, and for the split of a split colony. You will learn about these colony events in the Colony events vignette.

```
colony <- createColony(x = baseQueens[2])
colony
#> An object of class "Colony"
#> Id: 1
#> Location:
```

```
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 0
#> Number of drones: 0
#> Number of virgin queens: 1
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

We see all the above mentioned information in the printout of the `Colony` object. For this specific colony, we see that the ID of the colony is "8", the location is not set, and there is no queen (hence `NA`). There are consequently no fathers in the colony, nor any workers, drones or virgin queens. All the events are set to `FALSE` (you will learn more about events in the Colony events vignette) and the colony is not productive, since it does not include any individuals.

Let's now mate our virgin queen, so that she is promoted to a queen and can start laying eggs of her own workers and drones.

```
colony <- cross(colony, drones = baseDrones)
colony
#> An object of class "Colony"
#> Id: 1
#> Location:
#> Queen: 2
#> Number of fathers: 15
#> Number of workers: 0
#> Number of drones: 0
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

We see that the virgin queen is now a queen - hence we have a queen with the ID "2" and no virgin queens in our colony.

Next, let's build up our colony using the function `buildUp()` that adds in workers and drones. This function takes parameters `nWorkers` and `nDrones`, where we specify how many workers and drones to add. However, if these numbers are not specified in the function's call, the function uses the default numbers from the `SimParamBee` object (`SP$nWorkers` and `SP$nDrones`). This function also always turns the `production` status to `TRUE`, since it assumes we are building the colony up to its full-size.

```
buildUp(colony, nWorkers = 10, nDrones = 7)
#> An object of class "Colony"
#> Id: 1
#> Location:
#> Queen: 2
#> Number of fathers: 15
#> Number of workers: 6
#> Number of drones: 7
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
```

```
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
buildUp(colony)
#> An object of class "Colony"
#> Id: 1
#> Location:
#> Queen: 2
#> Number of fathers: 15
#> Number of workers: 66
#> Number of drones: 10
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
```

All the functions in SIMplyBee return objects, hence we need to save them as an object, otherwise they are lost.

```
colony <- buildUp(colony)
colony
#> An object of class "Colony"
#> Id: 1
#> Location:
#> Queen: 2
#> Number of fathers: 15
#> Number of workers: 73
#> Number of drones: 10
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
```

Colony structure

Lets explore our colony. In every colony we have different groups of individuals (castes). These include: queen, fathers, workers, drones, and virgin queens. The queen controls the colony, workers do all the hard work, drones disseminate queen's genes, and one of the virgin queens will eventually replace the queen. We also store fathers, which represent drones that the queen mated with. The fathers caste is effectively the drone sperm stored in queen's spermatheca. Storing fathers enables us to generate colony members on demand. SIMplyBee contains `n*()` functions to count the number of individuals in each caste, where `*` is `queen`, `fathers`, `workers`, `drones`, and `virginQueens`. Let's count how many individuals we have for each caste in our colony.

```
nQueens(colony)
#> [1] 1
```

```
nFathers(colony)
#> [1] 15
```

```
nWorkers(colony)
#> [1] 73
```

```
nDrones(colony)
#> [1] 10
```

```
nVirginQueens(colony)
#> [1] 0
```

Next, we can access the individuals of each caste with `get*()` functions. These functions leave the colony and its members intact (they do not change the colony) by copying the individuals.

```
(queen <- getQueen(colony))
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 1
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
```

```
(fathers <- getFathers(colony))
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 15
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
```

```
(workers <- getWorkers(colony))
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 73
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
```

```
(drones <- getDrones(colony))
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 10
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
```

```
(virginQueens <- getVirginQueens(colony))
#> NULL
```

As you see above, there are no virgin queens present in the colony at this moment, since the queen is active. Future colony events might change this.

Should you want to pull out, that is, remove castes or their members, have a look at `pull*()` functions. These functions return a list of objects: `pulled` being the pulled individuals (`Pop` object), and `remnant` being the remaining colony without the pulled individuals.

```
tmp <- pullWorkers(colony, n = 10)
colony <- tmp$remnant
colony
#> An object of class "Colony"
```

```
#> Id: 1
#> Location:
#> Queen: 2
#> Number of fathers: 15
#> Number of workers: 63
#> Number of drones: 10
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
```

```
pulledWorkers <- tmp$pulled
pulledWorkers
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 10
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
```

Next, you can obtain the caste of each individual with the `getCaste()` function. As already mentioned above, a similar group of functions are the `is*()` functions that check whether an individual is of specific caste. Let's now obtain the caste of colony members:

```
getCaste(queen)
#> [1] "queen"
```

```
getCaste(fathers)
#>  [1] "fathers" "fathers" "fathers" "fathers" "fathers" "fathers" "fathers"
#>  [8] "fathers" "fathers" "fathers" "fathers" "fathers" "fathers" "fathers"
#> [15] "fathers"
```

and so on. When you have a collection of bees at hand and you might not know their source, the `getCaste()` can be very useful:

```
bees <- c(queen, fathers[1:2], workers[1:2], drones[1:2])
getCaste(bees)
#> [1] "queen"   "fathers" "fathers" "workers" "workers" "drones"  "drones"
```

Complementary sex determining locus

The complementary sex determiner (*CSD*) locus, well, complements sex determination. Fertilised eggs that are heterozygous at the *CSD* locus develop into workers. On the other hand, homozygous eggs develop into unviable drones. These drones are usually discarded by workers. SIMplyBee does not store these unviable drones, but it does store their number in the queen's miscellaneous slot (`queen@misc`). Here, you can find the total number of workers and drones produced by the queen (`nWorkers` and `nDrones`) and how many of the diploid offspring were homozygous at the *CSD* (`nHomBrood`). There is also a `pHomBrood` slot, that represents the theoretical (expected) proportion of offspring that are expected to be homozygous based on queen's and father's *CSD* alleles. You can obtain `pHomBrood` and `nHomBrood` values with the corresponding `pHomBrood()` and `nHombrood()` functions that can be applied either on the queen (`Pop` class) or colony (`Colony` class) directly. You can obtain the entire `misc` slot with the `getMisc()` function.

```
getMisc(getQueen(colony))
#> [[1]]
#> [[1]]$fathers
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 15
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
#>
#> [[1]]$nWorkers
#> [1] 73
#>
#> [[1]]$nDrones
#> [1] 10
#>
#> [[1]]$nHomBrood
#> [1] 27
#>
#> [[1]]$pHomBrood
#> [1] 0.3
```

Technically, in SIMplyBee we represent the *CSD* locus as a series of bi-allelic single nucleotide polymorphisms that don't recombine. So, the *CSD* locus is represented as a non-recombining haplotype and different haplotypes represent different *CSD* alleles. By varying the number of sites within the *CSD* locus we can control the number of distinct alleles (see `help(SimParamBee)`).

We can retrieve information about *CSD* alleles with `getCsdAlleles()`. For details on where the *CSD* locus is and the number of distinct alleles, see `help(SimParamBee)`. Looking at the below output, the first row shows marker identifications (chromosome_locus) and the first column shows haplotype identifications (individual_haplotype). The alleles are represented with a sequence of 0's and 1's. You can see that the two sequences are different, meaning that the queen is heterozygous, as expected.

```
getCsdAlleles(queen)
#>     3_86 3_87 3_88 3_89 3_90
#> 2_1    1    0    1    1    1
#> 2_2    0    1    1    1    1
```

A keen geneticist would immediately inspect *CSD* alleles of fathers to check for any similarity with the queen's *CSD* alleles. Let's boost a chance of such an event by creating an inbreed colony. We will create a virgin queen from the current colony and mate her with her brothers. Oh, dear.

```
inbredColony <- createColony(x = createVirginQueens(x = colony, nInd = 1))
fathers <- selectInd(drones, nInd = SP$nFathers, use = "rand")
#> Warning in selectInd(drones, nInd = SP$nFathers, use = "rand"): Suitable
#> candidates smaller than nInd, returning 10 individuals
inbredColony <- cross(inbredColony, drones = fathers)
getCsdAlleles(inbredColony)
#> $queen
#>       3_86 3_87 3_88 3_89 3_90
#> 255_1    1    0    1    1    1
#> 255_2    0    1    1    1    1
#>
#> $fathers
#>       3_86 3_87 3_88 3_89 3_90
```

```
#> 247_1    0    1    1    1    1
#> 246_1    0    1    1    1    1
#> 253_1    1    0    1    1    1
#> 249_1    0    1    1    1    1
#> 251_1    0    1    1    1    1
#> 248_1    1    0    1    1    1
#> 245_1    1    0    1    1    1
#> 252_1    0    1    1    1    1
#> 254_1    0    1    1    1    1
#> 250_1    1    0    1    1    1
#>
#> $workers
#> NULL
#>
#> $drones
#> NULL
#>
#> $virginQueens
#> NULL
getCsdAlleles(inbredColony, unique = TRUE)
#> $queen
#>       3_86 3_87 3_88 3_89 3_90
#> 255_1    1    0    1    1    1
#> 255_2    0    1    1    1    1
#>
#> $fathers
#>       3_86 3_87 3_88 3_89 3_90
#> 249_1    0    1    1    1    1
#> 253_1    1    0    1    1    1
#>
#> $workers
#> NULL
#>
#> $drones
#> NULL
#>
#> $virginQueens
#> NULL
```

Can you spot any matches? Let's calculate the expected proportion of homozygous brood from this mating.

```
pHomBrood(inbredColony)
#> [1] 0.5
```

Let's see how many homozygotes will we observe. Note that inheritance is a random process, so a realised number of homozygotes will deviate from the expected proportion.

```
inbredColony <- addWorkers(inbredColony, nInd = 100)
inbredColony
#> An object of class "Colony"
#> Id: 2
#> Location:
#> Queen: 255
#> Number of fathers: 10
#> Number of workers: 49
```

```
#> Number of drones: 0
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
nHomBrood(inbredColony)
#> [1] 51
```

We tried adding 100 workers, but we only got 49. The difference of 51 is due to *CSD* homozygous brood. Let's add another set of workers to show variation in the realised numbers and accumulation of information.

```
inbredColony <- addWorkers(inbredColony, nInd = 100)
inbredColony
#> An object of class "Colony"
#> Id: 2
#> Location:
#> Queen: 255
#> Number of fathers: 10
#> Number of workers: 96
#> Number of drones: 0
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
nHomBrood(inbredColony)
#> [1] 104
```

In total we tried adding 200 workers. We got 96 workers and 104 homozygous brood. To see all this information, we can inspect the miscellaneous slot of the queen that contains the fathers population as well as the cumulative number of workers, drones, homozygous brood, and the expected proportion of homozygous brood.

```
getMisc(getQueen(inbredColony))
#> [[1]]
#> [[1]]$fathers
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 10
#> Chromosomes: 3
#> Loci: 300
#> Traits: 0
#>
#> [[1]]$nWorkers
#> [1] 96
#>
#> [[1]]$nDrones
#> [1] 0
#>
#> [[1]]$nHomBrood
#> [1] 104
#>
```

```
#> [[1]]$pHomBrood
#> [1] 0.5
```

References

Chen G.K., Marjoram P., Wall J.D. (2009) Fast and flexible simulation of DNA sequence data. Genome Research, 19(1):136–142. https://doi.org/10.1101/gr.083634.108

Wallberg, A., Han, F., Wellhagen, G. et al. (2014) A worldwide survey of genome sequence variation provides insight into the evolutionary history of the honeybee Apis mellifera. Nature Genetics, 46:1081–1088. https://doi.org/10.1038/ng.3077

Additional file 2 - Multiple colonies vignette

2022-12-15

Introduction

We have already introduced the Colony class that holds colony-specific information and caste individuals. However, when working with honeybees, we usually do not work with a single colony, but with apiaries or even whole populations of colonies. To cater for this, SIMplyBee provides a `MultiColony` class. It behaves as a list of `Colony` objects but with additional functionality - you can apply function directly to the `MultiColony` objects. A `MultiColony` can represent different apiaries or sub-populations in terms of either age of the queens or geographical location of the apiaries etc. This vignette demonstrates creating and working with `MultiColony` objects. First, we again load the package.

```
library(package = "SIMplyBee")
#> Loading required package: AlphaSimR
#> Loading required package: R6
#>
#> Attaching package: 'SIMplyBee'
#> The following object is masked from 'package:base':
#>
#>     split
```

Creating a MultiColony object

We create a `MultiColony` object with `createMultiColony()` function. Let's say you want to create a `MultiColony` object that represents a single apiary. The first option is to initialise an empty `MultiColony` object that represents an empty apiary without any colonies and individuals within them.

```
# Create an empty apiary
emptyApiary <- createMultiColony()
emptyApiary
#> An object of class "MultiColony"
#> Number of colonies: 0
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
```

Let's inspect the printout of the `MultiColony` object. This tells how many colonies are within, how many of them are `empty` and contain no individuals, how many are `NULL` objects, how many have experienced a split, swarm, supersedure, or a collapse (you can read more about these events in the Colony events vignette), and how many of them are productive, meaning that we can collect a production phenotype from them such as honey yield.

The second option is again to create an empty `MultiColony` object that represents an empty apiary without any individuals within, but with a defined number of colony slots.

```
# Create an empty apiary with 10 colony slots
emptyApiary1 <- createMultiColony(n = 10)
emptyApiary1
#> An object of class "MultiColony"
#> Number of colonies: 10
#> Are empty: 10
#> Are NULL: 10
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
```

The third option is to create a `MultiColony` object with a population of either virgin or mated queens. For this, we first have to initialise the simulation with founder genomes and creating a base population of virgin queens. We will use 10 virgin queens to produce drones and create a DCA.

```
# Create 20 founder genomes
founderGenomes <- quickHaplo(nInd = 30, nChr = 1, segSites = 100)
# Set up new global simulation parameters
SP <- SimParamBee$new(founderGenomes)
# Create a base population of 20 virgin queens
basePop <- createVirginQueens(founderGenomes)
# Create a DCA from the drones of the first 10 queens
DCA <- createDrones(basePop[1:10], nInd = 100)
```

We will now create an apiary with 10 virgin colonies with the `createMultiColony()` function by providing the second set of 10 virgin queens as the input parameter. Let's call this apiary `apiary1` and say that it is positioned at the location `(1,1)`.

```
# Create an apiary with the remaining virgin queens
apiary1 <- createMultiColony(x = basePop[11:20])
# Set the location of the apiary
apiary1 <- setLocation(apiary1, c(1,1))
```

Let's now use functions `isQueenPresent()` and `isVirginQueensPresent()` to confirm all the colonies are virgin.

```
# Check whether all the colonies are virgin
isQueenPresent(apiary1)
#>     1     2     3     4     5     6     7     8     9    10
#> FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
isVirginQueensPresent(apiary1)
#>    1    2    3    4    5    6    7    8    9   10
#> TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

MultiColony operations

Once we have a non-empty `MultiColony` object, we can do basic operations on it. First, we can select some colonies by either specifying their IDs, desired number or percentage of randomly selected colonies.

```
# Get the IDs of the colonies
getId(apiary1)
#>  [1]  1  2  3  4  5  6  7  8  9 10
# Select colonies according to IDs
selectColonies(apiary1, ID = c(1,2))
```

```
#> An object of class "MultiColony"
#> Number of colonies: 2
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
# Randomly select a given percentage of colonies
selectColonies(apiary1, p = 0.1)
#> Randomly selecting colonies: 1
#> An object of class "MultiColony"
#> Number of colonies: 1
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
```

Second, we can pull some colonies from the `MultiColony` object. This means, that the pulled colonies are removed from the original object. The function `pullColonies()` therefore returns two object - the pulled colonies and the remnant colonies.

```
# Pull one colony - returns a list with $remnant and $pulled nodes
pullColonies(apiary1, n = 1)
#> Randomly pulling colonies: 1
#> $pulled
#> An object of class "MultiColony"
#> Number of colonies: 1
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
#>
#> $remnant
#> An object of class "MultiColony"
#> Number of colonies: 9
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
```

Third, we can also remove some colonies from the `MultiColony` object with `removeColonies()` function.

```
removeColonies(apiary1, ID = 13)
#> Warning in removeColonies(apiary1, ID = 13): ID parameter contains come invalid
```

```
#> IDs!
#> An object of class "MultiColony"
#> Number of colonies: 10
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
```

These three functions can also select, pull, and remove colonies based on some values (phenotypes, genetic values ...). You can read more about that in the Quantitative genetics vignette.

Crossing a MultiColony

Next, we will cross all the virgin queens in the apiary with the `cross()` function to groups of drones that we collected from the DCA with the `pullDroneGroupsFromDCA()` function. We have to collect at least as many groups of drones as we have colonies in our `MultiColony`.

```
# Pull 10 groups of drones from the DCA
droneGroups <- pullDroneGroupsFromDCA(DCA, n = 10, nDrones = nFathersPoisson)
# Cross all virgin queens in the apiary to the selected drones
apiary1 <- cross(apiary1, drones = droneGroups)
# Check whether the queens are present (and hence mated)
isQueenPresent(apiary1)
#>    1    2    3    4    5    6    7    8    9   10
#> TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Once we have mated queens in the apiary, we can apply all the event functions directly to the `MultiColony` object: `buildUp()`, `downsize()`, `swarm()`, `split()`, `supersede()`, `collapse()` but also all the functions that either add, replace, or remove individuals from the castes. Let's say we want to build-up all the colonies in our apiary.

```
# Build-up all the colonies in the apiary1
apiary1 <- buildUp(apiary1, nWorkers = 1000, nDrones = 100)
```

Furthermore, we can use the `pullColonies()` or `selectColonies()` to subset the colonies that will for example swarm, collapse, or supersede (presented in the Colony events vignette), or the ones that we decided to split (check out the Colony events vignette).

Working with multiple MultiColony objects

Let's now initiate another `MultiColony` named as `apiary2` that is placed at location `(2,2)`. Here, we define different `MultiColony` object according to the location of the apiary, but the objects could also be defined according to the age of the queens (such as `age0`, `age1`...). `apiary2` contains only virgin queens and we want to mate them to a DCA made of drones from `apiary1`.

```
# Initiate apiary2 at the location (2,2)
apiary2 <- createMultiColony(basePop[21:30])
apiary2 <- setLocation(apiary2, c(2,2))
```

Since some time has passed, we want to first replace the drones in `apiary1` with new drones. We can do that with `replaceDrones()` function.

```
apiary1 <- replaceDrones(apiary1)
```

Now that we have a new set of drones, we can create a DCA with the function `createDCA()` and mate virgin queens in apiary2 to the DCA.

```
# Check whether all colonies in apiary2 are virgin
isQueenPresent(apiary2)
#>    11    12    13    14    15    16    17    18    19    20
#> FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
isVirginQueensPresent(apiary2)
#>   11   12   13   14   15   16   17   18   19   20
#> TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
# Create a DCA from all the drones in apiary
DCA <- createDCA(apiary1)
# Check how big is the DCA
DCA
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 1000
#> Chromosomes: 1
#> Loci: 100
#> Traits: 0
# Sample drones groups from the DCA
droneGroups <- pullDroneGroupsFromDCA(DCA,
                                      n = nColonies(apiary2),
                                      nDrones = nFathersPoisson)
# Cross virgin queens in apiary2 to selected drones
apiary2 <- cross(apiary2, drones = droneGroups)
```

To learn more about the `nFathersPoisson()` function and other similar functions, read the Sampliong functions vignette.

Additional file 3 - Colony events vignette

2022-12-15

Introduction

This vignette will introduce you to honeybee colony events. SIMplyBee implements the most important natural events like swarming, supersedure, and collapse of the colony, and a beekeeping management practice called splitting. All functions that implement colony events work both on `Colony` and `MultiColony` objects.

First, we start by loading the package.

```
library(package = "SIMplyBee")
#> Loading required package: AlphaSimR
#> Loading required package: R6
#>
#> Attaching package: 'SIMplyBee'
#> The following object is masked from 'package:base':
#>
#>     split
```

The code below generates 6 colonies so we can demonstrate various colony events.

```
founderGenomes <- quickHaplo(nInd = 30, nChr = 1, segSites = 100)
SP <- SimParamBee$new(founderGenomes)
basePop <- createVirginQueens(founderGenomes)
drones <- createDrones(basePop[1:10], n = 1000)
fatherGroups <- pullDroneGroupsFromDCA(drones, n = 30, nDrones = 10)

# Create Colony and MultiColony class, cross them and build them up
colony <- createColony(x = basePop[11])
colony <- cross(colony, drones = fatherGroups[[1]])
colony <- buildUp(colony, nWorkers = 100, nDrones = 20)

apiary <- createMultiColony(basePop[12:17])
apiary <- cross(apiary, drones = fatherGroups[2:7])
apiary <- buildUp(apiary, nWorkers = 100, nDrones = 20, exact = TRUE)
```

Swarming

Swarming is the process through which honeybee colonies produce new colonies. When a honeybee colony outgrows its hive, becomes too congested, or too populated for the queen's pheromones to spread among workers, then the swarming begins. The workers start building swarm cells for new virgin queens. When the queen is ready, she leaves the hive and is followed by about 70% of the workers in a massive cloud of flying bees, the swarm (Rangel and Seeley, 2012). The swarm will cluster on a nearby tree or bush and remain there until they find a suitable new home.

The virgin queens developing in the old hive are daughters of the queen that swarmed and are attended by the remnant workers that did not leave with the swarm. After few days, the new virgin queens begin to emerge. Typically, the first queen to emerge will kill the rest of virgin queens to assume the role as the new

queen for the colony. She will then go on a mating flight to find drones to mate with to begin laying eggs and rebuilding the workforce in the colony (Clemson Cooperative Extension, 2021; Rangel and Seeley, 2012).

In SIMplyBee, function `swarm()` simulates swarming (Figure 1). The function takes a `Colony` class object and a percentage `p` of workers that leave with the swarm. The function returns a list with two `Colony` class objects, `swarm` and `remnant`. The `swarm` contains the old queen and `p` percentage of workers that left the hive. The `remnant` contains the rest of workers `(1-p)`, all the drones, and virgin queens that are daughters of the old queen that swarmed.

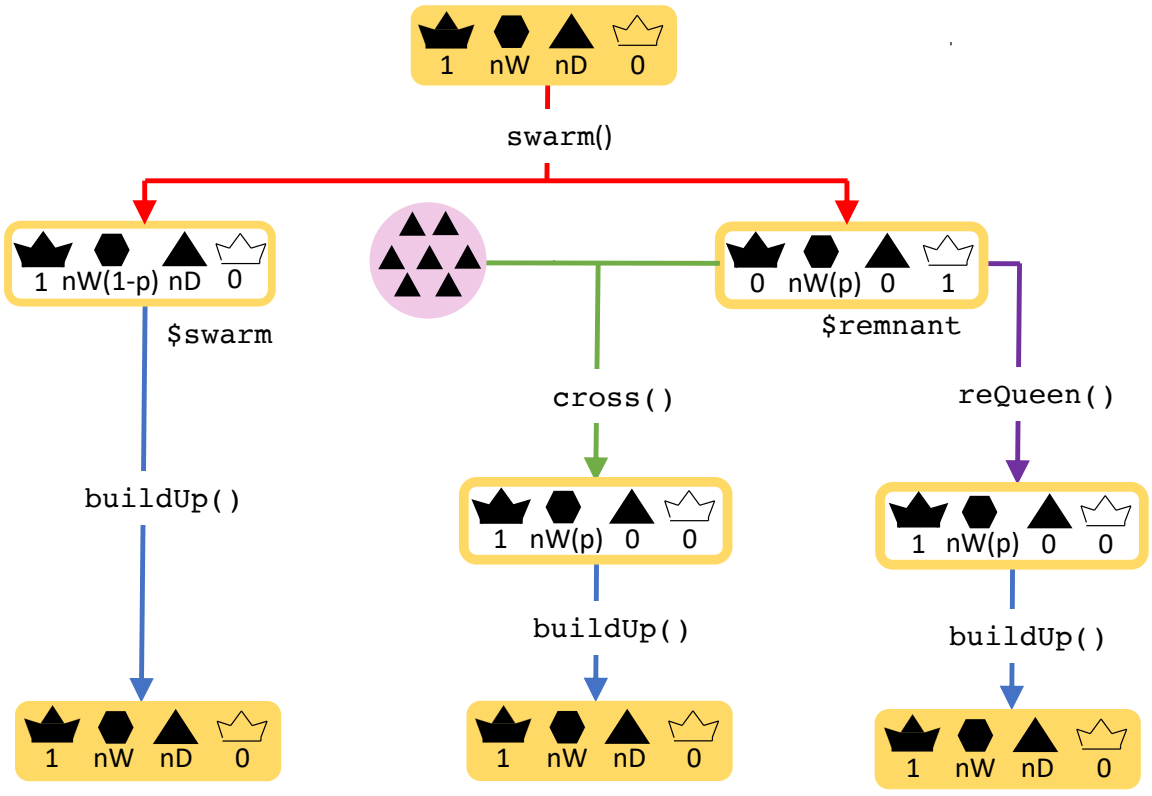


Figure 1: Swarm function.

- Swarming a Colony

Let's swarm our `colony`:

```
tmp <- swarm(colony, p = 0.4)
tmp
#> $swarm
#> An object of class "Colony"
#> Id: 8
#> Location:
#> Queen: 11
#> Number of fathers: 10
#> Number of workers: 38
#> Number of drones: 0
```

```
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
#>
#> $remnant
#> An object of class "Colony"
#> Id: 9
#> Location:
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 58
#> Number of drones: 20
#> Number of virgin queens: 1
#> Has split: FALSE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

We see that the remnant colony does not have a queen but has one virgin queen that will have to be mated. It also has 60 workers since we set `p` argument to 0.4, meaning that 40% of workers left with the swarm. All the drones remained in the remnant. Note that the `swarm` status is turned to `TRUE`.

The swarm contains the old queen, no virgin queens, and 40 workers, since we set the proportion `p` to 0.4. Same as in the remnant, the `swarm` status is turned to `TRUE` in the swarm.

Swarming also turns the `production` slot to `FALSE` for both, the swarm and the remnant colony, since we would not be able to collect honey from such colonies (or other products).

The swarm stays genetically identical to the "old" colony, although downsized. Assuming that we have caught the swarm, we assign it back to the original colony object. The remnant has a new queen and is hence genetically different from the original colony. Thus, we assigned it to a new colony object.

```
colony <- tmp$swarm
colony1 <- tmp$remnant
```

After swarming, the colony would usually build-up back to the full size and the virgin queens would mate. Building-up a colony turns the production status back to `TRUE`.

```
colony <- buildUp(colony)
colony
#> An object of class "Colony"
#> Id: 8
#> Location:
#> Queen: 11
#> Number of fathers: 10
#> Number of workers: 97
#> Number of drones: 100
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
```

```
#> Is productive: TRUE
```

Instead of setting the `p` every time we call the `swarm()` function, we can save the `swarmP` argument in the `SimParamBee` object. The `swarm()` function will then use this percentage if `p` is not set.

```
SP$swarmP
#> [1] 0.5
```

The default value is 0.5, but we can set any value we want.

```
SP$swarmP <- 0.35
SP$swarmP
#> [1] 0.35
```

You can also use a non-fixed `p` parameter by using the function `swarmPUnif` that samples the `p` from a uniform distribution between values 0.4 and 0.6 irrespective of the colony strength. You can read more about this in the Sampling functions vignette.

- Swarming a MultiColony

We swarm a `MultiColony` object in the same was we swarm a single `Colony` - with the `swarm()` function. The `swarm()` function is here applied to each colony in the `MultiColony` object with the same parameters. The function now returns a list with two `MultiColony` objects - one containing the swarms and the other containing the remnants.

```
tmp <- swarm(apiary)
tmp
#> $swarm
#> An object of class "MultiColony"
#> Number of colonies: 6
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 6
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
#>
#> $remnant
#> An object of class "MultiColony"
#> Number of colonies: 6
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 6
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
```

We see that we get six swarms and six remnants from the apiary with six colonies. We can inspect individuals colonies to ensure they swarmed according to the parameters. Let's inspect the swarm and remnant of the third colony.

```
tmp$swarm[[3]]
#> An object of class "Colony"
#> Id: 14
#> Location:
#> Queen: 14
```

```
#> Number of fathers: 10
#> Number of workers: 35
#> Number of drones: 0
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

```
tmp$remnant[[3]]
#> An object of class "Colony"
#> Id: 15
#> Location:
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 65
#> Number of drones: 20
#> Number of virgin queens: 1
#> Has split: FALSE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

We see that the the third colony was swarmed with `p` of 35% as specified in the `SimParamBee`, hence the swarm left with 35 workers and the old queen, and the remnant stayed behind with a new virgin queen and 65 workers.

Above, all the colonies in a `MultiColony` are swarmed with the same percentage. However, we can also specify a different `p` for each colony.

```
tmp <- swarm(apiary, p = c(0.3, 0.4, 0.5, 0.6, 0.7, 0.8))
```

If we now inspect the first and the second swarm, we see that each colony has a different percentage of workers that stayed or left.

```
tmp$swarm[[1]]
#> An object of class "Colony"
#> Id: 22
#> Location:
#> Queen: 12
#> Number of fathers: 10
#> Number of workers: 30
#> Number of drones: 0
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

```
tmp$swarm[[3]]
#> An object of class "Colony"
#> Id: 26
#> Location:
```

```
#> Queen: 14
#> Number of fathers: 10
#> Number of workers: 50
#> Number of drones: 0
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

If you want to track the genetics, you might want to assign the swarms back to the original apiary and create a new apiary from the remnants. However, if you want to track the position, the remnant actually stay in the original position and would hence be assigned back to the same apiary, while the swarm would be assigned to a new apiary or even be lost. Here, we will track the genetics and assign the swarm back to the original apiary and remnants to a new apiary.

```
apiary <- tmp$swarm
apiary <- buildUp(apiary)
apiary1 <- tmp$remnant
```

Split

Colony splitting is a common beekeeping technique to limit swarming. A percentage of workers, brood, and food stores are split away to create a new colony or combine two split. Old queen normally stays in the original (remnant) colony. We created a function `split()` that works on `Colony` and `Multicolony` objects (Figure 2). It accepts the `p` argument as a proportion of workers that will be split away for a new colony. The output of the function is a list of two `Colony` or `MultiColony` objects: remnant that contains the old queen, drones, and `(1-p)` workers; and split that doesn't contain a queen, but contains virgin queens and `p` workers. The `split()` function follows the same principles as the `swarm()`, hence we will limit explaining the outputs.

- Splitting a Colony

```
tmp <- split(colony, p = 0.3)
tmp
#> $split
#> An object of class "Colony"
#> Id: 34
#> Location:
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 29
#> Number of drones: 0
#> Number of virgin queens: 1
#> Has split: TRUE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
#>
#> $remnant
#> An object of class "Colony"
#> Id: 8
#> Location:
```

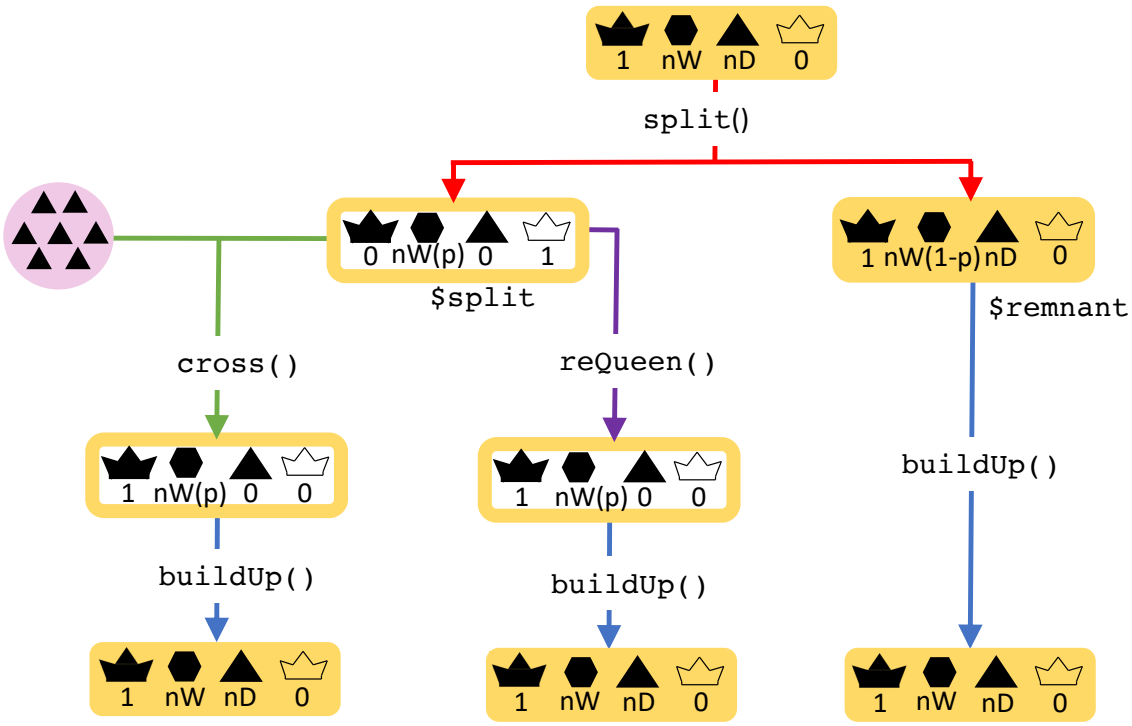


Figure 2: Split function

```
#> Queen: 11
#> Number of fathers: 10
#> Number of workers: 68
#> Number of drones: 100
#> Number of virgin queens: 0
#> Has split: TRUE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
```

We see that remnant contains the old queen and 70% of workers since we set `p` argument to 0.3, meaning that 30% of workers are removed in the split. The `split` status of the remnant colony is turned to `TRUE`. The `production` status of the remnant stays `TRUE`, because in reality we always split a colony in a way that does not threaten the production of the remnant colony.

If we inspect the split, we see that it contains 30% of the workers, the `split` status is turned to `TRUE`, and the `production` status is turned to `FALSE`, since in reality, these small colonies would not be productive.

We would usually consider the remnant as the original colony, since it tracks its genetics and location.

```
colony <- tmp$remnant
colony
#> An object of class "Colony"
#> Id: 8
#> Location:
#> Queen: 11
#> Number of fathers: 10
#> Number of workers: 68
#> Number of drones: 100
#> Number of virgin queens: 0
#> Has split: TRUE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
```

After a split, the colony would build-up back to their full-size.

```
colony <- buildUp(colony)
```

The `p` argument for splitting can be also saved in `SP` object, so we do not specify it each time we call the function - see `SimParamBee$splitP`.

- Splitting a MultiColony

We split a `MultiColony` object in the same way we split a `Colony` (as shown in swarm).

```
tmp <- split(apiary, p = 0.3)
tmp$remnant[[1]]
#> An object of class "Colony"
#> Id: 22
#> Location:
#> Queen: 12
#> Number of fathers: 10
#> Number of workers: 70
#> Number of drones: 100
#> Number of virgin queens: 0
```

```
#> Has split: TRUE
#> Has swarmed: TRUE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
```

We again see that in remnant we have the queen and 70 workers:

```
tmp$split[[1]]
#> An object of class "Colony"
#> Id: 35
#> Location:
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 30
#> Number of drones: 0
#> Number of virgin queens: 1
#> Has split: TRUE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: FALSE
```

and a virgin queen with 30 workers in split. We can use the vector of `p`, different for each colony same as shown for the `swarm()` function above.

After the split, we would assign the remnant colonies back to the apiary and build them up.

```
apiary <- tmp$remnant
apiary <- buildUp(apiary)
```

Supersedure

Supersedure is a replacement of the queen by one of her daughters without interference of the beekeeper. Supersedure is a natural way of re-queening a colony without swarming. There are many reasons for supersedure: poor physical condition of a queen, old age, diseases, depleted spermatheca, poorly bread queen, reduced pheromone output and many others (Hamdan, 2010).

Function `supersede()` removes the old queen and triggers the creation of new virgin queens from the brood (Figure 3). The function returns a single `Colony` or `MultiColony` object (not a list of two).

- Superseding a Colony

```
colony <- supersede(colony)
colony
#> An object of class "Colony"
#> Id: 8
#> Location:
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 92
#> Number of drones: 100
#> Number of virgin queens: 1
#> Has split: TRUE
#> Has swarmed: TRUE
#> Has superseded: TRUE
```

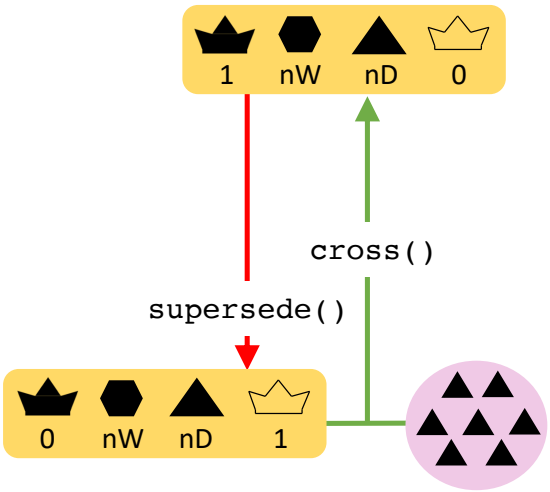


Figure 3: Supersede function

```
#> Has collapsed: FALSE
#> Is productive: TRUE
```

We see that after a supersedure, the old queen is removed and a new virgin queen is ready to mate. Hence, the next step in the simulation would be to cross this virgin queen. We also see that the `supersedure` status is set to `TRUE`. The `production` status stays set to `TRUE`, since the colony did not loose any individuals (it stayed at it's full-size).

- Superseding a MultiColony

The function `supersede()` works both on `Colony` and `MultiColony` classes. We supersede a `MultiColony` in a same way as a `Colony`.

```
apiary <- supersede(apiary)
apiary
#> An object of class "MultiColony"
#> Number of colonies: 6
#> Are empty: 0
#> Are NULL: 0
#> Have split: 6
#> Have swarmed: 6
#> Have superseded: 6
#> Have collapsed: 0
#> Are productive: 6
```

Collapse

Collapse of the colony is a term that describes the death of a colony - when all individuals within a colony die. There are many reasons for the collapse of a honey bee colony: diseases, starvation, queen problems, contamination with pesticides, etc. Colony losses can be high, possibly up to 60% of colonies per year. High

colony losses can significantly influence genetic structure of a population and hence genetic diversity and genetic gain.

Function `collapse()` simulates the collapse of a colony (Figure 4). This function does not remove individuals from the colony, but sets the `collapse` status of the colony to `TRUE` and `production` status to `FALSE`. This is to mimic reality, where bees would still be present in the colony, although being dead. This allows us to extract any genetic material from the colony even after collapse, say to study genetic causes related to the collapse. Future operations in terms of reproduction or simulation of events are not allowed with a collapsed colony.

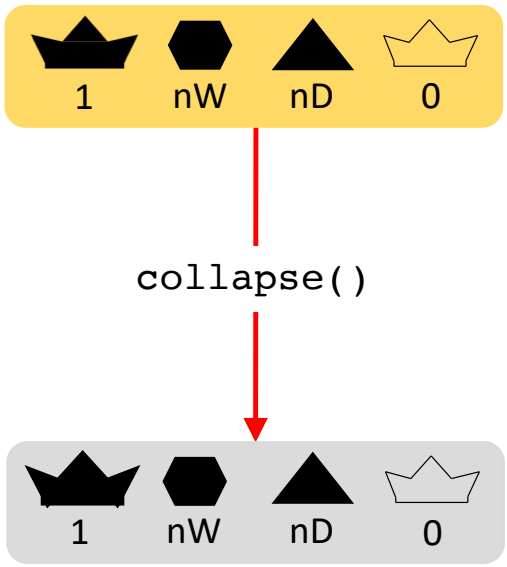


Figure 4: Collapse function

- Collapsing a Colony

```
colony <- collapse(colony)
colony
#> An object of class "Colony"
#> Id: 8
#> Location:
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 92
#> Number of drones: 100
#> Number of virgin queens: 1
#> Has split: TRUE
#> Has swarmed: TRUE
#> Has superseded: TRUE
#> Has collapsed: TRUE
#> Is productive: FALSE
```

- Collapsing a MultiColony

`collapse()` function is used when you want to keep collapsed colonies for subsequent analyses. If you don't need the collapsed colony, you can also simply select the surviving colonies with the `selectColonies()` function.

```
apiary <- collapse(apiary)
apiary[[3]]
#> An object of class "Colony"
#> Id: 26
#> Location:
#> Queen: NA
#> Number of fathers: 0
#> Number of workers: 100
#> Number of drones: 100
#> Number of virgin queens: 1
#> Has split: TRUE
#> Has swarmed: TRUE
#> Has superseded: TRUE
#> Has collapsed: TRUE
#> Is productive: FALSE
apiary
#> An object of class "MultiColony"
#> Number of colonies: 6
#> Are empty: 0
#> Are NULL: 0
#> Have split: 6
#> Have swarmed: 6
#> Have superseded: 6
#> Have collapsed: 6
#> Are productive: 0
```

References

Clemson Cooperative Extension (2021) Frequently asked questions about honey bee swarms. https://hgic.clemson.edu/factsheet/frequently-asked-questions-about-honey-bee-swarms/ (accessed on 2022-11-11)

Hamdan K. (2010) Natural Supersedure of Queens in Honey Bee Colonies. Bee World, 87(3):52-54. https://doi.org/10.1080/0005772X.2010.11417360

Rangel J., Seeley T. D. (2012) Colony fissioning in honey bees: size and significance of the swarm fraction. Insectes sociaux, 59(4):453-462. https://doi.org/10.1007/s00040-012-0239-5

Additional file 4 - Crossing vignette

2022-12-15

Introduction

This vignette shows how you can cross virgin queens in SIMplyBee. Here, we present how you can cross:

- single or multiple virgin queens (class `Pop`), virgin queen in a colony (class `Colony`), or all the virgin queens in an apiary or population (class `MultiColony`);
- cross either with pre-selected population of drones or according to a cross plan, and
- cross queens at an open drone congregation area (DCA) or at a mating station.

Start by loading the package:

```
library(package = "SIMplyBee")
#> Loading required package: AlphaSimR
#> Loading required package: R6
#>
#> Attaching package: 'SIMplyBee'
#> The following object is masked from 'package:base':
#>
#>     split
```

First, we create a founder population and some virgin queen, virgin colonies, and virgin apiaries that we will later cross.

```
# Simulate 40 founder genomes
founderGenomes <- quickHaplo(nInd = 50, nChr = 1, segSites = 100)
# Set global population paramaters
SP <- SimParamBee$new(founderGenomes)
# Create a base population of 40 virgin queens
basePop <- createVirginQueens(founderGenomes)

# Prepare populations with a single virgin queen
virginQueen1 <- basePop[1]
virginQueen2 <- basePop[2]
virginQueen3 <- basePop[3]
# Prepare populations with multiple virgin queens
virginQueens1 <- basePop[4:6]
virginQueens2 <- basePop[7:9]
virginQueens3 <- basePop[10:12]
# Prepare virgin Colony objects
colony1 <- createColony(basePop[13])
colony2 <- createColony(basePop[14])
colony3 <- createColony(basePop[15])
colony4 <- createColony(basePop[16])
# Prepare virgin MultiColony objects
apiary1 <- createMultiColony(basePop[17:21])
apiary2 <- createMultiColony(basePop[22:26])
```

```
apiary3 <- createMultiColony(basePop[27:31])
apiary4 <- createMultiColony(basePop[32:41])
```

We will now create a groups of drones from the remaining queens with 1,000 drones per queen that will represent a drone congregation area (DCA).

```
# Create a DCA from the remaining virgin queens
DCA <- createDrones(basePop[42:50], nInd = 1000)
```

Cross virgin queens at an open DCA

Cross by pre-selecting drone populations

We start by crossing our populations and colonies to pre-selected populations of drones. We pre-select the groups by pulling a desired number of drone packages from a DCA with the function `pullDroneGroupsFromDCA()`. This function requires you to specify a group of drones (`DCA`), how many groups you want to pull from the DCA (`n`), and how many drones per group you want (`nDrones`). For `nDrones`, you can either specify an integer or a sampling function, which results in a different number of drones in each of the pulled groups (you can read more about this in the Sampling functions vignette). These sampling functions are particularly useful in crossing simulations:

- `nFathersPoisson()`: samples the number of drones from a Poisson distribution with a default mean of 15 (the user can specify a different mean) - the output can contain the value 0 and
- `nFathersTruncPoisson()`: samples the number of drones from a zero truncated Poisson distribution with a default mean of 15 (the user can specif a different mean) - the output does not contain the value 0.

If these functions do not satisfy your needs, you can specify your own sampling function(s).

We can pull the drone groups out separately for each crossing or pull them out all at once.

```
# Pre-select drone (father) populations from a DCA
droneGroups <- pullDroneGroupsFromDCA(DCA, n = 20, nDrones = nFathersTruncPoisson)
```

Once we inspect these drone groups, we see there is a different number of drones in each group because we used a sampling function:

```
sapply(droneGroups, FUN = nInd)
#>  [1] 10 16 18 14 14  9 23 14 14 18 17 11 14 18 22 20 18 10 16 13
```

Now, we can cross our virgin queens to drone groups.

```
# A single virgin queen
virginQueen1 <- cross(virginQueen1, drones = droneGroups[[1]])
nFathers(virginQueen1)
#> [1] 10
```

```
# Multiple virgin queens
virginQueens1 <- cross(virginQueens1, drones = droneGroups[2:4])
nFathers(virginQueens1)
#> [1] 16 18 14
```

```
# A colony
colony1 <- cross(colony1, drones = droneGroups[[5]])
nFathers(colony1)
#> [1] 14
```

```
# An apiary
apiary1 <- cross(x = apiary1, drones = droneGroups[6:10])
nFathers(apiary1)
#>  5  6  7  8  9
#>  9 23 14 14 18
```

Cross according to a cross plan

Another option is to provide a cross plan with IDs of the virgin queens/colonies and drones, and a single drone population with all the drones listed in the cross plan. You can create a cross plan with the function `createRandomCrossPlan()`. This function creates a cross plan by randomly sampling a desired number of drone IDs from a DCA and assigning them to either virgin queen ID or colony ID. When crossing a virgin queen from a colony, you have to provide the colony ID, since there could be multiple virgin queens within the colony. In that case, the random selection of one virgin queen occurs within the `cross()` function. To create a cross plan you therefore have to provide the IDs of either the virgin queens or the colonies you want to cross (but not both in the same cross plan!!!), the drone population, and the number of drones you want to cross to a particular virgin queen. This can again be a fixed number or a sampling function. We can create a separate cross plan for each mating or create one combined cross plan for multiple matings (but can not have virgin queen and colony IDs in the cross plan at the same time!!!). Here, we again mate a single virgin queen, a population of virgin queens, virgin queens from a colony, and from an apiary.

```
# Create a combined cross for mating a single queen (virginQueen2) and a population
# of virgin queen (virginQueens2)
(crossPlanQueens <- createRandomCrossPlan(IDs = c(getId(virginQueen2),
                                                  getId(virginQueens2)),
                                          drones = DCA,
                                          nDrones = 15))
#> $`2`
#>  [1] "4542" "1760" "3616" "5276" "3105" "8024" "3146" "3164" "3656" "1261"
#> [11] "352"  "8976" "1007" "7894" "3500"
#>
#> $`7`
#>  [1] "6760" "4115" "5289" "1952" "2183" "883"  "7205" "4616" "7805" "938"
#> [11] "2495" "4245" "3620" "4812" "6895"
#>
#> $`8`
#>  [1] "2598" "7095" "5715" "3574" "7153" "3825" "3433" "8405" "3731" "2217"
#> [11] "4879" "3300" "4759" "5517" "834"
#>
#> $`9`
#>  [1] "7778" "4059" "3341" "8797" "5260" "4138" "802"  "6679" "2970" "3226"
#> [11] "5189" "1874" "2288" "2560" "1041"
```

We see that the created cross plan is a list with IDs corresponding to the virgin queen's IDs and the elements of each list being the IDs of the drones the virgin queen will mate with. Now we can cross these virgin queens by providing the `crossPlanQueens` to the `cross()` functions.

```
# Cross a single virgin queen
virginQueen2 <- cross(virginQueen2, drones = DCA, crossPlan = crossPlanQueens)
nFathers(virginQueen2)
#> [1] 15
# Cross multiple virgin queens
virginQueens2 <- cross(virginQueens2, drones = DCA, crossPlan = crossPlanQueens)
nFathers(virginQueens2)
#> [1] 15 15 15
```

As already mentioned, we need to create a separate plan for crossing virgin queens already in colonies, since here we need to provide colony IDs.

```
(crossPlanColonies <- createRandomCrossPlan(IDs = c(getId(colony2), getId(apiary2)),
                                            drones = DCA,
                                            nDrones = nFathersPoisson))
#> $`2`
#>  [1] "7907" "869"  "2862" "5255" "6954" "648"  "2174" "5448" "5654" "3151"
#> [11] "5723"
#>
#> $`10`
#>  [1] "1395" "5601" "7405" "7336" "8992" "5054" "3143" "4669" "2245" "2340"
#> [11] "6667" "474"  "1384" "2441" "5540" "6382" "8340" "1959"
#>
#> $`11`
#>  [1] "4896" "1859" "3259" "8021" "8072" "8630" "8301" "3699" "4572" "1090"
#> [11] "2799" "5356" "4800" "5500" "1527"
#>
#> $`12`
#>  [1] "2057" "6620" "3934" "3137" "913"  "1099" "1003" "1489" "4726" "1693"
#> [11] "111"  "3197" "7733"
#>
#> $`13`
#>  [1] "8113" "3951" "1248" "2618" "6107" "8772" "3437" "6433" "8158" "556"
#> [11] "287"  "1971" "2529" "1258" "7751" "245"  "1960" "6889" "1991" "4730"
#> [21] "1936" "5575"
#>
#> $`14`
#>  [1] "5899" "5545" "3734" "5483" "8555" "558"  "7735" "5694" "3087" "5556"
#> [11] "8361" "1191" "1004" "1409" "6354"
```

We again see that the cross plan is a list, now with colony IDs and elements of the list being the ID of drones the virgin queen within each colony will mate with. Now, we can cross our "colonies" by providing the `crossPlanColonies` to the `cross()` function.

```
# Cross a single colony
colony2 <- cross(colony2, drones = DCA, crossPlan = crossPlanColonies)
nFathers(colony2)
#> [1] 11
# Cross an apiary
apiary2 <- cross(x = apiary2, drones = DCA, crossPlan = crossPlanColonies)
nFathers(apiary2)
#> 10 11 12 13 14
#> 18 15 13 22 15
```

Cross virgin queens at a mating station

Mating virgin queens at a mating station is no different than mating them at an open DCA - the difference is in the DCA itself. In the case of open mating, the DCA consists of drones from multiple queens, all of which are usually unknown. In the case of a mating station, the DCA consists of drones coming from a sister group of drone producing queens (DPQ), the queen of which is known. This allows us to track the pedigree on the paternal side.

To simulate this situation, we first create a mating station DCA using `createMatingStationDCA()` function, which takes a single "sire" colony (queen of the DPQs). From the "sire" colony, the function first produces

a desired number of sister DPQs, and next produces a desired number of drones per DPQ. The produced drones represent the mating station's DCA.

```
# Create a DCA at a mating station from colony1
(matingStationDCA <- createMatingStationDCA(colony1, nDPQs = 20, nDronePerDPQ = 1000))
#> An object of class "Pop"
#> Ploidy: 2
#> Individuals: 20000
#> Chromosomes: 1
#> Loci: 100
#> Traits: 0
```

We see that the output of the function is a single population of 20,000 drones that represents the DCA. Once you have the DCA, you can cross virgin queens either by pulling out populations of drones or creating a mating plan as described above.

Here, we will mate a single colony (`colony3`) and a group of colonies (`apiary3`) at a mating station according to a cross plan.

```
# Mate only an apiary
crossPlanMatingStation <- createRandomCrossPlan(IDs = c(getId(colony3),
                                                        getId(apiary3)),
                                                drones = matingStationDCA,
                                                nDrones = nFathersTruncPoisson)
# Cross a colony
colony3 <- cross(colony3, crossPlan = crossPlanMatingStation, drones = matingStationDCA)
nFathers(colony3)
#> [1] 21
# Cross an apiary
apiary3 <- cross(apiary3, crossPlan = crossPlanMatingStation, drones = matingStationDCA)
nFathers(apiary3)
#> 15 16 17 18 19
#> 15  8  9 10 11
```

Cross virgin queens with different methods

It could happen that you have some virgin colonies in an apiary and you want to inseminate one of the virgin queens artificially with a single drone, take three of them to a mating station, and mate the rest of them openly at a local DCA. Since the cross plan is a named list, you can concatenate multiple cross plans into one. Let's mate the multicolony `apiary4` in such a manner.

```
# Create a single drone for single drone insemination
singleDrone = createDrones(colony2, nInd = 1)
# Create a cross plan for crossing some of the colonies in an open DCA,
# some with single drone, and some on a mating station
crossPlanApiary4 <- c(
  createRandomCrossPlan(IDs = getId(apiary4)[1],
                        drones = singleDrone,
                        nDrones = 1),
  createRandomCrossPlan(IDs = getId(apiary4)[2:6],
                        drones = DCA,
                        nDrones = nFathersTruncPoisson),
  createRandomCrossPlan(IDs = getId(apiary4)[7:10],
                        drones = matingStationDCA,
                        nDrones = nFathersTruncPoisson)
  )
```

```
apiary4 <- cross(apiary4,
                 crossPlan = crossPlanApiary4,
                 drones = c(singleDrone, DCA, matingStationDCA))
nFathers(apiary4)
#> 20 21 22 23 24 25 26 27 28 29
#>  1 17 10 19 16 12 14 20 13  8
```

Additional file 5 - Genomics vignette

2022-12-15

Introduction

This vignette demonstrates how SIMplyBee manages and manipulates the honey bee's genomic information. Specifically, it describes:

- how to obtain the genomic information,
- how to pool genotypes, and
- how to compute genomic relationship matrices.

Let's first create a colony.

```
library(package = "SIMplyBee")
#> Loading required package: AlphaSimR
#> Loading required package: R6
#>
#> Attaching package: 'SIMplyBee'
#> The following object is masked from 'package:base':
#>
#>     split
founderGenomes <- quickHaplo(nInd = 2, nChr = 3, segSites = 100)
SP <- SimParamBee$new(founderGenomes)
SP$setTrackRec(TRUE) # request recombination tracking

baseQueens <- createVirginQueens(founderGenomes)
baseDrones <- createDrones(x = baseQueens[1], nInd = 15)

colony <- createColony(x = baseQueens[2])
colony <- cross(colony, drones = baseDrones)
colony <- buildUp(colony)
```

Obtaining genomic information

Honeybees have a haplo-diploid inheritance system where queens and workers are diploid and drones are haploid. In SIMplyBee, we simulate drones as doubled-haploids, that is, as fully homozygous diploid individuals. This means that they have two identical sets of chromosomes. When they produce sperm, their gametes all have the same one set of chromosomes. Despite them being diploid, we generally return a haploid set of chromosomes from drones, unless specifically requested that you want the doubled-haploid genotype.

Following AlphaSimR, SIMplybee has a group of genome retrieval functions `get*Haplo/Geno()` which extract haplotypes and genotypes for all segregating sites (`SegSites`), quantitative trait loci (`QTL`), markers (`SNP`), and the identical by descent (`IBD`) haplotypes. Here, site, locus and marker are all synonyms for a position in the genome. These functions leverage AlphaSimR functionality, but work with SIMplyBee's `Colony` or `MultiColony` objects and in addition take the `caste` argument to extract information for a specific caste. Another argument you can use with this function is `collapse = TRUE/FALSE`. If `collapse = TRUE` then all of the information is collapsed together and a single matrix is returned, if `collapse = FALSE` we return a list by caste or by colony.

We recommend that you study the index of available `get*()` functions in SIMplyBee and read this vignette for a short demonstration.

```
help(SIMplyBee)
```

To show all this functionality, let's get haplotypes and genotypes across the segregating sites for the different castes using `getSegSitesGeno()` or `getSegSitesHaplo()`. The first row of the output shows marker identifications (chromosome_locus) and the first column shows haplotype identifications (individual_haplotype). The alleles are represented with a sequence of 0's and 1's. Let's first obtain the information at the segregating sites for the queen (we limit the output to the first 10 sites):

```
getSegSiteHaplo(colony, caste = "queen")[, 1:10]
#>     1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 2_1   1   1   1   0   0   1   0   1   1    0
#> 2_2   0   1   0   1   0   1   0   1   0    0
```

```
getSegSiteGeno(colony, caste = "queen")[, 1:10]
#>  1_1  1_2  1_3  1_4  1_5  1_6  1_7  1_8  1_9 1_10
#>    1    2    1    1    0    2    0    2    1    0
```

Now for the fathers:

```
getSegSiteHaplo(colony, caste = "fathers")[, 1:10]
#>      1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 16_1   0   1   1   1   1   0   1   1   0    0
#> 3_1    0   1   1   1   1   0   1   1   0    1
#> 4_1    1   1   1   1   0   1   1   1   0    1
#> 17_1   1   1   1   1   0   1   1   1   0    1
#> 6_1    1   1   1   1   0   1   1   1   0    1
#> 15_1   0   1   1   1   1   0   1   1   0    0
#> 10_1   0   1   1   1   1   0   1   1   0    0
#> 11_1   1   1   1   1   1   0   1   1   0    0
#> 14_1   0   1   1   1   1   0   1   1   0    0
#> 12_1   1   1   1   1   0   1   1   1   0    1
#> 7_1    1   1   1   1   0   1   1   1   0    1
#> 5_1    0   1   1   1   1   0   1   1   0    0
#> 8_1    0   1   1   1   1   0   1   1   0    0
#> 13_1   0   1   1   1   1   0   1   1   0    0
#> 9_1    1   1   1   1   0   1   1   1   0    0
```

```
getSegSiteGeno(colony, caste = "fathers")[, 1:10]
#>    1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 13   0   1   1   1   1   0   1   1   0    0
#> 4    1   1   1   1   0   1   1   1   0    1
#> 3    0   1   1   1   1   0   1   1   0    1
#> 17   1   1   1   1   0   1   1   1   0    1
#> 8    0   1   1   1   1   0   1   1   0    0
#> 6    1   1   1   1   0   1   1   1   0    1
#> 16   0   1   1   1   1   0   1   1   0    0
#> 15   0   1   1   1   1   0   1   1   0    0
#> 7    1   1   1   1   0   1   1   1   0    1
#> 5    0   1   1   1   1   0   1   1   0    0
#> 10   0   1   1   1   1   0   1   1   0    0
#> 12   1   1   1   1   0   1   1   1   0    1
#> 11   1   1   1   1   1   0   1   1   0    0
#> 9    1   1   1   1   0   1   1   1   0    0
#> 14   0   1   1   1   1   0   1   1   0    0
```

Since father are drones, and these are haploid, we get one row per father. We can retrieve the doublet-haploid (diploid implementation) state, if this is desired (showing just one father to show this clearly):

```
getSegSiteHaplo(colony, caste = "fathers",
                nInd = 1, dronesHaploid = FALSE)[, 1:10]
#>      1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 12_1   1   1   1   1   0   1   1   1   0    1
#> 12_2   1   1   1   1   0   1   1   1   0    1
```

```
getSegSiteGeno(colony, caste = "fathers",
               nInd = 1, dronesHaploid = FALSE)[, 1:10, drop = FALSE]
#>   1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 5   0   2   2   2   2   0   2   2   0    0
```

Now two workers:

```
getSegSiteHaplo(colony, caste = "workers", nInd = 2)[, 1:10]
#>      1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 22_1   1   1   1   0   0   1   0   1   1    0
#> 22_2   0   1   1   1   1   0   1   1   0    0
#> 92_1   1   1   1   0   0   1   0   1   1    0
#> 92_2   0   1   1   1   1   0   1   1   0    0
```

```
getSegSiteGeno(colony, caste = "workers", nInd = 2)[, 1:10]
#>    1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 59   2   2   2   1   0   2   1   2   1    1
#> 55   1   2   1   2   0   2   1   2   0    1
```

And finally four drones:

```
getSegSiteHaplo(colony, caste = "drones", nInd = 4)[, 1:10]
#>       1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 177_1   1   1   1   0   0   1   0   1   1    0
#> 191_1   0   1   0   1   0   1   0   1   0    0
#> 210_1   0   1   0   1   0   1   0   1   1    0
#> 171_1   0   1   0   1   0   1   0   1   0    0
```

```
getSegSiteGeno(colony, caste = "drones", nInd = 4)[, 1:10]
#>     1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 164   0   1   0   1   0   1   0   1   0    0
#> 129   0   1   0   1   0   1   0   1   0    0
#> 182   0   1   0   1   0   1   0   1   0    0
#> 121   1   1   1   0   0   1   0   1   1    0
```

You can also use `caste = "all"` to get the haplotypes and phenotypes from every individual in the colony. If the argument `collapse` is set to `FALSE`, then the function returns a list with haplotypes for each caste. Let's explore the structure of the output:

```
str(getSegSiteHaplo(colony, caste = "all", collapse = FALSE))
#> List of 5
#>  $ queen       : int [1:2, 1:300] 1 0 1 1 1 0 0 1 0 0 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:2] "2_1" "2_2"
#>   .. ..$ : chr [1:300] "1_1" "1_2" "1_3" "1_4" ...
#>  $ fathers     : int [1:15, 1:300] 1 0 0 0 1 1 1 0 0 1 ...
#>   ..- attr(*, "dimnames")=List of 2
```

```
#>   .. ..$ : chr [1:15] "6_1" "5_1" "13_1" "10_1" ...
#>   .. ..$ : chr [1:300] "1_1" "1_2" "1_3" "1_4" ...
#>  $ workers     : int [1:200, 1:300] 1 0 1 0 1 1 1 1 0 0 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:200] "63_1" "63_2" "49_1" "49_2" ...
#>   .. ..$ : chr [1:300] "1_1" "1_2" "1_3" "1_4" ...
#>  $ drones      : int [1:100, 1:300] 0 0 0 0 1 0 1 1 1 0 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:100] "153_1" "128_1" "137_1" "149_1" ...
#>   .. ..$ : chr [1:300] "1_1" "1_2" "1_3" "1_4" ...
#>  $ virginQueens: NULL
```

If the argument `collapse` is set to `TRUE`, the function returns a single matrix with haplotypes of all the individuals. The same behaviour is implemented for all the functions that extract genomic information

```
str(getSegSiteHaplo(colony, caste = "all", collapse = TRUE))
#>  int [1:317, 1:300] 1 0 1 0 0 1 0 1 1 0 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : chr [1:317] "2_1" "2_2" "4_1" "3_1" ...
#>   ..$ : chr [1:300] "1_1" "1_2" "1_3" "1_4" ...
```

```
getSegSiteHaplo(colony, caste = "all", collapse = TRUE)[1:10, 1:10]
#>      1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 2_1    1   1   1   0   0   1   0   1   1    0
#> 2_2    0   1   0   1   0   1   0   1   0    0
#> 9_1    1   1   1   1   0   1   1   1   0    0
#> 3_1    0   1   1   1   1   0   1   1   0    1
#> 17_1   1   1   1   1   0   1   1   1   0    1
#> 12_1   1   1   1   1   0   1   1   1   0    1
#> 8_1    0   1   1   1   1   0   1   1   0    0
#> 6_1    1   1   1   1   0   1   1   1   0    1
#> 10_1   0   1   1   1   1   0   1   1   0    0
#> 4_1    1   1   1   1   0   1   1   1   0    1
```

```
getSegSiteGeno(colony, caste = "all", collapse = TRUE)[1:10, 1:10]
#>    1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 2    1   2   1   1   0   2   0   2   1    0
#> 3    0   1   1   1   1   0   1   1   0    1
#> 9    1   1   1   1   0   1   1   1   0    0
#> 7    1   1   1   1   0   1   1   1   0    1
#> 14   0   1   1   1   1   0   1   1   0    0
#> 17   1   1   1   1   0   1   1   1   0    1
#> 5    0   1   1   1   1   0   1   1   0    0
#> 12   1   1   1   1   0   1   1   1   0    1
#> 10   0   1   1   1   1   0   1   1   0    0
#> 15   0   1   1   1   1   0   1   1   0    0
```

SIMplyBee also has shortcuts for these haplotype and genotype functions to make life a bit easier for the user:

- `getQueenSegSitesHaplo()`
- `getQueenSegSitesGeno()`
- `getFathersSegSitesHaplo()`
- `getFathersSegSitesGeno()`

- `getWorkersSegSitesHaplo()`
- `getWorkersSegSitesGeno()`
- `getDronesSegSitesHaplo()`
- `getDronesSegSitesGeno()`
- `getVirginQueensSegSitesHaplo()`
- `getVriginQueensSegSitesGeno()`

Similar aliases exist also for extracting information about quantitative trait loci (`QTL`), markers (`SNP`), and the identical by descent (`IBD`) haplotypes.

Pooling genotypic information

Unfortunately, in real life it's challenging to get the genotype of every individual honeybee and so SIMplyBee provides the function `getPooledGeno()` to imitate real life data. `getPooledGeno()` returns a pooled genotype from individual genotypes to mimic the genotyping of a pool of colony members. A comparison of pooled and individual genotypes also allows the user to compare the two and see the impact of pooled samples on results.

Firstly let's obtain the genotypes of the workers and of the queen so that they're easier to work with:

```
genoQ <- getSegSiteGeno(colony, caste = "queen")
genoW <- getSegSiteGeno(colony, caste = "workers")
```

The function `getPooledGeno()` required also the sex of individuals whose genotype are getting pooled (`F` for females and `M` for males).

```
sexW <- getCasteSex(colony, caste = "workers")
```

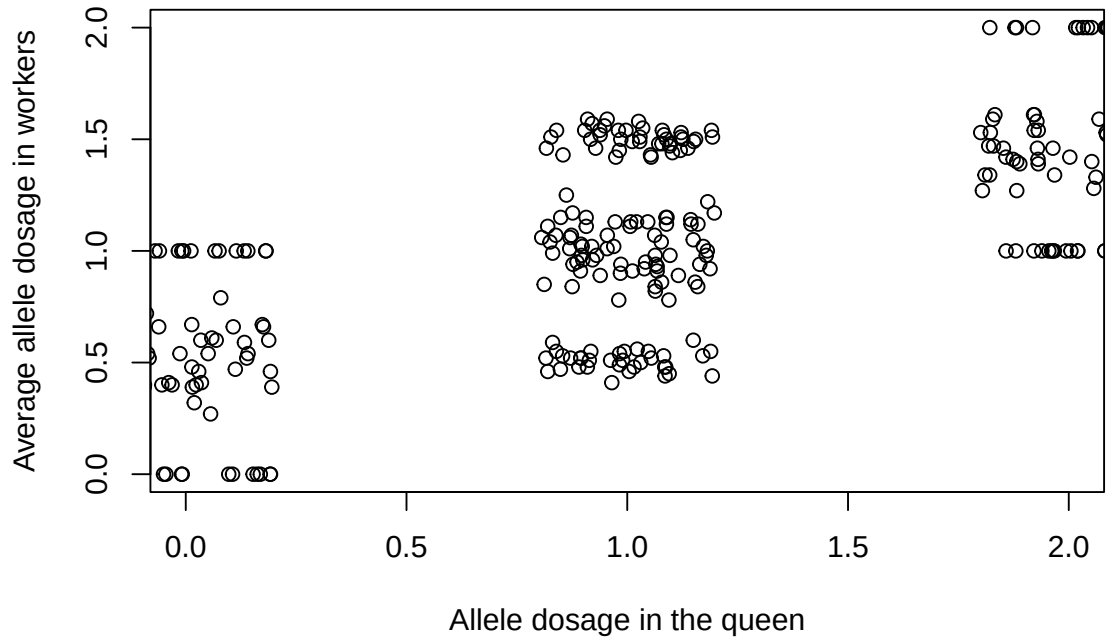
You have two options when choosing what kind of pooled genotypes you would like, using the `type =` argument. You can use `type = "mean"` for the average genotypes and `type = "count"` for the counts of reference and alternative alleles.

```
getPooledGeno(x = genoW, type = "count", sex = sexW)[, 1:10]
#>   1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_10
#> 0  98   0  47  55 141  59 100   0 146  160
#> 1 102 200 153 145  59 141 100 200  54   40
```

```
(poolW <- getPooledGeno(x = genoW, type = "mean", sex = sexW))[, 1:10]
#>  1_1  1_2  1_3  1_4  1_5  1_6  1_7  1_8  1_9 1_10
#> 1.02 2.00 1.53 1.45 0.59 1.41 1.00 2.00 0.54 0.40
```

Now lets plot and compare the pooled workers to the queen's genotype (note the use of jitter for queen's genotype on the x-axis so we can spread out the dots in the plot!).

```
plot(y = poolW, x = jitter(genoQ), ylim = c(0, 2), xlim = c(0, 2),
     ylab = "Average allele dosage in workers",
     xlab = "Allele dosage in the queen" )
```



Computing Genomic Relationship Matrices

This section introduces the calculations of IBD and IBS genomic relationship matrices, so let's have a quick reminder of what these mean. Identity-by-state (IBS) is a term used when two alleles, two segments or sequences of the genome are identical. Identity-by-descent (IBD) is when a segment of matching (IBS) DNA shared by two or more individuals has been inherited from a common ancestor.

Using IBD and IBS can allow a user to look into the relationships based on the genomic data. We'll demonstrate this by calculating some Genomic Relationship Matrices (GRM) using SIMplyBee's `calcBeeGRMIbs()` and `calcBeeGRMIbd()`.

Let's look at the `calcBeeGRMIbs()` first. This function returns a Genomic Relatedness Matrix (GRM) for honeybees from IBS genomic data (bi-allelic SNP represented as allele dosages) following the method for the sex X chromosome (Druet and Legarra, 2020).

To see this, let's obtain the genotypes and sex information of all individuals in the colony.

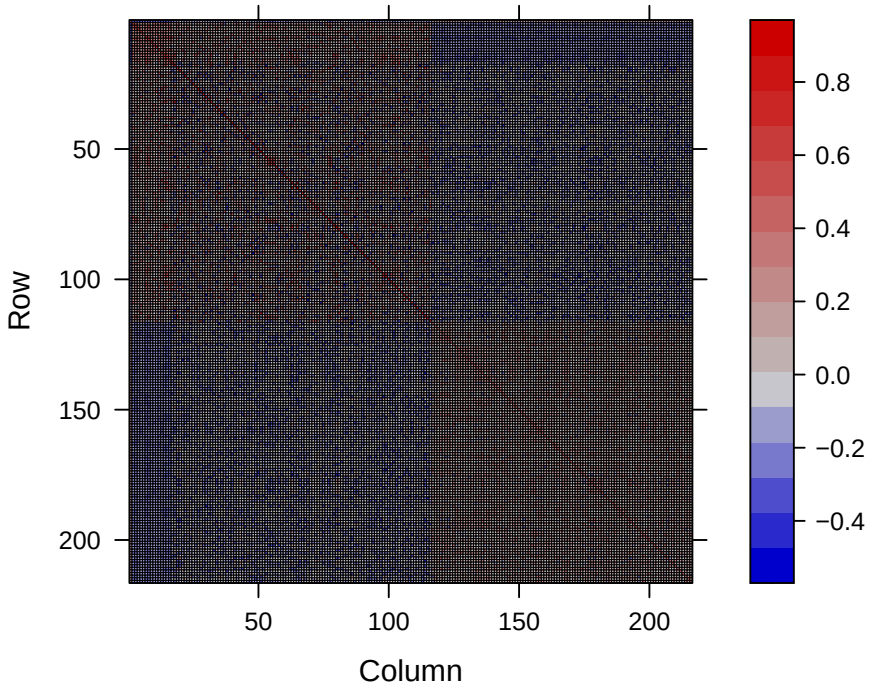
```
geno <- getSegSiteGeno(colony, collapse = TRUE)
sex <- getCasteSex(x = colony, collapse = TRUE)
```

Now let's calculate the IBS GRM, we will use the genotypes to calculate this:

```
GRM <- calcBeeGRMIbs(x = geno, sex = sex)
```

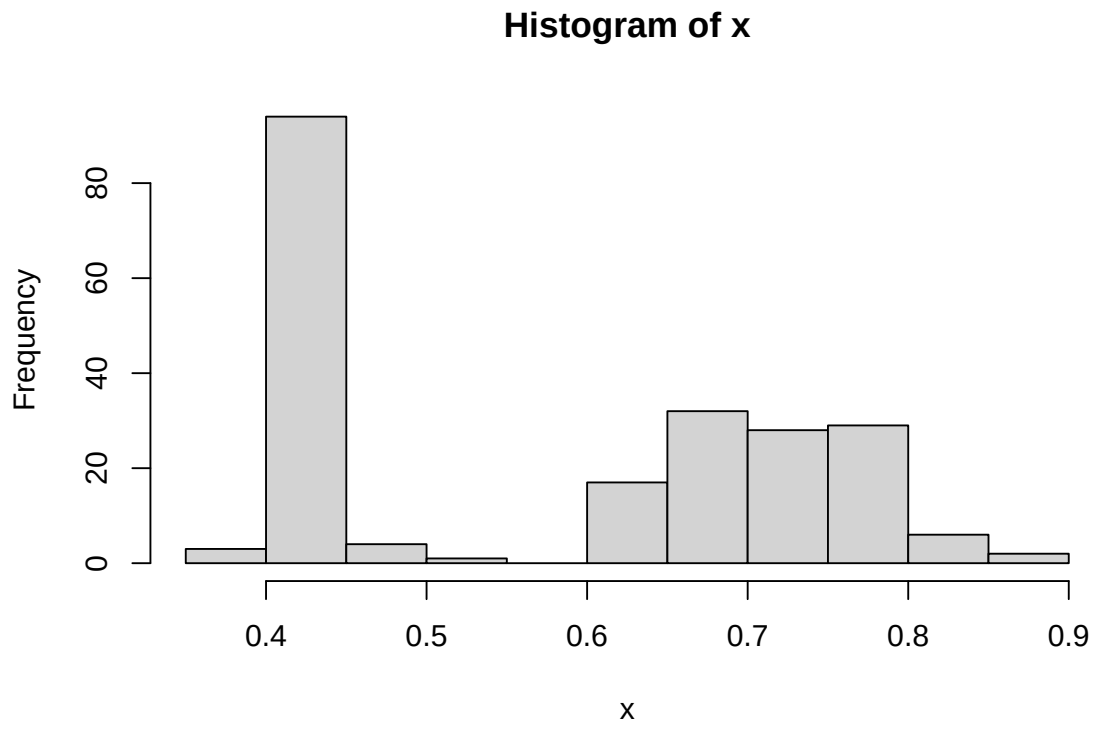
This produces a matrix that we can plot and summarise - its useful to summarise diagonal and off-diagonal values separately.

```
library("Matrix")
image(as(GRM, "Matrix"))
```



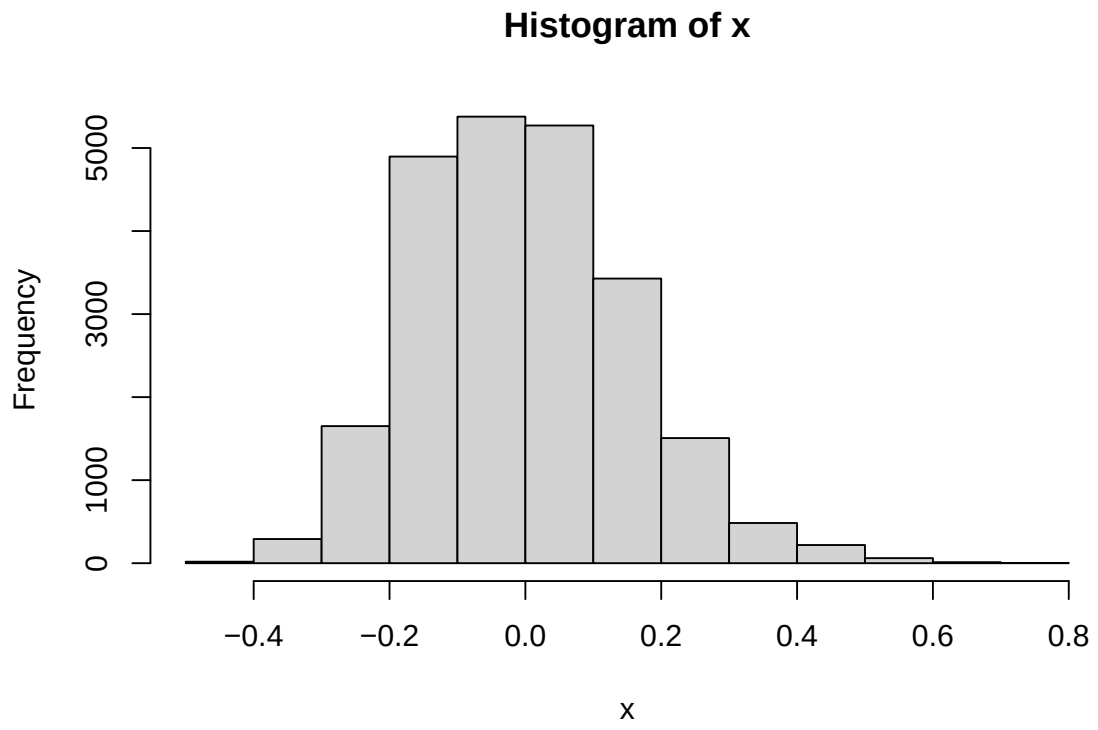
Dimensions: 216 x 216

```
x <- diag(GRM)
hist(x)
```




```
summary(x)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.3766  0.4280  0.6229  0.5793  0.7223  0.8750
```

```
x <- GRM[lower.tri(x = GRM, diag = FALSE)]
hist(x)
```

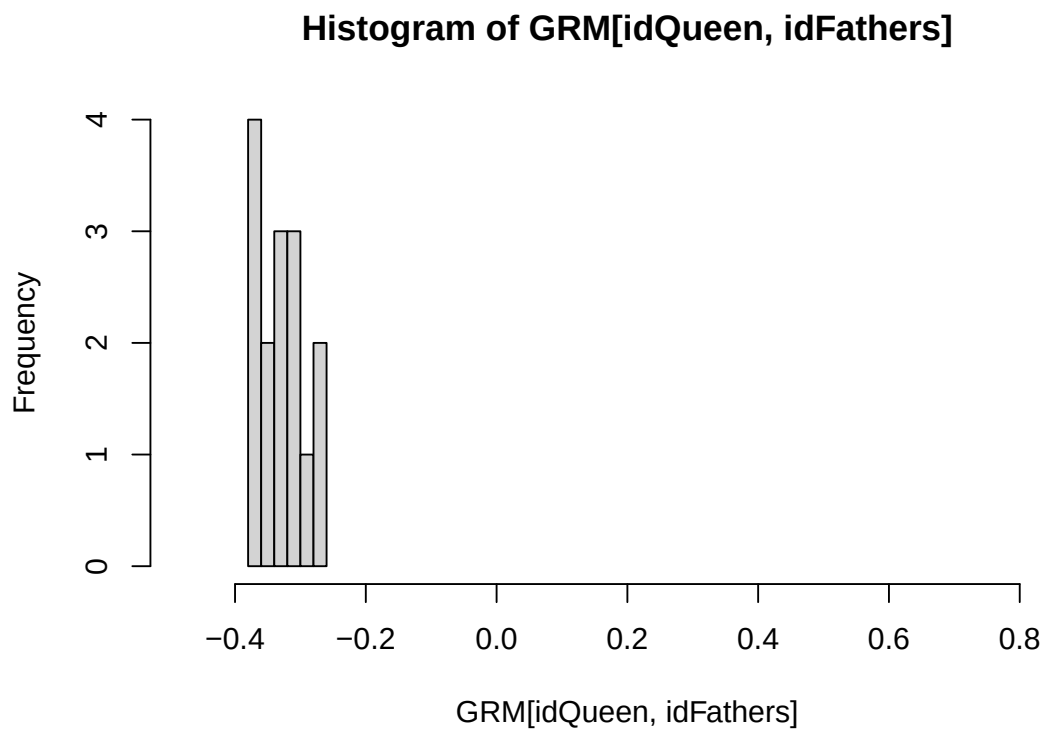



```
summary(x)
#>      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
#> -0.475355 -0.120799 -0.011382 -0.002694  0.097534  0.728822
```

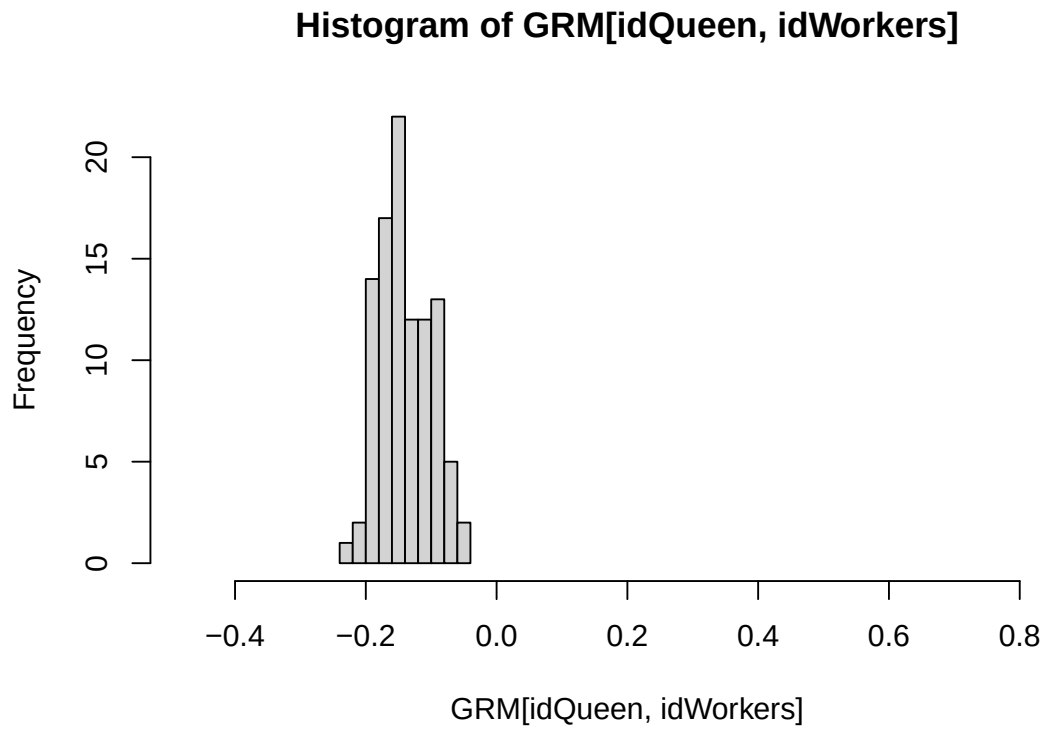
We can also inspect GRM elements between specific caste members:

```
ids <- getCasteId(colony)
idQueen <- ids$queen
idFathers <- ids$fathers
idWorkers <- ids$workers
idDrones <- ids$drones
idVirginQueens <- ids$virginQueens
mw <- "mw"
md <- "md"
```

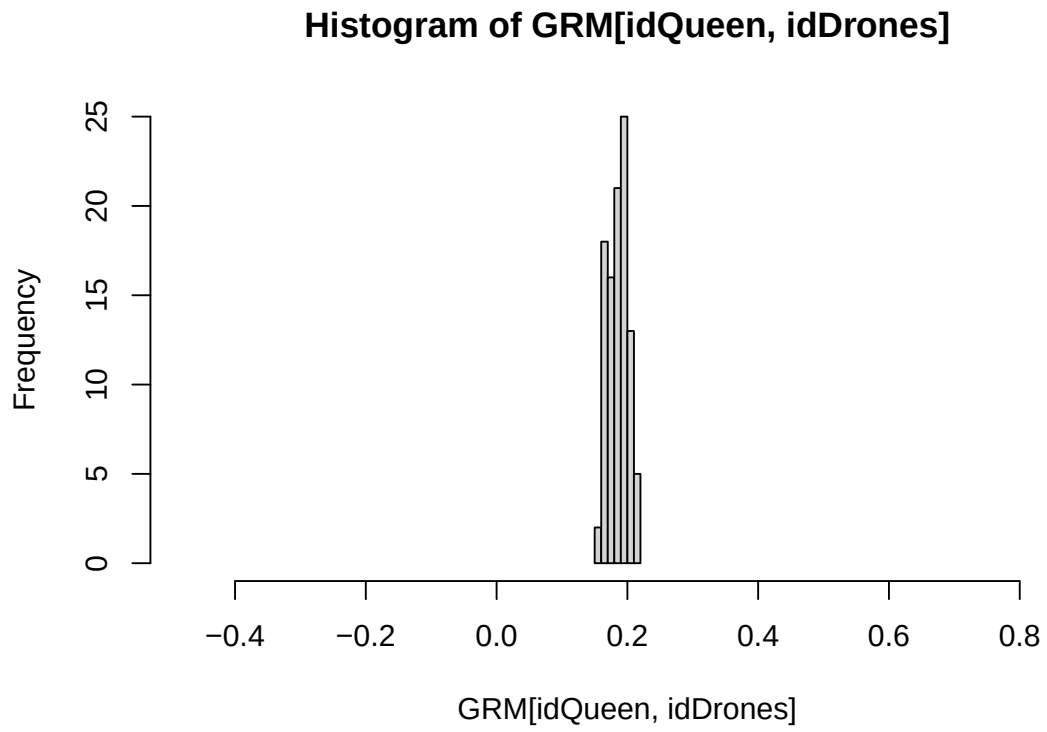
```
r <- range(GRM)
hist(GRM[idQueen, idFathers], xlim = r)
```




```
hist(GRM[idQueen, idWorkers], xlim = r)
```

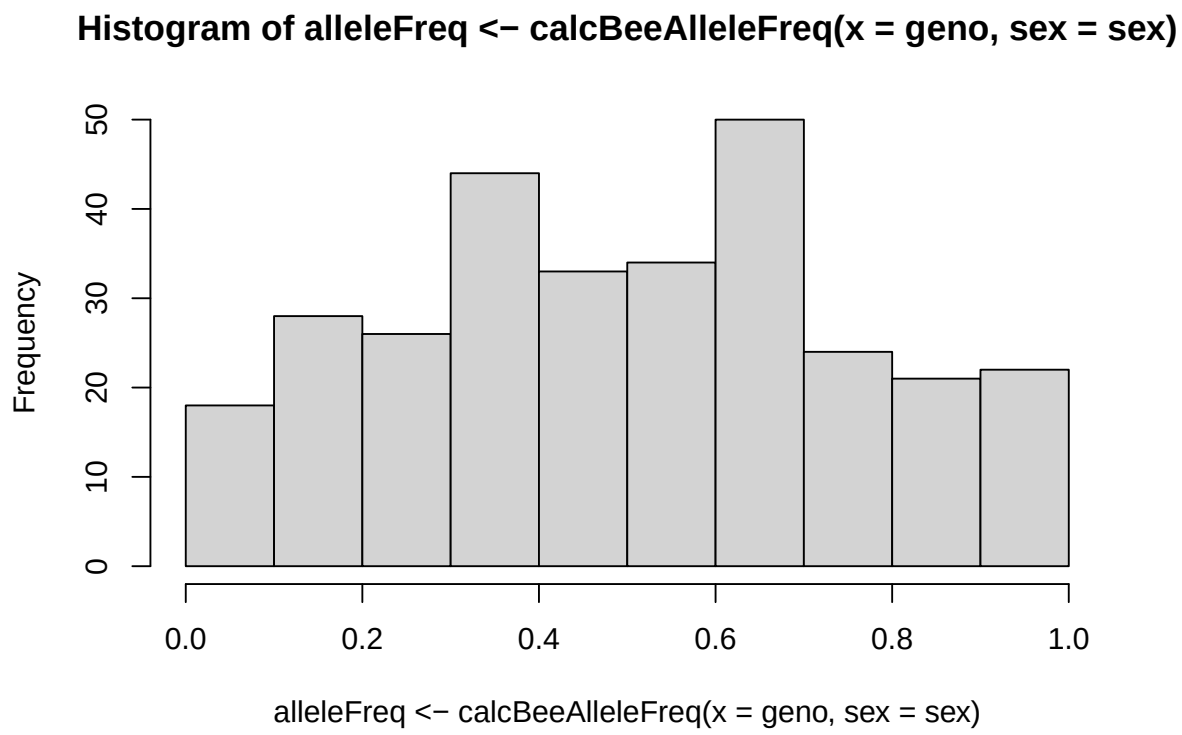



```
hist(GRM[idQueen, idDrones], xlim = r)
```



`calcBeeGRMIbs()` uses the `calcBeeAlleleFreq()` function to calculate allele frequencies for centering the honeybee genotypes. You can also use it in some other cases:

```
hist(alleleFreq <- calcBeeAlleleFreq(x = geno, sex = sex))
```



Now lets look at `calcBeeGRMIbd()`. This function creates Genomic Relatedness Matrix (GRM) for honeybees based on Identical-By-Descent (IBD) information. It returns a list with a matrix of gametic relatedness coefficients (between genomes) and a matrix of individual relatedness coefficients (between individuals). Please refer to Grossman and Eisen (1989), Fernando and Grossman (1989), Fernando and Grossman (1990), Van Arendonk, Tier, and Kinghorn (1994), and Hill and Weir (2011) for the background on this function.

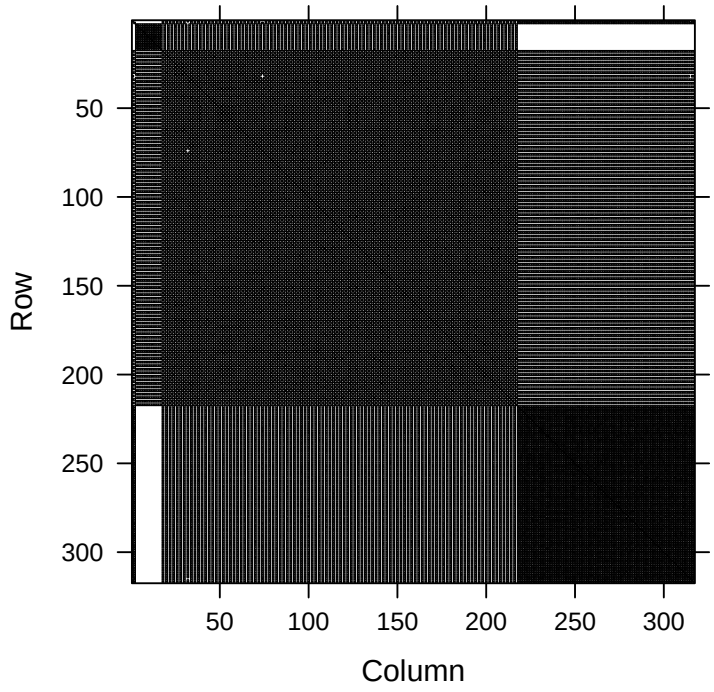
Now obtain the IBD haplotypes and compute IBD GRM.

```
haploQ <- getQueenIbdHaplo(colony)
haploF <- getFathersIbdHaplo(colony)
haploW <- getWorkersIbdHaplo(colony)
haploD <- getDronesIbdHaplo(colony)
haploV <- getVirginQueensIbdHaplo(colony)

haplo <- rbind(haploQ, haploF, haploW, haploD, haploV)
```

```
GRMs <- calcBeeGRMIbd(x = haplo)
```

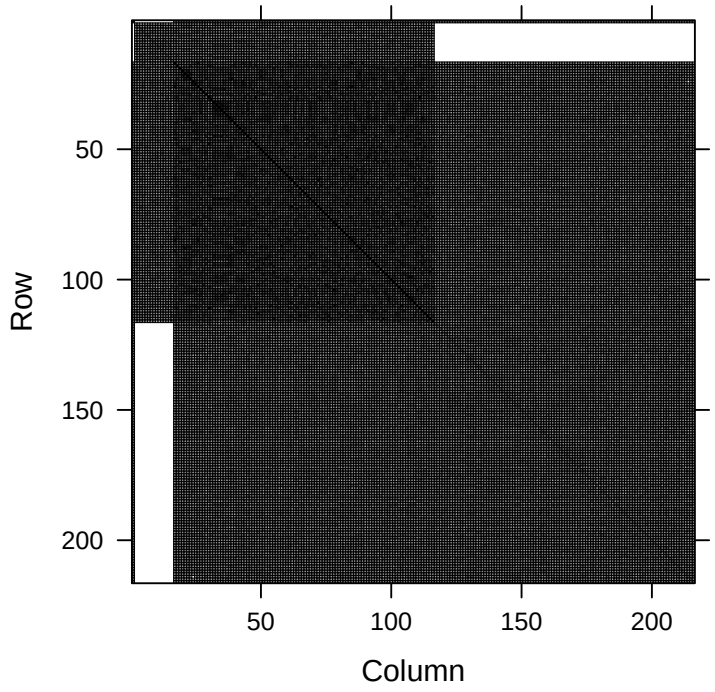
Let's view this matrix:

```
image(as(GRMs$genome, "Matrix"))
```



Dimensions: 317 x 317

```
image(as(GRMs$indiv, "Matrix"))
```



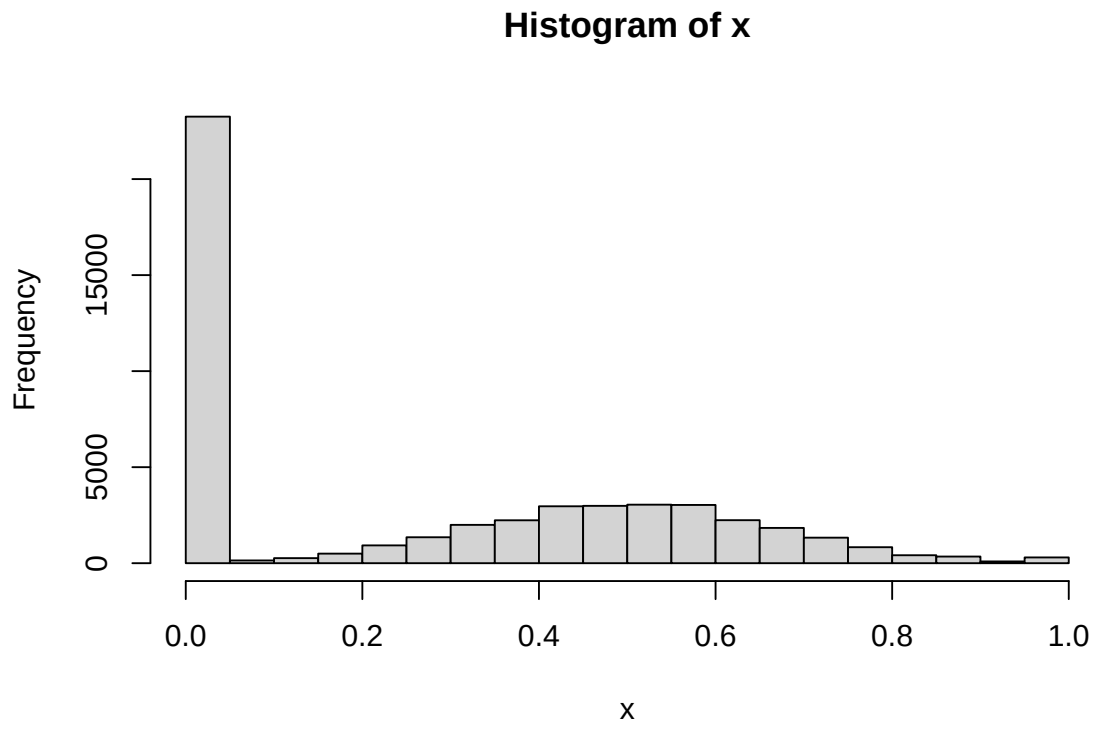
Dimensions: 216 x 216

Now we can look at the diagonal of the obtained matrices that represent 1 for a genomes and 1 + inbreeding coefficient individuals.

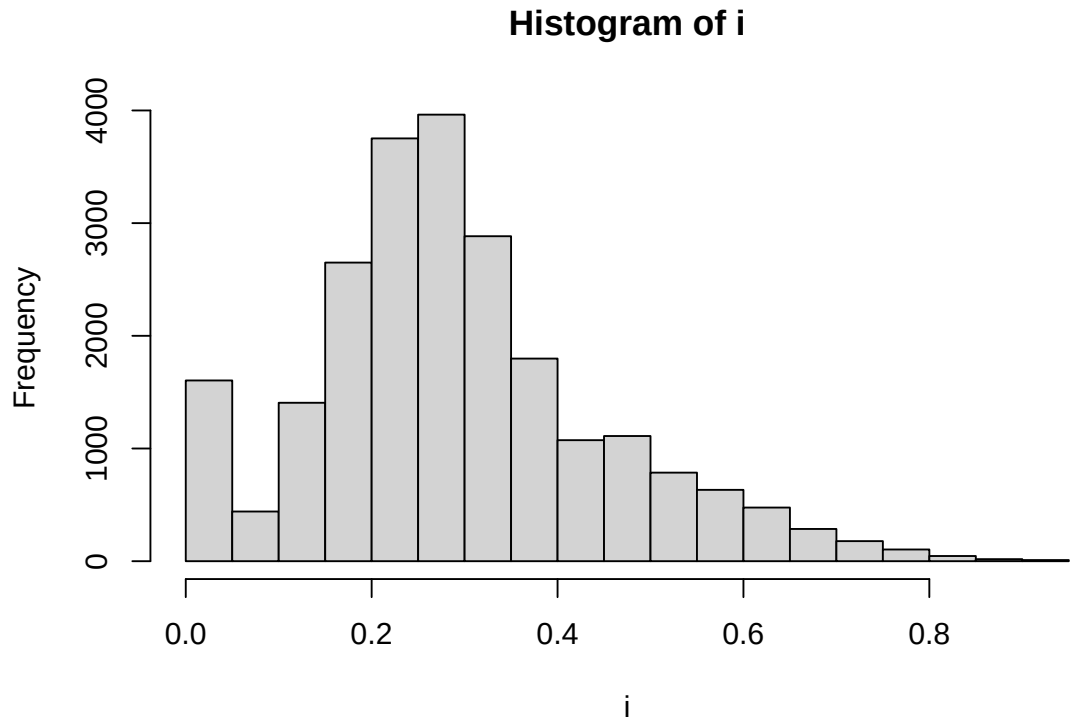
```
i <- diag(GRMs$genome)
summary(x)
#>      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
#> -0.475355 -0.120799 -0.011382 -0.002694  0.097534  0.728822

i <- diag(GRMs$indiv)
summary(i)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.5000  0.5000  0.5000  0.7338  1.0000  1.0000
```

And now the non-diagonals that represent the coefficients of relationship between genomes or between individuals.

```
x <- GRMs$genome[lower.tri(x = GRMs$genome, diag = FALSE)]
hist(x)
```




```
summary(x)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.0000  0.0000  0.2500  0.2703  0.5167  1.0000

i <- GRMs$indiv[lower.tri(x = GRMs$indiv, diag = FALSE)]
hist(i)
```



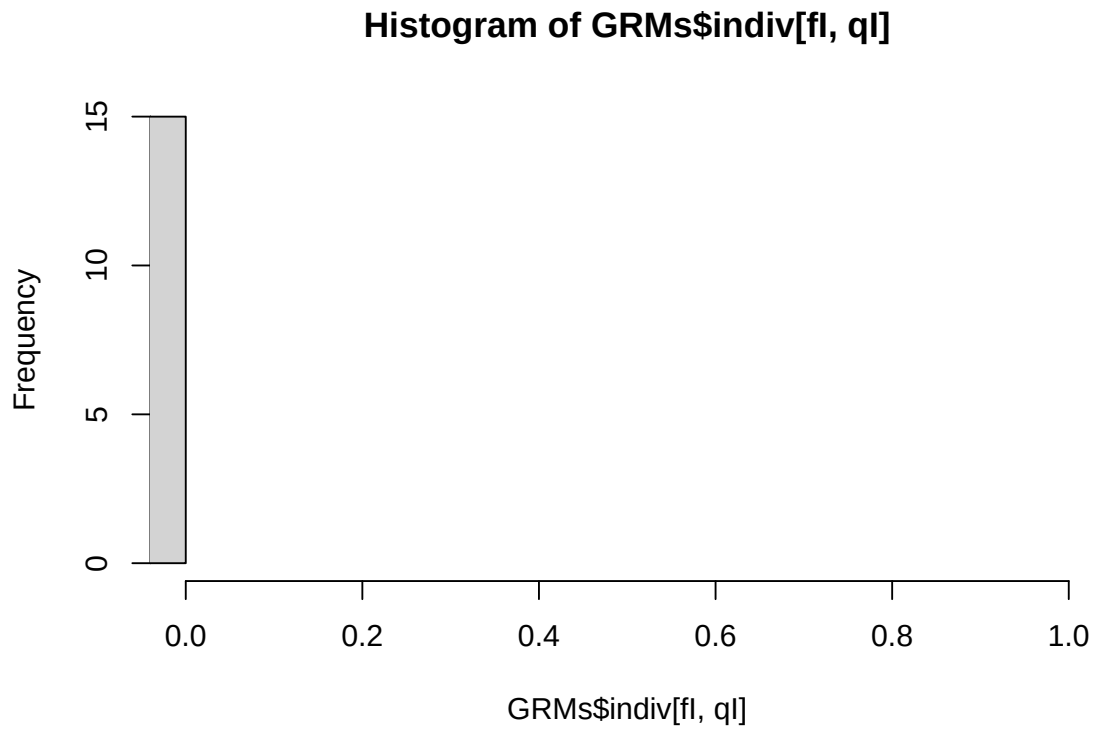

```
summary(i)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.0000  0.1950  0.2733  0.2915  0.3683  0.9500
```

Let's now compare compare relationships between caste members within a colony.

```
# Obtains caste member IDs
qI <- getQueen(colony)@id
fI <- sort(getFathers(colony)@id)
wI <- sort(getWorkers(colony)@id)
dI <- sort(getDrones(colony)@id)
r <- range(GRMs$indiv)
```

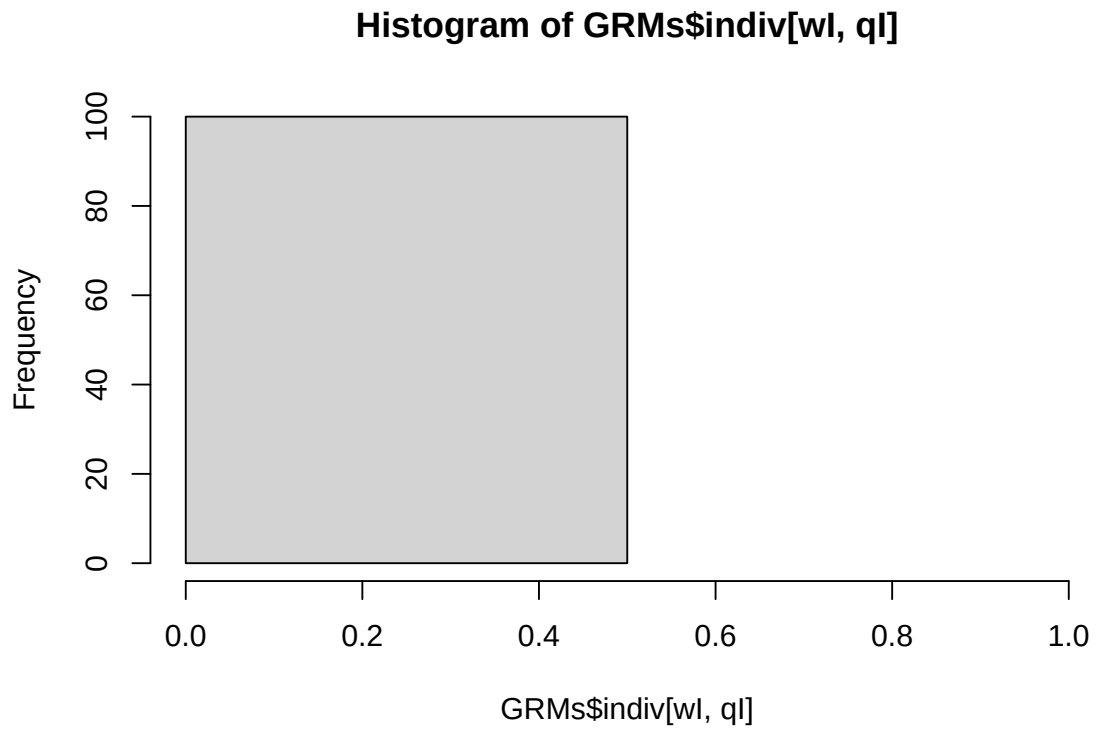
Compare queen and fathers:

```
hist(GRMs$indiv[fI, qI], xlim = r)
```



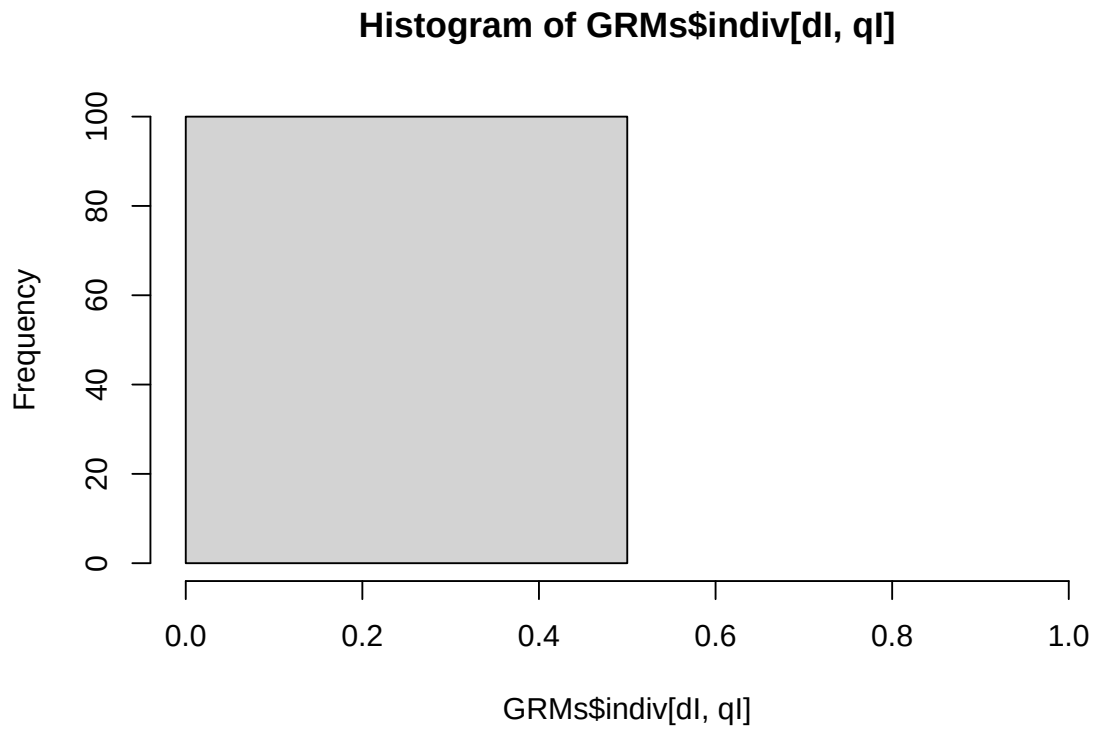
Queen and workers:

```
hist(GRMs$indiv[wI, qI], xlim = r)
```



Queen and drones:

```
hist(GRMs$indiv[dI, qI], xlim = r)
```



References

Druet and Legarra (2020) Theoretical and empirical comparisons of expected and realized relationships for the X-chromosome. Genetics Selection Evolution, 52:50. https://doi.org/10.1186/s12711-020-00570-6

Grossman and Eisen (1989) Inbreeding, coancestry, and covariance between relatives for X-chromosomal loci. The Journal of Heredity, 80(2):137–142. https://doi.org/10.1093/oxfordjournals.jhered.a110812

Fernando and Grossman (1989) Covariance between relatives for X-chromosomal loci in a population in disequilibrium. Theoretical and Applied Genetics, 77:311–319. https://doi.org/10.1007/bf00305821

Fernando and Grossman (1990) Genetic evaluation with autosomal and X-chromosomal inheritance. Theoretical and Applied Genetics, 80:75–80. https://doi.org/10.1007/bf00224018

Van Arendonk, Tier, and Kinghorn (1994) Use of multiple genetic markers in prediction of breeding values. Genetics, 137(1):319–329. https://doi.org/10.1093/genetics/137.1.319

Hill and Weir (2011) Variation in actual relationship as a consequence of Mendelian sampling and linkage. Genetics Research, 93(1):47–64. https://doi.org/10.1017/s0016672310000480

Additional file 6 - Quantitative genetics vignette

2022-12-15

Introduction

This vignette describes and demonstrates how SIMplyBee implements quantitative genetics principles for honeybees. Specifically, it describes three different examples where we simulate:

1. Honey yield - a single colony trait,
2. Honey yield and Calmness - two colony traits, and
3. Colony strength and Honey yield - two colony traits where one trait impacts the other one via the number of workers.

We start by loading SIMplyBee and quickly simulating genomes for some founder honeybees. Specifically, we will simulate genomes for 20 individuals with 16 chromosomes and 1000 segregating sites per chromosome.

```
library(package = "SIMplyBee")
#> Loading required package: AlphaSimR
#> Loading required package: R6
#>
#> Attaching package: 'SIMplyBee'
#> The following object is masked from 'package:base':
#>
#>     split
library(package = "ggplot2")
founderGenomes <- quickHaplo(nInd = 20, nChr = 16, segSites = 1000)
```

Honey yield

This section shows how to simulate one colony trait, honey yield, that is influenced by the queen and workers as well as the environment. We will achieve this by:

a. setting base population quantitative genetic parameters,
b. inspecting individual values in the base population,
c. inspecting individual values in a colony,
d. calculating colony value,
e. calculating multi-colony values, and
f. selecting on colony values.

Base population quantitative genetic parameters

AlphaSimR, and hence SIMplyBee, simulates each individual with its corresponding genome, and quantitative genetic and phenotypic values. To enable this simulation, we must set base population quantitative genetic parameters for the traits of interest in the global simulation parameters via `SimParamBee`. We must set:

1) the number of traits,
2) the number of quantitative trait loci (QTL) that affect the traits,
3) the distribution of QTL effects,

4) trait means, and
5) trait genetic and environmental variances - if we simulate multiple traits, we must also specify genetic and environmental covariances between the traits.

In honeybees, the majority of traits are influenced by the queen and workers. There are many biological mechanisms for these queen and workers effects. Depending on which caste is the main driver of the trait (the queen or workers), we also talk about direct and indirect effects. For example, for honey yield, workers directly affect honey yield by foraging, while the queen indirectly affects honey yield by stimulating workers via pheromone production. The queen and workers effects for a trait can be genetically and environmentally independent or correlated (usually negatively).

Here, we will simulate two traits to represent the queen and workers effects on honey yield. From this point onward we will use the terms the queen effect and queen trait interchangeably. The same applies to workers effect and workers trait. These two effects (=traits) will give rise to honey yield trait. We will assume that colony honey yield is approximately normally distributed with the mean of 20 kg and variance of 4 kg^2, which implies that most colonies will have honey yield between 14 kg and 26 kg (see `hist(rnorm(n = 1000, mean = 20, sd = sqrt(4)))`). Traits like honey yield have a complex polygenic genetic architecture, so we will assume that this trait is influenced by 100 QTL per chromosome (with 16 chromosomes, this gives us 1600 QTL in total).

We will first initiate global simulation parameters and set the mean of queen effects to 10 kg with genetic variance of 1 kg^2, while we will set the mean of workers effects to 10 kg with genetic variance of 1 kg^2. The mean and the variance for the worker effect are proportionally scaled by the expected number of workers in a colony. The mean and variance for the queen effect is assumed larger than for the workers effect, because there is one queen and many workers in colony and we assume that workers effects "accumulate". Deciding how to split the colony mean between queen and workers effects will depend on the individual to colony mapping function, which we will describe in the Colony value sub-section.

```
# Global simulation parameters
SP <- SimParamBee$new(founderGenomes)

nQtlPerChr <- 100

# Genetic parameters for queen and workers effects - each represented by a trait
mean <- c(10, 10 / SP$nWorkers)
varA <- c(1, 1 / SP$nWorkers)
```

We next set genetic correlation between the queen and workers effects to -0.5 to reflect the commonly observed antagonistic relationship between these effects. With all the quantitative genetic parameters defined, we now add two additive traits to global simulation parameters and name them `queenTrait` and `workerTrait`. These parameters drive the simulation of QTL effects. Read about all the other trait simulation options in AlphaSimR via: `vignette(topic = "traits", package="AlphaSimR")`.

```
corA <- matrix(data = c( 1.0, -0.5,
                        -0.5,  1.0), nrow = 2, byrow = TRUE)
SP$addTraitA(nQtlPerChr = nQtlPerChr, mean = mean, var = varA, corA = corA,
             name = c("queenTrait", "workersTrait"))
```

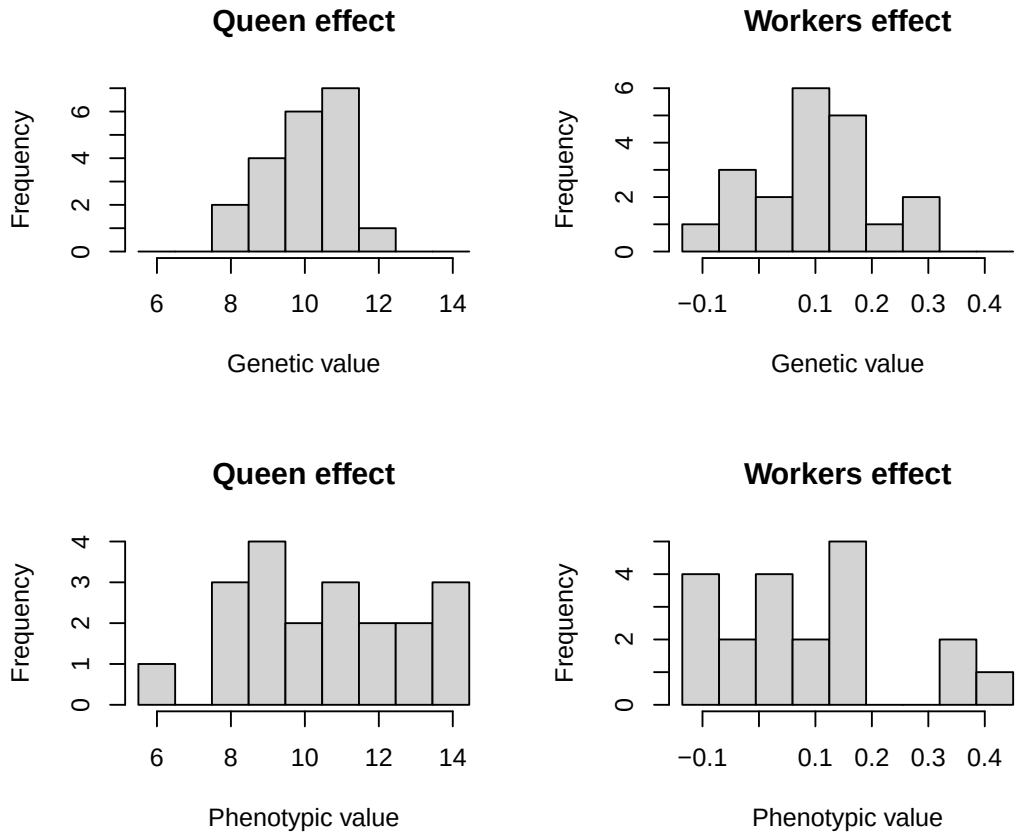
Finally, we set the environmental variance of the queen and workers effects to 3 kg^2 and we again scale the worker variance by the expected number of workers. Contrary to the negative genetic correlation, we here assume that environmental correlation between the queen and workers effects is slightly positive, 0.3. This is just an example! These parameters should be based on literature or simulation scenarios of interest.

```
varE <- c(3, 3 / SP$nWorkers)
corE <- matrix(data = c(1.0, 0.3,
                        0.3, 1.0), nrow = 2, byrow = TRUE)
SP$setVarE(varE = varE, corE = corE)
```

Individual values in the base population

Now we create a base population of virgin queens. Since we defined two traits, all honeybees in the simulation will have genetic and phenotypic values for both traits. The genetic values are stored in the `gv` slot of each `Pop` object, while phenotypic values are stored in the `pheno` slot.

```
#>       queenTrait workersTrait
#> [1,]  10.788385   0.18465787
#> [2,]  10.929577  -0.05781942
#> [3,]  10.727263   0.22147453
#> [4,]  10.292713   0.04216888
#> [5,]   9.384368   0.14211954
#> [6,]   8.711415   0.26930206
#>       queenTrait workersTrait
#> [1,]  13.917449  0.375823283
#> [2,]  13.532658  0.049644589
#> [3,]  10.450736 -0.030217317
#> [4,]   9.228794 -0.006395749
#> [5,]  10.263605  0.176029007
#> [6,]   5.498231  0.050032906
```

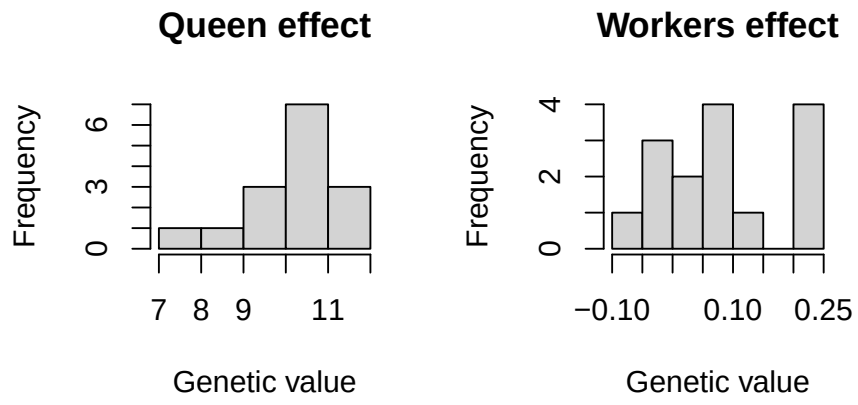


Note that these are virgin queens, yet we obtained queen and workers effect values for them! Is this wrong? No! Virgin queens carry DNA with genes that are differentially expressed in different castes, which would be only showed in their phenotype. Hence, virgin queens have genetic values for the queen and worker effects, but they might never actually express these effects. In this simulation virgin queens also obtained phenotypic

values for both of the effects. This is technically incorrect because virgin queens don't express genes for the worker effect at all, and they also do not express the queen effect, not until they become the queen of a colony. We can treat these phenotypic values for virgin queens as values that we could see if these virgin queens would express these traits. We will show later in the Colony value sub-section how we use these traits from different castes. If existence of these phenotypic values for certain castes is a hindrance, we can always remove them for population or colony objects by modifying the corresponding slots as required.

As with the virgin queens, drones also carry DNA with genes that are expressed in different castes. Therefore, drones will also have the queen and workers effect genetic (and phenotypic values) for honey yield even though they do not contribute to this trait in a colony.

```
#>      queenTrait workersTrait
#> [1,]   9.938580  0.042295154
#> [2,]  10.854940  0.233638511
#> [3,]  11.290960  0.061548334
#> [4,]   8.680186  0.095505665
#> [5,]  10.766090 -0.005435782
#> [6,]  10.306705 -0.020331858
```



Individual values in a colony

We continue by creating a colony from one base population virgin queen, crossing it, and adding some workers.

```
colony <- createColony(x = basePop[6])
colony <- cross(x = colony, drones = drones)
colony <- addWorkers(x = colony, nInd = 50)
colony
#> An object of class "Colony"
#> Id: 1
#> Location:
#> Queen: 6
#> Number of fathers: 15
#> Number of workers: 50
#> Number of drones: 0
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
```

```
#> Has collapsed: FALSE
#> Is productive: FALSE
```

We can access the genetic and phenotypic values of colony members with functions `getGv()` and `getPheno()`, both of which have the `caste` argument (see more via `help(getGv)`).

```
getGv(colony, caste = "queen")
#>   queenTrait workersTrait
#> 6   8.711415    0.2693021
getGv(colony, caste = "workers") |> head(n = 4)
#>    queenTrait workersTrait
#> 36   9.093787  0.168633369
#> 37   9.434525  0.189607063
#> 38   9.573273  0.282235542
#> 39  11.107806  0.009656202

getPheno(colony, caste = "queen")
#>   queenTrait workersTrait
#> 6   5.498231   0.05003291
getPheno(colony, caste = "workers") |> head(n = 4)
#>    queenTrait workersTrait
#> 36   3.528048   0.04455145
#> 37   9.656141   0.35082489
#> 38   8.893331   0.28633569
#> 39   9.263199  -0.13089175
```

For convenience, there are also alias functions for accessing the genetic and phenotypic values of each caste directly.

```
getQueenGv(colony)
#>   queenTrait workersTrait
#> 6   8.711415    0.2693021
getWorkersGv(colony) |> head(n = 4)
#>    queenTrait workersTrait
#> 36   9.093787  0.168633369
#> 37   9.434525  0.189607063
#> 38   9.573273  0.282235542
#> 39  11.107806  0.009656202

getQueenPheno(colony)
#>   queenTrait workersTrait
#> 6   5.498231   0.05003291
getWorkersPheno(colony) |> head(n = 4)
#>    queenTrait workersTrait
#> 36   3.528048   0.04455145
#> 37   9.656141   0.35082489
#> 38   8.893331   0.28633569
#> 39   9.263199  -0.13089175
```

Some phenotypes, such as honey yield, are only expressed if colony is at full size. This is achieved by the `buildUp()` colony event function that adds worker and drones and hence turns on the `production` status of the colony (to `TRUE`). SIMplyBee includes a function `isProductive()` to check the production status of a colony.

```
# Check if colony is productive
isProductive(colony)
```

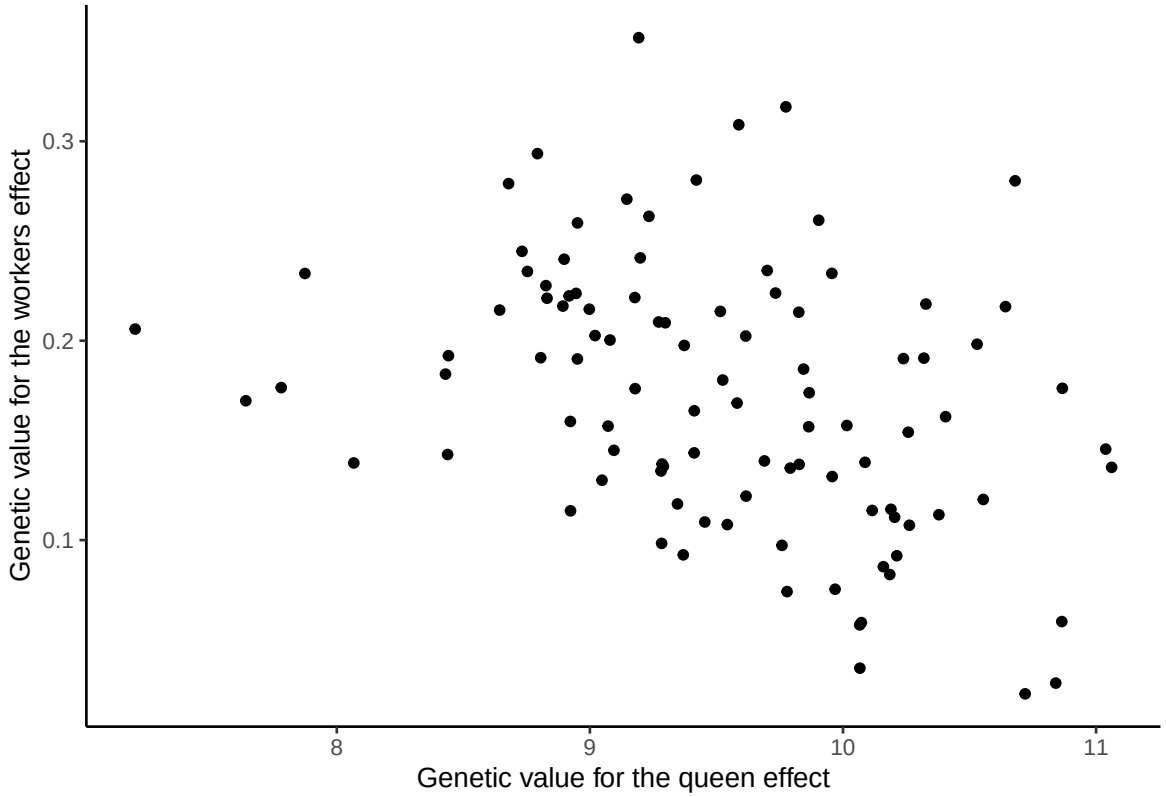
```
#> [1] FALSE

# Build-up the colony and check the production status again
colony <- buildUp(colony)
colony
#> An object of class "Colony"
#> Id: 1
#> Location:
#> Queen: 6
#> Number of fathers: 15
#> Number of workers: 100
#> Number of drones: 100
#> Number of virgin queens: 0
#> Has split: FALSE
#> Has swarmed: FALSE
#> Has superseded: FALSE
#> Has collapsed: FALSE
#> Is productive: TRUE
isProductive(colony)
#> [1] TRUE
```

For the ease of further demonstration, we now combine workers' values into a single data.frame.

```
# Collate genetic and phenotypic values of workers
df <- data.frame(id = colony@workers@id,
                 mother = colony@workers@mother,
                 father = colony@workers@father,
                 gvQueenTrait = colony@workers@gv[, "queenTrait"],
                 gvWorkersTrait = colony@workers@gv[, "workersTrait"],
                 pvQueenTrait =  colony@workers@pheno[, "queenTrait"],
                 pvWorkersTrait = colony@workers@pheno[, "workersTrait"])
head(df)
#>   id mother father gvQueenTrait gvWorkersTrait pvQueenTrait pvWorkersTrait
#> 1 86      6     29     9.543084     0.10777863    10.293754     -0.1009606
#> 2 87      6     32     9.779104     0.07416383     7.501888      0.2059583
#> 3 88      6     22     9.199141     0.24152505     8.632661      0.5426398
#> 4 89      6     28     8.917791     0.22245858     6.425176      0.2531411
#> 5 90      6     34     9.192954     0.35193982     7.557333      0.4963278
#> 6 91      6     31     9.689607     0.13970435     8.715310      0.1888455
```

To visualise correlation between queen and workers effects in workers, we plot these effect values against each other.

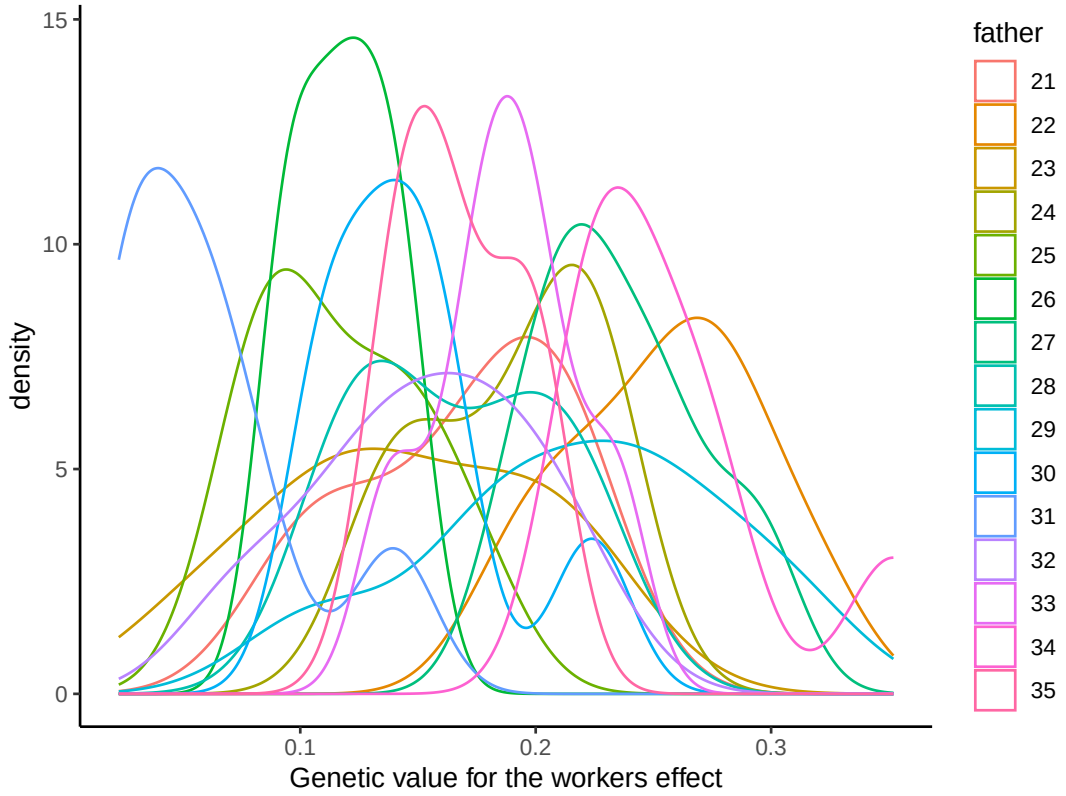
```
# Covariation between queen and workers effect genetic values in workers
p <- ggplot(data = df, aes(x = gvQueenTrait, y = gvWorkersTrait)) +
  xlab("Genetic value for the queen effect") +
  ylab("Genetic value for the workers effect") +
  geom_point() +
  theme_classic()
print(p)
```



In SIMplyBee, we know genetic values of all individuals, including drones that the queen mated with (=fathers in a colony)!

```
# Variation in patriline genetic values
getFathersGv(colony)
#>    queenTrait workersTrait
#> 21   9.938580  0.042295154
#> 22  10.854940  0.233638511
#> 23  11.290960  0.061548334
#> 24   8.680186  0.095505665
#> 25  10.766090 -0.005435782
#> 26  10.306705 -0.020331858
#> 27   9.122339  0.233144933
#> 28  10.032936  0.132336360
#> 29  10.486507  0.208555785
#> 30  10.498850 -0.002344818
#> 31  11.340227 -0.092942020
#> 32  11.521382  0.024364225
#> 33   7.488110  0.051033702
#> 34   9.378083  0.235324488
#> 35  10.605020  0.093297513
```

Knowing the father of each worker, we inspect variation in the distribution of genetic values of worker by the patriline (workers from a single father drone) for the workers effect.



Colony value

However, in honeybees we usually don't observe values on individuals, but on a colony. SIMplyBee provides functions for mapping individual values to a colony value. The general function for this is `calcColonyValue()`, which can combine any value and trait from any caste. There are also aliases `calcColonyGv()` and `calcColonyPheno()`. These functions require users to specify the so-called mapping function (via the `FUN` argument). The mapping function specifies queen and workers traits (potentially also drone traits) and what function we want to apply to each of them before mapping them to the colony value(s). We can also specify whether the colony value(s) depend on the production status. For example, if a colony is not productive, its honey yield would be 0 or unobserved. SIMplyBee provides a general mapping function `mapCasteToColonyValue()` and aliases `mapCasteToColonyGv()` and `mapCasteToColonyPheno()`. These functions have arguments to cater for various situations. By default, they first calculate caste values: leave the queen's value as it is, sum workers' values, potentially sum drones' values, and lastly sum all these caste values together into a colony value. Users can provide their own mapping function(s) too!

We now calculate honey yield for our colony - a single value for the colony.

```
# Colony phenotype value
calcColonyPheno(colony, queenTrait = "queenTrait", workersTrait = "workersTrait")
#>          [,1]
#> [1,] 22.56731
help(calcColonyPheno)
help(mapCasteToColonyPheno)
```

These colony values are not stored in a colony, because they change as colony changes due to various events. For example, reducing the number of workers will reduce the colony honey yield.

```
# Colony phenotype value from a reduced colony
removeWorkers(colony, p = 0.5) |>
  calcColonyPheno(queenTrait = "queenTrait", workersTrait = "workersTrait")
#>          [,1]
#> [1,] 11.18727
```

Please note that we assumed that the queen contributes half to colony honey yield and workers contribute the other half. This means that removing workers will still give a non-zero honey yield! This shows that we have to design the mapping between individual, caste, and colony values with care!

```
# Colony phenotype value from a reduced colony
removeWorkers(colony, p = 0.99) |>
  calcColonyPheno(queenTrait = "queenTrait", workersTrait = "workersTrait")
#>          [,1]
#> [1,] 5.986204
```

Finally, note that SIMplyBee currently does not provide functionality for breeding values, dominance deviations, and epistatic deviations at caste and colony levels, despite the availabiliy of AlphaSimR `bv()`, `dd()`, and `aa()` functions. This is because we have to check or develop theory on how to calculate these values across active colonies and hence we currently advise against the use of AlphaSimR `bv()`, `dd()`, and `aa()` functions with SIMplyBee as the output of these functions could be easily misinterpreted.

MultiColony values

The same functions can be used on a `MultiColony` class object. Let's create an apiary.

```
apiary <- createMultiColony(basePop[7:20])
drones <- createDrones(basePop[1:5], nInd = 100)
droneGroups <- pullDroneGroupsFromDCA(drones, n = nColonies(apiary), nDrones = 15)
apiary <- cross(apiary, droneGroups)
apiary <- buildUp(apiary)
```

We can extract the genetic and phenotypic values from multiple colonies in the same manner as from a single colony, by using `get*Gv()` and `get*Pheno()` functions. The output of these function is a named list with values for each colony or a single matrix if we set the `collapse` argument to `TRUE`.

```
getQueenGv(apiary) |> head(n = 4)
#> $`2`
#>   queenTrait workersTrait
#> 7   9.305151    0.1412799
#>
#> $`3`
#>   queenTrait workersTrait
#> 8   11.54926   0.09952709
#>
#> $`4`
#>   queenTrait workersTrait
#> 9     8.1036     0.278294
#>
#> $`5`
#>    queenTrait workersTrait
#> 10   8.925681    0.1522489
getQueenGv(apiary, collapse = TRUE) |> head(n = 4)
#>    queenTrait workersTrait
#> 7    9.305151   0.14127993
#> 8   11.549261   0.09952709
```

```
#> 9    8.103600   0.27829399
#> 10   8.925681   0.15224885
```

In a similar manner, we can calculate colony value for all the colonies in our apiary, where the row names of the output represent colony IDs.

```
colonyGv <- calcColonyGv(apiary)
colonyPheno <- calcColonyPheno(apiary)
data.frame(colonyGv, colonyPheno)
#>    colonyGv colonyPheno
#> 2  20.82047   18.680565
#> 3  20.76831   21.628231
#> 4  24.57669   25.744683
#> 5  19.14472   21.761066
#> 6  17.20801   16.473422
#> 7  22.52309   23.442279
#> 8  19.48094   15.751048
#> 9  11.08495   12.919046
#> 10 20.32237   21.316871
#> 11 22.92625   21.111922
#> 12 14.47827   11.128967
#> 13 18.09847   21.082192
#> 14 11.35714    8.597806
#> 15 23.57225   25.261447
```

Selection on colony values

Since the aim of selection is to select the best individuals or colonies for the reproduction, we could select the best colony in our apiary based on either genetic or phenotypic value for grafting the new generation of virgin queens. We can use the function `selectColonies()` that takes a matrix of colony values (the output of `calcColonyValue()` function). The default behavior is to select the colonies with the highest value (argument `selectTop` set to `TRUE`), but you can also select the colonies with the lowest values (argument `selectTop` set to `FALSE`).

```
# Select the best colony based on gv
selectColonies(apiary, n = 1, by = colonyGv)
#> An object of class "MultiColony"
#> Number of colonies: 1
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 0
# Select the best colony based on phenotype
selectColonies(apiary, n = 1, by = colonyPheno)
#> An object of class "MultiColony"
#> Number of colonies: 1
#> Are empty: 0
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
```

```
#> Are productive: 0
```

The same functionality is implemented in `pullColonies()` and `removeColonies()`.

Honey yield and Calmness

In this section we expand simulation to two uncorrelated colony traits with queen and workers effects, honey yield and calmness. We follow the same recipe as in the previous section where we simulated only one colony trait.

We first reinitialize the global simulation parameters because we will define new traits. For honey yield we will use the same parameters as before, while for calmness trait we will assume that the trait is scored continuously in such a way that negative values are undesirable and positive values are desirable with zero being population mean. We will further assume the same variances for calmness as for honey yield, and a genetic (and environmental) correlation between the queen and workers effects of -0.4 (and 0.2) for calmness. We assume no genetic or environmental correlation between honey yield and calmness. Beware, this is just an example to show you how to simulate multiple colony traits - we have made up these parameters - please use literature estimates in your simulations!

```
# Global simulation parameters
SP <- SimParamBee$new(founderGenomes)

nQtlPerChr <- 100

# Quantitative genetic parameters - for two traits, each with the queen and workers effects
meanP <- c(10, 10 / SP$nWorkers, 0, 0)
varA <- c(1, 1 / SP$nWorkers, 1, 1 / SP$nWorkers)
corA <- matrix(data = c( 1.0, -0.5,  0.0,  0.0,
                        -0.5,  1.0,  0.0,  0.0,
                         0.0,  0.0,  1.0, -0.4,
                         0.0,  0.0, -0.4,  1.0), nrow = 4, byrow = TRUE)
SP$addTraitA(nQtlPerChr = 100, mean = meanP, var = varA, corA = corA,
             name = c("yieldQueenTrait", "yieldWorkersTrait",
                      "calmQueenTrait", "calmWorkersTrait"))

varE <- c(3, 3 / SP$nWorkers, 3, 3 / SP$nWorkers)
corE <- matrix(data = c(1.0, 0.3, 0.0, 0.0,
                        0.3, 1.0, 0.0, 0.0,
                        0.0, 0.0, 1.0, 0.2,
                        0.0, 0.0, 0.2, 1.0), nrow = 4, byrow = TRUE)
SP$setVarE(varE = varE, corE = corE)
```

We continue by creating a base population of virgin queens and from them an apiary with 10 full-sized colonies.

```
basePop <- createVirginQueens(founderGenomes)
drones <- createDrones(x = basePop[1:5], nInd = 100)
apiary <- createMultiColony(basePop[6:20])
droneGroups <- pullDroneGroupsFromDCA(drones, nColonies(apiary), nDrones = 15)
apiary <- cross(x = apiary, drones = droneGroups)
apiary <- buildUp(apiary)
apiary
#> An object of class "MultiColony"
#> Number of colonies: 15
#> Are empty: 0
```

```
#> Are NULL: 0
#> Have split: 0
#> Have swarmed: 0
#> Have superseded: 0
#> Have collapsed: 0
#> Are productive: 15
```

We can again inspect the genetic (and phenotypic) values of all individuals in each colony and whole apiary with `get*Gv()` and `get*Pheno()` functions. Now, the output contains four traits representing the queen and workers effect for honey yield and calmness. These functions also take an `nInd` argument to sample a number of individuals along with their values.

```
getQueenGv(apiary) |> head(n = 4)
#> $`1`
#>   yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 6        9.540833        0.09555984       1.394206     -0.001118209
#>
#> $`2`
#>   yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 7        9.469601         0.2713463      -1.156479        0.0667444
#>
#> $`3`
#>   yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 8        9.374259         0.2227225       1.319143      0.001601704
#>
#> $`4`
#>   yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 9        8.981484       -0.002116586       1.089913       -0.1551117
getWorkersPheno(apiary, nInd = 3) |> head(n = 4)
#> $`1`
#>     yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 521        10.68829         0.4492840     -0.3032404        0.1680091
#> 522        12.00711         0.1531743      1.4555905       -0.3103395
#> 523        10.50278         0.2923529      0.5945246       -0.3491241
#>
#> $`2`
#>     yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 721       10.462600        0.26374262      -3.029606        0.4142178
#> 722        9.608215        0.05049904      -1.941136        0.5086108
#> 723        9.592244        0.18510651      -1.488955       -0.3405637
#>
#> $`3`
#>     yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 921        9.192794        -0.1022689      -2.318974       0.06265507
#> 922        7.896632         0.1270552      -1.281463      -0.15481262
#> 923        8.019727         0.1081714       3.012734       0.07678534
#>
#> $`4`
#>      yieldQueenTrait yieldWorkersTrait calmQueenTrait calmWorkersTrait
#> 1121       10.046189         0.1526824     -0.7891866      -0.21855027
#> 1122        9.453486         0.2662015      0.3324839      -0.04823473
#> 1123       11.474695         0.1208822      1.0039044      -0.41267355
```

Now, we calculate colony genetic and phenotypic values for all colonies in the apiary. Since we are simulating

two traits, honey yield and calmness, we have two ways to calculate corresponding colony values. The first way is to use the default `mapCasteToColony*()` function in `calcColony*()` and only define additional arguments as shown here:

```
colonyValues <- calcColonyPheno(apiary,
                                queenTrait = c("yieldQueenTrait", "calmQueenTrait"),
                                workersTrait = c("yieldWorkersTrait", "calmWorkersTrait"),
                                traitName = c("yield", "calmness"),
                                checkProduction = c(TRUE, FALSE)) |> as.data.frame()
colonyValues
#>       yield  calmness
#> 1  17.46859  5.448026
#> 2  25.12284  8.574206
#> 3  25.44408  2.007797
#> 4  10.48484 -6.048520
#> 5  13.11196 -3.648724
#> 6  23.49879 -5.755791
#> 7  24.81994  5.771627
#> 8  10.93667  7.516247
#> 9  11.41759  1.343587
#> 10 19.07101 -1.446990
#> 11 22.28974  1.755019
#> 12 16.24151  1.982780
#> 13 19.20846 -3.015932
#> 14 11.95638  8.735068
#> 15 24.09200 -1.759276
```

The second way is to create our own mapping function. An equivalent outcome to the above is shown below just to demonstrate use of your own function, but we are simply just reusing `mapCasteToColonyPheno()` twice;)

```
myMapCasteToColonyPheno <- function(colony) {
  yield <- mapCasteToColonyPheno(colony,
                                 queenTrait = "yieldQueenTrait",
                                 workersTrait = "yieldWorkersTrait",
                                 traitName = "yield",
                                 checkProduction = TRUE)
  calmness <- mapCasteToColonyPheno(colony,
                                    queenTrait = "calmQueenTrait",
                                    workersTrait = "calmWorkersTrait",
                                    traitName = "calmness",
                                    checkProduction = FALSE)
  return(cbind(yield, calmness))
}
colonyValues <- calcColonyPheno(apiary, FUN = myMapCasteToColonyPheno) |> as.data.frame()
colonyValues
#>       yield  calmness
#> 1  17.46859  5.448026
#> 2  25.12284  8.574206
#> 3  25.44408  2.007797
#> 4  10.48484 -6.048520
#> 5  13.11196 -3.648724
#> 6  23.49879 -5.755791
#> 7  24.81994  5.771627
#> 8  10.93667  7.516247
```

```
#> 9  11.41759  1.343587
#> 10 19.07101 -1.446990
#> 11 22.28974  1.755019
#> 12 16.24151  1.982780
#> 13 19.20846 -3.015932
#> 14 11.95638  8.735068
#> 15 24.09200 -1.759276
```

Again, we can now select the best colony based on the best phenotypic value for either yield, calmness, or an index of both. Let's say that both traits are equally important so we select on a weighted sum of both of them - we will use the AlphaSimR `selIndex()` function that enables this calculation along with scaling. We will represent the index such that it has a mean of 100 and standard deviation of 10 units.

```
colonyValues$Index <- selIndex(Y = colonyValues, b = c(0.5, 0.5), scale = TRUE) * 10 + 100
bestColony <- selectColonies(apiary, n = 1, by = colonyValues$Index)
getId(bestColony)
#> [1] 2
```

We see that we selected colony with ID "4", but we would be selecting a different colony based on different selection criteria (yield, calmness, or index).

Strength and honey yield

In this section we change simulation to two traits where the phenotype realisation of the first trait affects the phenotype realisation of the second trait. Specifically, we will assume that queen's fecundity, and hence the number of workers, is under the genetic affect of the queen and her environment. Furthermore, we will assume as before that colony honey yield is due to the queen effect and workers effect. Since the value of the workers effect depends on then number of workers, we obtain correlation between fecundity and honey yield, even if these traits would be uncorrelated on the queen level. We emphasise that this is just an example and the biology of these traits might be different.

We follow the same logic as before and simulate three traits that will contribute to two colony traits, queen's fecundity, that is colony strength, and honey yield. We assume that fecundity is only due to the queen (and not the workers), hence we simulate only the queen effect for this trait. For honey yield we again assume that both the queen and workers contribute to the colony value. For speed of simulation we only simulate 100 workers per colony on average and split honey yield mean between the queen and workers. We measure fecundity with the number of workers, which is a count variable and for such variables Poisson distribution is a good model. This distribution has just one parameter (lambda) that represents both the mean and variance of the variable. To this end we set phenotypic variance to 100 and split it into 25 for genetic and 65 for environmental variance. As before we warn that these are just exemplary values to demonstrate the code functionality and do not necessarily reflect published values!

```
# Global simulation parameters
SP <- SimParamBee$new(founderGenomes)

# Quantitative genetic parameters
# - the first trait has only the queen effect
# - the second trait has both the queen and workers effects
nWorkers <- 100
mean <- c(nWorkers, 10, 10 / nWorkers)
varA <- c(25, 1, 1 / nWorkers)
corA <- matrix(data = c(1.0,  0.0,  0.0,
                        0.0,  1.0, -0.5,
                        0.0, -0.5,  1.0), nrow = 3, byrow = TRUE)
SP$addTraitA(nQtlPerChr = 100, mean = mean, var = varA, corA = corA,
```

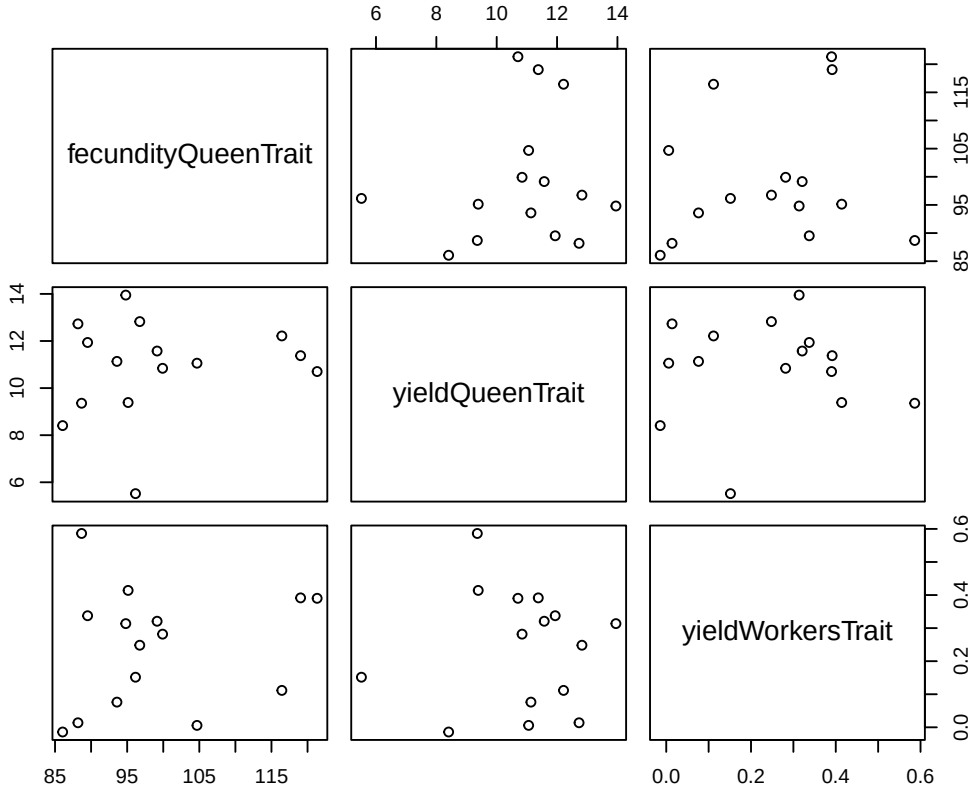
```
            name = c("fecundityQueenTrait", "yieldQueenTrait", "yieldWorkersTrait"))

varE <- c(75, 3, 3 / nWorkers)
corE <- matrix(data = c(1.0, 0.0, 0.0,
                        0.0, 1.0, 0.3,
                        0.0, 0.3, 1.0), nrow = 3, byrow = TRUE)
SP$setVarE(varE = varE, corE = corE)
```

We continue by creating an apiary with 10 colonies.

```
basePop <- createVirginQueens(founderGenomes)
drones <- createDrones(x = basePop[1:5], nInd = 100)
apiary <- createMultiColony(basePop[6:20])
droneGroups <- pullDroneGroupsFromDCA(drones, nColonies(apiary), nDrones = 15)
apiary <- cross(x = apiary, drones = droneGroups)
```

Let's explore queen's genetic and phenotypic values for fecundity and honey yield. The below printouts show quite some variation in fecundity between queens at the genetic, but particularly phenotypic level. This is a small example, so we should not put too much into correlations between these three variables. However, if you restart this simulation many times, you will notice zero correlation on average between `fecundityQueenTrait` and the other two traits and negative correlation on average between `yieldQueenTrait` and `yieldWorkersTrait`. Just like we defined in the global simulation parameters.

```
#>    fecundityQueenTrait yieldQueenTrait yieldWorkersTrait
#> 6            108.88870        8.111167        0.18171695
#> 7             95.17647       11.914948       -0.11438644
#> 8             98.54288        9.458258        0.14737806
#> 9            105.08133        9.869568        0.16769221
#> 10           105.57552        9.491473        0.09531919
#> 11            91.77573       10.778029        0.11964755
#> 12           103.97353       11.008658        0.08624144
#> 13            98.00120       10.441387        0.13840539
#> 14            99.37330        8.820901        0.19543398
#> 15           104.52665       10.002896        0.16241357
#> 16            92.55539       10.090902        0.10528649
#> 17            99.16269       10.297400       -0.01234176
#> 18            97.64239       10.324049        0.03814645
#> 19           103.25623        9.001674        0.28418611
#> 20            98.09218        7.954785        0.25127178
#>                     fecundityQueenTrait yieldQueenTrait yieldWorkersTrait
#> fecundityQueenTrait           1.0000000      0.15424563        0.15095948
#> yieldQueenTrait               0.1542456      1.00000000        0.01782012
#> yieldWorkersTrait             0.1509595      0.01782012        1.00000000
```



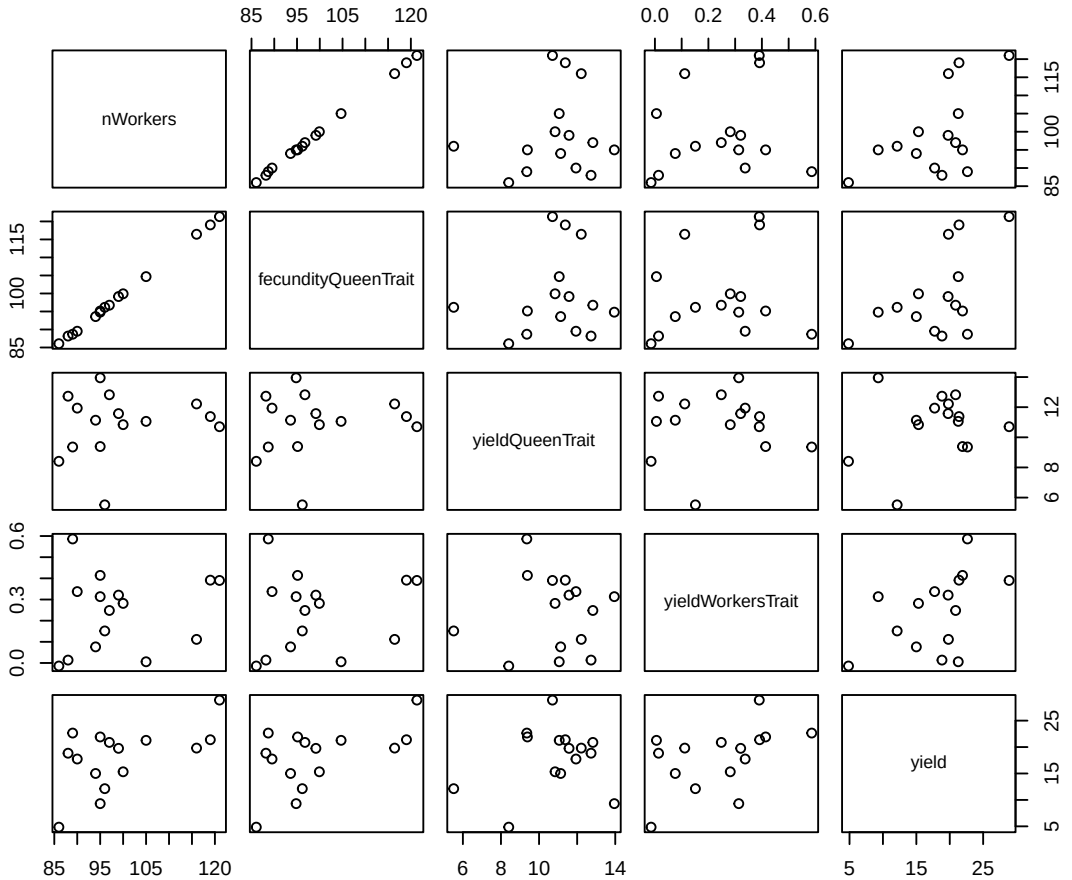
We next build-up colonies in the apiary. But instead of building them all up to the same fixed number of workers, we build them up according to queen's fecundity. For that we use the sampling function `nWorkersColonyPhenotype()`, that samples the number of workers based on phenotypes of colony members, in our case `fecundityQueenTrait` in queens. Correspondingly, each colony will have a different number of workers. Read more about this function in it's help page.

```
apiary <- buildUp(apiary, nWorkers = nWorkersColonyPhenotype,
                  queenTrait = "fecundityQueenTrait")
cbind(nWorkers = nWorkers(apiary), queenPheno)
#>    nWorkers fecundityQueenTrait yieldQueenTrait yieldWorkersTrait
#> 1       121           121.32917       10.703465       0.390117447
#> 2        95            94.80643       13.950397       0.313545028
#> 3        96            96.15619        5.517601       0.151471707
#> 4        97            96.74393       12.824597       0.248187194
#> 5       105           104.67734       11.058364       0.005778195
#> 6        88            88.17772       12.729043       0.013687268
#> 7       116           116.43964       12.215153       0.111575243
#> 8       100            99.90959       10.840850       0.281694643
#> 9        99            99.14828       11.574551       0.320719568
#> 10      119           119.03696       11.377387       0.391416464
#> 11       90            89.51533       11.937927       0.337426566
#> 12       86            86.05471        8.407536      -0.014260286
#> 13       94            93.58208       11.134411       0.076145806
#> 14       95            95.13048        9.387953       0.413992635
```

```
#> 15        89             88.68424        9.357136        0.586243390
help(nWorkersColonyPhenotype)
```

To compute the colony value for honey yield, we again employ the `calcColonyPheno()` function. Correlating the queen and colony values we will now see a positive correlation because our individual to colony mapping function sums workers effect across all workers and the more workers there are the larger the sum.

```
#>                      nWorkers fecundityQueenTrait yieldQueenTrait
#> nWorkers            1.0000000           0.9997623      0.16070057
#> fecundityQueenTrait 0.9997623           1.0000000      0.15424563
#> yieldQueenTrait     0.1607006           0.1542456      1.00000000
#> yieldWorkersTrait   0.1551816           0.1509595      0.01782012
#> yield               0.5587129           0.5560590      0.20135821
#>                     yieldWorkersTrait     yield
#> nWorkers                   0.15518156 0.5587129
#> fecundityQueenTrait        0.15095948 0.5560590
#> yieldQueenTrait            0.01782012 0.2013582
#> yieldWorkersTrait          1.00000000 0.4747823
#> yield                      0.47478229 1.0000000
```



Additional file 7 - Sampling functions vignette

2022-12-15

Introduction

SIMplyBee includes functions to sample various values that are expected to vary between colonies and events. These functions are used to sample numbers, usually individuals, and proportions. We can use the functions, pass them to other functions, or save them in the `SimParamBee` object so they can be used by default by other functions.

We start by loading the package:

```
library(package = "SIMplyBee")
#> Loading required package: AlphaSimR
#> Loading required package: R6
#>
#> Attaching package: 'SIMplyBee'
#> The following object is masked from 'package:base':
#>
#>     split
```

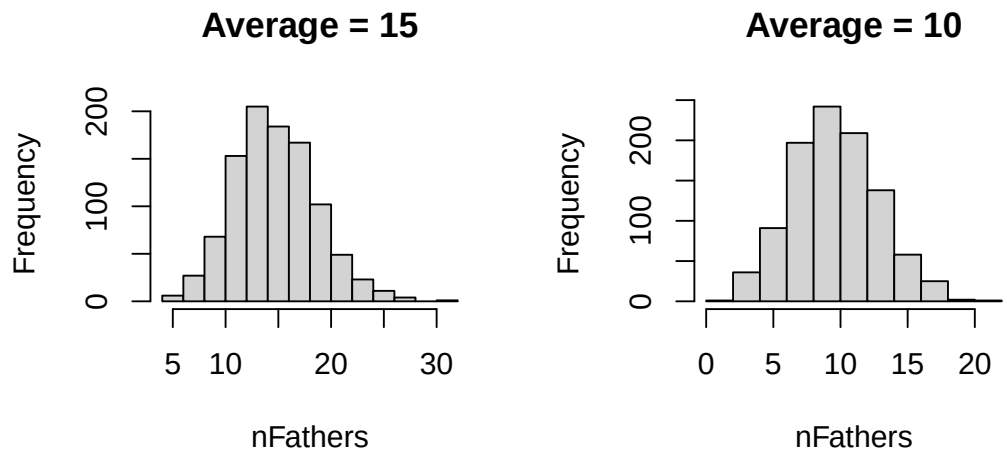
Functions to sample numbers

First, there are functions to sample the number of caste individuals from either a Poisson or truncated Poisson distribution: `n*Poisson()` and `n*TruncPoisson()`, where `*` is either `Workers`, `Drones`, `VirginQueens`, or `Fathers`. Most SIMplyBee functions that take the number of individuals as an argument can accept these sampling functions as an input, meaning that the output of such function calls will be stochastic. These functions are useful when you want to sample a variable number of individuals around a mean, as for example when mating virgin queens with a variable number of drones.

Let's start a simulation by creating a DCA and an apiary with 10 virgin colonies:

```
founderGenomes <- quickHaplo(nInd = 20, nChr = 1, segSites = 100)
SP <- SimParamBee$new(founderGenomes)
basePop <- createVirginQueens(founderGenomes)

# Create a DCA from the first 10 base virgin queens
DCA <- createDrones(x = basePop[1:10], nInd = 100)

# Create an apiary with 10 virgin colonies
apiary <- createMultiColony(basePop[11:20])
```

From the literature we know that virgin queens on average mate to 17 drones, but the actual number varies around this mean. Some mate with 10 drones, some with 20, etc. To resemble this variation, we can use the function `nFathersPoisson()` to sample variable number of drones from a DCA. The default average for this function is 15, but you can use any value you want. Let's use this function to sample 1,000 values and inspect the distribution and the mean.



Let's now use this functionality to sample a variable number of drones from the DCA to mate with each of the 10 virgin queens.

```
droneGroups <- pullDroneGroupsFromDCA(DCA = DCA, n = 10, nDrones = nFathersPoisson)
apiary <- cross(apiary, drones = droneGroups)
```

And inspect the number of fathers in each of the colony and their mean.

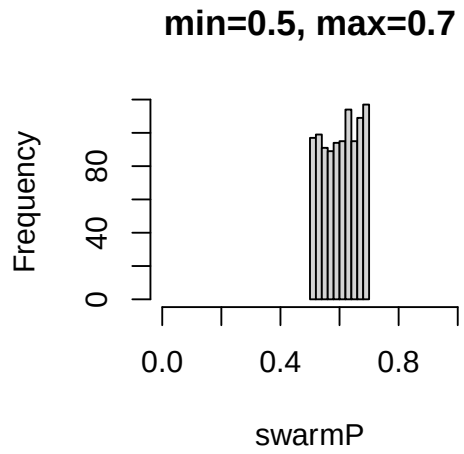
```
nFathers(apiary)
#>  1  2  3  4  5  6  7  8  9 10
#> 16 17 20 17 16 14 16 15 13 12
mean(nFathers(apiary))
#> [1] 15.6
```

Second, we have a group of functions that will sample the number of individuals according to the colony phenotype, whatever that might be. These functions are named `n*ColonyPhenotype()`, where `*` is either is either `Workers`, `Drones`, or `VirginQueens`. An example of this would be sampling the number of workers and drones according to queen's fecundity or honey yield. An example of this can be seen in the quantitative genetics vignette in the "Strength and honey yield" example.

Functions that sample proportions

SIMplyBee also includes functions to sample the proportions of workers that leave or are removed when downsizing, splitting, or swarming a colony from either a uniform distribution or from a beta distribution that accounts for the number of individuals in a colony (colony strength). These functions are named `*PUnif()`, where `*` can be either `swarm`, `split`, or `downsize`. There is an additional function, `splitPColonyStrength()`, that determines the number of workers to be removed in a split according to the colony strength.

Let's say we want to swarm all the colonies in our apiary with a variable percentage of workers that leave. We want to sample this percentage from an uniform distribution with the mean of 0.6. For this, we use `swarmPUnif()` function that takes a `min` and a `max` values and sample a value between them. By default, the `min` is set to 0.4 and `max` to 0.6. Let's use this function to sample a 1,000 values between 0.5 and 0.7 and inspect the mean.



Let's now swarm all the colonies in our apiary with a variable percentage of workers that leave.

```
apiary <- buildUp(apiary, nWorkers = 1000, nDrones = 100)
tmp <- swarm(apiary, p = swarmPUnif(n = 10, min = 0.5, max = 0.7))
nWorkers(tmp$swarm)
#>  11  13  15  17  19  21  23  25  27  29
#> 668 589 575 661 664 539 599 656 626 548
```

We see that each colony swarmed with a different percentage between 0.5 and 0.7.

3 The principles of expected and realised genetic relatedness in honeybees

This chapter contains the manuscript *The principles of expected and realised genetic relatedness in honeybees* that has been submitted for publication at the Heredity journal (see preprint: `https://www.biorxiv.org/content/10.1101/2024.05.26.595938v1`) and is currently under consideration.

This manuscript demonstrates the principles of genetic relatedness in honeybees by analysing the genetic and pedigree information from a SIMplyBee simulation, comparing closed and hybrid populations. Relatedness coefficients are commonly employed to assess genetic similarities both within and across population and among their individual members. While the haplo-diploid inheritance system in honeybees is well established, accurately interpreting different types of relatedness coefficients based on pedigree and genotype information poses a significant challenge for researchers and practitioners involved in honeybee breeding. We evaluated the genetic relatedness between individuals within a colony, between queens of the same population, and between queens of different populations. The findings of this work emphasise the need for better understanding and standardising methodologies for computing relatedness coefficients to ensure accurate comparability in relatedness studies.

The manuscript is a joint work of Laura Strachan[1], Jernej Bubnič[2], Gertje Petersen[3], Gregor Gorjanc[1] and Jana Obšteter[2]

[1] The Roslin Institute and Royal (Dick) School of Veterinary Studies, University of Edinburgh, Easter Bush Research Centre, Midlothian EH25 9RG, UK; [2] Department of Animal Science, The Agricultural Institute of Slovenia, Ljubljana, Slovenia; [3] Abacusbio, 442 Moray Place, Central Dunedin, Dunedin 9016, New Zealand.

3.1 Author contribution statement

The study was initiated by Gregor and supervised by both Gregor and Jana. The simulation plan was developed, the code was written, data analysis was conducted, and the initial draft of the manuscript was prepared by Laura. Regular meetings with Gregor and Jana provided guidance, during which coding issues and unusual results were discussed. All authors contributed to the revision of the manuscript and approved

the final version, which was subsequently submitted to the Heredity journal for peer review. The manuscript was edited by Laura following the reviewers' feedback.

Manuscript Status: In review

3.2 Abstract

Monitoring honeybee genetic variability is essential to manage global and local genetic diversity. Coefficients of relatedness are regularly used to measure genetic similarity within and between populations and their individuals. Although the haplo-diploid inheritance of honeybees is well understood, interpreting the various types of relatedness coefficients based on pedigree and genotype data is a challenge for researchers and practitioners in honeybee breeding. To demonstrate the principles of genetic relatedness in honeybees and its different individual-based estimators, we simulated three honeybee populations each containing 400 colonies over 10 years using the stochastic simulator SIMplyBee. We kept two populations closed and hybridised the third one by importing drones from one of the closed populations. We evaluated the relatedness between individuals within a colony, between queens of the same population, and between queens of different populations. We calculated three types of relatedness: expected identity by descent using pedigree information, realised identity by descent using pedigree and genotype information, and identity by state using genotype information. Our results demonstrated an alignment of mean relatedness across different types when calculated using the same founder population, regardless of their data source. Identity by state relatedness varied significantly when calculated with different founder populations. Although this is an expected result, it shows that caution is needed when comparing values between studies using different populations with different allele frequencies. We expectedly showed increased relatedness over time in closed populations and decreased in the hybrid population. Our results underscore the significance of understanding the methodology for computing relatedness coefficients.

3.3 Introduction

Monitoring and managing genetic variability is crucial to safeguarding genetic diversity in local and global populations. The loss of genetic variability diminishes the fitness of extant populations and their long-term adaptability, which are essential in dynamic environments. Meanwhile, in breeding programmes, monitoring and managing genetic variability is central to i) provide sufficient variability to feed short-term genetic improvements aligned with current breeding goals and long-term genetic improvements in anticipation of future breeding goals; and ii) validate relationships that aid in identifying and distributing superior genetic material.

Conservation by utilisation is considered the most effective approach for supporting and managing genetic variability. A crucial aspect of this strategy involves breeding programmes to enhance economically important traits while managing diversity, as

intensive selection can lead to its decline (Falconer and Mackay, 1996; Charlesworth and Charlesworth, 1999; Uzunov et al., 2017). Inbreeding depression, characterised by reduced individual genetic variability, can result in reduced performance. This is attributed to factors such as the expression of partially recessive deleterious mutations or increased homozygosity at loci expressing overdominance (Falconer and Mackay, 1996; Charlesworth and Charlesworth, 1999). Furthermore, increased relatedness and inbreeding enhance genetic drift, driving random changes in allele frequency and linkage, potentially resulting in unforeseen changes in economically important traits. To manage such losses of genetic diversity, it is essential to optimise parental selection and mating allocations based on relatedness between individuals (Wright, 1921).

The Western Honeybee (*Apis mellifera L.*) is a globally important economic species that plays a vital role in pollination services and food production (Klein et al., 2007; Gallai et al., 2009). Recent research has sparked a growing interest in monitoring and managing genetic diversity within honeybee populations, due to emerging studies demonstrating a decline in genetic variability of honeybee subspecies when comparing historical and current samples (Espregueira Themudo et al., 2020). Reduced genetic variability negatively affects honeybee population by increasing brood losses (Woyke, 1962; Zayed et al., 2006), compromising colony efficiency (Oxley et al., 2010; Espregueira Themudo et al., 2020), leading to inbreeding depression for fitness traits (Antolin, 1999; Henter, 2003; Zayed, 2009), and impairing long-term adaptive capacity, in turn increasing the risk of extinction. The genetic variability of honeybees is impacted by abiotic factors, including climate changes, the reduction of suitable habitats, agricultural intensification with increased agrochemical usage, and an increasing burden of pathogens and parasites (Zayed, 2009; Jaffé et al., 2010).

Several biotic factors prevent inbreeding and maintain genetic diversity in honeybee populations. First being the sex-determining system, whereby diploid eggs heterozygous at the complementary sex determiner (*CSD*) locus develop into diploid females, diploid homozygotes at *CSD* produce sterile or non-viable diploid drones, and haploid hemizygotes develop into functional drones (Woyke, 1962; Beye et al., 2006). The second mechanism to prevent inbreeding is polyandry, whereby a virgin queen mates with multiple males during her mating flights. Consequently, three types of sisters can be present in the diploid offspring. Half-sisters have unrelated fathers, full-sisters have related fathers (brother drones produced by the same queen), and super-sisters share the same father (Crow and Roberts, 1950; Crozier, 1970; Laidlaw Jr and Page Jr, 1984). Third is population management. Introgressive hybridisation driven by natural mating at geographic subspecies borders or import of non-native subspecies increases the genetic diversity of a population but can affect its fitness (De la Rúa et al., 2013; Espregueira Themudo et al., 2020). In contrast, selective breeding practices generally decrease genetic variation by rearing preferred families. Although, such practices can also increase the frequency of rare alleles and transitionally increase genetic variance (Crow, 2010). Furthermore, most of genetic variance reduced due to selection within a generation is restored via recombination between generations (Bulmer, 1971; Lara et al., 2022). Reduced genetic variability negatively affects honeybee population by increasing brood losses (Woyke, 1962; Zayed et al., 2006), compromising colony efficiency (Oxley et al., 2010; Espregueira Themudo et al., 2020), leading to inbreeding

depression for fitness traits (Antolin, 1999; Henter, 2003; Zayed, 2009), and impairing long-term adaptive capacity, in turn increasing the risk of extinction.

To understand genetic diversity within a population, Wright (1921) introduced coefficients of relationship and inbreeding within a known pedigree. He based this work on diploid individuals and additive gene action such that the genetic value of an individual i (a_i) is the sum of values of its two alleles: $a_i = a_{i,1} + a_{i,2}$. Wright's coefficient of relationship is the correlation between the genetic values of two individuals, denoted as

$$R_{i,j} = Cor\left(a_i, a_j\right),$$

while Wright's coefficient of inbreeding is the correlation between the genetic values of the two alleles of an individual, expressed as

$$F_i = Cor\left(a_{i,1}, a_{i,2}\right)$$

Wright (1921) showed how to calculate these coefficients by tracing all pedigree lineages between relatives. Cotterman (1940) and Malécot (1948) have later shown that Wright's coefficients measure the expected identity by descent (IBD) of individual's genomes within a pedigree relative to the pedigree founders, while observing individuals' genomes would enable measuring identity by state (IBS).

Emik and Terrill (1949) developed a recursive algorithm to calculate the relatedness coefficients for all pairs of individuals within a pedigree. This algorithm works with an $n \times n$ covariance coefficient matrix $\mathbf{A}$ for n individuals and sets this matrix to an identity matrix between pedigree founders. Diagonal elements of the matrix are defined as:

$$\mathbf{A}_{i,i} = Var\left(a_i\right)/\sigma_a^2 = 1 + F_i,$$

with values in the range of $[1, 2]$ since F_i is the correlation coefficient.
Off-diagonal elements of the matrix are defined as:

$$\begin{aligned}\mathbf{A}_{i,j} &= \frac{\mathrm{Cov}(a_i, a_j)}{\sigma_a^2} \\ &= \frac{1}{2}\left(\mathrm{Cor}(a_{i,1}, a_{j,1}) + \mathrm{Cor}(a_{i,1}, a_{j,2}) + \mathrm{Cor}(a_{i,2}, a_{j,1}) + \mathrm{Cor}(a_{i,2}, a_{j,2})\right),\end{aligned}$$

with values in the range of $[0, 2]$.
F_i is estimated as half the off-diagonal element between the parents of i: $1/2\mathbf{A}_{m(i),f(i)}$.
Off-diagonal elements are estimated as:

$$\mathbf{A}_{i,j} = 1/2\left(\mathbf{A}_{i,m(j)} + \mathbf{A}_{i,f(j)}\right)$$

Completing the matrix $\mathbf{A}$ enables calculating Wright's inbreeding coefficients F_i and relationship coefficients $R_{i,j}$. However, it is now common to use just elements of $\mathbf{A}$, which Henderson (1976) named as the numerator relationship matrix.

Pedigree-based relatedness coefficients exhibit two important limitations. First, they measure expected relatedness between individuals, wherein the expectation is predicted

based on the recombination and segregation processes of genomes between parents and progeny. However, there is substantial variation in the realised relationships between pairs of relatives due to these stochastic processes (Hill and Weir, 2011). Second, pedigree-based coefficients measure relatedness relative to the assumed unrelated founder population at the start of the pedigree. Hence pedigree-based coefficients are limited by the unobserved inheritance of genomes, assumed unrelated founders, and pedigree depth. Nevertheless, pedigree-based relatedness measures expected IBD, which is expectation of the realised IBD, which is in turn expectation of the IBS, all relative to the base population of pedigree founders.

With the availability of genome-wide genotype (genomic) data, we can nowadays observe actual relatedness between individuals due to IBS, which is ultimately driven by mutations and their recombination and segregation across generations (Visscher et al., 2006; VanRaden, 2008; Hill and Weir, 2011). To calculate the genotype-based numerator relationship matrix, let $\mathbf{w}_i$ represent a row vector of individual i's genotypes across n_m markers, where each element of $\mathbf{w}_i$ is encoded as the number of alternative alleles present at each marker. Similarly to the pedigree-based numerator relationship matrix, elements of genotype-based numerator relationship matrix are calculated from the observed covariance between genotypes of pairs of individuals: $\mathbf{w}_i\mathbf{w}_j^T$ (VanRaden, 2008). In this calculation, the genotypes are often centred and scaled to present genotype-based relatedness relative to a chosen base/reference population (VanRaden, 2008; Yang et al., 2010; Speed and Balding, 2015). The centring is based on allele frequency calculated from the chosen base/reference population. A common approach for estimating these allele frequencies is to use those from the current population as proxies for base population values. However, this method assumes the absence of inbreeding and a low mean relationship within the sample, which may not always hold true (Goudet et al., 2018). When inbreeding is present or the sample exhibits high relatedness, this assumption can lead to biased estimates of relatedness, affecting downstream analyses. Alternative methods, such as explicitly accounting for historical allele frequencies or leveraging external reference panels, may help mitigate these biases. Coupled with pedigrees, the availability of genomic data also allows observing the realised IBD, by reconstructing the inheritance of genomes between generations of a pedigree (Elston and Stewart, 2008; Whalen et al., 2018).

Estimating relatedness among honeybees has to take into account their haplo-diploid inheritance. Grossman and Eisen (1989) and Grossman and Fernando (1989) developed a general method for calculating pedigree-based relatedness for the X chromosome, which has haplo-diploid inheritance and can also be used for individual honeybees. Recently, Druet and Legarra (2020) developed a general method for calculating genotype-based relatedness matrix for the X chromosome. They also presented a method for calculating a "hybrid" relatedness matrix that combines pedigree and genotype data when some individuals are not genotyped.

Although the genome-wide genotype data is now becoming increasingly available for honeybee researchers, the current landscape of honeybee genetic studies reveals a gap in the utilisation and comparison of methodologies in constructing genomic relationship matrices (Tarpy et al., 2023). While some GWAS studies briefly mention the calcula-

tion of genotype-based relatedness, they lack explicit discussion on the methodologies employed and leave room for assumptions (Avalos et al., 2020; Eynard et al., 2024). In the majority of these studies, the allele frequency used are derived from pooled samples typically collected from a wide range of locations, possibly encompassing different subspecies and their hybrids. Consequently, this large genetic diversity challenges the comparison of the resulting genotype-based relatedness to classic pedigree-based relatedness.

Moreover, previous research mostly focused on pooled samples or colony-based relationships. With the increasing availability of genomic data and the advance of honeybee genetics research many studies would benefit from individual honeybee genotype information, such as, paternity verification (**?**), as well as drone quality (**?**), and honeybee behaviour (**?**). Therefore, there is a need for studies to explicitly delineate the methods used in calculating relatedness among individual honeybees, particularly focusing on allele frequency and IBS calculations, to estimate the impact of chosen reference populations on the relatedness values. This clarification will aid in better study comparisons and improve the management of genetic variability in honeybee populations.

The overarching aim of this contribution is to demonstrate genetic relatedness among individual honeybees within a colony, within a population, and comparatively between populations, using simulation. Specifically, we aim to: (i) demonstrate the principles of genetic inheritance in honeybees by evaluating relatedness within a colony; (ii) compare relatedness calculated from different sources of information and using different methodology; and (iii) measure the impact of hybridisation between honeybee populations.

We also demonstrate how the choice of allele frequencies impacts results to highlight the importance of disclosing methodology.

3.4 Materials and Methods

We used the stochastic simulator SIMplyBee (Obšteter et al., 2023) to simulate a 10-year cycle of two closed honeybee populations and their hybrid. We inspected whole-genome and *CSD* locus relatedness values within a colony, between queens of the same population, and between queens of different populations. We computed three types of relatedness values based on: (i) expected identity by descent (eIBD) information, (ii) realized identity by descent (rIBD) information, and (iii) identity by state (IBS) information.

3.4.1 Simulation of the honeybee population

Simulation of founder population

We simulated the founder haplotypes using the Markovian Coalescent Simulator (MaCS; (Chen et al., 2009)) that is integrated into SIMplyBee via AlphaSimR (Gaynor et al., 2021). We simulated 400 founder whole-chromosome haplotypes of the "Carniolan honeybee" (*Apis mellifera carnica*) and 800 founder whole-chromosome haplotypes of

the "European dark bee" (*Apis mellifera mellifera*), using historical population splits and effective population sizes (Ne) from the demographic model described in Wallberg et al. (2014). This model provides estimates of past divergence events and population sizes, allowing us to generate realistic genetic variation in the simulated populations. The mutation rate was set to 3.4e-09 (Yang et al., 2015), and the recombination rate to 2.3e-07 (Beye et al., 2006). We simulated 16 chromosomes and retained 1000 segregating sites per chromosome to monitor the relatedness. In addition, we simulated the *CSD* locus as a non-recombining region of DNA, located on the third chromosome, with 128 possible alleles allocated at random (Zareba et al., 2017; Obšteter et al., 2023). From the founder haplotypes, we created a founder population of unrelated diploid virgin queens, 400 *A.m.carnica* and 800 *A.m.mellifera*. We subsequently edited the *CSD* locus for all the virgin queens to be heterozygous and hence viable.

Throughout the remainder of the simulation we have removed any diploid individuals homozygous at the *CSD* locus. The founder queens were assigned to three populations, each with 400 virgin queens: two *A.m.mellifera* populations, one intended for within-species mating (Mel) and one for hybridisation (MelCross), and one *A.m.carnica* population (Car) for within-subspecies mating (Figure 3.1). Within each population, we randomly selected 300 virgin queens as the virgin queens of future colonies, and 100 as the drone-producing queens (DPQs). We initiated each population by mating the 300 virgin queens with an average of 15 drones produced by the 100 DPQs, and creating 300 colonies.

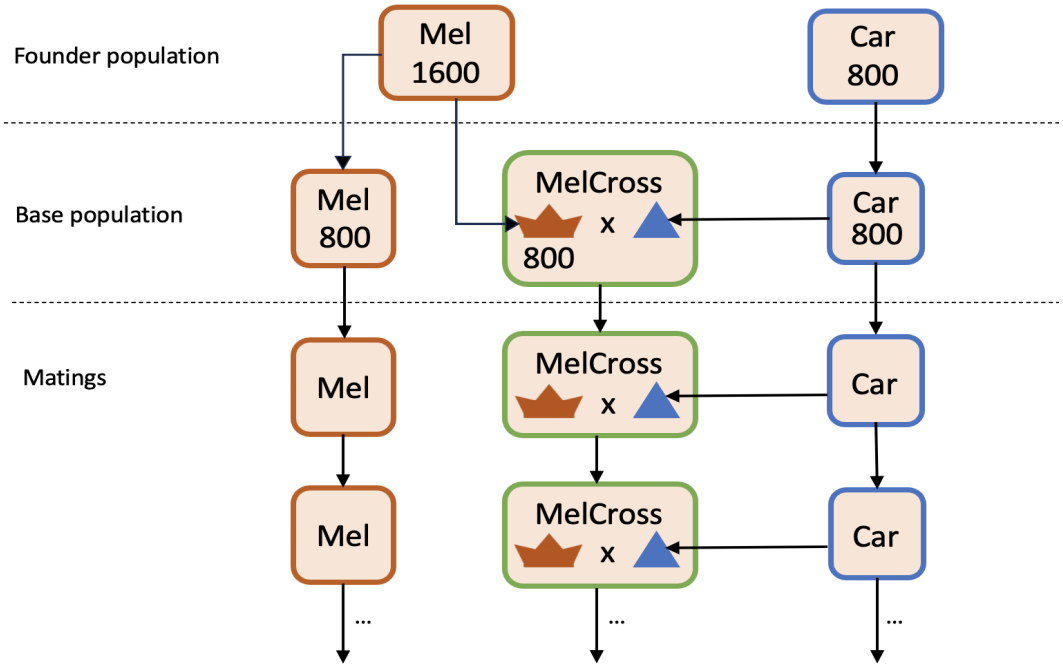


Figure 3.1: A flow diagram of the simulation, showing the founder population, base population and descending generations with the pathway of matings highlighted with arrows. 1600 *A.m.mellifera* and 800*A.m.carnica* founder haplotypes are used to create three founder populations of unrelated virgin queens, 400 *A.m.carnica* and 400 *A.m.mellifera* and a hybrid MelCross population contain 400 *A.m.mellifera* virgin queens. The *A.m.mellifera* and *A.m.carnica* populations had a closed population breeding system, while the hybrid MelCross population virgin queens were mated with *A.m.carnica* drones.

10-year breeding programme

We conducted a 10-year simulation, where one year represented the annual series of events of a honeybee colony with three periods that represented spring, summer, and winter. The simulation incorporated key colony events including split, swarm, supersedure, and collapse, with probabilities varying based on the specific period as outlined in Table 3.1. For details on how SIMplyBee simulates colony events, see Obšteter et al. (2023).

In period 1, all colonies were built up by adding 50 drones and 10 workers followed by a managed split. Subsequently, a subset of colonies with one or two-year-old queens experienced additional events based on probabilities (Table 3.1). The split and swarm events led to an increase in the number of simulated colonies, generating a "split" or "swarmed" colony with a new queen and a parent "remnant" colony with the previous queen. Given the managed nature of a split, the virgin queens of "split" colonies were replaced with donor virgin queens generated from 5 one-year-old queens, representing artificial selection by beekeepers. All virgin queens resulting from split, swarm, and supersedure events were mated, through a random natural selection with drones from a drone congregation area (DCA) that consisted of drones from colonies with one- or

Table 3.1: Probability of events during the yearly cycle

Period of yearly cycle	Event	Probability
Period 1	Split	1.00
	Swarm	0.05
	Supersedure	0.05
	Collapse	0.10
Period 2	Swarm	0.01
	Supersedure	0.05
	Collapse	0.10
Period 3	Collapse age 0 colonies	0.25
	Collapse age 2 colonies	0.30

two-year-old queens. Mating schemes differed between closed and hybrid populations, the former mating only with drones of the same population and the latter mating with a mix of 50% of their own MelCross drones and 50% of Car drones (Figure 3.1).

Period 2 varied only slightly from Period 1. In Period 2, no managed splits were performed and all colony events occured with probabilities outlined in Table 3.1. Virgin queen matings were identical to that of Period 1.

As Period 3 represents winter, the only colony event that took place was a collapse to mimic winter colony losses. The probability of a collapse was determined by the queens' age (Table 3.1).

To maintain a population size of 300 colonies, we kept all colonies with one-year-old queens. If these summed up to less than 300, we "topped-up" populations with remnants from splits. Any excess colonies were removed from the population, which represented sale or export. Any colonies containing queens over the age of two were removed to simulate a more realistic managed honeybee queen life-cycle.

3.4.2 Calculations of relatedness

We used three relatedness metrics based on different sources of information: (i) expected IBD (eIBD) by using pedigree data; (ii) realised IBD (rIBD) by using pedigree and recombination data; and (iii) IBS by using genotype data. We express relatedness based on the pedigree or genomic numerator relationship matrix, which is equal to twice the coefficient of kinship (Wright, 1921; Henderson, 1976), specific for haplo-diploid inheritance (Grossman and Eisen, 1989). Hence, we report relatedness values in the range of $[0, 2]$.

We calculated the eIBD relatedness based on the collected pedigree of all simulated individuals over all the populations and years. To manage the compute and memory requirements, we have first used R package *nadiv* (Wolak, 2012) to compute the sparse inverse of pedigree-based relationship matrix for haplo-diploid inheritance following Grossman and Fernando (1989). We next used the indirect method of Colleau (2002)

to calculate the dense eIBD relationship matrix only for individuals of interest.

We calculated the rIBD relatedness based on the origin of alleles. Alleles' origins were tracked with SIMplyBee/AlphaSimR since founders by labelling the alleles of founders' whole-chromosome haplotypes with unique codes (from 1 to twice the number of founders) and tracking their recombination and segregation through the pedigree. We then calculated the rIBD relationship matrix for individuals of interest as twice the proportion of shared alleles IBD between pairs of individuals using the SIMplyBee function `calcBeeGRMIbd` (Van Arendonk et al., 1994; Obšteter et al., 2023).

We calculated the IBS relatedness from genotype data on 1000 segregating sites per chromosome. We used SIMplyBee function `calcBeeGRMIbs` that implements the method described by Druet and Legarra (2020) for computing haplo-diploid relationship matrix. To demonstrate the impact of different allele frequencies on the IBS relatedness, we used: (i) a colony's own allele frequency (ColonyAF) estimated from all individuals within a colony, (ii) allele frequencies of subspecies-specific founder queens (SingleAF), (iii) allele frequencies of founder queens of both subspecies, *A.m.mellifera* and *A.m.carnica*, combined (MultiAF), or (iv) allele frequency of 0.5 (0.5AF).

We show all three relatedness metrics among individuals in year 1 and 10 and in three "dimensions": (i) between individual honeybees of different castes within a colony, (ii) between queens of the same population, and (iii) between queens across different populations. For the within-colony "dimension", we have first randomly selected a colony in each of the populations in years 1 and 10 and computed the within-colony relatedness between the queen, 1000 workers, 200 drones, any virgin queens present after colony events, and all father-drones that have mated with the queen. For the within-population "dimension", we have computed the relatedness between all the queens of the same population (Car, Mel, and MelCross). We inspected the impact of hybridisation by inspecting separately the off-diagonal and diagonal elements of relationship matrices, which respectively represent the relatedness between same-population queens and the relatedness of the queens to themselves. For the between-population "dimension", we have inspected the relatedness between queens of different populations by analysing the off-diagonal elements of relationship matrices.

R code for the simulations, relatedness calculations, and data analysis is available at `https://github.com/HighlanderLab/lstrachan_honeybee_relatedness`.

3.5 Results and Discussion

3.5.1 Results Summary

Principles of honeybee genetic inheritance were demonstrated by inspecting relatedness within a colony in year 1 and 10 of closed population breeding.

Overall our results showed that the expected and realised IBD and IBS relatedness, over the 10-year simulation period, exhibit an increase in closed populations and a decrease in the hybrid population. Relatedness in all castes was shown to increase with years of closed population breeding due to increasing relatedness between queens and fathers. This closed population breeding increased the relatedness the most between workers and the least between drones. When comparing different metrics, calculations using the same pedigree founder population and reference population resulted in an alignment of mean relatedness, regardless of them being computed with pedigree or genotype data. However, when we assumed different reference populations for IBS relatedness we observed a large amount of variation in the resulting values, concluding that the results obtained can be easily misinterpreted depending on the allele frequency used in the calculations. These findings suggest that special attention has to be paid to understanding the used methodologies when interpreting the results.

3.5.2 Relatedness within a colony

We start by comparing relatedness between different caste individuals within a pure bred colony, with relatedness based on eIBD, rIBD, and IBS calculated using allele frequencies of the founder *A.m.carnica* queens (SingleAF). We first present the results for year 1 by examining the worker-to-worker (WW) relatedness, followed by workers-to-drones (WD), and drones-to-drones (DD) relatedness. To inspect the effects of closed-population breeding, we next compare these results to year 10. We visualise these results in Figure 3.2 and present the corresponding mean and standard deviation of each relatedness metric for different types of sisters in Table 3.2.

In year 1, eIBD relatedness between workers (WW) was seen at three distinct values (Figure 3.2). These values correspond to the different types of sisters in a colony, with 0.25 denoting half-sisters with unrelated fathers, 0.50 denoting full-sisters whose fathers are brothers (from the same queen), and 0.75 denoting super-sisters who share the same father. These values correspond to the literature on relatedness between honeybee workers (Crow and Roberts, 1950; Crozier, 1970; Laidlaw Jr and Page Jr, 1984) and more general literature on haplo-diploid inheritance (Grossman and Eisen, 1989; Druet and Legarra, 2020). eIBD relatedness between workers and drones (WD) within a colony is 0.25 (Figure 3.2) because they are only related through the maternal side. Lastly, eIBD relatedness between drones (DD) is 0.25 (Figure 3.2), again because all of their haploid genomes are inherited from the queen.

In year 1, the mean rIBD relatedness for different relatives corresponded to the mean eIBD relatedness (Table 3.2, Figure 3.2). Unlike the eIBD, rIBD relatedness takes into account recombination and segregation of chromosomes since the founders and captures

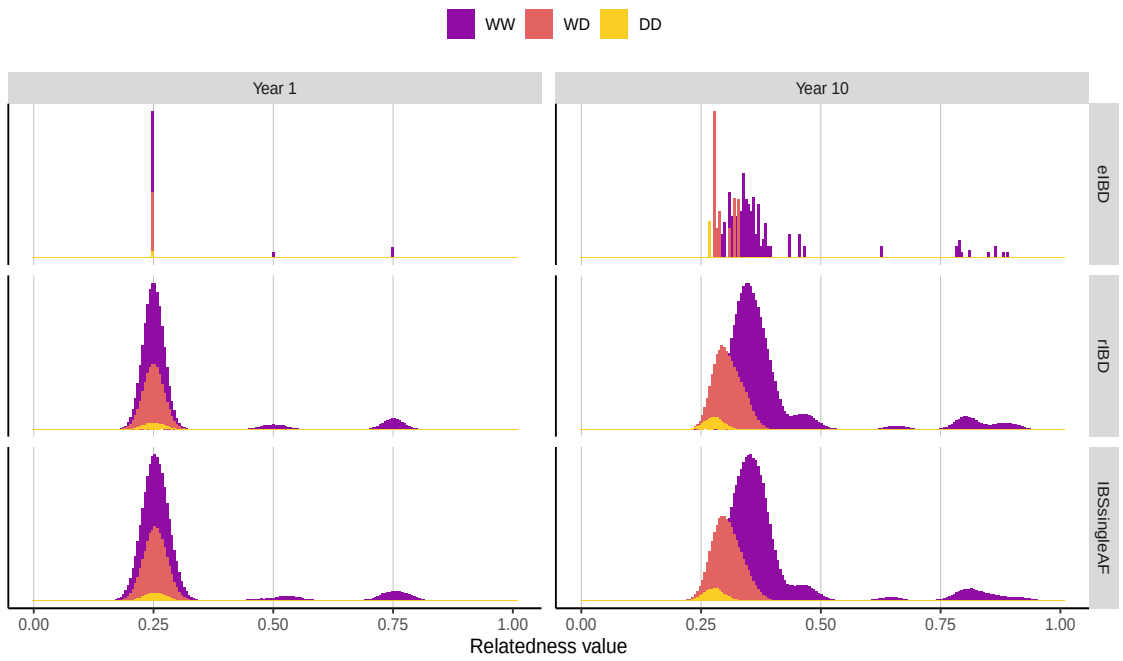


Figure 3.2: Expected identity-by-descent (eIBD), realised identity-by-descent (rIBD), and identity-by-state (IBS) relatedness within an A.m. carnica colony in year 1 (left) and year 10 (right) of the simulation. Expected IBD (eIBD) represents pedigree-based relatedness, estimating the probability that two individuals share alleles inherited from a common ancestor. Realised IBD (rIBD) accounts for actual genetic sharing due to recombination and segregation, which may differ from expected values. IBS measures genetic similarity based on observed genotype data, without considering inheritance, and was calculated here using allele frequencies of the founder A.m. carnica queens (SingleAF). The figure shows relatedness between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD). Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons.

distribution of realised relatedness due to these processes around the expected values (Table 3.2).

As shown in Figure 3.2, the mean IBSSingleAF relatedness for different relatives coincided with the means of eIBD and rIBD relatedness because all three metrics assumed the same founder/reference population. This calculation of IBS relatedness, IBSSingleAF, used allele frequencies from the population of founder *A.m.carnica* queens (IBSsingleAF). Despite the same mean, IBSsingleAF relatedness exhibited a larger standard deviation of values than the eIBD relatedness (Table 3.2). This larger variation is a result of the recent and ancestral (pre-pedigree) recombination and segregation of alleles captured by the genotype data in the IBSsingleAF calculations.

In year 10, the mean relatedness for all types of relatives increased compared to year 1 (Figure 3.2, Table 3.2), an expected consequence of a closed population. WW relationships had the largest increase in mean relatedness compared to year 1, whereas DD relationships the smallest increase, a difference which can be attributed to the honeybee's haplo-diploidy. Workers are diploid, meaning that relatedness compares

Table 3.2: Mean and standard deviation of expected identity-by-descent (eIBD) relatedness, realised identity-by-descent (rIBD) relatedness, and identity-by-state (IBS) relatedness between workers by sister type within an A.m. carnica colony in year 1 and year 10 of the simulation. Expected IBD (eIBD) represents the pedigree-based relatedness, reflecting the probability of inheriting alleles from a common ancestor, while realised IBD (rIBD) accounts for actual genetic sharing due to recombination and segregation. IBS measures genetic similarity based on observed genotype data, regardless of ancestry, and was calculated here using allele frequencies of the founder A.m. carnica queens (SingleAF). The mean and standard deviation were computed in R across all worker pairs within the colony to capture the distribution of relatedness values.

		Mean (Standard Deviation)		
Year	Sister type	eIBD	rIBD	IBSsingleAF
1	Half-sister	0.25 (0.00)	0.25 (0.02)	0.25 (0.03)
	Full-sister	0.50 (0.00)	0.48 (0.03)	0.49 (0.03)
	Super-sister	0.75 (0.00)	0.75 (0.02)	0.75 (0.03)
10	Half-sister	0.70 (0.05)	0.68 (0.05)	0.70 (0.05)
	Full-sister	0.92 (0.03)	0.89 (0.02)	0.90 (0.00)
	Super-sister	1.06 (0.04)	1.04 (0.03)	1.04 (0.03)

two alleles that are on average increasingly more similar over time through both the maternal and paternal side in a closed population. Notably, the numerator relationship includes a variance coefficient for each genome and twice the covariance coefficient between the two alleles Wright (1921); Henderson (1976); Grossman and Eisen (1989). Whereas, relatedness among haploid drones compares one genome exhibiting relatedness only through maternal side. We also observed an increase in the standard deviation of relatedness in year 10 compared to year 1, due to the random process of inheritance accumulating variation over time.

Relatedness at the CSD locus within a colony

We further inspected the relatedness at the *CSD* locus. We show this in Figure 3.3, which we present in the same manner as Figure 3.2 by showing the relatedness values at the locus in years 1 and 10.

The CSD locus is crucial in determining sex determination in honeybees, where allelic interactions at this locus dictate whether an individual is female or male. This system involves a form of balancing selection, where homozygous diploid drones (males) are selected against because they fail to survive, thus impacting relatedness patterns at the locus. The eIBD measures expected relatedness across the genome according to pedigree and therefore does not account for the balancing selection at the *CSD* locus. Hence, the eIBD relatedness at the *CSD* locus (Figure 3.3) was identical to the eIBD relatedness for the whole genome (Figure 3.2). The rIBD relatedness displayed the three possible relatedness values at the locus for worker-to-worker (WW): a relatedness value of 0.0, 0.5, and 1.0 respectively representing sharing 0, 1, and 2 alleles out of 4 alleles of

two diploid workers. Namely, relatedness larger than 1.0 is not possible between workers given that homozygotes ("diploid drones") at the locus are killed upon detection by nursing workers.

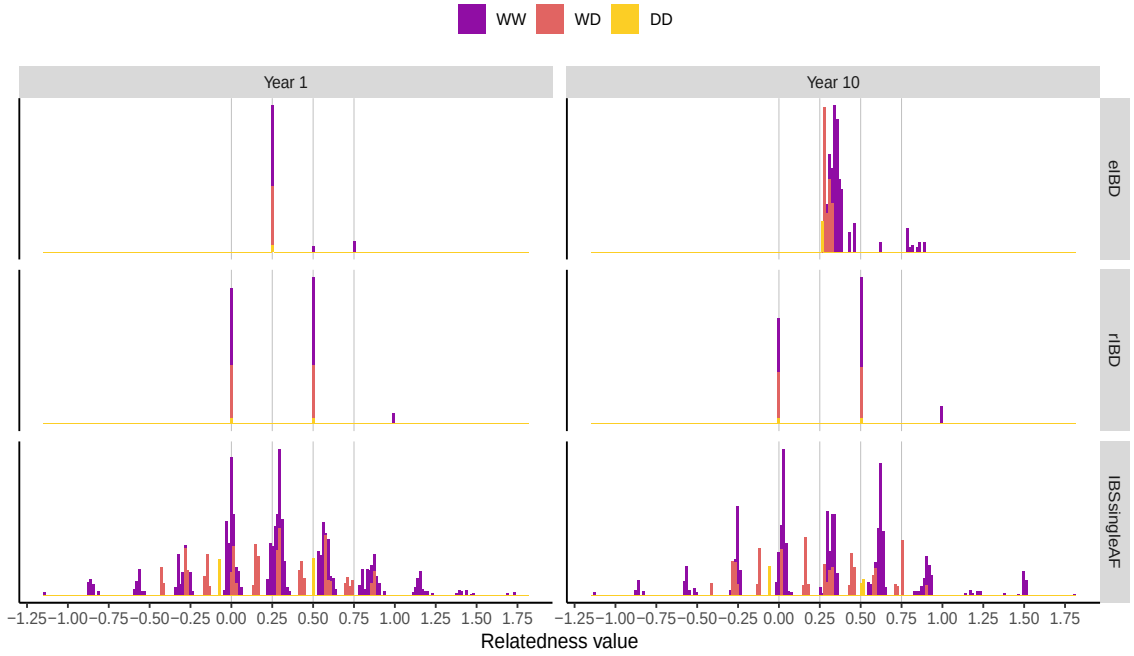


Figure 3.3: Expected and realised identity-by-descent (IBD) and identity-by-state (IBS) relatedness for the complementary sex determination (*CSD*) locus within a *A.m.carnica* colony in year 1 (left) and 10 (right) of the simulation. IBS was calculated using allele frequencies of the founder *A.m.carnica* queens (SingleAF). The figure shows relatedness between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD). Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons.

In year 1, the majority of workers shared 0 or 1 allele, and only a small proportion shared 2 alleles,reflecting the selection against homozygous diploid drones. and the selection against homozygous diploid drones. The balancing selection ensures that the colony maintains genetic diversity and avoids the production of homozygous drones. rIBD and eIBD differed in values because eIBD reports the expected relatedness among workers from the same queen, whereas rIBD reports the realised relatedness and two workers can inherit different alleles from the same queen and hence share no IBD alleles from the maternal side. Worker-to-drones (WD) relatedness was only 0.0 or 0.5, because drones potentially share 0 or 1 allele at the locus out of 3 alleles of one worker and one drone. Note that relatedness is calculated by comparing haploid genomes' and averaged over all possible comparisons (Grossman and Eisen, 1989; Druet and Legarra, 2020). Drone-to-drones (DD) only had relatedness of 0.0 or 0.5, because drones are haploid and therefore could potentially share 0 or 1 allele at the locus out of 2 alleles of two haploid drones (Grossman and Eisen, 1989; Druet and Legarra, 2020). IBSsingleAF relatedness had a wide distribution of values in year 1, with the mean

(standard deviation) relatedness of 0.29 (0.40) for WW relationships, 0.27 (0.28) for WD relationships, and 0.28 (0.21) for DD relationship. The wide range of IBS values was a consequence of the *CSD* locus spanning 7 segregating sites, which gave rise to a large number of possible combinations of alleles at the sites and hence different IBS relatedness values.

As inbreeding increased over the next 10 years within the closed population, we observed an expected rise in allele sharing in WW and decreased frequency of WW pairs with no shared alleles, both measured by rIBD. This increase in allele sharing was also observed in WD relationships, because the queen and the fathers became more related over generations. Previous studies report that increased allele sharing, resulting from inbreeding, increases the likelihood of a lethal homozygous (spotty) brood which impacts brood numbers and possibly colony survival (Lerner, 1954; Crozier and Page, 1985; Woyke, 1980).

Because homozygous individuals at the *CSD* locus are killed, we expected lower mean eIBD relatedness compared to rIBD relatedness in year 10 for WW relationships at the whole-genome level (Figure 3.2, Table 3.2), as indicated by the mean difference between eIBD and rIBD at the *CSD* locus (Figure 3.3). However, we did not observe this expected difference at the whole-genome level, as the selective elimination of homozygous diploid drones at the CSD locus reduced the impact of inbreeding on the overall genome-wide relatedness.. To explore this further, a queen was mated with her own male offspring to create a more inbred colony where progeny will be homozygous at the *CSD* 50% of time. In this case, the increased presence of homozygous diploid males resulted in a decreased mean rIBD relatedness at the whole-genome level (1.24) and at the csd chromosome level (1.19) compared to eIBD mean relatedness (1.25).

Relatedness of the queen with her offspring within a colony

In addition to siblings, we inspected the relatedness values of the queen with her offspring (Supplementary Figure 3.7). In year 1, both queen-to-workers (QW) and queen-to-drones (QD) relatedness had a mean of 0.5 regardless of the metric used to compute relatedness. Both IBD values align with the paternal-offspring relationship on mammalian X chromosome described by Druet and Legarra (2020).

The relatedness of 0.5 is expected for QW relationships because workers inherit half of their diploid genome from the queen and the remaining half from their fathers (relatedness compares worker's two alleles to queen's two alleles), while for QD relationships drones receive all of their haploid genome from the queen (relatedness compares drone's one genome to queen's two alleles).

In year 10, we saw the same trend as shown in the sibling relationships (Figure 3.2); an increase in relatedness due to inbreeding within the closed population. Specifically, inbreeding increased similarity between the queen's genomes and the drones she mates with, which in turn increased relatedness between the queen and her offspring. We observed a higher increase in relatedness for QW relationship compared to QD relationship, which is expected due to queens and workers having a diploid genome, drones having a haploid genome, and how the relatedness compares genome combinations of

pairs of individuals.

Comparison of IBS relatedness within a colony

We next compare IBS relatedness between different caste individuals within a purebred colony, calculated using different allele frequencies. We visualise these results in Figure 3.4 and present the corresponding mean and standard deviation of each relatedness calculation for different types of sisters in Table 3.3).

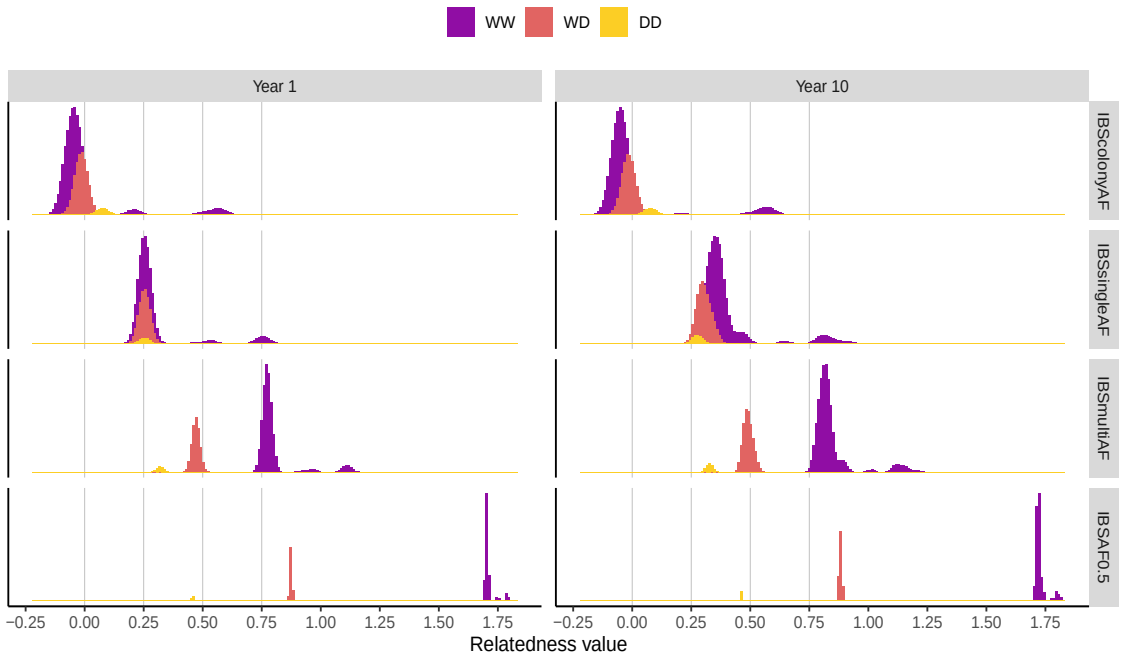


Figure 3.4: Comparison of identity-by-state (IBS) relatedness within an *A.m.carnica* colony in year 1 (left) and 10 (right) of the simulation. IBS relatedness was calculated using four different allele frequencies: the colony allele frequency (ColonyAF), the founder population allele frequency of *A.m.carnica* queens (SingleAF), the founder population allele frequency of *A.m.carnica* and *A.m.mellifera* queens (MultiBF), and allele frequency of 0.5 (0.5AF). The figure shows relationships between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD). Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons.

Table 3.3: Mean and standard deviation of identity-by-state (IBS) relatedness between workers by sister type within a *A.m.carnica* colony in year 1 and year 10 of the simulation. IBS relatedness were calculated using four different allele frequencies: the colony allele frequency (ColonyAF), the founder population allele frequency of *A.m.carnica* queens (SingleAF), the founder population allele frequency of *A.m.carnica* and *A.m.mellifera* queens (MultiAF), and allele frequency of 0.5 (0.5AF).

		Mean (Standard Deviation)			
Year	Sister type	IBSsingleAF	IBScolonyAF	IBSmultiAF	IBSAF0.5
1	Half-sister	0.25 (0.03)	-0.03 (0.03)	0.78 (0.02)	1.70 (0.01)
	Full-sister	0.49 (0.03)	0.22 (0.03)	0.94 (0.02)	1.75 (0.01)
	Super-sister	0.75 (0.03)	0.56 (0.04)	1.11 (0.02)	1.79 (0.01)
10	Half-sister	0.70 (0.05)	-0.05 (0.05)	1.03 (0.05)	1.77 (0.01)
	Full-sister	0.90 (0.00)	0.29 (0.04)	1.17 (0.02)	1.80 (0.00)
	Super-sister	1.04 (0.03)	0.53(0.06)	1.29 (0.04)	1.83 (0.01)

We present Figure 3.4 in the same manner as Figure 3.2, examining within-colony relatedness for WW, WD, and DD relationships in year 1 followed by year 10. We compare how IBS relatedness, using different allele frequencies, compares relative to IBSsingleAF, as described up to this point. The results show significant differences in IBS relatedness depending on the used allele frequencies.

ColonyAF: Alike to IBSsingleAF, year 1 relatedness between workers (WW) using the colony's own allele frequencies (IBScolonyAF) shows three peaks (Figure 3.4), representing the three sister types (Table 3.3). However, the mean IBScolonyAF relatedness in year 1 is 0 because it is calculated relative to the colony's own allele frequency. The mean relatedness of half-sisters indicates that they are on average less related than a random pair of individuals in the colony. In comparison, full sisters exhibit above average relatedness and even more so super-sisters. Using colony's own allele frequencies to compute IBS relatedness shows an intriguing "flip" in drone relatedness (DD) compared to using other allele frequencies. The mean IBScolonyAF relatedness between drones is higher than the mean IBScolonyAF relatedness between half-sisters (first peak of WW), which is in contrast with other IBS calculations where drones appear less or equally related than the half-sisters. An explanation for this is that the queen serves as the common genetic denominator for the colony, representing the colony's "internal" allele frequency. In contrast, fathers introduce "external" alleles, contributing to the genetic diversity to the colony beyond queen's alleles. Therefore, mean IBScolonyAF relatedness among drones is higher, because they have exclusively queens' "internal" alleles, whereas IBScolonyAF relatedness among workers (WD) and workers and drones (WD) is lower due to fathers' "external" alleles. This difference reduces with IBSsingleAF and flips with IBSmultiAF since both use allele frequencies from founder population(s) across multiple queens, as well as with IBSAF0.5 where all allele frequencies are assumed 0.5.

It is also important to highlight that IBSColonyAF relatedness is the only method that included males in the reference population for computing allele frequencies. Since males are haploid, their inclusion leads to a reduction in the mean values of relatedness in the reference population, and an increase in the relatedness values among other individuals since they are compared to a lower average. Supplementary Figure 3.8 demonstrates this, where IBSColonyNM relatedness assuming allele frequency computed without considering males consistently exhibits higher mean relatedness across all relationships in comparison to IBSColonyAF.

MultiAF: When using the founder allele frequencies across both subspecies, *A.m.mellifera* and *A.m.carnica* (IBSMultiAF), the mean relatedness was the highest among all IBS variants. This arises because the reference/founder population we compare the colony to is on average less related compared to other reference/founder populations, such as the single subspecies population used for IBSSingleAF. Namely, for IBSMultiAF we calculated allele frequencies across two subspecies that have diverged from a common ancestor approximately 300,000 years ago (Wallberg et al., 2014), meaning that individual colonies deviate substantially from this common reference population giving high IBS relatedness.

AF0.5: Using allele frequencies of 0.5 resulted in the widest range of within-colony mean IBS relatedness (IBSAF0.5) among all the reference/founder populations. However, because the mean relatedness deviates the most from 0 we can infer that the actual allele frequency of the colony, ColonyAF, differs considerably from 0.5. IBSAF0.5 relatedness had the least variation in mean values between different sister types and the lowest standard deviation within sister type (Table 3.3).

Year 10: In year 10, the mean IBS relatedness increased for all relationships compared to year 1 (Figure 3.4, Table 3.3), regardless of the allele frequencies used in calculations, as would be expected in a closed mating population. While the true biological increase in relatedness is consistent and reflects the population's genetic structure, the apparent magnitude of this increase varies depending on the allele frequencies used for IBS calculations. The mean IBS relatedness of all variants increased the most for WW relationships, whereas DD relationships had the smallest increase as described before due to diploid and haploid comparisons. Comparing values from year 1 to year 10, we observed the largest increase in relatedness for all relatives when using allele frequencies calculated from the founder *A.m.carnica* queens (IBSsingleAF), and the smallest increase when using allele frequencies of 0.5. Using colony's own allele frequency (IBSColonyAF) exhibited no increase in the mean relative relatedness from year 1 to 10, though the standard deviation did increase in WW relatedness (Table 3.3). The reason for this is that the IBScolonyAF relatedness in year 10 is based on colony allele frequency from the year 10, hence the relatedness is again relative to the average relatedness in the colony and centered on 0. On the other hand, IBSsingleAF and IBSmultiAF relatedness use the same allele frequencies in year 1 and year 10 meaning that the inspected colony is compared to the same reference/founder population.

3.5.3 Relatedness of queens within the same populations

IBS relatedness between queens within the same population is shown in Figure 3.5a and the queens' to themselves in Figure 3.5b. IBS relatedness was calculated using allele frequencies of the founder queens containing *A.m.carnica* and *A.m.mellifera* (MultiAF). We present relatedness values from year 10 because all queens in year 1 show "background" relatedness between the most recent common ancestor and the founding population in year 1.

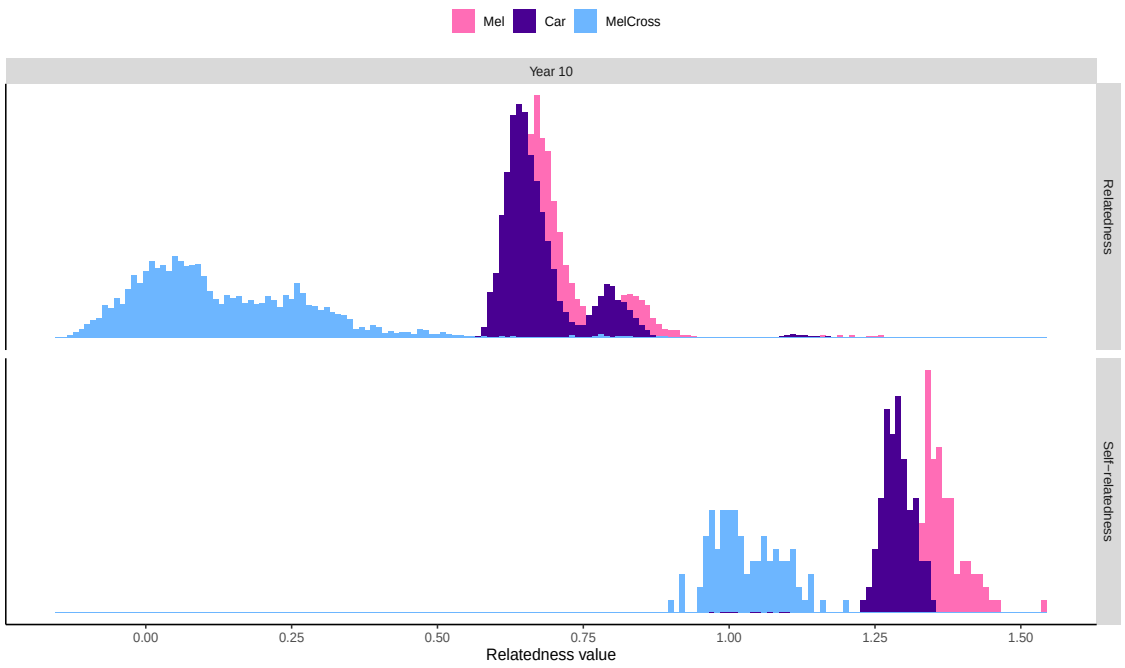


Figure 3.5: Identity-by-state (IBS) relatedness in year 10: a) between queens within two closed populations *A.m.mellifera* and *A.m.carnica* and one hybridising population *MelCross* (top-row); b) the self-relatedness of these queens (bottom-row). IBS relatedness was calculated using allele frequencies of the *A.m.carnica* and *A.m.mellifera* founder queens (MultiAF).

The hybridised MelCross queens are less related than queens in the closed populations in both between queens and in queens' with themselves. In this simulation, we mated MelCross queens with 50% of their own and 50% imported drones from *A.m.carnica*. We used this high proportion of import for demonstration only. Given the reported UK statistics, 260,000 managed hives were accounted for in 2020 with 21,000 imported queens, hence a more realistic import percentage would be $\sim 10\%$ (gov.uk, 2021; BeeBase, 2022). However, our results show decreased relatedness in hybrid populations compared to the closed populations. Previous studies showed that hybridisation of the populations increases the genetic variance and genetic diversity, and reduces the risk of inbreeding depression, which have all been shown to increase overall population's health and adaptability (Brückner, 1976; Crozier and Fjerdingstad, 2001; Seeley and Tarpy, 2006).

Since both closed populations, Mel and Car, have the same number of individuals and undergo the same events throughout the simulation, we would expect the same relatedness values of both populations. In this replicate of the simulation, the Mel queens' self-relatedness exhibit higher values in comparison to the Car queens (Figure 3.5b), which we concluded was stochastic, with some simulation replicates showing Car self-relatedness to be higher.

3.5.4 Relatedness of queens between different populations

The IBSMultiAF relatedness between queens of the different populations after 10 years is visualised in Figure 3.6 The relatedness was calculated using allele frequencies of the founder queens containing *A.m.carnica* and *A.m.mellifera*. Alike to Figure 3.5, we only present year 10 of the simulation in Figure 3.6.

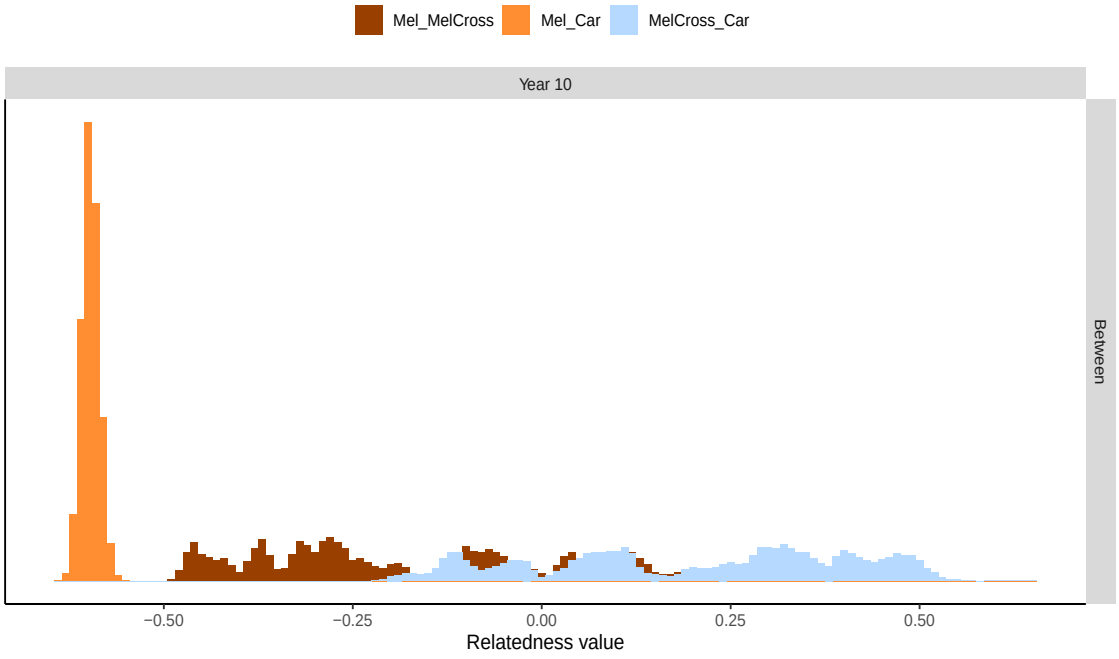


Figure 3.6: Identity-by-state (IBS) relatedness of queens between three different populations in year 10 of the simulation. Comparisons were made between the closed populations (Mel_Car) and between each closed population with the hybrid population (Mel_MelCross and MelCross_Car). IBS relatedness was calculated using allele frequencies of the *A.m.carnica* and *A.m.mellifera* founder queens (MultiAF).

We observed the lowest relatedness between the *A.m.mellifera* queens with the *A.m.carnica* (Mel_Car) queens populations (Figure 3.6), because these two populations have different founders and never cross. Relatedness between Mel queens and MelCross queens (Mel_MelCross) was higher compared to relatedness between Mel and Car population queens (Mel_Car). Although Mel and MelCross populations do not interact in the simulation, they come from the same founder population of *A.m.mellifera* genomes. Relatedness between queens in the hybrid MelCross population and the Car population (MelCross_Car) had the highest values. This is due to the continuous mating of MelCross queens with Car drones, introducing *A.m.carnica* through gradual accumulation within the MelCross population (Eckert et al., 2008).

3.6 Implications

One of the aims of this study was to investigate the impact of inbreeding and hybridisation on the relatedness of honeybee populations. Whilst reviewing the literature, we noticed a lack of studies with clear statement of relatedness metrics and underlying methods. We addressed the question both with the simulations and calculations presented in the results. These results will provide valuable insights into the genetic health and diversity of populations whilst highlighting the need for clear methods for assessing relatedness.

3.6.1 Imports debate within the beekeeping community

There is an ongoing debate within the beekeeping community around the benefits and consequences of importing queens, which impacts performance, fitness, and genetic diversity of receiving honeybee populations. Two of the most commonly imported subspecies are *A.m.ligustica* and *A.m.carnica*, given their desired gentleness and productivity in their native ranges and beyond (Lodesani and Costa, 2003). Advocates of importing queens hence argue that the lower costs of queens, gentler dispositions, higher pathogen or parasite resistance, and a longer period of availability outweigh any potential negative repercussions (Schiff and Sheppard, 1995; Muñoz et al., 2014; Bieńkowska et al., 2021). Imports are also argued to promote the increase of genetic diversity within the local area, however not enough studies have looked into local honeybee diversity and identified problems in the populations to claim that an increase is essential. Previous studies have indicated that such an increase in genetic diversity in honeybee populations can contribute to improved population health, honey yield, and adaptability in the fast-changing ecosystems (Tarpy, 2003; Mattila and Seeley, 2007; Simone-Finstrom et al., 2016). The opposing side posits that the introduction of non-native subspecies may lead to the extinction of the native population through hybridisation, a loss of local adaptation essential to survival in changing-environments, and to potential introduction of diseases to which native subspecies are susceptible (Moritz et al., 2005; Le Conte and Navajas, 2008; Byatt et al., 2016). Such detrimental impacts were seen during the introduction of the African honeybee *Apis mellifera scutellata* into Brazil, as described by Kent (1988). Eckert et al. (2008) suggest that whilst hybridisation may increase local genetic variability, it could lead to an overall

decrease in the global variability through reduction or even loses of local populations. Further investigations are warranted to explore the extent of this limitation in honeybees. Multiple organisations are striving to preserve and conserve native populations, maintaining their purity as much as possible. Conservation schemes focusing on *A.m.mellifera* and located throughout Europe, such as the Native Irish Honey Bee Society (NIHBS, 2012) and the Scottish Native Honey Bee Society (SNHBS, 2017).

The need for simulation tools such as SIMplyBee (Obšteter et al., 2023) is therefore crucial in assessing the potential consequences of queen importation on both managed and wild honeybee populations. These tools allow for the evaluation of spillover effects resulting from neighbouring apiaries' breeding practices, which may unintentionally introduce genetic changes or disease risks into local and wild populations. By incorporating ecological and genetic factors, simulations can provide insights into how importation influences colony dynamics over time and help inform sustainable management strategies.

3.6.2 Impact of the methodology

A comparative examination of different relatedness metrics revealed that the absolute relatedness values depend heavily on the source of information used and which population is considered as a founding/reference. This underscores the importance of understanding the methodology when comparing publications and their results. While previous research has focused on other species, such as cattle (e.g. Makgahlela et al., 2013; Wientjes et al., 2017), there is a gap in the literature about this topic in honeybees. Classic genetic studies report relatedness between individual honeybees (Brascamp et al., 2014, 2018), while most recent quantitative genetic studies evaluate relatedness at the colony level. With the challenge of recording pedigrees in honeybees and growing use of genome-wide genotype data, we have set out to connect the classic relatedness studies with pedigrees to this new source of information. To this end, our results connect the well established eIBD relatedness based on pedigree with rIBD relatedness based on capturing recombination and segregation of genomes within a pedigree, and IBS relatedness based on genotype data. While eIBD and IBS are easily calculated from respectively pedigree and genotype data, rIBD requires identification of recombination and segregation events pedigree and genotype data, which is not trivial from observed data (e.g. Elston and Stewart, 2008; Whalen et al., 2018). All reported IBS relatedness metrics in this study are correct, but the assumed reference population from which allele frequencies are calculated has a large influence and care must be taken when comparing IBS relatedness metrics across studies. To calculate the IBS relatedness we used the VanRaden method 1 (VanRaden, 2008), which weighs all loci equally. As highlighted by Goudet et al. (2018), the choice of the most suitable marker-based estimator depends on the purpose of kinship estimation. If the objective is to estimate heritability and the proportion of related individuals is small, genomic relationship matrices (GRM) using VanRaden's method 2 would have been preferable (VanRaden, 2008). These considerations are particularly relevant in honeybee breeding programs, where the choice of estimator can impact the interpretation of genetic relationships and subsequent management decisions. There is a large number of other

IBS relatedness methods (Speed and Balding, 2015), which also have implied assumptions, such as weighing loci differently according to their allele frequency (giving larger weight to loci with rare alleles) (Yang et al., 2010) or linkage disequilibrium between loci (equalizing weights of loci in linkage) (Speed et al., 2012). We have focused on individual relatedness throughout and left colony level relatedness for future work.

3.6.3 Simulation limitations

To streamline and expedite the simulation and calculation run-time, we only used 1000 segregating sites per chromosome, which likely underestimated the relatedness at the whole-genome level (e.g. Makgahlela et al., 2013; Eynard et al., 2015). A founder population containing 1600 *A.m.mellifera* and 800*A.m.carnica* founder haplotypes were used to create three founder populations of unrelated virgin queens, 800 *A.m.carnica* and 800 *A.m.mellifera* and a hybrid MelCross population contain 800 *A.m.mellifera* virgin queens. These numbers are purely demonstrative, and while they reflect a large-scale breeding program focused on maintaining genetic diversity, real-world honeybee breeding programs typically start with a smaller founder population, often ranging from a few hundred to a thousand individuals per subspecies. Furthermore, we only created 1000 workers and 50 drones per colony. In real colonies, there can be up to 60,000 workers and a few thousand drones present in a colony at one time (Bodenheimer, 1937; Page and Peng, 2001). Whilst our numbers are smaller, they can be treated as a representative sample of a realistic setting.

This sentiment can also be applied to our choice of the ratio of 300 virgin queens mated with drones bred from 100 DPQs, but in real-world breeding programs, the ratio may be different. In practice, breeding programs tend to use a lower number of drone-producing queens (DPQs) relative to virgin queens. A more typical ratio might be 1 DPQ for every 10-20 virgin queens, depending on the goals of the program and the breeding practices. Moreover, we ran the simulation for only 10 years, which given the generational interval of 1-2 years is not too short a time period to see results. Our 10-year period was simply for demonstration purposes but it did enable us to show the trends of inbreeding and hybridisation. The import percentage used in this example was set unrealistically high, fixed to 50% and sourced from only one foreign population. This deliberate exaggeration aimed to highlight the impacts of hybridisation on the relatedness values within the colonies. In reality, imported animals would likely originate from numerous source populations. As described above, a more realistic import percentage would be $\sim$ 10% (BeeBase, 2022; gov.uk, 2021), though this number would likely fluctuate depending on the time of year and location of the hives.

The demographic model outlined in Wallberg et al. (2014) provides valuable insights into the genetic structure of wild honeybee populations, particularly with respect to the low linkage disequilibrium (LD) observed, where r^2 values are typically less than 0.1 at a genomic distance of 1 kb. Here, r^2 represents the correlation between alleles at two genetic loci, quantifying the degree to which alleles are inherited together rather than assorting independently, and values below 0.1 suggest relatively low LD and high genetic diversity in outbred populations. However, this model primarily reflects larger, outbred populations. In contrast, in closed honeybee breeding nuclei with small popu-

lation sizes, we expect higher r^2 values due to factors such as inbreeding, genetic drift, and reduced effective population size (Ne). These factors contribute to increased LD and reduced genetic diversity, particularly over short genomic distances, which may result in a more constrained genetic landscape compared to the larger, more diverse populations studied by Wallberg et al. (2014). This difference underscores the importance of considering population structure when simulating or interpreting genetic data in managed honeybee breeding programs.

3.7 Conclusions

This paper demonstrated the principles of honeybee genetic relatedness using simulation, exhibiting the potential of simulations in the study of quantitative methods as well as establishment or optimisation of honeybee breeding programmes. We evaluated the relatedness within a single colony, between queens of the same population, and between queens of different populations, demonstrating an increase in relatedness in closed breeding populations as a consequence of inbreeding as well as a decrease in relatedness within a hybrid population. We found a lot of variation in relatedness depending on the underlying method, which can lead to misinterpretations of the results and highlights the importance of understanding the methodology and assumptions made in calculations. Hence, a complete information of data sources and methodologies used is needed to avoid misinterpretations when calculating and comparing relatedness across studies.

To translate these findings into practical applications, honeybee breeders should adopt standardised methodologies for estimating and reporting relatedness coefficients, ensuring comparability across studies and breeding programs. This includes clearly defining reference populations, specifying marker-based estimation methods, and considering the implications of different genetic models. By aligning methodological approaches, breeders can make more informed decisions to manage genetic diversity, mitigate inbreeding risks, and optimise selection strategies for resilient and productive honeybee populations.

3.8 Acknowledgements

LS and GG acknowledge funding from BBSRC ISP grants

(BBS/E/D/30002275, BBS/E/RL/230001A, and BBS/E/RL/230001C) and support from the BBSRC DTP (EASTBio) CASE PhD studentship with AbacusBio. LS, JO, JB, and GG acknowledge support from the Slovenian Research Agency's research project L4-2624. JO acknowledges support from the Slovenian Research Agency's research programme P4-0133. JB acknowledges support from the Slovenian Research Agency's PhD studentship 1000-20-0401 and the Slovenian Research Agency's research programme P4-0431.

3.9 Conflict of Interest

The authors declare no conflicts of interest.

3.10 Data Archiving

The simulation R package SIMplyBee is available at
`http://www.SIMplyBee.info` with the underlying GitHub repository
`https://github.com/HighlanderLab/SIMplyBee` and on CRAN
`https://cran.r-project.org/package=SIMplyBee`.
The simulation script is available at
`https://github.com/HighlanderLab/lstrachan_honeybee_relatedness`

3.11 Supplementary Figures

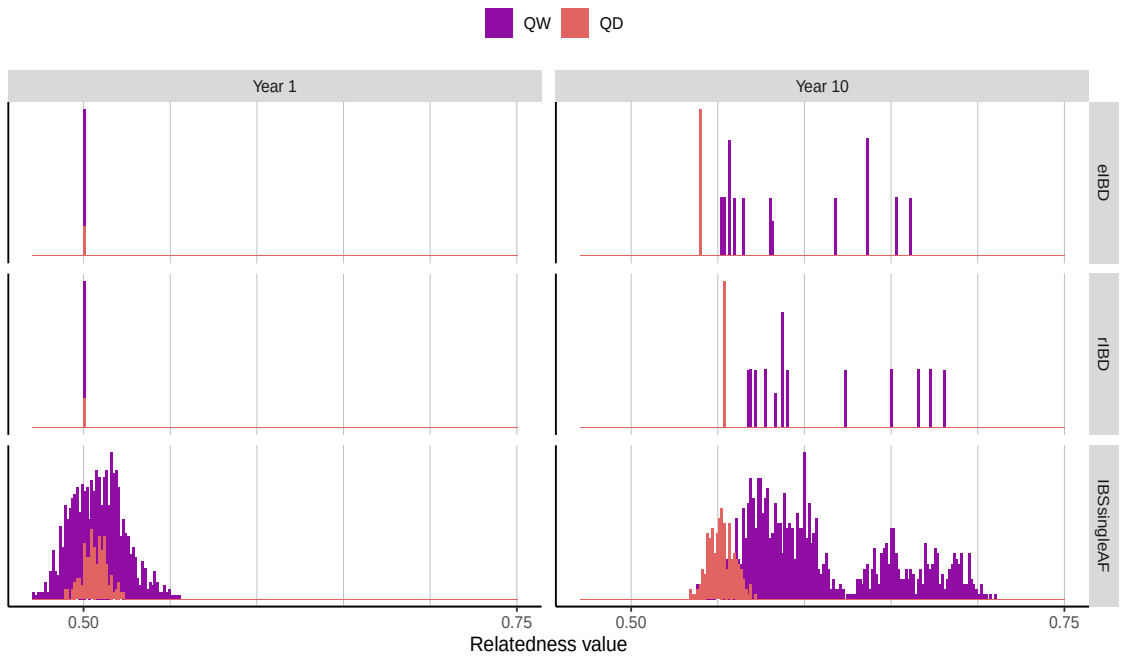


Figure 3.7: Expected and realised identity-by-descent (IBD) and identity-by-state (IBS) relatedness within an *A.m.carnica* colony in year 1 (left) and 10 (right) of the simulation. IBS was calculated using allele frequencies of the founder *A.m.carnica* queens (SingleAF). The figure shows relatedness between queen-to-workers (QW) and queen-to-drones (QD). Vertical lines at every 0.05 between relatedness values 0.5 and 0.75 were added to facilitate comparisons.

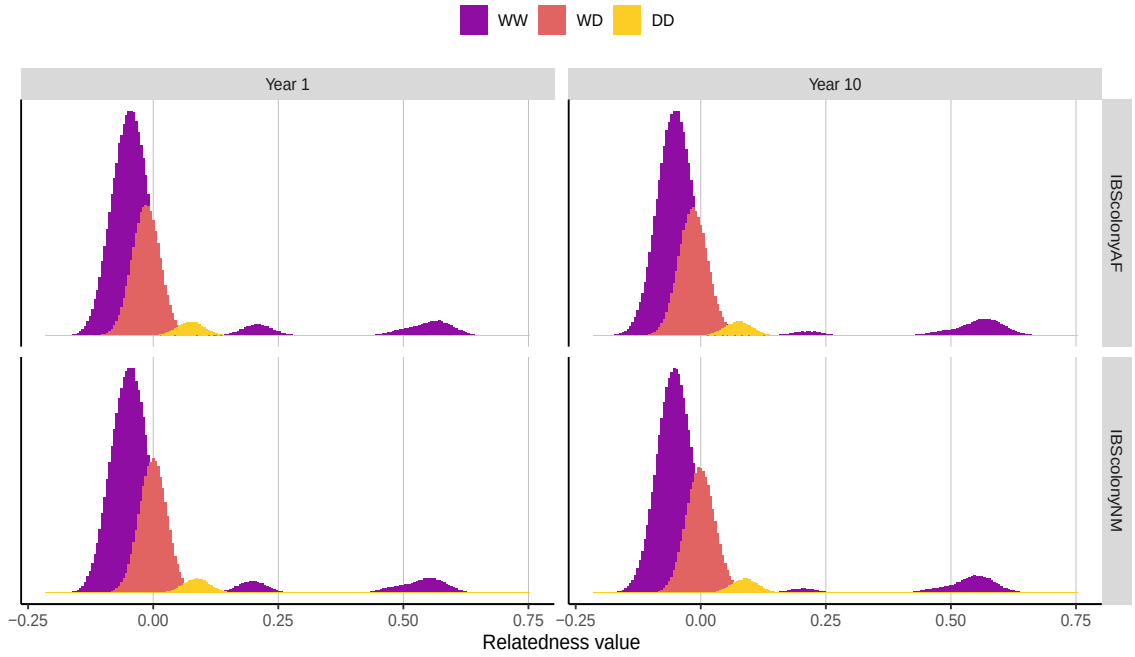


Figure 3.8: Comparison of identity-by-state (IBS) relatedness within an *A.m.carnica* colony calculated using two different allele frequencies - colony allele frequency (ColonyAF) including drones and colony allele frequency without drones (ColonyNM). The figure shows relatedness between workers-to-workers (WW), workers-to-drones (WD), and drones-to-drones (DD) in year 10 of the simulation. Vertical lines at 0.00, 0.25, 0.50, and 0.75 represent the expected within-colony relatedness of a non-inbred colony to facilitate comparisons.

4 Optimizing Genotype Phasing and Patriline Determination in Honeybees Through Pedigree Reconstruction

This chapter contains the manuscript *Optimizing Genotype Phasing and Patriline Determination in Honeybees Through Pedigree Reconstruction* which aims to evaluate pedigree reconstruction and patriline determination in honeybees using both real and simulated data.

Tracking inheritance patterns in breeding programs is crucial for managing genetic diversity, avoiding inbreeding, and ensuring accurate pedigree records, especially for economically important species like the Western honeybee *Apis mellifera*. Honeybee breeders face challenges in obtaining accurate pedigree information due to the species' complex reproductive biology and mating control difficulties. To address this, pedigree reconstruction software uses genomic data, but there is a need for methods that can maternally and paternally assign parent-of-origin to haplotypes estimated via genotype phasing. This study assesses the reliability of different pedigree reconstruction methods for paternal assignments using different SNP array sizes through simulation. We also perform trio genotype phasing and develop a method for determining the parental origin of haplotypes. Based on this, we develop and evaluate a method for determining the number of patrilines in a honeybee colony. The results of this study will enhance the accuracy of pedigree reconstruction in honeybees, improving our ability to manage genetic diversity and avoid inbreeding depression in breeding programmes. Additionally the methods developed could contribute to the broader understanding of honeybee genetic and reproductive biology, aiding efforts to monitor and manage honeybee breeding programmes

The manuscript is a joint work of Laura Strachan[1], Jernej Bubnič[2], Janez Prešern[2], Gregor Gorjanc[1] and Jana Obšteter[2]

[1] The Roslin Institute and Royal (Dick) School of Veterinary Studies, University of Edinburgh, Easter Bush Research Centre, Midlothian EH25 9RG, UK; [2] Department of Animal Science, The Agricultural Institute of Slovenia, Ljubljana, Slovenia;

4.1 Author contributions

This worked was initiated and planned by myself, Gregor and Jana, after which I was the primary contributor to this work, with Gregor and Jana providing support and guidance when required. I was responsible for the development of all code and the processing and analysis of the data for this study. The scripts utilized have been archived in a GitHub repository, which can be accessed at `https://github.com/HighlanderLab/lstrachan_patrilines`. I wrote the first draft of this manuscript, with Jana as my main editor and all other authors contributing to the final version of the manuscript.

Keywords: Honeybee, Genomic data, Pedigree reconstruction, Phasing, Haplotype, Patrilines

4.2 Abstract

Methods for tracking inheritance patterns in breeding programmes is essential for managing genetic diversity, avoiding inbreeding, and ensuring accurate pedigree records that enable genetic improvement. This is the case in particular with economically important species like the Western honeybee *Apis mellifera*. However, due to their complex reproductive biology, honeybee breeders face challenges in acquiring accurate pedigree information due difficulties with controlling mating. Pedigree reconstruction softwares incorporate genomic information to verify or improve pedigree records, which when used with other genomic techniques such as genotype phasing and parent-of-origin haplotype assignments, provide more precise methods for identifying patrilines. We identified the needs for a method that take haplotypes from phased genotype data, where parent-of-origin is often still unknown, and assign them based on calculation of parental contribution. We evaluated pedigree reconstruction and patriline determination in honeybees using both real and simulated data. Specifically, we assessed the accuracy of different pedigree reconstruction methods, performed genotype phasing using the reconstructed pedigree and developed custom R functions to assign haplotype parent-of-origin and determine the number of patrilines per queen. Our results highlight that different software tools and SNP array sizes significantly impact the accuracy of paternal assignments and pedigree reconstruction. In real data, software sensitivity varied, and discrepancies between simulated and actual results underscored challenges in replicating real-world genetic complexity. Additionally, our haplotype assignments revealed deviations from expected Mendelian sampling values, suggesting phasing errors and emphasizing the need for improved methods and data quality in genetic analysis.

4.3 Introduction

A crucial part of every breeding programme is monitoring relatedness to better understand population dynamics and guide breeding decisions. One of the primary uses of relatedness monitoring is to examine and manage genetic variability, to ensure both short- and long-term genetic gain for economically important traits in fast-changing

environments (Espregueira Themudo et al., 2020). By evaluating variability over time, breeders can identify trends and adapt their breeding strategies to optimize trait improvement and maintain a balance between genetic gain and genetic diversity (Schwartz et al., 2007; Hoban et al., 2021). In practice, this is mainly done by monitoring and managing the rates of inbreeding and co-ancestry to make suitable adjustments in the mating scheme to avoid inbreeding depression, which can lead to reductions in fitness and adaptability (Woolliams and Toro, 2007; Fernández and Bennewitz, 2017). Accurate parentage information is key components in these operations. Monitoring relatedness, hence, benefits from a verified pedigree to identifying familial bonds. An accurate pedigree is essential for ensuring precise relatedness calculations and breeding value estimations and selection by providing detailed ancestral information (Wang, 2004a).

The Western honeybee (*Apis mellifera*) has a vital role in pollination services and food production, which makes this species of global economic importance. Consequently, in recent decades, there has been an increased interest in establishing new or optimising already-established honeybee breeding programmes (Gallai et al., 2009; Klein et al., 2007). Due to the complex reproductive characteristics of honeybees, acquiring a population's pedigree information is challenging. A honeybee virgin queen will mate only once in her lifetime on mating flights upon reaching sexual maturity. She will travel to a drone-congregation-area (DCA), where thousands of local male drones gather, and mate with typically around 17 mates (Woyke, 1963), storing a mixture of their sperm in her storage organ, the spermatheca. In a non-controlled environment, this polyandrous mating makes it impossible to visually determine paternity and record the complete pedigree (Remolina and Hughes, 2008; Woyke, 1962).

Honeybee breeders use several mating methods to control mating: artificial insemination, geographical isolation, temporal isolation, and biological isolation. Artificial insemination of a virgin queen provides full control over mating. However, this technique is still uncommon in honeybee breeding, due to its high costs and required training (Gillard and Oldroyd, 2020). Nonetheless, artificial insemination could become more popular with the use of known or estimated breeding values, as it provides unparalleled precision in controlling genetic contributions. This approach is particularly valuable in regions where drone congregation areas (DCAs) are scarce, such as colder climates, where controlled breeding simulations are likely to see greater interest and uptake (Hasnat, 2018). Geographical isolation entails establishing mating stations in a remote location, such as an island or alpine valley, to which virgin queens and drone-producing queens are relocated. This reduces the chance of any unknown drone from feral or managed colony mating with the virgin queens (Uzunov et al., 2017). Temporal isolation, achieved using methods such as the Horner system, manipulate both light and temperature to control the flight times of selected virgin queens and drones (Oxley et al., 2010). Biological isolation involves overflowing the local mating radius with selected drones, increasing the chances of mating with the desired drones. However, there are two main challenges with mating control in honeybee breeding populations. Firstly, despite these techniques being available, they are rarely implemented due to logistical challenges, such as high cost and specialized training requirements. Secondly, even when these techniques are implemented, excluding artificial insemination, they are

unable to determine the exact matings and the resulting number of patrilines, which are the distinct paternal lineages or individual drones, a queen mated with.

In situations where pedigree information is incomplete, analysing genomic data provides a means to determine familial relationships and construct pedigrees. By leveraging genomic information, we can improve the quality and accuracy of pedigrees by verifying existing pedigrees, identifying previously unknown relatives, and determining the number of patrilines. Software for pedigree reconstruction commonly analyse genetic markers such as SNPs and can assign parentage with high precision (Wang, 2019). In practical honeybee research, genotyping is typically performed on drone-producing queens (DPQs) rather than individual drones (Uzunov et al., 2017). As a result, pedigree reconstruction software typically assigns "sires" based on DPQ genotypes rather than the actual drones that fathered the workers. The true paternal drones can only be identified if they are directly genotyped, such as through artificial insemination (Borgia, 1980; Bull, 1981). Pedigree reconstruction software often employ likelihood, relatedness, and exclusion methods to infer pedigree. Exclusion methods determine relatedness by eliminating genotype combinations that are incompatible with specific familial relationships (Calus, 2010). Alternatively, relatedness methods estimate kinship coefficients or pairwise relatedness between individuals to determine familial relationships (Manichaikul et al., 2010). Comparatively, likelihood methods measure the likelihood of parentage by considering all possible pedigree structure arrangements and the probabilities of these configurations across all individuals and loci. Likelihood methods, whilst computationally slower, offer significantly greater power compared to exclusion methods. They can make use of information from heterozygous genotypes, which are disregarded in exclusion methods, thereby inferring familial relationships more accurately (Thompson, 1986; Huisman, 2017; Whalen et al., 2019).

Analysing genomic data can also provide a way of determining the patrilines. Beekeepers and researchers seek to determine the number of patrilines in a honeybee colony for many reasons. First, to assess the efficiency of mating control that aims to inseminate virgin queens with drones of a desired origin and to prevent mating with unknown and feral drones. Hence, assessing the efficiency of mating control requires extracting patriline haplotypes and distinguishing between them to quantify the percentage of them coming from the desired (installed) source (Carroll et al., 2023; Wossler et al., 2006). Second, honeybee breeders and researchers seek to determine the number of patrilines to assess the genetic diversity of a colony, or population. An increased number of patrilines indicates that the queen has mated with more drones, increasing the genetic diversity and strength of the colony leading to increased fitness and adaptability (Tarpy et al., 2013; Ryals et al., 2024).

Traditional methods for estimating patriline numbers included direct mating observations, sperm counts and sperm volume measurements, which were prone to error (Roberts, 1944; Woyke, 1962; Comparini and Biasiolo, 1991). Modern techniques, like SNP arrays offer more accurate and comprehensive genetic profiles by assessing thousands of less-variable loci, addressing the limitations of microsatellite genotypes, which rely on a smaller number of loci and are prone to sampling and sequencing errors

(Tarpy et al., 2013; Mroczek et al., 2024). However, SNP arrays have limitations, such as their inability to detect structural variants or provide the whole-genome coverage necessary for analysing complex genomic rearrangements. In contrast, methods such as whole-genome sequencing offers a more detailed view of genetic variation, including structural variants and rare mutations, though it is often more costly and computationally intensive (Mroczek et al., 2024; Fuentes-Pardo and Ruzzante, 2017).

A method for determining patriline number involves isolating paternal haplotypes from offspring workers and analysing their similarities. This process requires genotype phasing and parent-of-origin assignments to the derived haplotypes. Pedigree-based phasing software, which uses familial relationships and inheritance patterns to track the transmission of alleles across generations, typically assigns parent-of-origin to the haplotypes; AlphaImpute (Whalen and Hickey, 2020) is an example. Alternatively, population-based phasing software, which uses statistical methods and reference panels from unrelated individuals to infer haplotypes, typically does not assign parent-of-origin to the haplotypes; Beagle (Browning and Browning, 2007) is an example. However, incorporating pedigree information into population-based phasing can improve phasing accuracy. Unfortunately, researchers may receive phased data that does not contain parent-of-origin assignments, requiring manual calculations to determine them. To our knowledge, there are no publicly available methods that input haplotypes estimated using phased SNP genotype data and manually allocate haplotypes maternally or paternally.

The principle aim of this paper is to evaluate pedigree reconstruction and patriline determination in honeybees using both real and simulated data.

Specifically, we aim to:

i) assess the reliability of different pedigree reconstruction methods for paternal assignments using different SNP array sizes with simulation,

ii) perform trio genotype phasing using the reconstructed pedigree and develop a method for determining the parental origin of the estimated haplotypes, and

iii) using information above, develop and evaluate a method for determining the number of patrilines in a honeybee colony

4.4 Materials and Methods

We obtained real genotype data from a honeybee mating control experiment and simulated genotype data from a 2-generation honeybee population simulation. Using simulated data, we evaluated the accuracy and limitations of pedigree reconstruction using 4 different software, 4 different SNP array sizes, and 2 datasets: without and with added genotyping errors. We used the method that proved highly accurate on simulated data, and provided the highest number of sire assignments in the real data, to reconstruct the pedigree information of the real data.

Next, we phased the real and the simulated data with the highest SNP array using

Beagle 4.0 (Browning and Browning, 2007), incorporating the reconstructed pedigree information. The haplotypes estimated during phasing were inputted into a custom R function to assign parent-of-origin to haplotypes, from which we extracted all paternal haplotypes and determined the number of fathers that mated with each queen.

R code for the simulations, the custom functions and data analysis is available at `https://github.com/HighlanderLab/lstrachan_patrilines`

4.4.1 Real data

We obtained the genotype data of 288 Slovenian honeybees (*Apis mellifera carnica*), genotyped on a 4K SNP chip (Momeni et al., 2021) as part of the collaborative European initiative "Joint Effort for Honey Bee Conservation and Selection", shortened to BeeConSel (BeeConSel.eu, 2020). The project aimed to evaluate mating controls and included installing 4 drone-producing queens (DPQs) in an isolated Alpine valley (Krma) in Slovenia, relocating 8 virgin colonies for mating, and genotyping the queens, 30 worker offspring per queen, and a pooled sample of the DPQs, presented in Table 4.1.

Table 4.1: Genotyped honeybee samples from the BeeConSel project

Sample Type	Number of Samples
Total Slovenian honeybees genotyped (*Apis mellifera carnica*)	288
Genotyped queens after matings	8
Genotyped worker offspring per queen	30
Genotyped pooled sample of the mating station DPQs	1

We used PLINK (Purcell et al., 2007) to perform quality controls on the data: we filtered out all individuals with more than 10% missing genotypes, and SNPs with more than 10% missing data and minor allele frequency lower than 1%. The resulting data included 280 individuals and 1722 SNPs, with a total genotype call rate of 0.94. The quality control also removed one DPQ and one queen. Therefore, the worker offspring of the excluded queen were also removed from the dataset. Ultimately, we had 247 individuals with 1722 SNPs.

4.4.2 Simulated data

We simulated genotypes using the R package SIMplyBee (Obšteter et al., 2023) that matches the real data. We simulated the genomes of 10 *Apis mellifera carnica* founder individuals, according to the demographic model described in Wallberg et al. (2014), each with 16 chromosomes and 1000 segregating sites (Obšteter et al., 2023). The mutation and recombination rates were kept at the default values of $3.4e^{-09}$ and $2.3e^{-07}$ respectively, as described in SIMplyBee (Obšteter et al., 2023). In addition, we simulated the *CSD* locus, located on the third chromosome, with 128 possible alleles (Zareba et al., 2017; Obšteter et al., 2023). From the founder genomes, a base population of 10

virgin queens was established. To simulate a sister group of drone-producing colonies, first, a base population virgin queen was mated with drones from another base population virgin queen. Second, the mated queen produced 4 virgin queens, which all remained un-mated since no mating is required to produce drones in the simulation. The 4 virgin drone-producing queens each produced 50 drones, which were grouped to mimic a drone-congregation-area (DCA). The remaining 8 virgin queens were mated with the drones in the DCA. The number of father-drones mating with virgin queens was determined using a Poisson distribution with an average of 15. Each of the 8 mated queens produced 30 workers per colony. All pedigree information of the parent-offspring trios (worker/queen/DPQ) was recorded, as were the actual father-drones.

We simulated 4 different SNP arrays to identify the lower limits of SNP number required for accurate paternal assignment (Table 4.2). The largest array of 1728 SNPs (Array-1) was generated to best match the 1722 SNP array used in the Slovenian data collection post quality control. Note that we did not generate an exact SNP array size that matched the real data, as the simulated number of SNPs per chromosome was constant and therefore the total had to be a multiple of 16. From the large SNP array, we created smaller arrays with 800 SNPs (Array-2), 160 SNPs (Array-3), and 16 SNPs (Array-4). These smaller SNP sizes were chosen at random, simply to represent a range of sizes and tests the lower limits of paternal identification.

Table 4.2: SNP array sizes modeled in the simulated data

SNP Array	SNPs per chromosome	SNP array size
1	1	16
2	10	160
3	50	800
4	108	1728

First, we obtained the true simulated genotype data without errors. Next, we introduced errors into the simulated genotypes to better reflect the characteristics of real data, where such errors often occur during a genotype calling process. To incorporate errors into the simulated genotypes, we created a custom R function that introduced two types of errors, genotyping errors and sampling errors. Sampling errors are caused by the under-representation of a population's diversity and were initially implemented across each individual and locus with a probability of 0.005, setting some genotype values to missing.

Genotyping errors were then introduced, occurring with a probability of error1 (0.001). This error rate matched the default rate used in Beagle 4.1 (Browning and Browning, 2016). If an error did occur, as determined by a binomial trial with probability error1, the locus genotype was changed using the parameters error2 (0.00001) and missing (0.005):

- If the genotype was 0, it was changed to 1, 2, or missing with the probabilities 1 - error2 - missing, error2 and missing respectively.
- If the genotype was 1, it was changed to 0, 2, or missing with the probabilities 1 - error2 - missing, 1 - error2 - missing and missing respectively.
- If the genotype was 2, it was changed to 0, 1, or missing with the probabilities error2, 1 - error2 - missing and missing respectively.

The probability of genotyping errors causing missingness (0.005) was calculated from the real data.

The simulated data then underwent the same quality control in PLINK (Purcell et al., 2007) as the real data.

4.4.3 Paternity assignment

We inputted the real data and the simulated datasets with and without genotyping errors into four different pedigree reconstruction software: Sequoia, Colony, AlphaAssign, and KING. Given that the father-drone information in the real data's pedigree was unknown, and the DPQs act as sires in a mammalian sense, we provided DPQ genotypes as potential sires in all pedigree reconstruction software. We used simulated data to assess the accuracy of the pedigree reconstruction using four different software: Sequoia, Colony, AlphaAssign, and KING. Descriptions of these software and their implementation are shown in Table 4.3. Pedigree reconstruction software have limitations and may either fail to assign a sire completely or fail to assign the correct sires when unfavourable data is provided. Using simulated data allowed us to evaluate the accuracy of sire assignments and identify the software limitations.

Next, we extracted assigned sires from the software output files:

- Sequoia: all assigned sires and their log-likelihood ratios.
- Colony: all assigned sires and their likelihood confidence levels.
- AlphaAssign: a binary column (chosen) where 1 indicates that a candidate has been chosen as the most likely sire for the offspring, determined by a likelihood score.
- KING: sire-worker kinship coefficients equal or greater than 0.2 and IBS0 value smaller than 0.005. IBS0 is the proportion of SNP with zero Identity-by-State (IBS), or allele mismatches, meaning that the alleles we sampled shared alleles at over 99.5% of the genetic markers. These thresholds were selected following methods described by Parejo et al. (2024), where kinship and IBS0 ranges were analysed for both known worker-DPQ pairings and incorrect pairings.

We used the simulated datasets to evaluate the accuracy of the sire assignments. We computed the overall accuracy of each software's paternal assignments in the simulated

dataset, including all SNP arrays and both genotyping scenarios, expressed as:

$$\frac{\text{Number of correct sires assigned}}{\text{Number of candidate sires assigned}} \times 100$$

We reconstructed the pedigree information of the real data using AlphaAssign, as it demonstrated high accuracy of sire assignments and the highest number of assigned sires when using the simulated data.

Tool	Sequoia	AlphaAssign	Colony	KING
Reference	(Huisman, 2017)	(Whalen et al., 2018)	(Jones and Wang, 2010)	(Manichaikul et al., 2010)
Method	Likelihood	Likelihood	Likelihood	Likelihood & Relatedness
Data Requirements	– Many SNP markers – High MAF – Low genotype missingness – Low genotyping error rates – Low linkage disequilibrium	– Genotype matrix – List of potential fathers	– Genotype matrix – Individual information (ID/-Sex/Potential parents/Potential sibships) – Reproductive parameters	– Genotype matrix – Many SNP markers for accurate relatedness estimates
Key Features	– No pre-assigned parentage used – Identifies sibling clusters – Assigns 'dummy' parents to sibships where a true parents cannot be determined – Handles different breeding system complexities – Takes possible genotyping errors into account	– Infers sire-offspring relationships – Handles different mating systems – Takes possible genotyping errors into account – Handles missing genotypes	– Can incorporate pre-assigned parentage and sibling information – Handles different mating systems – Takes possible genotyping errors into account – Handles missing genotypes – Allows multiple independent runs – Extensive output files	– Estimates kinship coefficients – Computes the proportion of IBS between individuals' genomes – Designed to handle large datasets – Integrated into PLINK 2.0
Input	– Genotype matrix of all individuals – 'Life History' file (IDs, sex, birth year). Workers and queens assigned female & DPQs assigned male to best assign "paternity".	– Pedigree file (offspring ID, sire ID, dam ID with unknown sires as 0) – Genotype matrix – List of potential sires (DPQs) – Bash script with run-type set to "likelihood" with default thresholds	– Detailed input file including information such as mating system type – Genotype matrix of offspring, dams, and sires – Pedigree information including exclusions (maternal, paternal, and sibship)	– Requires PLINK binary files or VCF files

Table 4.3: A table containing the information of four pedigree reconstruction software: Sequoia, AlphaAssign, Colony, and KING

4.4.4 Phasing

Next, we phased real and simulated genotype data from the largest SNP array size with and without genotyping errors, with Beagle 4.0 (Browning and Browning, 2007). We incorporated the AlphaAssign reconstructed pedigree to improve phasing accuracy. In both real and simulated data, all queens and DPQs had unknown parentage, only workers had known pedigree. We set the Beagle 4.0 windows parameter to 200 and overlap parameters to 50, to better reflect our SNP array. We phased each chromosome individually by performing 200 burn-in iterations, 200 phasing iterations, and 1000 imputation iterations. We will refer to the haplotypes derived by the phased genotypes as estimated haplotypes.

4.4.5 Assigning haplotype origin

Next, we determined which haplotype of the offspring was maternal and which paternal (parent-of-origin) with a custom R function. We will refer to it as the haplotype assignment function. We ran this function with 4 types of data:

- Simulated haplotypes without errors. These haplotypes were perfectly phased, with the first haplotype always being maternal and the second one paternal. We will refer to these as true haplotypes.
- Estimated haplotypes derived from the simulated genotype data without errors, phased with Beagle4.0
- Estimated haplotypes derived from the simulated genotype data with errors phased with Beagle4.0
- Estimated haplotypes derived from the real data genotype phased with Beagle4.0

The function iterated through all 16 chromosomes to compute the difference between the offspring's haplotypes (coded as 0/1) and each of the parents' genotypes (coded as 0/1/2). Similarly to finding opposing homozygotes in parentage assignment, a difference of either 1 or -2 indicated an allelic `MISMATCH` between the offspring and the parent, and values of 0 or -1 indicated a `MATCH` (Table 4.4).

	Parent Geno = 0	**Parent Geno = 1**	**Parent Geno = 2**
Offspring Haplo = 0	0	-1	-2
Offspring Haplo = 1	1	0	-1

Table 4.4: Comparison of offspring haplotypes with parent genotypes.

We summarised the differences across loci with a power mean of allelic mismatches:

$$Score = \frac{\sum(\texttt{MISMATCH})^2}{length(\texttt{MISMATCH})},$$

where the higher the score, the less likely the haplotype was inherited from that parent. The scores were checked using conditional arguments to improve the parent-of-origin assignment accuracy for each offspring haplotype:

- In cases where one haplotype-parent genotype pairing had the absolute lowest `MISMATCH` score, the haplotype was assigned to the tested parent, and the other haplotype was assigned to the opposite parent. For example, if the test between the first haplotype of the offspring and the mother's genotype had the lowest score, then the offspring's first haplotype was assigned as "maternal" and the second haplotype, by default, was assigned as "paternal".
- When the lowest `MISMATCH` was equal, but occured between two different haplotype-parent genotype tests, then no conflict occurs and both are assigned to the tested parent. For example, if the scores when testing the offspring's first haplotype to the mother's genotype and the offspring's second haplotype to the father's genotype were both the lowest and equal, there was no conflict and the offspring's first haplotype was assigned "maternal" and the offspring's second haplotype was assigned "paternal".
- In cases when a haplotype had the same `MISMATCH` score with both parent genotypes, the lowest score on the other haplotype was used to determine the parent-of-origin. For example, when the first offspring haplotype had equal score with both parent genotypes, the second haplotype was checked. If this haplotype had smaller `MISMATCH` score with the mother's genotype, then the second haplotype was assigned "maternal" and the first haplotype was assigned "paternal".
- If all `MISMATCH` scores were equal the function was halted with error.

For a practical example of this function, consider an offspring with haplotypes $[0, 1, 0, 1]$, a mother with genotypes $[0, 1, 2, 0]$, and a father with genotypes $[1, 0, 1, 1]$. Using Table 4.4, we calculate mismatches for the mother: $0 - 0 = 0$ (`MATCH`), $1 - 1 = 0$ (`MATCH`), $0 - 2 = -2$ (`MISMATCH`), and $1 - 0 = 1$ (`MISMATCH`), yielding $[0, 0, -2, 1]$. Squaring these differences gives $[0, 0, 4, 1]$, and the power mean score is

$$\text{Score} = \frac{0 + 0 + 4 + 1}{4} = 1.25.$$

For the father: $0 - 1 = 1$ (`MISMATCH`), $1 - 0 = -1$ (`MATCH`), $0 - 1 = -1$ (`MATCH`), and $1 - 1 = 0$ (`MATCH`), yielding $[1, -1, -1, 0]$. Squaring these gives $[1, 1, 1, 0]$, and the score is

$$\text{Score} = \frac{1 + 1 + 1 + 0}{4} = 0.75.$$

The lower score for the father indicates the haplotype is paternally derived.

4.4.6 Evaluating the accuracy of the haplotype assignment function

To assess the accuracy of the haplotype parent-of-origin assignments, we calculated the gametic Mendelian sampling values of both the simulated data and the real data.

Wright's pedigree model depicts the genetic value of an individual g_i as a sum of the average of parental genetic values, $g_m(i)$ and $g_p(i)$, and a Mendelian sampling value r_i (Wright, 1921). In diploid organisms, this can be expressed using genotypic information as:

$$g_i = {}^1\!/\!_2 g_{m(i)} + {}^1\!/\!_2 g_{p(i)} + r_i.$$

In our analysis, we extended this model to measure deviations in Mendelian sampling at a haplotype level, denoted as:

$$g_{i,maternal} = {}^1\!/\!_2 g_{m(i),1} + {}^1\!/\!_2 g_{m(i),2} + r_{i,maternal},$$
$$g_{i,paternal} = {}^1\!/\!_2 g_{p(i),1} + {}^1\!/\!_2 g_{p(i),2} + r_{i,paternal}.$$

where $g_{i,maternal}$ and $g_{i,paternal}$ represent the genetic values of the maternally and paternally assigned offspring haplotypes. $g_{m(i),1}$ and $g_{m(i),2}$ are the genetic values of the mother's haplotypes, and similarly $g_{p(i),1}$ and $g_{p(i),2}$ are those of the father. The terms $r_{i,maternal}$ and $r_{i,paternal}$ represent the Mendelian sampling value for the respective maternal and paternal offspring's haplotypes. Setting the reference allele to 0, the alternative allele to 1, and allele substitution effect to 1, the genetic value variable here represent the number of alternative alleles, their parent average (expectation) and Mendelian sampling (deviations from expectation). We iterated through every offspring in the pedigree, extracting their parents and calculating the sum of $r_{i,maternal}$ and the sum of $r_{i,paternal}$ to provide an overall measure Mendelian sampling value across all SNPs for each individual. Accurate assignments of haplotype parent-of-origin should result in Mendelian sampling terms with a mean of 0 for both maternally and paternally assigned haplotypes, affirming that the assignments are correct.

4.4.7 Evaluating the accuracy of the phasing

After confirming the haplotype parent-of-origin using true haplotypes, any deviations observed in the phased simulated datasets suggested phasing errors. Deviations in the real data could additionally indicate deviations from Mendelian inheritance patterns, but such patterns were not simulated, meaning that deviations in the simulated data were more likely a result of phasing errors. To evaluate the accuracy of phasing by Beagle4.0 (Browning and Browning, 2007), we compared the estimated haplotypes to the true haplotypes following parental origin assignments. We systematically compared the SNP sequences of the maternally and paternally assigned haplotypes, counting the number of matches and calculating the overall match percentage to quantify the phasing accuracy.

4.4.8 Determining the number of fathers

We used a custom R function to determine the number of drones a queen mated with by quantifying the number of distinct paternal haplotypes in her offspring, that is the number of patrilines. We will refer to this function as the patriline function. Our function operates in three steps:
1) For each queen, we extracted all of her workers' paternal haplotypes. We constructed

a similarity matrix by comparing each worker's paternal haplotype (i) to the potential father haplotypes (j), calculated as:

$$similarity_{[i,j]} = \frac{\sum(haplotype_i \equiv haplotype_j)}{n},$$

where n was the number of loci, and missing data was excluded from comparisons. In the simulated data, we had the true father-drone haplotypes, while in the real data, we considered all eight DPQ haplotypes and treated each one as a separate potential paternal haplotype. Workers were assigned to a patriline-group if the similarity score between their paternal haplotype and a potential father's haplotype surpassed a predefined *father threshold*.

2) Workers who could not be assigned to any patriline-group in the first step were iteratively compared to workers already assigned a patrilines. If an unmatched worker had a similarity score greater than a predefined *sister threshold* when compared to a worker from an established patriline-group, they were deemed super-sisters and added to the patriline-group. The similarity score between the workers was obtained using the same equation as described above, where $haplotype_i$ and $haplotype_j$ are now two paternal haplotypes of workers i and j.

3) Any remaining workers that could not be assigned in either of the previous steps were compared to each other. If pairs of workers exhibited a similarity score above the *sister threshold*, they were grouped together into a new patriline-group. To get the final number of estimated patrilines, we counted the number of patriline-groups per colony.

We ran the patriline function with the true haplotypes, simulated data with and without genotyping errors, and real data. To identify the threshold that would produce the most reliable results with the real data, we ran all datasets with a similarity *father threshold* of 1.0 and 0.95, requiring haplotypes to be 100% or 95% identical to be considered a match. For every *father threshold* we also tested four similarity *sister threshold*, 1.0, 0.95, 0.85, and 0.75, where worker haplotypes had to exceed the thresholds to be deemed sisters from the same father. In total, this produced 8 estimates of the number of patrilines per colony, each derived by applying different similarity thresholds.

We further assessed the proportion of estimated patrilines in the simulated datasets as:

$$\text{Proportion of estimated patrilines}(\%) = \left(\frac{\text{Number of estimated patrilines}}{\text{Number of known patrilines}}\right).$$

We determined the accuracy of the patriline number determination in the simulated datasets as:

$$\text{Accuracy of estimated patrilines}(\%) = \left(\frac{\text{Number of correct patrilines}}{\text{Number of estimated patrilines}}\right) \times 100.$$

The proportion and accuracy of the real data could not be assessed due to the lack of information on the number of actual patrilines.

4.5 Results

4.5.1 Results Summary

Using both real and simulated data, we evaluated pedigree reconstruction and patriline determination in honeybees. Overall, we observed a trade-off in accuracy and speed in pedigree reconstruction software when using simulated dataset. Our simulated results also showed that SNP array size significantly impacted the ability to accurately reconstruct the pedigree, with differences between the software being exacerbated at lower SNP arrays. Deviation in Mendelian sampling were observed across the phased datasets, suggesting that phasing errors occured. The accuracy of patriline determination in the phased simulated datasets was the most accurate at haplotype similarity father threshold 0.95. However, the application of this threshold to the real data predicted an unusually high number of patrilines relative to the number of workers per colony. Deviations between the simulated and real present in all of our results highlight the challenges of simulating real data.

4.5.2 Accuracy of parentage assignment using pedigree reconstruction software

The simulation revealed that while pedigree reconstruction was not significantly impacted by the introduction of genotyping errors, its was notably influenced by SNP array size. Furthermore, the differences between pedigree reconstruction software were exacerbated at lower SNP arrays.

The number of candidate sires assigned to the simulated datasets, using each pedigree reconstruction software, and their accuracies are shown in Figure 4.1. In both scenarios, with and without genotyping errors, all four software assigned a candidate sire to all 240 workers, with 100% accuracy, when using Array-4 (1700 SNPs) and Array-3 (800 SNPs). Using a smaller SNP array, Array-2 (160 SNPs), did not affect the performance of Sequoia, AlphaAssign, and Colony, since they still assigned a sire to all 240 workers with 100% accuracy in scenarios with and without genotyping errors. However, KING only assigned a sire to 183 workers, with 88% accuracy, in simulated data without genotyping errors, and a sire to 182 workers, with 87% accuracy, with genotyping errors. Decreasing the SNP array size even more and using Array-1 (16 SNPs) introduced significant differences in the number of candidate sires assigned and their accuracy between the software. Sequoia assigned 35 and 39 sires, with and without genotyping errors, respectively, both with a 77% accuracy. Colony assigned 239 sires in both genotyping scenarios, however, the accuracy dipped to 70% in both cases. AlphaAssign was unable to assign any sires using Array-1 in any of the simulated data scenarios. KING assigned 269 candidate sires, with 40% accuracy, in simulated data without genotyping errors and 275 candidate sires, with 41% accuracy, with genotyping errors. These numbers both extended the total number of worker offspring (240). Upon

inspection, it was discovered that there were occurrences of worker offspring being assigned two candidate sires, where kinship values were identical and a conclusive sire was unable to be assigned.

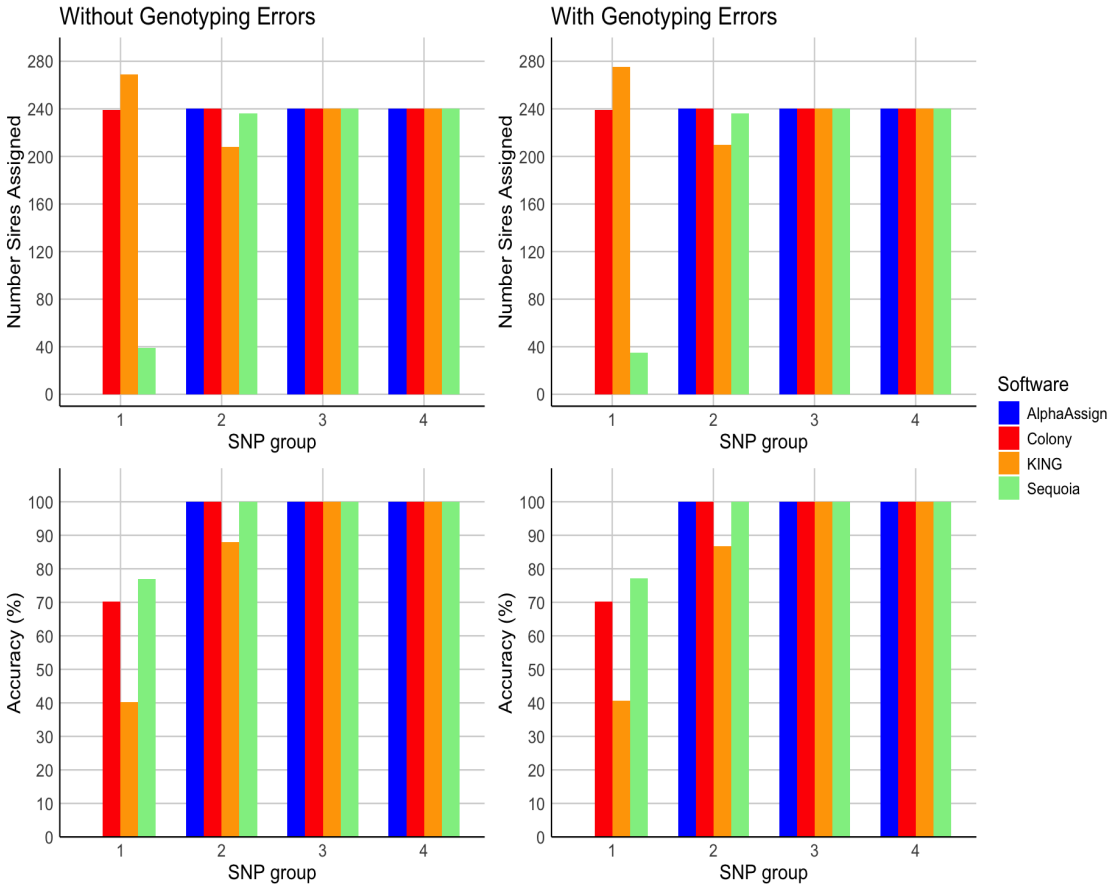


Figure 4.1: Number of sires assigned (top) and accuracy (bottom) by four pedigree reconstruction software using simulated data, with and without genotyping error (top row). SNP arrays had: 16 SNPs (1), 160 SNPs (2), 800 SNPs (3) and 1700 SNPs (4).

When using real data, the number of assigned sires was significantly lower than for the simulated data with corresponding SNP array size (Figure 4.2). AlphaAssign assigned the highest number of sires (127), followed by Colony (87), Sequoia (48), and KING (28). None of these software assigned multiple candidate sires to a single worker. We reconstructed the pedigree information of the real data using the assigned sires designated by AlphaAssign, as it demonstrated 100% accuracy when testing it on the largest simulated SNP array and also assigned the highest number of sires in the real data.

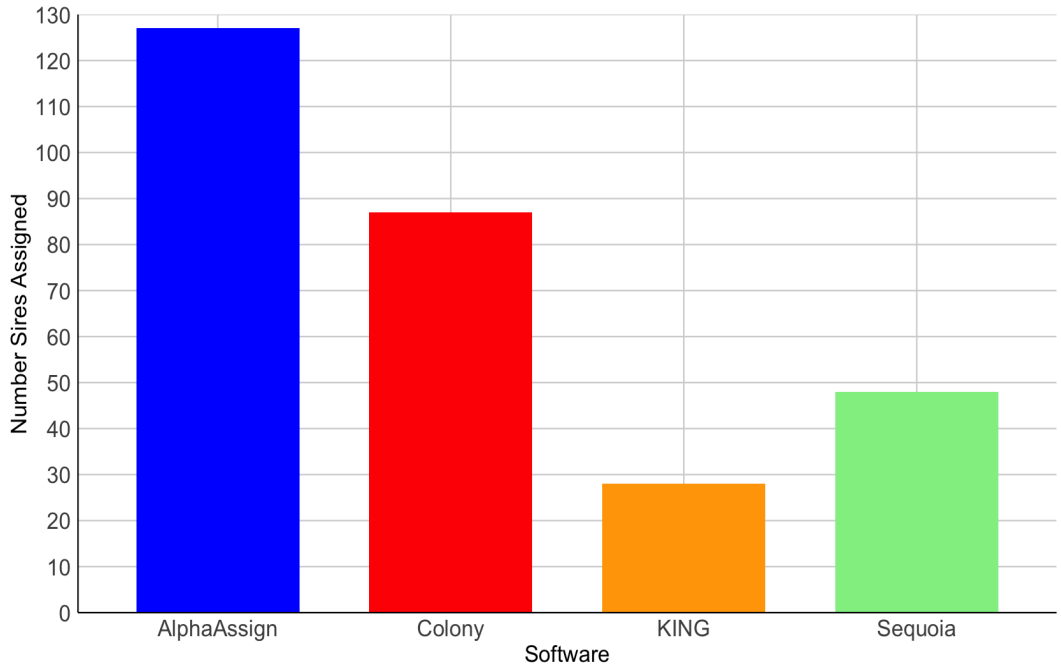


Figure 4.2: Number of sires assigned by four pedigree reconstruction software using real data with 1700 SNPs.

Table 4.5 presents the runtimes of all pedigree reconstruction software, using real and simulated data. Very little variation was seen in the runtimes between the simulated data with or without genotyping errors. KING and AlphaAssign were the fastest and completed their runs in under 1 second, regardless of the dataset or the SNP array. When running Sequoia with simulated data, the fastest runtime was 8 seconds using Array-2 in both genotyping scenarios. Runtime increased incrementally with SNP array size, extending up to 675% using Array-4 in both scenarios. Array-1 was an outlier in both simulated scenarios, taking approximately the same amount of time as Array-4. When run using real data, Sequoia took significantly longer than the simulated data, taking 1.18 hrs (70.73 minutes). Colony took the longest with the runtime increasing with SNP array size. Its minimum run time was 87 minutes using Array-1 without genotyping errors, and its maximum was 32.67 hours (1960.2 minutes) using Array-4 without genotyping errors. The Colony runtime using the real data was 29.75 hrs (1785 minutes), matching its simulated counterpart.

SNP group	Colony	Sequoia	AlphaAssign	KING	Data Type	Genotyping errors
1	87.0	1.1	0.0	0.0	Simulated	FALSE
2	440.4	0.1	0.0	0.0	Simulated	FALSE
3	519.0	0.5	0.0	0.0	Simulated	FALSE
4	1960.2	1.0	0.0	0.0	Simulated	FALSE
1	115.2	1.0	0.0	0.0	Simulated	TRUE
2	388.8	0.1	0.0	0.0	Simulated	TRUE
3	1052.4	0.5	0.0	0.00.0	Simulated	TRUE
4	1792.2	1.0	0.0	0.0	Simulated	TRUE
4	1785.0	70.7	0.0	0.0	Real	TRUE

Table 4.5: Run-time, in minutes, of the pedigree reconstruction software COLONY, Sequoia, AlphaAssign, and KING in real data and simulated data, with and without genotyping errors.

4.5.3 Phasing and Haplotype parent-of-origin assignment

The assignment of the haplotype parent-of-origin on the simulated true haplotypes had 100% accuracy. We also evaluated the haplotype assignment of all datasets by computing the haplotype Mendelian sampling values (Figure 4.3), which were expected to have a mean of 0 when haplotypes were correctly assigned. The true haplotypes ($True$) had an average of Mendelian sampling values of -0.54 (range -33 to 33) for the maternally assigned haplotypes, and 0.2 (range -29.5 to 28.5) for the paternally assigned haplotypes. The estimated haplotypes of the simulated data without genotyping errors (Est_nGE), had the average Mendelian sampling value for the maternally and paternally assigned haplotypes of -0.69 (range -31 to 32) and -0.14 (range -55.5 to 47.5), respectively. In contrast, the estimated haplotypes of the simulated data with genotyping errors (Est_GE) had an average Mendelian sampling value of -0.4 (range -30.5 to 33.5) for the maternally assigned haplotypes and -0.18 (range -55.5 to 51.5) for the paternally assigned haplotypes. For the estimated haplotypes of the real data (Est_Real) we saw a deviation from the expected mean of 0. The average Mendelian sampling value for the maternally assigned haplotypes was 0.31 (range -35.0 to 31.5) and 6.1 (range -23.5 to 32.5) for the paternally assigned haplotypes.

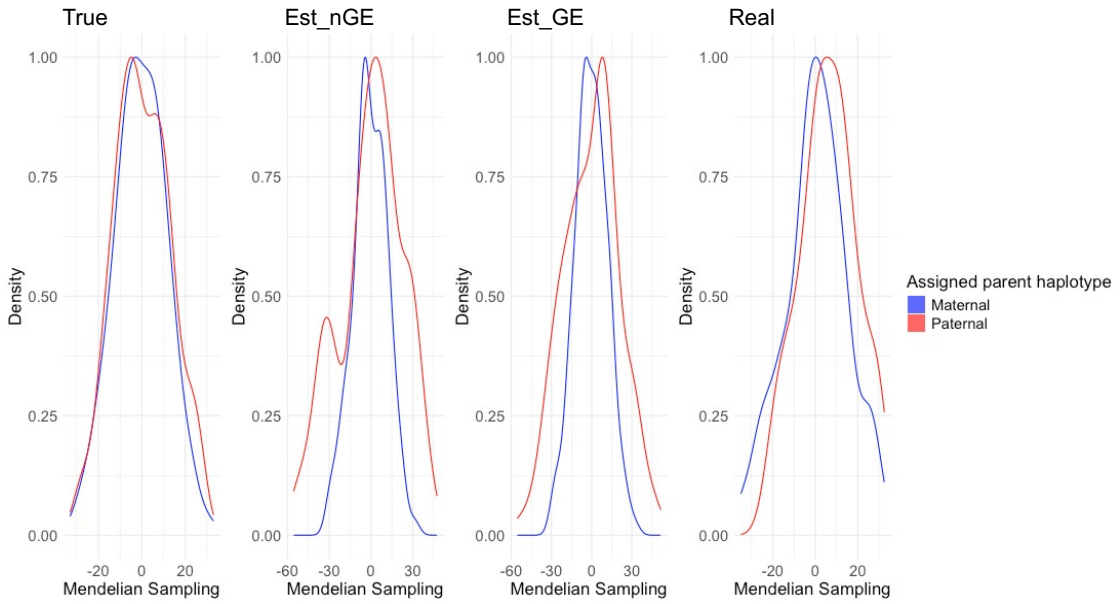


Figure 4.3: Summarised Mendelian sampling deviations of the maternal and paternal haplotypes for each offspring, representing the overall Mendelian sampling values across all SNPs. Four haplotype scenarios are shown: True haplotypes taken directly from the simulation ($True$), estimated haplotypes of the simulated data without genotypic errors (Est_nGE), and with genotyping errors (Est_GE), and estimated haplotypes of the real data (Est_Real).

When comparing the true maternal haplotype to the maternally assigned haplotypes of the simulated data without genotyping errors, the match was 99.6%, as was the case when comparing paternally assigned haplotypes. In contrast, the true maternal haplotype had a 85.1% match with maternally assigned haplotypes of the simulated data with genotyping errors, while the match for the paternal counterpart was 80.1%.

4.5.4 Determining the number of patrilines

The numbers of patrilines per colony determined from the estimated haplotypes of the simulated data, Est_nGE and (Est_GE), are presented in Figure 4.4, using various sister thresholds (1.0, 0.95, 0.85, and 0.75) and father thresholds (1.0 and 0.95). A similarity father threshold of 1.0 consistently resulted in inaccurate estimation of patriline numbers. Specifically, a father threshold of 1.0 paired with a sister threshold of 1.0 led to the incorrect conclusion that each of the 30 offspring had a unique father in five of the eight of the simulated dataset without genotyping errors (Est_nGE) and seven of the eight colonies with genotyping errors (Est_GE). This overestimated the actual number of patrilines by up to threefold (Supplementary Figure 4.6). However, the accuracy of the estimated patrilines was extremely low with a father threshold of 1.0 and a sister threshold of 1.0, averaging 2.6% in Est_nGE and 3.8% in Est_GE. Results also showed that with a father threshold of 1.0, the proportion of estimated patrilines decreased as the sister thresholds decreased, incrementally getting more cor-

rect (Supplementary Figure 4.6). While this trend represented an improvement, the estimated number of patrilines remained overestimated even at sister threshold of 0.75. Nevertheless, the accuracy improved with lowering of the sister thresholds, reaching an average accuracy of 43.8% in the (Est_nGE) and 75% in the (Est_GE) at sister threshold 0.75. When a father threshold of 0.95 was applied, Figures 4.4 and Supplementary 4.6 estimated the number of patrilines with 100% accuracy across all sister thresholds, in both (Est_nGE) and (Est_GE) datasets. The number of patrilines determined using true haplotypes is not displayed in Figure 4.4, as they consistently achieved 100% accuracy across all similarity thresholds.

The number of patrilines determined in the real dataset is shown in Figure 4.5. Since the father-drone information was unavailable for the real data, we were unable to determine the accuracy of determining the patriline number. Unlike the simulated data, which had 30 worker offspring per queen, the number of offspring in the real data varied, ranging from 8 to 21. Changing the father threshold did not impact the estimated number of patrilines when the sister threshold remained constant. However, varying the sister threshold led to significant differences in the estimated number of patrilines. Specifically, using sister thresholds of 1.0 and 0.95 led to the conclusion that each of the offspring had a unique patriline in all and six out of eight colonies, respectively. As expected, the number of estimated patrilines decreased by lowering the sister threshold. The most significant decrease occurred when the number of estimated patrilines dropped from 21 using a sister threshold of 1.0 to 12 when using sister thresholds of 0.75 and 0.85, representing a 42.9% decrease in the number of estimated patrilines.

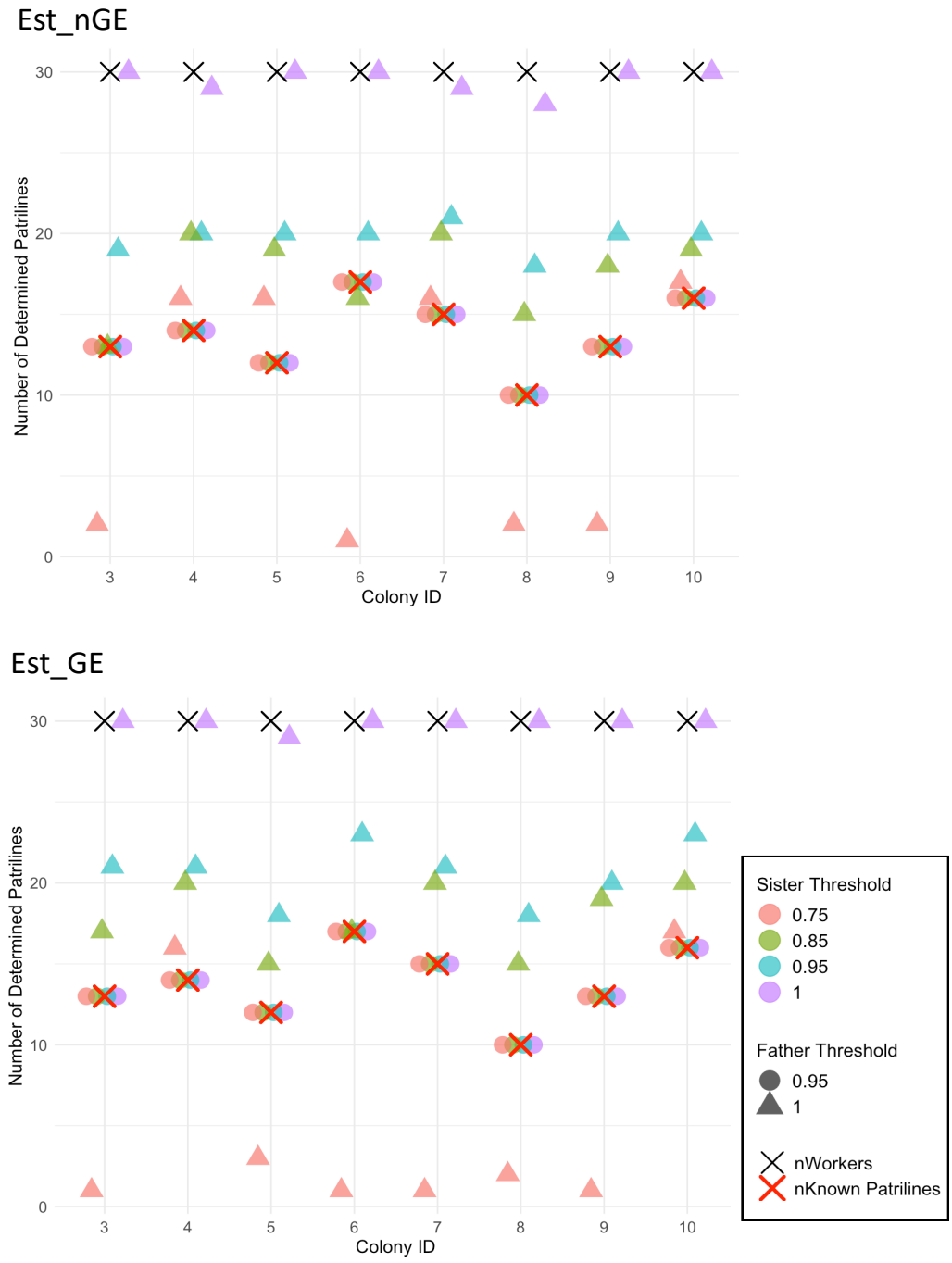


Figure 4.4: The determined number of patrilines per colony (Colony Id) in simulated data. We used the estimated haplotypes of the simulated data without genotyping errors (Est_nGE), and the estimated haplotypes of the simulated data with genotyping errors (Est_GE). We also show the actual number of fathers in the simulated data (Known nPatrilines) and number of workers (nWorkers) available per queen used to determine the estimated number of patrilines. Similarity father thresholds of 1.0 and 0.95 were used in all calculations, where haplotypes had to be 100% and 95% identical, respectively, to be categorised as a match. Similarity sister thresholds of 1.0, 0.95, 0.85 and 0.75 were used to determine which worker haplotypes were sisters.

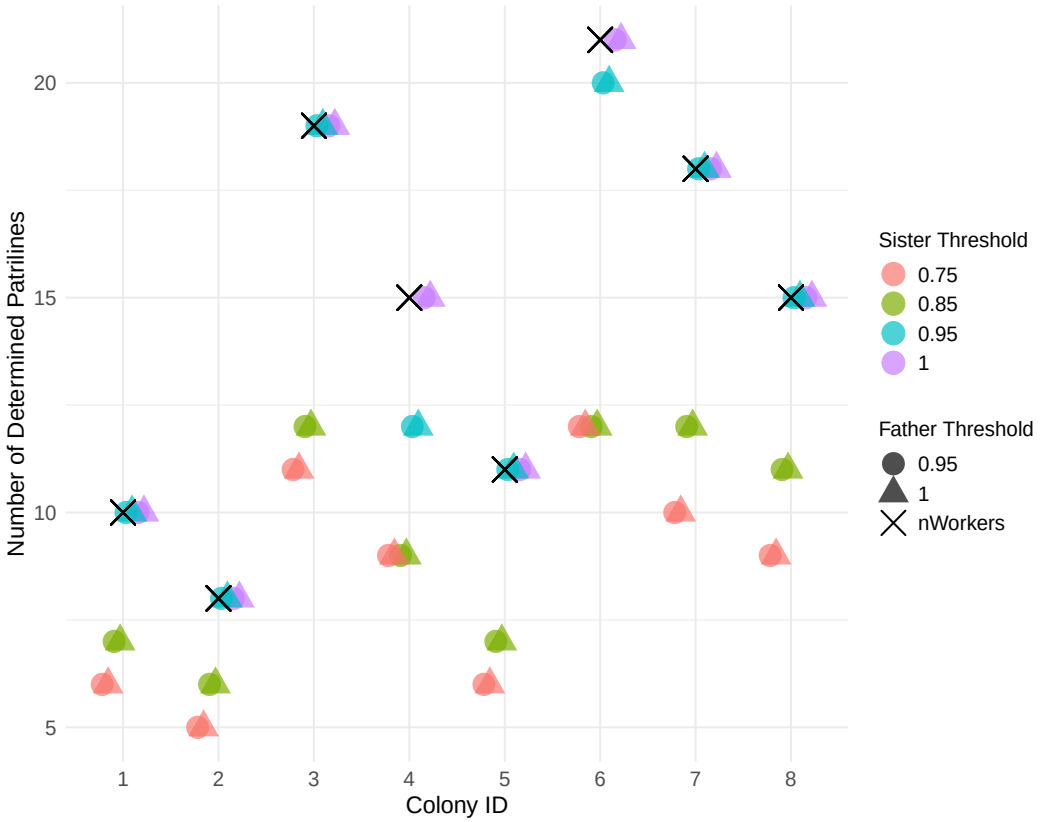


Figure 4.5: The determined number of patrilines per colony (Colony Id) using the estimated haplotypes of the real dataset. We also show the number of workers (nWorkers) available per queen used to determine the estimated number of patrilines. Similarity father thresholds of 1.0 and 0.95 were used in all calculations, where haplotypes had to be 100% and 95% identical, respectively, to be categorised as a match. Similarity sister thresholds of 1.0, 0.95, 0.85 and 0.75 were used to determine which worker haplotypes were sisters.

4.6 Discussion

This study aimed to test and develop tools for paternity assessment and assignment in honeybees by evaluating pedigree reconstruction and patriline determination, using simulated and real data. In the following section, we first discuss the accuracy of paternal assignments using different software and SNP array sizes. Next, we discuss the accuracy of the haplotype parent-of-origin assignments and possible explanations for deviation in gametic Mendelian sampling values, as well as phasing and file conversion limitations. Finally, we discuss the limitations of patriline number determination before exploring the implications of the study.

4.6.1 Pedigree reconstruction

One of the aims of this study was to assess the reliability of four pedigree reconstruction software (Colony, AlphaAssign, Sequoia, and KING) for paternal assignments. In

the simulated data overall, Colony exhibited the most consistent performance across different SNP array sizes and genotyping scenarios, maintaining high accuracy and a stable number of candidate sires assigned. This reliability can likely be attributed to its complex input file, allowing users to incorporate detailed information, such as sibship/parental inclusions or exclusions, to aid sire assignment (Wang, 2004a). However, this accuracy came at a cost, since Colony's input file was the most complex to setup and the runtime was significantly longer than other software. For instance, Colony's runtime extended up to 32.67 hours in the largest SNP array without genotyping errors, whereas AlphaAssign and KING had a runtime below 1 second in all datasets.

These findings align with the observations of Rentof et al. (2024), who compared Colony and Sequoia in reconstructing lateral pedigrees of sea lamprey, utilizing more individuals but fewer loci. They found that Sequoia's speed was markedly faster (less than 5 minutes) compared to Colony's (approximately a week). However, they also observed that while Sequoia's efficiency was advantageous, its accuracy in more complex kinship scenarios may be compromised, particularly in species like sea lamprey with many half-siblings. We also observed this in our results, where Sequoia performed well with larger SNP arrays, but assigned a candidate sire to fewer workers and had a reduced accuracy using smaller SNP arrays.

AlphaAssign also showed high accuracy with larger SNP array, but was unable to assign any candidate sires when using the smallest SNP array with only 16 SNPs (1 SNP per chromosome), highlighting the software's SNP array size limitation. Parejo et al. (2024), who employed KING for paternity assignments in honeybee using a 4.5K SNP chip on a larger dataset (384 samples), reported high accuracy (97.7%) in samples with known ancestry. This high accuracy partly corresponds with our findings showing that while KING's performance benefits from larger datasets with more SNP arrays, it is strongly influenced by the SNP array size. In larger datasets, KING's simplicity, requiring only a VCF file, might be an advantage over software that require complex inputs (Manichaikul et al., 2010). However, in our smaller-scale datasets, with fewer samples and less SNPs, KING exhibited greater variability in performance between SNP arrays when compared to other tested software.

It is important to clarify that the "sires" we were aiming to assign were drone-producing-queens (DPQs), which establish the DPQs, and not the actual drones that fathered the workers. Drones possess only one copy of the genome inherited from their DPQ and are only able to transmit non-recombined genetic information from their maternal lineage (Borgia, 1980; Bull, 1981). This effectively makes drones "flying gametes" while the DPQs represent the recombining "paternal" lineage for workers, acting like mammalian fathers (Dzierzon, 1845). Furthermore, in practice, honeybee breeders would typically obtain DPQ genotypes but not specific drone genotypes (Uzunov et al., 2017). Honeybee mating naturally occurs on-the-wing at elevated altitudes, after which the drones die, preventing breeders from obtaining specific drone information through observation alone. Consequently, breeding must rely on DPQ information to construct the paternal pedigree.

Sire assignments in real data

Using the same SNP array size (1700 SNPs), the number of workers with a candidate sire assigned differed significantly between the real and simulated datasets. All 240 workers in the simulated data were assigned a sire whereas in the real data with 235 workers, AlphaAssign performed best by assigning a candidate sire to 127 workers, accounting for only 54% of the offspring. This indicates different levels of sensitivity and specificity among the software programs when dealing with real data, which can be influenced by several factors often not fully captured in simulations.
In the simulated datasets, founder genotypes are generated based on a demographic model, in our case that of Wallberg et al. (2014). However, demographic models will likely not capture the impact of local historical events, such as bottlenecks or admixture. Furthermore, coalescent simulators do not simulate selection or recent breeding. Such events can result in increased inbreeding rates, more rare alleles, and unique patterns of linkage disequilibrium (LD), not fully represented by the simulated data (Falconer and Mackay, 1996; Cridland et al., 2017; Nordborg and Tavaré, 2002). We can mitigate this by simulating a burn-in phase that simulates recent breeding and population management before evaluating future scenarios. The default recombination and mutation rates in the simulation may also not fully match those of the real population. This could result in genotype differences and, consequently, discrepancies in candidate sire assignments, even when using the same dataset sizes (Yang et al., 2010; Beye et al., 2006). Additionally, the demographic model in Wallberg et al. (2014) may not fully capture all the complexity of LD in real populations. Moreover, while the founders in the simulated dataset have shared ancestry, since their genomes are generated from the same coalescent simulation, we might not have simulated sufficient relatedness among the founders to accurately reflect the relatedness observed in the real data. Furthermore, although SIMplyBee is a stochastic simulator, it may not fully capture all sources of randomness observed in the real data (Obšteter et al., 2023). Additionally, the LD of the real data could have been estimated using tools like PLINK, but this would require a denser SNP array than the 1700 SNPs we had available post quality control. Overall, these factors highlight the challenges of accurately mimicking real data with simulations, even when the simulation is designed to closely replicate real characteristics. The low number of workers with candidate sire assignments in the real data also suggests that the quality of data was poor in comparison to the simulated data, indicating that more rigorous quality controls should have been implemented. However, the discrepancy may also stem from more complex issues specific to real datasets, including missing or incomplete genotyping, phasing errors, and sampling biases, all of which can significantly reduce the accuracy of sire assignments. These issues, often exacerbated in real-world settings, are harder to control for than in simulations, and they can lead to incorrect assignments.

4.6.2 Parent-of-origin haplotype assignments

We found a gap in the literature for a method that assigns parent-of-origin to estimated haplotypes, derived through phasing genotypes. There are some phasing software that assign haplotype parent-of-origin during phasing, like AlphaImpute (Whalen

and Hickey, 2020). However, it is not always possible to use these specific software, since they may not be tailored to specific needs or species, or they might lack the flexibility to incorporate additional information, such as pedigree data. Researchers may also not receive the raw data required to perform phasing themselves. This can be due to several reasons, such as privacy or ethical concerns, or because the raw data is processed separately in a collaborative setting, with only the phased data being shared among collaborators (Bonomi et al., 2020; Byrd et al., 2020).

Determining the accuracy of our haplotype function was challenging, as phasing introduces errors in both real and simulated data. However, using gametic Mendelian sampling values to evaluate the assignments provides an effective method for evaluating the function's accuracy. The average Mendelian sampling values of the true haplotypes, in both maternally and paternally assigned haplotypes, were close to the expected mean of 0, indicating that offspring haplotypes assigned as "maternal" were the combination of their queen's haplotypes, and the ones assigned as "paternal" were a combination of their DPQ's haplotypes.

Our calculations of Mendelian sampling deviations using haplotype information are more accurate compared to calculations using genotypic information, as it allows for the precise assessment of the genetic contributions and recombination events. By leveraging this level of precision, we can gain deeper insights into how traits and genes are passed through generations.

If phasing had been 100% accurate we would expect to see the same Mendelian sampling values in the estimated haplotypes of the simulated data without genotyping errors as in the true haplotypes. However, slight deviations were observed, with the average Mendelian sampling value positive in maternally assigned haplotypes and negative in paternally assigned haplotypes, suggesting that an error had been introduced in the phasing process, since no Mendelian deviations had been simulated in the true haplotypes. These deviations from the mean 0 were also observed in the estimated haplotypes of the simulated data with genotyping errors. Unexpectedly, the Mendelian sampling values of the estimated haplotypes of the simulated data with genotyping errors were more similar to the true data than those without genotyping errors. This suggests that the imputation processes performed by the software inadvertently corrected some of the genotyping errors.

Mendelian sampling values of the real data showed deviations from the expected mean in both maternally and paternally assigned haplotypes. Additionally, the range of Mendelian sampling values observed in the real data was wider compared to simulated haplotypes, indicating potential complexities in real-world genetic inheritance that were not fully captured by the simulation. Such deviations in the real data could arise from factors like unexpected deviations from Mendelian inheritance, errors in assigning parentage, or inaccuracies in the data itself.

In this context, "unusual Mendelian inheritance patterns" refer to cases where offspring do not inherit genetic material in the expected ratios predicted by Mendel's laws of inheritance. For example, these patterns might involve skewed allele segregation, genetic mechanisms altering typical inheritance ratios, or reproductive anomalies. In honey-

bees, deviations as a result of unusual Mendelian inheritance have only been reported in the African subspecies *Apis mellifera capensis*, whose workers exhibit thelytoky, a form of asexual reproduction that produces female offspring (Chapman et al., 2015). However, these abnormal Mendelian inheritance patterns have not been reported in *Apis mellifera carnica*. Alternatively, a skewed distribution of Mendelian sampling values may indicate phasing issues in both the real and simulated data. As we did not simulate any unusual Mendelian inheritance patterns, any deviations in the simulated data are likely due to phasing errors. Honeybees are known to exhibit high recombination rates relative to their physical genome length (Beye et al., 2006), which increases the likelihood of crossover events during meiosis and complicates phasing. Although the overall number of Morgans in honeybees is not significantly larger than in other species, this occurs over a much smaller physical genome length. As a result, the recombination breaks the genome into smaller physical pieces, making phasing a more complex puzzle. Recombination may also be concentrated in certain regions of the genome, which could amplify the effects of gene conversion, where genetic material is non-reciprocally transferred between homologous chromosomes in the queen or DPQ (Liu et al., 2015; Marais, 2003). This could also amplify the effects of gene conversion, where genetic material is non-reciprocally transferred between homologous chromosomes in the queen or DPQ (Liu et al., 2015; Marais, 2003). This would also result in unexpected allele combinations in the offspring. Moreover, high recombination rates could amplify the effects of chromosomal abnormalities, such as translocation or inversion, and also cause deviation by disrupting the segregation and recombination processes during meiosis (Rueppell et al., 2016; Salmela et al., 2022).

The phasing could be improved with an increased number of high-quality SNPs and samples. More data would reduce the number of phasing errors by providing more data points and genetic variation information which would aid the phasing algorithm in distinguishing between different haplotypes (Browning and Browning, 2007; Wang, 2004a; Wertenbroek et al., 2024). Incorporating phased reference genomes for honeybees could also improve phasing accuracy (Browning et al., 2021).

We also used an older version of the phasing software Beagle (Beagle4.0), as it was able to incorporate the reconstructed pedigree information (Browning and Browning, 2007). We theorised that more recently developed phasing software would have improvements that could reduce the phasing inaccuracies on small datasets such as ours. To address this, we also tested phasing using Beagle 4.1 (Browning and Browning, 2016) which does not include a pedigree input, however, this proved to have identical results to those of Beagle 4.0 (Browning and Browning, 2007), suggesting no improvement in accuracy. This finding indicates that the incorporation of pedigree information did not impact the phasing accuracy in this particular dataset.

File conversion inconsistencies

When calculating Mendelian sampling using a PLINK PED file (Purcell et al., 2007), derived from the phased VCF output file generated by Beagle4.0 (Browning and Browning, 2007), we observed significant deviations in the Mendelian sampling values, shown in Supplementary Figure 4.7. Upon further investigation, we identified that during the

conversion from VCF to PLINK PED format the haplotypes of certain markers (but not all) were incorrectly converted from 0/1 to A/C/G/T notation. This effectively disrupted the phase and introduced errors in the haplotypes. These findings highlight the importance of rigorous validation of file conversions to endure the integrity of allele order post-phasing and avoid potential errors in downstream analysis.

4.6.3 Number of patriline determination

In our study, we estimated the number of patrilines by counting distinct paternal haplotypes in a colony. However, with the presence of genotyping and phasing errors, determining if two haplotypes are identical is not straightforward. We evaluated two different similarity father thresholds (1.0 and 0.95) that two haplotypes needed to meet to be marked as coming from the same drones. Additionally, we evaluated four different similarity sister thresholds (1.0, 0.95, 0.85 and 0.75) that two haplotypes needed to meet to be marked as super-sisters from the same father.

Minor discrepancies were observed between the estimated haplotypes derived from the phasing of the simulated genotype data with and without genotyping errors. Similar to the Mendelian sampling results, the estimated haplotypes from phased simulated genotypes with genotyping errors were slightly more accurate that those without, suggesting that they were less affected by the phasing process. In the simulated data, a similarity father threshold of 0.95 consistently provided the most reliable results across all sister thresholds, as the number of patrilines was identified with 100% accuracy in all cases. This indicates that the father threshold had a more significant influence on the results than the sister threshold. However, we also demonstrated that the highest father threshold (1.0), when paired with the lowest sister threshold (0.75), yielded the most accurate estimate suggesting that a balance between the similarity thresholds can be achieved.

In contrast, the real data showed the opposite pattern, with the father threshold having no effect on the estimated patriline numbers, while discrepancies arose between the sister thresholds. Despite the simulated data suggesting that a father threshold of 0.95 should yield accurate patriline estimates, applying this threshold to the real data produced improbably high estimates, far exceeding the expected number of patrilines based on colony size. In fact, seven of the eight colonies indicated that nearly every worker originated from a different father, which is highly unlikely. These discrepancies highlight the challenge of translating simulated expectations into real-world scenarios, as simulations may not fully account for the biological and genetic complexities present in the real data. Factors such as incomplete sampling, hidden population structure, or more complex mating patterns may have contributed to these unexpected results. Additionally, the high recombination rate observed in honeybees may have led to phasing errors that influenced patriline estimates. These phasing errors can complicate the identification of true genetic inheritance patterns, making it challenging to accurately reconstruct patrilines. These discrepancies underscore the need for a more flexible threshold when analyzing real-world data, as the strict 0.95 father threshold suggested by the simulations did not align with the observed patterns in the actual colonies.

The number of patrilines that can be determined also depends upon the number of available paternally assigned haplotypes. Initially, the real dataset had approximately 30 workers per queen. While providing the highest number, AlphaAssign assigned only 127 sires, resulting in a 54% reduction in the number of offspring that we could use to undergo haplotype parent-of-origin assignments and determine the number of fathers that had mated with each queen. This reduction lowered the worker count per queen to a range of 8-21, potentially limiting the genetic diversity captured and contributing to inaccurate patriline identification. Unlike the simulated data, which consistently had 30 workers per queen and thus captured the full spectrum of patrilines, the smaller sample size in the real data may have missed certain patrilines, particularly those from fathers contributing fewer offspring. This suggests that inadequate sampling in the real data could have impacted the accuracy of patriline estimates. Increasing the sample size would likely improve the ability to detect patrilines and provide a more comprehensive view of the colony's genetic diversity, underscoring the importance of robust sampling in such analyses.

4.6.4 Implementations

Improving sire assignment accuracy and enabling better determination of patrilines can help breeders avoid inbreeding and select for desirable traits more effectively. A primary benefit of this is the significant enhancement in mating control, the increasing accuracy of breeding values, and the ability to implement more effective breeding programmes. These refined methods allow for precise selection of mating pairs based on genetic analysis, optimizing genetic diversity, increasing genetic gain, and improving desirable traits.

Pedigree reconstruction is essential for managing genetic diversity, achieving genetic gain, and understanding the inheritance patterns of specific traits. One of its key functions is verifying high-value individuals, whose desirable genetic traits can significantly impact the success of a breeding programme. This verification ensures that breeders can reliably identify and select the best individuals to pass on their valuable traits, while also enhancing the accuracy of breeding value estimation . In doing so, pedigree reconstruction not only safeguards the genetic quality of breeding populations, ensures breeding line integrity, and accelerates genetic gain (Wang, 2019; Manichaikul et al., 2010). Additionally, pedigree reconstruction supports demographic research by providing insights into the population structure and historical inheritance patterns (Wallberg et al., 2014). Our evaluation of different pedigree reconstruction software, tested with various SNP array sizes, not only guides honeybee breeders and researchers to select the software that offers the most reliable and precise sire assignments but also helps set realistic expectations for their datasets. For instance, with an array of only 16 SNPs, the accuracy of the results may be limited. Breeders should be cautious when interpreting these results and instead opt for a larger SNP array, which provides greater stability and reliability for analysis. Assessing the number of patriline within a population is essential for evaluating the effectiveness of controlled queen mating and ensuring desired genetic contributions. Additionally, determining patriline diversity helps gauge the genetic diversity of the population, which enhances its fitness and adaptability. However,

determining the number of patrilines is a notorious job and not many effective methods have been developed. Our methods for determine patriline numbers will potentially improve breeding programmes through a better understanding of a population's genetic diversity, reproductive dynamics, and mating patterns. Determining the parental origin of haplotype is not only a benefit to honeybee breeding but also has a broader applications. Identifying haplotype parental origins can reveal hereditary patterns for specific traits or disease-causing alleles (Feng et al., 2011). This process also enable computations of Mendelian sampling at a gametic level and provide more accurate determinations of the patriline numbers. Additionally, parental haplotype identification is essential for studying imprinted genes, which are expressed deferentially depending of the parent-of-origin (Howey et al., 2015).

Our haplotype parent-of-origin assignment function and patriline determination function are not specific to honeybees and can be applied to any phased datasets. The functions offer significant advantages over traditional methods by being quick, easy to use, and reliable. They are well-suited for use as confirmatory tests or supplementary analysis when genotype data is collected, allowing research to validate findings quickly without extensive experimentation. Additionally, the ability to calculate Mendelian sampling at the gametic level enhances understanding of inheritance patterns, benefiting both practical breeding efforts and research on trait heritability (Hill and Weir, 2011; Bateson, 1902). However, further testing is needed to assess the limitations and consistency of these methods across different datasets to ensure their reliability before recommending widespread use.

4.6.5 Limitations

Our results revealed some limitations of our study. Firstly, despite SIMplyBee being an advanced holistic simulator, it highlighted the challenges of accurately replicating real data through simulations. Although our simulation framework is robust, discrepancies between simulated and real datasets may arise due to inherent differences in how biological processes are captured in simulations versus actual field data. Simulated data assumes a controlled genetic structure with clear inheritance patterns, but actual colonies experience additional variation that cannot be fully captured in simulations. For instance, real-world data showed a significantly lower rate of sire assignment than simulated data (only 54% of workers were assigned a sire using AlphaAssign), suggesting that the real colonies may have had more fragmented or complex mating patterns than assumed in the simulations. Furthermore, the simulated data did not fully account for factors such as inbreeding, rare alleles, or historical demographic events, which can have a significant impact on genetic diversity. Secondly, our findings are still subject to various types of errors, including sampling, sequencing, biases in the source data, genotyping, and missing genotypes in the real data, which can not be fully captured in the simulated data. Additionally, in both real and simulated datasets our results were exposed to phasing errors. Phasing errors can have a profound impact on downstream analyses, particularly in patriline determination. These errors may cause the incorrect assignment of genetic relationships, leading to distortions in the patriline structure and reducing the accuracy of relatedness calculations. Although we assessed phasing

accuracy mainly through match percentages, it's crucial to recognize that match percentages alone might not capture all biases, particularly in smaller datasets. In such cases, phasing errors may not be immediately evident, but they can significantly affect the accuracy and reliability of genetic conclusions. Thirdly, like a chain reaction, the effectiveness of our methods are influenced by the amount and quality of data provided. Specifically, increases in the number of SNPs and a larger sample of workers from a colony are crucial for capturing the full genetic diversity of patrilines. Variations in SNP density were also noted as a factor that could influence the accuracy of our findings. For pedigree reconstruction sire assignment, we suggest that the minimum SNP size should be greater than 160 SNPs, as the accuracy of sire assignments was impacted at this size, but 100% accuracy was achieved at 800 SNPs, indicating that a higher SNP density provides greater accuracy. The minimum number of workers required to get accurate results is still to be determined. Our work suggests that 30 workers per queen is sufficient for pedigree reconstruction, but this may not be enough for our patriline calculations, where increasing the worker count could help minimize the potential for phasing errors and give more data per patrilines, thus improving the precision of the results.

Nevertheless, in comparison to other commercially bred species, genomic research in honeybees is still in its early days. Given that the whole honeybee genome was only sequenced in 2006 (Consortium, 2006), and the largest SNP chip, containing 100K markers, was developed in 2020 (Jones et al., 2020), future advances in genomic testing will likely address some of these limitations. Larger datasets with higher SNP coverage and more workers will enhance our ability to capture genetic diversity, accurately reconstruct pedigrees, and improve phasing accuracy, ultimately providing more precise parent-of-origin assignments.

4.7 Future work

To maximize the benefits of these methods, formalizing them for broader adoption and standardized application would be advantageous. This entails developing clear protocols, validating their effectiveness, and integrating them into standard practices to ensure consistent and reliable application in various research and practical settings. Future work will include testing these functions with larger datasets, by incorporating additional worker samples, and larger SNP chip densities to identify any limitations. It would also be useful to investigate whether pedigree reconstruction is necessary for phasing, especially in larger datasets, and to explore if these functions can perform accurately with only maternal pedigree information, potentially bypassing the need for full pedigree reconstruction. Additionally, estimating the effective size of drone-congregation areas (DCAs) in regions with low bee populations could provide valuable insights into local breeding dynamics and genetic diversity. The potential introgression of non-native genetics following queen imports also warrants further investigation to understand the impact on local populations. Furthermore, examining mating preferences and their effect on the genetic structure of populations could contribute to a more comprehensive understanding of honeybee breeding patterns.

4.8 Conclusions

This study aimed to test and develop tools for improved management and evaluation of honeybee mating through improvements in pedigree reconstruction, haplotype parent-of-origin assignments, and patriline determination. The development of more accurate tools for tracking genetic inheritance and paternity assignment in honeybees has the potential to greatly enhance mating control, improve genetic gain, and support the maintenance of genetic diversity, which is crucial for colony health and adaptability.

4.8.1 Supplementary Figures

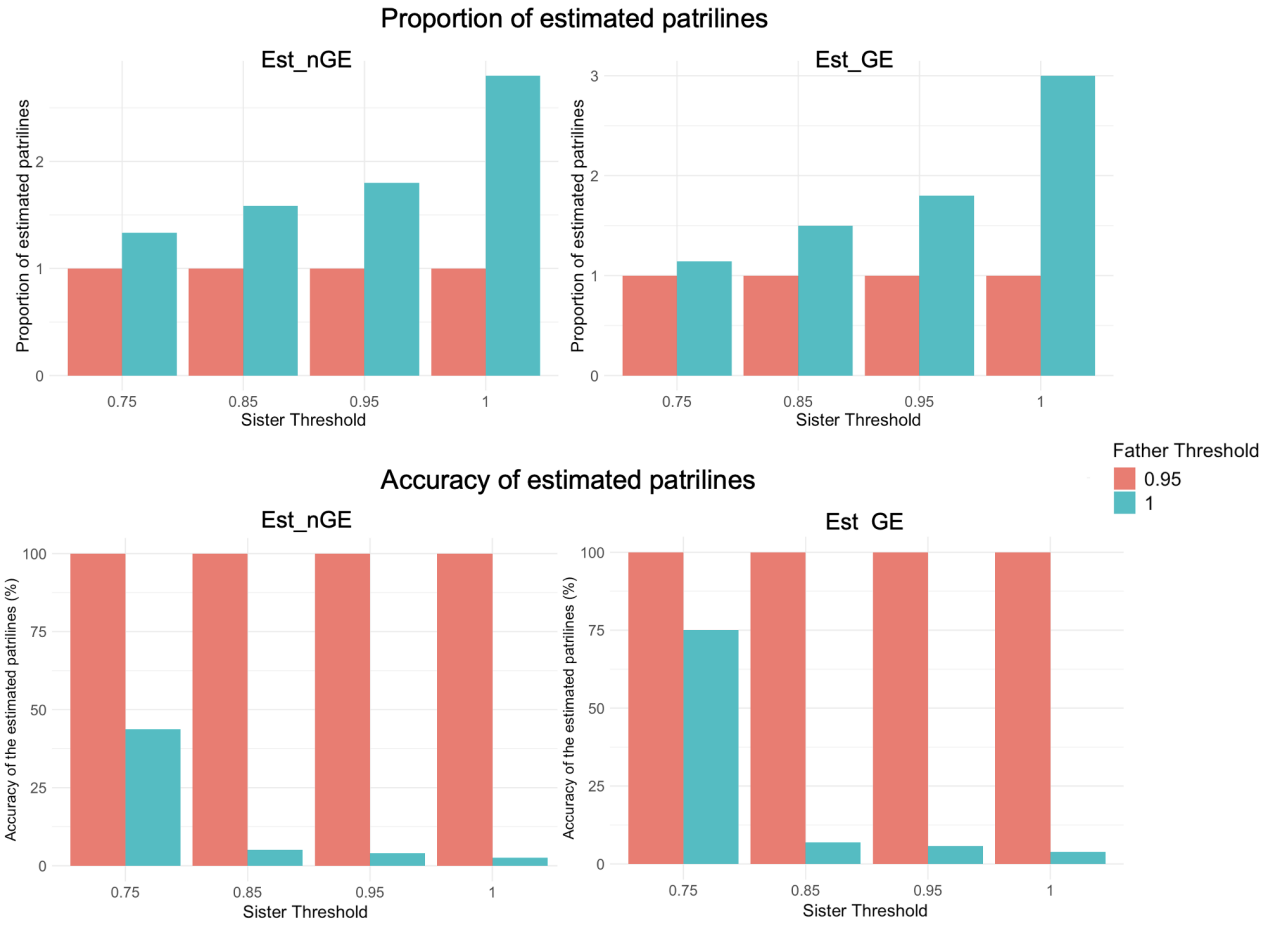


Figure 4.6: The proportion of known patrilines was estimated (top row) and the accuracy of these estimated patrilines (bottom row). We used the estimated haplotypes of the simulated data without genotyping errors (Est_nGE) (left), and the estimated haplotypes of the simulated data with genotyping errors (Est_GE) (right). Similarity father thresholds of 1.0 and 0.95 were used in all calculations, where haplotypes had to be 100% and 95% identical, respectively, to be categorised as a match. Similarity sister thresholds of 1.0, 0.95, 0.85 and 0.75 were used to determine which worker haplotypes were sisters.

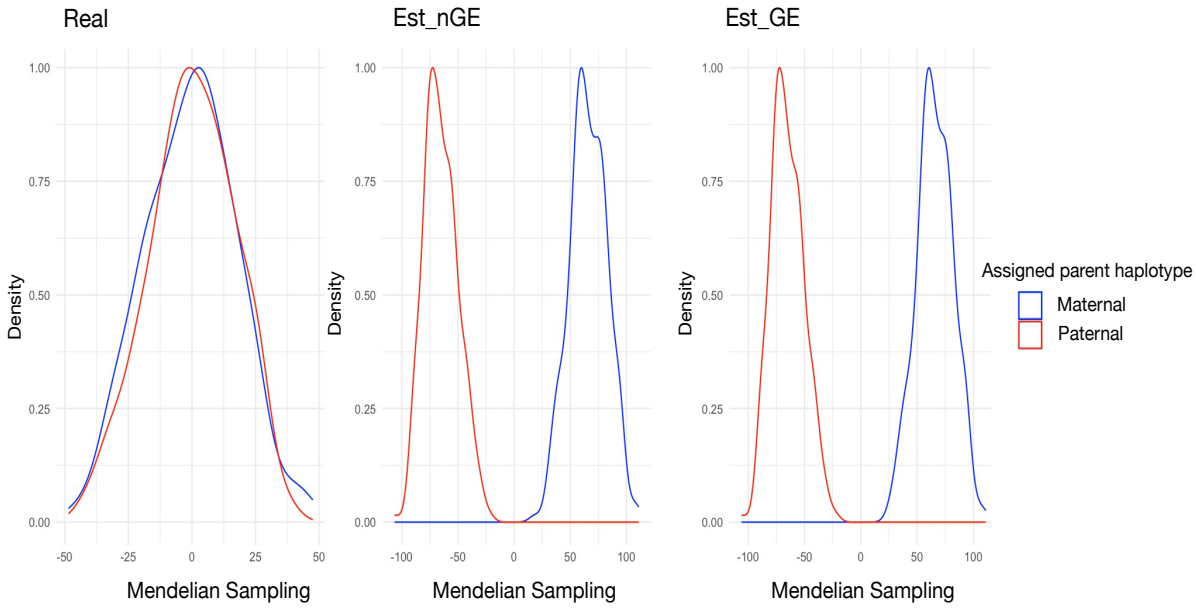


Figure 4.7: Summed Mendelian sampling deviations of the maternal and paternal haplotypes for each offspring, representing the overall Mendelian sampling values across all SNPs. Three haplotype scenarios are shown: True haplotypes taken directly from the simulation ($Real$), phased haplotypes without genotypic errors (Est_nGE) and phased haplotypes with genotyping errors (Est_GE). Est_nGE and Est_GE VCF files were converted to PLINK PED format and run through the the haplotype assignment function.

5 General Discussion

5.1 Thesis motivation and objectives

The growing interest in the economically important honeybee species *Apis mellifera* has sparked an increasing investment in establishing new, or optimising pre-established breeding programmes (Gallai et al., 2009; Klein et al., 2007). A critical aspect of breeding programs is the monitoring and management of genetic diversity, as well as achieving genetic gain through the tracking of inheritance patterns (Schwartz et al., 2007; Hoban et al., 2021). However, the biological complexities of honeybees present significant challenges in this regard. While measurements of genetic relatedness using pedigree and genomic data are commonplace in the breeding programmes of well established species, their utilisation and methodologies in honeybees vary and require further refinement. Simulations are valuable tools that allow one to test numerous hypotheses quickly, facilitating the comparison of simulated and real data, and thereby provide more accurate, reliable predictions to inform breeding decisions.

This thesis aimed to advance our understanding of honeybee genetics and relatedness, and develop new tools to contribute to the sustainable improvement of managed honeybee populations.

Each of my three research chapters dealt with different aspects of this overarching aim:

Chapter 2: The aim of Chapter 2 was to develop a novel, open-source stochastic simulator tailored specifically for honeybee population management and breeding programmes, capable of simulating individual bees with their associated genomes and quantitative traits.

Chapter 3: The aim of Chapter 3 was to demonstrate the principles of genetic relatedness between individual honeybees within a colony, within a population, and comparatively between populations, using SIMplyBee. Our specific objectives were to illustrate the principles of genetic inheritance in honeybees by assessing relatedness within a colony, compare relatedness estimates derived from different data sources and methodologies, and evaluate the effects of hybridization between honeybee populations.

Chapter 4: The primary aim of Chapter 4 was to assess the accuracy of pedigree reconstruction and patriline determination in honeybees using both simulated and real data. The chapter's specific objectives were to assess the effectiveness of different pedigree reconstruction methods for paternal assignments across various

SNP array sizes, to phase trio genotypes and develop a method for identifying the parental origin of estimated haplotypes, and to use this information to create and test a method for estimating the number of patrilines in a honeybee colony.

In the following sections, I will first summarize the findings of each individual study (5.2) and compose a broader discussion (5.2.1) across chapters 2, 3, and 4. Subsequently, I will present challenges, perspectives, recommendations for future work, and practical considerations (5.3), before concluding with the final remarks (5.4).

5.2 Summary of Chapter findings

In **Chapter 2**, we described the development and implementation of the holistic simulator SIMplyBee (Obšteter et al., 2023). SIMplyBee provides a publicly available R package and is supported by its website `www.simplybee.info`, which contains function descriptions, demonstrations, and educational vignettes. SIMplyBee is an extension of the stochastic simulator AlphaSimR (Gaynor et al., 2021), and upgrades the latter with three honeybee-specific classes holding: global simulation parameters (`SimParamBee`), a honeybee colony (`Colony`), and multiple colonies (`MultiColony`). We also addressed aspects of complex honeybee biology, developing functions to simulate the honeybee genome, haplodiploid inheritance, social organisation, complementary sex determination, polyandry, colony events, and quantitative genetics at the individual- and colony-levels (Obšteter et al., 2023). SIMplyBee hence allows users to simulate honeybee populations and population management programmes, generating individuals' genomes and the corresponding individual-level quantitative values as well as colony-level values.

In **Chapter 3**, we demonstrated the principles of genetic inheritance in honeybees by inspecting relatedness coefficients. We simulated the interactions of three honeybee populations over a ten year period, with two breeding scenarios: closed mating and hybrid mating. From this, we evaluated three types of genetic relatedness using different information: i) expected Identity-By-Descent (IBD) from pedigree data, ii) realised IBD from pedigree and genotypic data, and iii) Identity-By-State (IBS) from only genotypic data. Over a 10-year simulation, we observed that expected and realised IBD relatedness coefficients both increased in a closed purebred population and decreased in a hybridising population. All caste members within a colony in the closed population were found to have increased relatedness after the 10-year period due to an increased relatedness between queens and fathers as inbreeding occured, with the greatest increase occurring between workers and the lowest between drones. Our comparison of various methods for calculating relatedness coefficients revealed that using the same founder and reference populations resulted in consistent mean relatedness values, irrespective of whether pedigree or genotypic data was being used. However, substantial variation was observed when different founder populations were assumed for IBS relatedness calculations, indicating that the choice of methodology significantly impacted our relatedness value outcomes. These results highlight three key points: the importance of maintaining consistent methodology in relatedness studies to prevent misinterpretation, the need for careful interpretation of results, and the necessity of

considering the specific methods used when drawing conclusions.

In **Chapter 4** we evaluated pedigree reconstruction and patriline determination in honeybees. We used real data gathered from the mating experiment BeeConSel (BeeConSel.eu, 2020), and simulated data with different SNP array sizes, simulated using SIMplyBee (Obšteter et al., 2023). Our simulation revealed overall differences in the performance of pedigree reconstruction software, with these differences being more pronounced when using smaller SNP arrays. We also observed a trade-off between accuracy and speed, particularly at lower SNP array sizes, where faster software tended to be less precise. In the real dataset, AlphaAssign outperformed other software by assigning sires to 54% of the workers, whereas all workers in the simulated data were assigned a sire, reflecting challenges in applying simulation models to real-world data. This, coupled with its 100% accuracy with the largest simulated SNP array size, made AlphaAssign the chosen tools for providing pedigree information in the phasing process. We confirmed 100% accuracy in assigning haplotype parent-of-origin in simulated true haplotypes, while some deviations were observed with estimated haplotypes of both real and simulated data, as measured via the Mendelian sampling values. These deviations could indicate the introduction of errors during genotype phasing that could be mitigated using datasets with larger sample sizes, higher SNP array densities, or a phased reference dataset. In the real data, such deviations could also indicate data errors, parentage assignment errors, or unusual Mendelian inheritance patterns. Patriline determination in all simulated datasets had 100% accuracy with haplotype similarity father threshold of 0.95, across all sister similarity thresholds. However, accuracy significantly dropped when outside of the optimal father similarity threshold range and was heavily influenced by sister similarity thresholds. In contrast, in the real data, when using a father similarity threshold of 0.95, we predicted a improbably high number of partilines given the small number of workers in the dataset. This disparity between the real and simulated results highlighted the challenges of replicating real data.

5.2.1 Relevance of the thesis findings

In **Chapter 2** we recognized that, while many stochastic simulators have been developed for commercially important mammalian and plant species (Sargolzaei and Schenkel, 2009; Gaynor et al., 2021; Pook et al., 2020), they are not suitable for honeybee populations due to significant differences in biology and social structure (Sargolzaei and Schenkel, 2009; Gaynor et al., 2021; Pook et al., 2020). Additionally, we found that existing honeybee-specific simulators often lack the necessary features for modelling complex breeding programs, focus on population-level dynamics rather than individual genetic data, or are not publicly accessible. Mathematical models specific to honeybees have been available since the pioneering BEEPOP model developed by DeGrandi-Hoffman et al. (1989). This model incorporated colony population dynamics driven by biology and environmental factors. Becher et al. (2013) later reviewed 31 of these honeybee mathematical models, categorising them into 8 'colony models' which addressed within-hive colony dynamics, 11 'Varroa models' which addressed the interactions between honeybees and Varroa mite infestations and 12 'foraging models' which addressed foraging in diverse landscapes. However, this review was restricted to models of single colonies and omitted population or meta-population models. These older honeybee models are effective for addressing specific questions when detailed input data is available, but they often fall short in providing a large-scale or in-depth perspective. Chen et al. (2021) updated this information, providing a review of the models developed since the Becher et al. (2013) paper. The more recent honeybee simulator, BeeSim, developed by Plate et al. (2019), is designed to model quantitative genetics in honeybees. However, it focuses on simulating quantitative values at a colony level and does not account for colony events such as swarming. This is particularly important because swarming is a natural process that significantly impacts the genetic structure of honeybee populations. The failure to simulate swarming events limits the model's ability to accurately capture the dynamics of genetic transfer between colonies, potentially skewing predictions related to genetic diversity and inbreeding rates. In addition, BeeSim is not publicly accessible and open source, limiting its use to the authors and contributors, and restricting its broader application by the scientific community.

In contrast, our SIMplyBee simulator integrates a holistic approach to modelling honeybee populations and breeding programs. It combines detailed genetic and breeding simulations with the flexibility needed to deliver a clearer and more comprehensive picture of large-scale dynamics and long-term outcomes. SIMplyBee can be used to predict the impact of different breeding schemes on genetic diversity, inbreeding rates, effective population sizes, and genetic gain. It can also be used to evaluate the assumptions of current quantitative genetic models. The key feature of SIMplyBee that enables these capabilities, is its ability to simulate individual honeybees and their genomes, which, although may seem counterproductive as honeybees function on a colony level, offers several advantages for understanding genetic interactions and optimizing breeding outcomes. Such advantages include the ability to simulate the quantitative values of individual workers to assess within hive interactions, or the possibility to develop or improve upon methods that compute genetic relationships in honeybees for systems that deviate from the commonly studied breeding design. By focusing on individuals and their genomes, SIMplyBee can investigate the effects of phenotyping schemes, mat-

ing control designs, selection strategies, and conservation practices on genetic diversity, providing valuable insights for both breeding and conservation efforts. SIMplyBee can also model the effects of swarming events, which can drastically alter colony structure and genetic composition, providing a more accurate representation of population dynamics over time. This is a crucial feature for understanding how genetic diversity evolves in response to natural processes. By providing all of these functionalities at an individual honeybee level, it can however be slower than other simulators.

Beekeepers could utilize SIMplyBee outputs to make informed breeding decisions aimed at improving colony health and productivity. For example, by simulating different mating control strategies, beekeepers can optimize queen selection to enhance resistance to diseases such as American Foulbrood or Chalkbrood and to pests such as Varroa mites (Hansen and Brødsgaard, 1999; Aronstein and Murray, 2010; Rosenkranz et al., 2010). Additionally, the ability to predict inbreeding rates allows beekeepers to implement strategic queen replacement programmes, preventing detrimental genetic bottlenecks. SIMplyBee can also assist in assessing the potential impact of introducing new genetic lines into a population, helping beekeepers maintain genetic diversity while avoiding unwanted hybridization effects. Furthermore, by modelling different selection strategies, beekeepers can improve desirable traits such as honey production, overwintering survival, and temperament, ensuring a more sustainable and productive beekeeping operation.

A recent addition to SIMplyBee, which was not yet developed when **Chapter 2** was published, is a geo-spatial component that enables users to define both mating radii for each colony and the ability to simulate environmental effects that vary with location. Since colony location and local interactions play such an important role in mating success and overall colony performance, we believe that this addition will further improve the accuracy and realism of simulated breeding programmes. This feature is now available in the latest version of SIMplyBee, available on CRAN.

Chapter 3 contributed to the thesis' overarching aim by providing further understanding of honeybee genetics and relatedness. We acknowledged that researchers face challenges when interpreting various relatedness coefficients derived from both pedigree and genomic data, particularly when dealing with diverse information sources as is often seen in honeybees. To address this, we effectively used SIMplyBee to present the impact of inbreeding within a closed mating system and a hybridisation with open mating on the relatedness of honeybees. The relevance of these findings extends to an ongoing debate within the beekeeping community regarding the importation of queens. Previous research indicates that increasing genetic diversity in honeybee population can enhance the performance, health, and adaptability in changing environments (Tarpy, 2003; Mattila and Seeley, 2007; Simone-Finstrom et al., 2016). Proponents of importation argue the benefits of better tempered, higher yield producing, and cheaper to purchase bees, outweigh any negative impact caused by the foreign subspecies introduction (Schiff and Sheppard, 1995; Muñoz et al., 2014; Bieńkowska et al., 2021). On the other hand, critics warn that hybridisation between native and non-native species may threaten native subspecies by reducing local adaptations, introducing disease, and potentially causing population declines (Moritz et al., 2005; Le Conte and Navajas, 2008;

Byatt et al., 2016). These concerns underscore the need for further research into the impacts of queen importation.
Furthermore, a comparative examination of different relatedness metrics in **Chapter 3** revealed that relatedness values depend heavily on the source of information used and which population is considered as a founder or reference population. Upon reviewing the literature, it became evident that there was a significant gap in studies that clearly articulated the metric and methods used to assess genetic relatedness. Specifically, we identified a lack of comparative analyses of the methodologies employed for constructing genomic relationship matrices. This gap is critical, as these differences in methodologies can lead to skewed results and misinterpretations when comparing findings across studies. Additionally, honeybees introduce further complexity due to their multiple social levels—individual, colony, and population—which adds an extra layer of difficulty to accurately assessing genetic relatedness. The results of Chapter 3 offer valuable insights into the genetic health and diversity of honeybee populations, emphasizing the necessity for clear and consistent methods when assessing genetic relatedness.

In **Chapter 4**, contributes to the thesis' overarching aim by advancing our understanding of current pedigree reconstruction tools and developing new patriline determination methods, ultimately supporting the sustainable improvement of managed honeybee populations. We demonstrated the effectiveness of pedigree reconstruction software and the development of improved patriline determination methods. Improving sire assignments and accurately determining patrilines can help breeders avoid inbreeding and efficiently select for desirable traits, leading to enhanced breeding programmes. Pedigree reconstruction plays a crucial role in managing genetic diversity, verifying high-value individuals, and understanding trait inheritance in breeding programmes. Correct parentage enhances the accuracy of breeding value estimations by providing precise information on ancestry and genetic relationships, which improves the estimation of an individual's genetic potential and the prediction of future breeding outcomes. By confirming the genetic quality of individuals, breeders can reliably select the best candidates for mating, ensuring breeding line integrity and accelerating genetic gain. Additionally, by reducing errors in parentage assignment, breeders can more reliably enhance traits of interest and improve the overall genetic quality of the population. Furthermore, pedigree reconstruction supports demographic research by providing insights into population structures and historical inheritance patterns, contributing to a better understanding of genetic dynamics. Our assessment of pedigree reconstruction software across various SNP array sizes helps honeybee breeders and researchers choose reliable tools and set realistic data expectations. With smaller SNP arrays, such as 16 markers, accuracy may be limited, requiring cautious interpretation. By assessing each software's strengths and weaknesses, we provide insights to improve pedigree reconstruction practices, supporting better decisions for honeybee health, productivity, and adaptability.
Researchers and breeders evaluate patriline numbers to assess the success of controlled queen mating and ensure desired genetic contributions. Our methods for determining patriline diversity can enhance breeding programs by providing a clearer understanding of genetic diversity, reproductive dynamics, and mating patterns, ultimately improv-

ing population fitness and adaptability. In addition to honeybee breeding, haplotype parental origin determination has broader applications, such as identifying trait inheritance patterns and disease-causing alleles. The newly developed functions for haplotype parental origin assignments and patriline determination are versatile, easy to use, and applicable across phased datasets, offering faster and more reliable results. These functions are ideal for validating results and conducting supplementary analyses, offering a swift way to confirm findings and gain deeper insights into inheritance patterns, thus supporting both breeding practices and research into trait heritability.

5.3 Limitations and Future work

5.3.1 Simulation limitation and future plans for SIMplyBee

A section of this thesis' overarching aim was to develop new tools to contribute to the sustainable improvement of managed honeybee populations. The simulator tool that we developed, SIMplyBee, is a flexible platform that allows the simulation of detailed genetic and breeding scenarios. However, whilst simulators offer many advantages, it is important to note that they are simplifications of complex biological processes that rely on underlying mathematical models and do therefore have their limitations. Simulations are built upon many assumptions, including those made during the generation of founder genomes. SIMplyBee uses the demographic model of Wallberg et al. (2014) to generate a group of founder genomes. While demographic models can incorporate historical events such as population bottlenecks or migrations, they are often based on assumptions made of a whole species or subspecies across a wide geographical range (Coale and Trussell, 1996). In contrast, real honeybees datasets typically have small sample sizes and reflect local adaptions that may not be captured by demographic models. In a real dataset, "founder" genomes refer to the earliest known ancestors in the pedigree, which are related due to their shared ancestry (Hogg et al., 2019). Without detailed information about the local history of a population, it's difficult to predict the impact of events that could result in increased inbreeding rates and the more frequent occurrence of rare alleles (Falconer and Mackay, 1996; Cridland et al., 2017). Additionally, while SIMplyBee is a stochastic simulator designed to replicate the randomness of genetic events, it may not fully capture the true randomness observed in the real data (Obšteter et al., 2023). For instance, the default recombination and mutation rates used in SIMplyBee might not perfectly reflect those seen in a real population. Beye et al. (2006) reported that the genome exhibits recombination rates 10 times higher (23.3 cM/Mb) than those previously calculated for other higher eukaryotes, with an average 5.7 recombination events occurring per chromosome pair during meiosis. If not simulated accurately, these very high recombination rates, given the genome size, could result in significantly different genomes to the ones observed in real data. This underscores the importance of recognizing that, while simulations are valuable, time- and cost-effective tools allowing researchers to test numerous breeding scenarios, accurately replicate real data remains a challenge (Ripley, 2009; Nelson, 2010). These challenges were highlighted in Chapter 4, where we observed deviations in results between the simulated and real datasets, even though the simulation was designed to best match

the real data. Therefore, calibration against real data is essential for simulation reliability, requiring systematic testing and comparison to real data to assess the accuracy of its randomness. Such tests should be performed on SIMplyBee in the future to set the most accurate default parameters for users to receive reliable outputs.

Another technical challenge we encountered during the development of SIMplyBee was in implementing the computation of statistical components of honeybee genetic values, specifically breeding values (BV), dominance deviations (DD), and epistasis deviations (AA). Breeding values indicate an individual's genetic potential trait contribution to its offspring, whereas dominance deviations capture the effects of interactions between alleles as a specific locus, and epistasis deviations represent the interactions between different loci (Falconer, 1985; Viana, 2005). Since SIMplyBee is an extension of AlphaSimR (Gaynor et al., 2021), we can adopt the existing AlphaSimR's functions, `bv()`, `dd()`, and `aa()` to calculate these values. However, these functions assume that population is in Hardy-Weinberg equilibrium, meaning that the allele and genotype frequencies are expected to remain constant over generations in the absence of evolutionary influences (Falconer, 1985; Mayo, 2008). These statistical values (BV, DD, AA) are calculated relative to the specific population being analysed, meaning they depend upon the populations' genetic information at the time of the calculation (Falconer and Mackay, 1996; Viana, 2005). For a honeybee simulation, reporting values relative to each colony would make the results difficult to compare, as each colony's values would be unique and not useful for broader analysis. Alternatively, we would have to create a large "meta" population object of all currently living honeybees to provide a more stable and comparable reference population. However, this approach is complex and requires further development in SIMplyBee through the collaboration of quantitative geneticists and honeybee researchers who have access to real datasets to find the optimal solution.

Additional future work for SIMplyBee includes the development of a new honeybee demographic model that incorporates a wider range of honeybee subspecies and improves parameter estimates (Obšteter et al., 2022). This process will begin with a detailed analysis of the demography of different honeybee subspecies, including evaluating the distribution of the five main lineages and quantifying gene flow between populations. A key step will be integrating genomic data, ecological factors, and historical records to refine subspecies differentiation and better understand population structure. In addition, the model will require the development of more accurate parameter estimates by calibrating it with real-world population data. As part of the validation process, future efforts will focus on running simulations and comparing the model's predictions with empirical data to ensure it effectively captures the genetic diversity, gene flow, and evolutionary dynamics of honeybee populations.

As stated throughout this thesis, SIMplyBee is specifically designed for honeybees and closely mimics their biology. While it has the potential to simulate other closely related species with similar mating behaviours, such as bumblebees, I recommend users interested in more common livestock species consider AlphaSimR, of which SIMplyBee is an extension, as it supports a broader range of species. We also aim to improve the computation efficiency of SIMplyBee by increasing its speed and reducing its memory

usage. To generate individual honeybee genomes, SIMplyBee currently uses MaCS (Chen et al., 2009), however the package may in the future incorporate novel simulators such as msprime (Kelleher et al., 2016; Baumdicker et al., 2022) and SLIM (Haller and Messer, 2019), which may introduce more flexibility and power. Both of these simulators are used by the public stdpopsim library, a repository for population genetic simulation models, which will improve as honeybee genomic information becomes more available, thus improving the simulation accuracy (Adrion et al., 2020; Lauterbur et al., 2022). We have identified certain functions, such as our colony splitting function `split()`, that were deemed too slow. There are several optimizations to speed up these lagging functions that will allow SIMplyBee to operate more efficiently and effectively. These optimizations include offloading certain functions to C++, an alternative programming language, to significantly reduce computation times. For instance, we are utilizing some of the AlphaSimR's C++ functionality and have developed one C++ function of our own. Additionally, we have simplified certain loops to reduce unnecessary iterations and improve performance. In the future, SIMplyBee will continue optimizing by moving more function to C++ and exploring parallelization techniques, further enhancing the simulator's speed and overall efficiency. Additionally, simulating large colony sizes, whole-genomes or a full-scale breeding programmes can be highly computationally demanding.

These strains on computational resources were the reason why we down-scaled our simulation in Chapter 3. Our simulation described in Chapter 3 only used 1000 segregating sites per chromosomes, which likely underestimated the relatedness at the whole-genome level (Makgahlela et al., 2013; Eynard et al., 2015). In real colonies, there can be up to 70,000 workers and several thousand drones present in a colony at peak time (Bodenheimer, 1937; Page and Peng, 2001). However, in this experiment, we simulated only 1000 workers and 50 drones per colony. While these numbers are a gross under-representation of an actual colony, they are relatively large compared to the sample sizes found in typical honeybee datasets. This must be taken into account when interpreting the results of this simulation, as they might not fully capture the genetic diversity seen within a full-size colony. Moreover, the simulation was only conducted over a 10-year period. Considering the generational interval of 1-2 years, this time frame is not too short to see results. However, our 10-year duration was primarily for demonstrative purposes to show the trends of inbreeding and hybridisation. For more accurate breeding simulations, future work should span over much longer periods of time to better understand the trends and determine the most effective breeding strategies. The import percentage used in the Chapter 3 simulation was set unrealistically high at 50% and imports were sourced from a single foreign population. This exaggeration was intended to highlight the impact of hybridisation on relatedness within the colonies. In real-world scenarios, imported bees would typically come from various source populations. A more realistic import percentage would be $\sim 10\%$ (BeeBase, 2022; gov.uk, 2021), though this number would vary depending on the country of import, seasonal factors, and hive location. With more time and resources for increased computational power, future work with SIMplyBee would aim to simulate accurate breeding programmes, taking into account realistic colony sizes, importation rates and the duration of breeding programmes. This would allow for the testing of a variety of hypotheses and provide honeybee breeders and commercial beekeepers with

reliable, precise results to inform their breeding decisions.

In Chapter 4, the real dataset comprised of only 247 individuals with 1722 SNPs after quality controls, and simulated datasets of a similar size were used to evaluate the results of the real data and test the limitations of the numerous processes they underwent. The real data was also subject to potential errors such as sampling, sequencing, and genotyping errors, which we aimed to replicate in the simulation by introducing similar errors into one dataset. However, mimicking these errors identically with simulation is still a challenge. Additionally, in both real and simulated datasets our results were exposed to phasing errors. Our analysis of phased datasets demonstrated how phasing errors affected gametic Mendelian sampling values. The effectiveness of our methods was closely related to the quantity and quality of data, with larger SNP arrays and more extensive workers samples being essential to capture the full genetic diversity of a colony. Despite advances in genomic research, honeybee studies are still relatively new compared to other species. As genomic acquisition methods improve and datasets become more comprehensive, our methods will be better able to provide accurate and reliable results. Further research is needed to determine the minimum number of SNPs required for optimal performance of our parent-of-origin assignment and patriline determination functions, achieved through systematic testing of the methods. Another important consideration for the future of honeybee breeding is the role of artificial insemination. This technique allows complete control over sire selection, thereby significantly improving the accuracy of genetic predictions in breeding programs. While natural mating in open environments introduces variability and limits precision, artificial insemination enables beekeepers to selectively breed for desired traits with higher certainty. This is particularly valuable for conservation programs of native bees in colder or remote regions, where controlled breeding is essential to prevent genetic dilution or maintain local adaptations. The integration of artificial insemination within breeding simulations in SIMplyBee could enhance the predictive power of genetic models, offering more refined strategies for managing inbreeding, hybridization, and genetic diversity across diverse environmental conditions. To fully leverage the methods described in Chapter 4, it would be beneficial to formalize them for wider use and standardize their application. This involves creating clear guidelines, validating their effectiveness, and incorporating them into routine practices to ensure their consistent and reliable use across different research and practical environments. Future efforts should focus on testing these methods with larger datasets by including more worker samples and increasing SNP chip densities to identify any potential limitations. To achieve this, collaborations with honeybee breeders, researchers, and commercial beekeepers with access to large-scale datasets from breeding programs will be crucial. Collaborative initiatives could involve sharing genomic data from different regions or colonies to increase genetic diversity, including more varied environmental conditions to assess the robustness of these methods in diverse settings, and leveraging new genomic technologies like next-generation sequencing to further enhance dataset quality. Additionally, it would be valuable to examine whether pedigree reconstruction is essential for phasing in larger datasets, and to assess if these methods can accurately perform with only maternal pedigree information, possibly eliminating the need for complete pedigree reconstruction.

5.4 Conclusion

In this thesis, we first we developed SIMplyBee, a novel honeybee-specific simulator. This tools offers a comprehensive approach to simulating honeybee populations and breeding programs. By enabling detailed genetic and breeding simulations, SIMplyBee helps evaluate long-term outcomes like genetic diversity, inbreeding, and genetic gain. By focusing on individual-level and colony-level bees and their corresponding genomes, it offers insights into genetic interactions, breeding strategies, and colony dynamics that go beyond colony-level models. Although this individual-level detail can slow the simulation, it delivers valuable understanding of breeding practices and conservation efforts.

Next, we demonstrated the principles of genetic relatedness at multiple levels: within individual bees in a colony, across a honeybee population, and between different populations. Using SIMplyBee, we explored how inbreeding and hybridization over time affect relatedness, contributing to the debate on queen importation and its impact on genetic diversity. We highlighted the difficulties in interpreting relatedness from both pedigree and genomic data. This emphasizes the need for consistent methods to evaluate genetic relatedness, particularly due to the challenges posed by honeybees' multi-tiered social structure.

Finally, we used SIMplyBee to advance the understanding of pedigree reconstruction tools and developed new methods for patriline determination, thus contributing to the sustainable improvement of managed honeybee populations. By evaluating various pedigree reconstruction software, we assisted researchers and breeders in selecting reliable tools and interpreting data accurately. We also developed versatile methods for haplotype parental origin assignment and patriline determination that offer broad applications in breeding and trait research. Our work emphasized that accurate sire assignments and patriline identification are crucial for avoiding inbreeding, selecting desirable traits, and enhancing breeding programs by improving the precision of genetic value estimations and reducing errors.

Beyond its utility for breeding programmes, SIMplyBee has the potential to contribute significantly to global pollinator conservation efforts and ecosystem service management. By simulating the genetic consequences of different management strategies, researchers can assess the long-term viability of conservation breeding programmes aimed at preserving endangered honeybee populations. The tool can also be used to evaluate how selective breeding impacts traits such as disease resistance, foraging efficiency, and climate adaptability, providing data-driven guidance for sustainable beekeeping worldwide. Furthermore, SIMplyBee could assist in modelling genetic exchange between managed and wild honeybee populations, helping to mitigate risks associated with hybridization and genetic bottlenecks. By integrating geo-spatial components, the simulator could also be adapted to study how landscape changes influence honeybee mating success, colony resilience, and pollination networks, thereby supporting efforts to maintain biodiversity and ensure food security on a global scale.

Overall, this thesis has enhanced our knowledge of honeybee genetics and relatedness

while creating innovative tools designed to support the sustainable advancement of managed honeybee populations.

References

Jeffrey R Adrion, Christopher B Cole, Noah Dukler, Jared G Galloway, Ariella L Gladstein, Graham Gower, Christopher C Kyriazis, Aaron P Ragsdale, Georgia Tsambos, Franz Baumdicker, Jedidiah Carlson, Reed A Cartwright, Arun Durvasula, Ilan Gronau, Bernard Y Kim, Patrick McKenzie, Philipp W Messer, Ekaterina Noskova, Diego Ortega-Del Vecchyo, Fernando Racimo, Travis J Struck, Simon Gravel, Ryan N Gutenkunst, Kirk E Lohmueller, Peter L Ralph, Daniel R Schrider, Adam Siepel, Jerome Kelleher, and Andrew D Kern. A community-maintained standard library of population genetic models. *eLife*, 9:e54967, June 2020. ISSN 2050-084X. doi: 10.7554/eLife.54967. URL `https://doi.org/10.7554/eLife.54967`.

Marcelo A. Aizen and Lawrence D. Harder. The Global Stock of Domesticated Honey Bees Is Growing Slower Than Agricultural Demand for Pollination. *Current Biology*, 19(11):915–918, June 2009. ISSN 0960-9822. doi: 10.1016/j.cub.2009.03.071. URL `https://www.sciencedirect.com/science/article/pii/S0960982209009828`.

Mohamed Alburaki, Sibyle Moulin, Hélène Legout, Ali Alburaki, and Lionel Garnery. Mitochondrial structure of Eastern honeybee populations from Syria, Lebanon and Iraq. *Apidologie*, 42(5):628–641, September 2011. ISSN 1297-9678. doi: 10.1007/s13592-011-0062-4. URL `https://doi.org/10.1007/s13592-011-0062-4`.

M. Delia Allen. The Effect of a Plentiful Supply of Drone Comb on Colonies of Honeybees. *Journal of Apicultural Research*, 4(2):109–119, January 1965a. ISSN 0021-8839. doi: 10.1080/00218839.1965.11100114. URL `https://doi.org/10.1080/00218839.1965.11100114`.

M. Delia Allen. The Production of Queen Cups and Queen Cells in Relation to the General Development of Honeybee Colonies, and its Connection with Swarming and Supersedure. *Journal of Apicultural Research*, 4(3):121–141, January 1965b. ISSN 0021-8839. doi: 10.1080/00218839.1965.11100115. URL `https://doi.org/10.1080/00218839.1965.11100115`.

MH Allsopp and HR Hepburn. Swarming, supersedure and the mating system of a natural population of honey bees (Apis mellifera capensis). *Journal of Apicultural Research*, 36(1):41–48, 1997. Publisher: Taylor & Francis.

Eric C Anderson and John Carlos Garza. The Power of Single-Nucleotide Polymorphisms for Large-Scale Parentage Inference. *Genetics*, 172(4):2567–2582, April 2006.

ISSN 1943-2631. doi: 10.1534/genetics.105.048074. URL https://doi.org/10.1534/genetics.105.048074.

RH Anderson. The laying worker in the cape honeybee, apis mellifera capensis. *Journal of Apicultural Research*, 2(2):85–92, 1963.

Sreten Andonov, Cecilia Costa, Aleksandar Uzunov, Patrizia Bergomi, Daniela Lourenco, and Ignacy Misztal. Modeling honey yield, defensive and swarming behaviors of Italian honey bees (Apis mellifera ligustica) using linear-threshold approaches. *BMC Genetics*, 20(1):78, October 2019. ISSN 1471-2156. doi: 10.1186/s12863-019-0776-2. URL https://doi.org/10.1186/s12863-019-0776-2.

Michael F. Antolin. A genetic perspective on mating systems and sex ratios of parasitoid wasps. *Population Ecology*, 41(1):29–37, 1999.

M. C. Arias and W. S. Sheppard. Molecular Phylogenetics of Honey Bee Subspecies (*Apis* melliferaL.) Inferred from Mitochondrial DNA Sequence. *Molecular Phylogenetics and Evolution*, 5(3):557–566, June 1996. ISSN 1055-7903. doi: 10.1006/mpev.1996.0050. URL https://www.sciencedirect.com/science/article/pii/S1055790396900500.

Maria C Arias and Walter S Sheppard. Phylogenetic relationships of honey bees (Hymenoptera: Apinae: Apini) inferred from nuclear and mitochondrial DNA sequence data. *Molecular phylogenetics and evolution*, 37(1):25–35, 2005. Publisher: Elsevier.

Katherine A Aronstein and K Daniel Murray. Chalkbrood disease in honey bees. *Journal of invertebrate pathology*, 103:S20–S29, 2010.

Gary N Atlin and Bethany Fallon Econopouly. Simple deterministic modeling can guide the design of breeding pipelines for self-pollinated crops. *Crop Science*, 62(2):661–678, 2022.

Arián Avalos, Miaoquan Fang, Hailin Pan, Aixa Ramirez Lluch, Alexander E. Lipka, Sihai Dave Zhao, Tugrul Giray, Gene E. Robinson, Guojie Zhang, and Matthew E. Hudson. Genomic regions influencing aggressive behavior in honey bees are defined by colony allele frequencies. *Proceedings of the National Academy of Sciences*, 117(29):17135–17141, July 2020. doi: 10.1073/pnas.1922927117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1922927117.

François Balloux, Harald Brunner, Nicolas Lugon-Moulin, Jacques Hausser, and Jéarôme Goudet. Microsatellites can be misleading: an empirical and simulation study. *Evolution*, 54(4):1414–1422, 2000.

B. Basso, T. Kistler, T. Gerez, and F. Phocas. Genetic analysis of royal jelly production and behaviour traits of honeybees. Rotterdam, The Netherlands, July 2022.

William Bateson. *Mendel's Principles of Heredity*. Courier Corporation, March 1902. ISBN 978-0-486-14837-3.

Franz Baumdicker, Gertjan Bisschop, Daniel Goldstein, Graham Gower, Aaron P Ragsdale, Georgia Tsambos, Sha Zhu, Bjarki Eldon, E Castedo Ellerman,

Jared G Galloway, Ariella L Gladstein, Gregor Gorjanc, Bing Guo, Ben Jeffery, Warren W Kretzschumar, Konrad Lohse, Michael Matschiner, Dominic Nelson, Nathaniel S Pope, Consuelo D Quinto-Cortés, Murillo F Rodrigues, Kumar Saunack, Thibaut Sellinger, Kevin Thornton, Hugo van Kemenade, Anthony W Wohns, Yan Wong, Simon Gravel, Andrew D Kern, Jere Koskela, Peter L Ralph, and Jerome Kelleher. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics*, 220(3):iyab229, March 2022. ISSN 1943-2631. doi: 10.1093/genetics/iyab229. URL https://academic.oup.com/genetics/article/doi/10.1093/genetics/iyab229/6460344.

Matthias A Becher, Juliet L Osborne, Pernille Thorbek, Peter J Kennedy, and Volker Grimm. Towards a systems approach for understanding honeybee decline: a stocktaking and synthesis of existing models. *Journal of Applied Ecology*, 50(4):868–880, 2013. Publisher: Wiley Online Library.

Matthias A. Becher, Volker Grimm, Pernille Thorbek, Juliane Horn, Peter J. Kennedy, and Juliet L. Osborne. BEEHAVE: a systems model of honeybee colony dynamics and foraging to explore multifactorial causes of colony failure. *Journal of Applied Ecology*, 51(2):470–482, 2014. ISSN 1365-2664. doi: 10.1111/1365-2664.12222. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1365-2664.12222.

BeeBase. Hive count » APHA - National Bee Unit - BeeBase, 2022. URL https://www.nationalbeeunit.com/bees-and-the-law/hive-count/.

BeeBreed.eu. Beebreed.eu. URL https://www2.hu-berlin.de/beebreed/ZWS/.

BeeConSel.eu. BeeConSel.eu, 2020. URL https://beeconsel.eu. Publication Title: BeeConSel.

Rudi C Berkelhamer. Intraspecific genetic variation and haplodiploidy, eusociality, and polygyny in the Hymenoptera. *Evolution*, pages 540–545, 1983. Publisher: JSTOR.

Richard Bernstein. Realisation of genomic selection in the honey bee, 2022.

Richard Bernstein, Manuel Du, Zhipei G Du, Anja S Strauss, Andreas Hoppe, and Kaspar Bienefeld. First large-scale genomic prediction in the honey bee. *Heredity*, 130(5):320–328, 2023. Publisher: Springer International Publishing Cham.

Martin Beye, Martin Hasselmann, M.Kim Fondrk, Robert E Page, and Stig W Omholt. The Gene csd Is the Primary Signal for Sexual Development in the Honeybee and Encodes an SR-Type Protein. *Cell*, 114(4):419–429, August 2003. ISSN 00928674. doi: 10.1016/S0092-8674(03)00606-8. URL https://linkinghub.elsevier.com/retrieve/pii/S0092867403006068.

Martin Beye, Irene Gattermeier, Martin Hasselmann, Tanja Gempe, Morten Schioett, John F Baines, David Schlipalius, Florence Mougel, Christine Emore, and Olav Rueppell. Exceptionally high levels of recombination across the honey bee genome. *Genome research*, 16(11):1339–1344, 2006. ISSN 1088-9051.

K. Bienefeld and F. Pirchner. Heritabilities for several colony traits in the honeybee (Apis mellifera carnica). *Apidologie*, 21(3):175–183, 1990. ISSN 0044-8435.

doi: 10.1051/apido:19900302. URL http://www.apidologie.org/10.1051/apido:19900302.

Kaspar Bienefeld, Klaus Ehrhardt, and Friedrich Reinhardt. Genetic evaluation in the honey bee considering queen and worker effects – A BLUP-Animal Model approach. *Apidologie*, 38(1):77–85, January 2007. ISSN 0044-8435, 1297-9678. doi: 10.1051/apido:2006050. URL http://link.springer.com/10.1051/apido:2006050.

Małgorzata Bieńkowska, Aleksandra Splitt, Paweł Węgrzynowicz, and Robert Maciorowski. The Buzz Changes within Time: Native Apis mellifera mellifera Honeybee Subspecies Less and Less Popular among Polish Beekeepers Since 1980. *Agriculture*, 11(7):652, July 2021. ISSN 2077-0472. doi: 10.3390/agriculture11070652. URL https://www.mdpi.com/2077-0472/11/7/652.

Lelania Bilodeau and Christine Elsik. A scientific note defining allelic nomenclature standards for the highly diverse complementary sex-determiner (csd) locus in honey bees. *Apidologie*, 52(4):749–754, 2021.

F. S. Bodenheimer. Studies in Animal Populations. II. Seasonal Population-Trends of the Honey-Bee. *The Quarterly Review of Biology*, 12(4):406–425, December 1937. ISSN 0033-5770. doi: 10.1086/394540. URL https://www.journals.uchicago.edu/doi/abs/10.1086/394540.

Luca Bonomi, Yingxiang Huang, and Lucila Ohno-Machado. Privacy challenges and research opportunities for genomic data sharing. *Nature genetics*, 52(7):646–654, 2020. Publisher: Nature Publishing Group US New York.

Gerald Borgia. Evolution of haplodiploidy: models for inbred and outbred systems. *Theoretical population biology*, 17(2):103–128, 1980. Publisher: Elsevier.

E. W. Brascamp, R. F. Veerkamp, and P. Bijma. Estimation of genetic parameters and breeding values in honey bees. In *Proceedings of the 10th World Congress on Genetics Applied to Livestock Production (10th WCGALP)*, pages 17–22. 10th World Congress of Genetics Applied to Livestock Production, 2014. URL https://www.researchgate.net/profile/Pim-Brascamp/publication/268109842_Estimation_of_genetic_parameters_and_breeding_values_in_honey_bees/links/56975b3508aec79ee32a6387/Estimation-of-genetic-parameters-and-breeding-values-in-honey-bees.pdf.

Evert W Brascamp and Piter Bijma. Methods to estimate breeding values in honey bees. *Genetics Selection Evolution*, 46(1):53, December 2014. ISSN 1297-9686. doi: 10.1186/s12711-014-0053-9. URL http://gsejournal.biomedcentral.com/articles/10.1186/s12711-014-0053-9.

Pim Brascamp, Tieme Wanders, Yvonne Wientjes, and P Bijma. *Prospects for genomic selection in honey-bee breeding*. May 2018.

Tom D Breeze, Robin Dean, and Simon G Potts. The costs of beekeeping for pollination services in the UK – an explorative study. *Journal of Apicultural Research*, 56(3):310–317, May 2017. ISSN 0021-8839, 2078-6913. doi: 10.

1080/00218839.2017.1304518. URL https://www.tandfonline.com/doi/full/10.1080/00218839.2017.1304518.

Mark J. F. Brown and Robert J. Paxton. The conservation of bees: a global perspective. *Apidologie*, 40(3):410–416, May 2009. ISSN 0044-8435, 1297-9678. doi: 10.1051/apido/2009019. URL http://dx.doi.org/10.1051/apido/2009019.

Brian L Browning and Sharon R Browning. Genotype imputation with millions of reference samples. *The American Journal of Human Genetics*, 98(1):116–126, 2016. Publisher: Elsevier.

Brian L Browning, Xiaowen Tian, Ying Zhou, and Sharon R Browning. Fast two-stage phasing of large-scale sequence data. *The American Journal of Human Genetics*, 108(10):1880–1890, 2021. Publisher: Elsevier.

Sharon R Browning and Brian L Browning. Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. *The American Journal of Human Genetics*, 81(5):1084–1097, 2007. Publisher: Elsevier.

Dorothea Brückner. The Influence of Genetic Variability on Wing Symmetry in Honeybees (Apis mellifera). *Evolution*, 30(1):100–108, 1976. ISSN 0014-3820. doi: 10.2307/2407677. URL https://www.jstor.org/stable/2407677.

Buckfast.dk. Breeding – Buckfast Denmark. URL https://buckfast.dk/breeding/.

J. J. Bull. Coevolution of Haplo-Diploidy and Sex Determination in the Hymenoptera. *Evolution*, 35(3):568–580, 1981. ISSN 0014-3820. doi: 10.2307/2408203. URL https://www.jstor.org/stable/2408203.

M. G. Bulmer. The Effect of Selection on Genetic Variability. *The American Naturalist*, 105(943):201–211, May 1971. ISSN 0003-0147. doi: 10.1086/282718. URL https://www.journals.uchicago.edu/doi/abs/10.1086/282718.

M. A. Byatt, N. C. Chapman, T. Latty, and B. P. Oldroyd. The genetic consequences of the anthropogenic movement of social bees. *Insectes Sociaux*, 63(1):15–24, February 2016. ISSN 1420-9098. doi: 10.1007/s00040-015-0441-3. URL https://doi.org/10.1007/s00040-015-0441-3.

James Brian Byrd, Anna C Greene, Deepashree Venkatesh Prasad, Xiaoqian Jiang, and Casey S Greene. Responsible, practical genomic data sharing that accelerates research. *Nature Reviews Genetics*, 21(10):615–629, 2020. Publisher: Nature Publishing Group UK London.

M. P. L. Calus. Genomic breeding value prediction: methods and procedures. *animal*, 4(2):157–164, February 2010. ISSN 1751-732X, 1751-7311. doi: 10.1017/S1751731109991352. URL https://www.cambridge.org/core/journals/animal/article/abs/genomic-breeding-value-prediction-methods-and-procedures/ED7841BD91A97F1E7FC48F6598EB0710.

Emma L Carroll, Mike W Bruford, J Andrew DeWoody, Gregoire Leroy, Alan Strand, Lisette Waits, and Jinliang Wang. Genetic and genomic monitoring with minimally

invasive sampling methods. *Evolutionary applications*, 11(7):1094–1119, 2018. Publisher: Wiley Online Library.

Mark J. Carroll, Nicholas J. Brown, Zachary Ruetz, Vincent A. Ricigliano, and Kirk E. Anderson. Honey bee retinue workers respond similarly to queens despite seasonal differences in Queen Mandibular Pheromone (QMP) signaling. *PLOS ONE*, 18(9):e0291710, September 2023. ISSN 1932-6203. doi: 10.1371/journal.pone.0291710. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0291710.

J Jesus Ceron-Rojas, José Crossa, Vivi N Arief, Kaye Basford, Jessica Rutkoski, Diego Jarquín, Gregorio Alvarado, Yoseph Beyene, Kassa Semagn, and Ian DeLacy. A Genomic Selection Index Applied to Simulated and Real Data. *G3 Genes|Genomes|Genetics*, 5(10):2155–2164, October 2015. ISSN 2160-1836. doi: 10.1534/g3.115.019869. URL https://doi.org/10.1534/g3.115.019869. _eprint: https://academic.oup.com/g3journal/article-pdf/5/10/2155/40571450/g3journal2155.pdf.

Nadine C Chapman, Madeleine Beekman, Michael H Allsopp, Thomas E Rinderer, Julianne Lim, Peter R Oxley, and Benjamin P Oldroyd. Inheritance of thelytoky in the honey bee Apis mellifera capensis. *Heredity*, 114(6):584–592, 2015. Publisher: Nature Publishing Group.

Brian Charlesworth and Deborah Charlesworth. The genetic basis of inbreeding depression. *Genetics Research*, 74(3):329–340, December 1999. ISSN 1469-5073, 0016-6723. doi: 10.1017/S0016672399004152. URL https://www.cambridge.org/core/journals/genetics-research/article/genetic-basis-of-inbreeding-depression/98A68D3BB0DD50888B4018445BADBC8C.

Gary K. Chen, Paul Marjoram, and Jeffrey D. Wall. Fast and flexible simulation of DNA sequence data. *Genome Research*, 19(1):136–142, January 2009. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.083634.108. URL https://genome.cshlp.org/content/19/1/136.

Jun Chen, Gloria DeGrandi-Hoffman, Vardayani Ratti, and Yun Kang. Review on mathematical modeling of honeybee population dynamics. *Mathematical Biosciences and Engineering*, 18(6), 2021.

Sammy Kiprotich Cheruiyot, H Michael G Lattorff, Ruth Kahuthia-Gathu, Jenard Patrick Mbugi, and Elliud Muli. Varroa-specific hygienic behavior of apis mellifera scutellata in kenya. *Apidologie*, 49:439–449, 2018.

Soochin Cho, Zachary Y. Huang, Daniel R. Green, Deborah R. Smith, and Jianzhi Zhang. Evolution of the complementary sex-determination gene of honey bees: Balancing selection and trans-species polymorphisms. *Genome Research*, 16(11):1366–1375, November 2006. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.4695306. URL https://genome.cshlp.org/content/16/11/1366.

Mark R. Christie, Patrick G. Meirmans, Oscar E. Gaggiotti, Robert J. Toonen,

and Crow White. Disentangling the relative merits and disadvantages of parentage analysis and assignment tests for inferring population connectivity. *ICES Journal of Marine Science*, 74(6):1749–1762, July 2017. ISSN 1054-3139. doi: 10.1093/icesjms/fsx044. URL https://doi.org/10.1093/icesjms/fsx044.

Ansley Coale and James Trussell. The development and use of demographic models. *Population studies*, 50(3):469–484, 1996. Publisher: Taylor & Francis.

Jean-Jacques Colleau. An indirect approach to the extensive calculation of relationship coefficients. *Genetics Selection Evolution*, 34(4):409, July 2002. ISSN 1297-9686. doi: 10.1186/1297-9686-34-4-409. URL https://doi.org/10.1186/1297-9686-34-4-409.

Antonio Comparini and Adriano Biasiolo. Genetic discrimination of Italian bee, *Apis* mellifera ligustica versus Carniolan bee, *Apis* mellifera carnica by allozyme variability analysis. *Biochemical Systematics and Ecology*, 19(3):189–194, June 1991. ISSN 0305-1978. doi: 10.1016/0305-1978(91)90002-H. URL https://www.sciencedirect.com/science/article/pii/030519789190002H.

Honeybee Genome Sequencing Consortium. Insights into social insects from the genome of the honeybee Apis mellifera. *Nature*, 443(7114):931–949, October 2006. ISSN 0028-0836. doi: 10.1038/nature05260. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2048586/.

James M Cook and Ross H Crozier. Sex determination and population biology in the Hymenoptera. *Trends in Ecology & Evolution*, 10(7):281–286, 1995. Publisher: Elsevier Current Trends.

Matteo Cortellari, Alessio Negro, Arianna Bionda, Silverio Grande, Alberto Cesarani, Antonello Carta, Nicola Macciotta, Stefano Biffani, and Paola Crepaldi. Using pedigree and genomic data toward Better Management of Inbreeding in italian dairy Sheep and Goat Breeds. *Animals*, 12(20):2828, 2022. Publisher: MDPI.

Charles William Cotterman. *A calculus for statistico-genetics*. PhD Thesis, The Ohio State University, 1940. URL https://etd.ohiolink.edu/acprod/odb_etd/ws/send_file/send?accession=osu1298297334&disposition=inline.

Julie M. Cridland, Neil D. Tsutsui, and Santiago R. Ramírez. The Complex Demographic History and Evolutionary Origin of the Western Honey Bee, Apis Mellifera. *Genome Biology and Evolution*, 9(2):457–472, February 2017. ISSN 1759-6653. doi: 10.1093/gbe/evx009. URL https://doi.org/10.1093/gbe/evx009.

James F. Crow. On epistasis: why it is unimportant in polygenic directional selection. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1544):1241–1244, April 2010. doi: 10.1098/rstb.2009.0275. URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2009.0275.

James F. Crow and William C. Roberts. Inbreeding and homozygosis in bees. *Genetics*, 35(6):612, 1950.

R. H. Crozier. On the potential for genetic variability in haplo-diploidy. *Genetica*, 41 (1):551–556, 1970.

R. H. Crozier and R. E. Page. On being the right size: male contributions and multiple mating in social Hymenoptera. *Behavioral Ecology and Sociobiology*, 18(2):105–115, December 1985. ISSN 1432-0762. doi: 10.1007/BF00299039. URL https://doi.org/10.1007/BF00299039.

Ross H. Crozier and Else J. Fjerdingstad. Polyandry in social Hymenoptera — disunity in diversity? *Annales Zoologici Fennici*, 38(3/4):267–285, 2001. ISSN 0003-455X. URL https://www.jstor.org/stable/23735845.

Marina Souza Cunha, Danon Clemes Cardoso, Maykon Passos Cristiano, Lucio Antônio de Oliveira Campos, and Denilce Meneses Lopes. The Bee Chromosome database (Hymenoptera: Apidae). *Apidologie*, 52(2):493–502, April 2021. ISSN 1297-9678. doi: 10.1007/s13592-020-00838-2. URL https://doi.org/10.1007/s13592-020-00838-2.

Raffaele Dall'Olio, Alberto Marino, Marco Lodesani, and Robin FA Moritz. Genetic characterization of italian honeybees, apis mellifera ligustica, based on microsatellite dna polymorphisms. *Apidologie*, 38(2):207–217, 2007.

Pilar De la Rúa, Rodolfo Jaffé, Raffaele Dall'Olio, Irene Muñoz, and José Serrano. Biodiversity, conservation and current threats to European honeybees. *Apidologie*, 40(3):263–284, May 2009. ISSN 0044-8435, 1297-9678. doi: 10.1051/apido/2009027. URL http://link.springer.com/10.1051/apido/2009027.

Pilar De la Rúa, Rodolfo Jaffé, Irene Muñoz, José Serrano, Robin F. A. Moritz, and F. Bernhard Kraus. Conserving genetic diversity in the honeybee: Comments on Harpur et al. (2012). *Molecular Ecology*, 22(12):3208–3210, 2013. ISSN 1365-294X. doi: 10.1111/mec.12333. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/mec.12333.

G. DeGrandi-Hoffman, S. A. Roth, G. L. Loper, and E. H. Erickson. BEEPOP: A honeybee population dynamics simulation model. *Ecological Modelling*, 45(2):133–150, April 1989. ISSN 0304-3800. doi: 10.1016/0304-3800(89)90088-4. URL https://www.sciencedirect.com/science/article/pii/0304380089900884.

Olivier Delaneau, Cédric Coulonges, and Jean-François Zagury. Shape-IT: new rapid and accurate algorithm for haplotype inference. *BMC bioinformatics*, 9:1–14, 2008. Publisher: Springer.

Deutsch, Huey, Sheldon, and Ghalambor. Impacts of climate warming on terrestrial ectotherms across latitude \textbar PNAS, 2008. URL https://www.pnas.org/doi/abs/10.1073/pnas.0709472105.

Oxford English Dictionary. Simulation definition: Dictionary. URL https://www.oed.com/dictionary/simulation_n?tab=factsheet#22546636.

Kathleen A. Dogantzis, Tanushree Tiwari, Ida M. Conflitti, Alivia Dey, Harland M. Patch, Elliud M. Muli, Lionel Garnery, Charles W. Whitfield, Eckart Stolle, Ab-

dulaziz S. Alqarni, Michael H. Allsopp, and Amro Zayed. Thrice out of Asia and the adaptive radiation of the western honey bee. *Science Advances*, 7(49): eabj2151, December 2021. ISSN 2375-2548. doi: 10.1126/sciadv.abj2151. URL https://www.science.org/doi/10.1126/sciadv.abj2151.

Dolezal, Dvorsky, Kopecky, Mudrak, Capkova, and Liancourt. Functionally distinct assembly of vascular plants colonizing alpine cushions suggests their vulnerability to climate change \textbar Annals of Botany \textbar Oxford Academic, 2019. URL https://academic.oup.com/aob/article/123/4/569/5239867.

Tom Druet and Andres Legarra. Theoretical and empirical comparisons of expected and realized relationships for the X-chromosome. *Genetics Selection Evolution*, 52 (1):1–17, 2020.

Manuel Du, Richard Bernstein, Andreas Hoppe, and Kaspar Bienefeld. A theoretical derivation of response to selection with and without controlled mating in honeybees. *Genetics Selection Evolution*, 53:1–14, 2021a.

Manuel Du, Richard Bernstein, Andreas Hoppe, and Kaspar Bienefeld. Short-term effects of controlled mating and selection on the genetic variance of honeybee populations. *Heredity*, 126(5):733–747, May 2021b. ISSN 1365-2540. doi: 10.1038/s41437-021-00411-2. URL https://www.nature.com/articles/s41437-021-00411-2.

Johann Dzierzon. Gutachten über die von Herrn Direktor Stöhr im ersten und zweiten Kapitel des General-Gutachtens aufgestellten Fragen. *Eichstädter Bienenzeitung*, 1 (109-113):119–121, 1845.

Mehmet Ali Döke, Maryann Frazier, and Christina M Grozinger. Overwintering honey bees: biology and management. *Current Opinion in Insect Science*, 10:185–193, August 2015. ISSN 2214-5745. doi: 10.1016/j.cois.2015.05.014. URL https://www.sciencedirect.com/science/article/pii/S2214574515000930.

C. G. Eckert, K. E. Samis, and S. C. Lougheed. Genetic variation across species' geographical ranges: the central–marginal hypothesis and beyond. *Molecular Ecology*, 17(5):1170–1188, 2008. ISSN 1365-294X. doi: 10.1111/j.1365-294X.2007.03659.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-294X.2007.03659.x.

Dirk Eddelbuettel and James Joseph Balamuta. Extending *R* with C++: A Brief Introduction to Rcpp. *The American Statistician*, 72(1):28–36, January 2018. ISSN 0003-1305, 1537-2731. doi: 10.1080/00031305.2017.1375990. URL https://www.tandfonline.com/doi/full/10.1080/00031305.2017.1375990.

R.C. Elston and J. Stewart. A General Model for the Genetic Analysis of Pedigree Data. *Human Heredity*, 21(6):523–542, August 2008. ISSN 0001-5652. doi: 10.1159/000152448. URL https://doi.org/10.1159/000152448.

L. Otis Emik and Clair E. Terrill. Systematic procedures for calculating inbreeding coefficients. *Journal of Heredity*, 40(2):51–55, 1949.

Michael S Engel. The taxonomy of recent and fossil honey bees (hymenoptera: Apidae; apis). *Bulletin of the American Museum of Natural history*, 1999.

Gonçalo Espregueira Themudo, Alba Rey-Iglesia, Lucía Robles Tascón, Annette Bruun Jensen, Rute R. da Fonseca, and Paula F. Campos. Declining genetic diversity of European honeybees along the twentieth century. *Scientific Reports*, 10 (1):10520, June 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-67370-2. URL https://www.nature.com/articles/s41598-020-67370-2.

Sonia E. Eynard, Jack J. Windig, Grégoire Leroy, Rianne van Binsbergen, and Mario PL Calus. The effect of rare alleles on estimated genomic relationships from whole genome sequence data. *BMC Genetics*, 16(1):24, March 2015. ISSN 1471-2156. doi: 10.1186/s12863-015-0185-0. URL https://doi.org/10.1186/s12863-015-0185-0.

Sonia E. Eynard, Alain Vignal, Benjamin Basso, Kamila Canale-Tabet, Yves Le Conte, Axel Decourtye, Lucie Genestout, Emmanuelle Labarthe, Fanny Mondet, and Bertrand Servin. Reconstructing queen genotypes by pool sequencing colonies in eusocial insects: Statistical Methods and their application to honeybee. *Molecular Ecology Resources*, 22(8):3035–3048, 2022. ISSN 1755-0998. doi: 10.1111/1755-0998.13685. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1755-0998.13685.

Sonia E. Eynard, Fanny Mondet, Benjamin Basso, Olivier Bouchez, Yves Le Conte, Benjamin Dainat, Axel Decourtye, Lucie Genestout, Matthieu Guichard, François Guillaume, Emmanuelle Labarthe, Barbara Locke, Rachid Mahla, Joachim de Miranda, Markus Neuditschko, Florence Phocas, Kamila Tabet, Alain Vignal, and Bertrand Servin. Sequence-based genome-wide association studies reveal the polygenic architecture of Varroa destructor resistance in Western honey bees Apis mellifera, April 2024. URL https://www.biorxiv.org/content/10.1101/2024.02.16.580755v3.

D. S. Falconer. A note on Fisher's 'average effect' and 'average excess'. *Genetical Research*, 46(3):337–347, December 1985. ISSN 0016-6723, 1469-5073. doi: 10.1017/S0016672300022825. URL https://www.cambridge.org/core/product/identifier/S0016672300022825/type/journal_article.

D. S. Falconer and T. F. C. Mackay. Introduction to quantitative genetics. Essex. *UK: Longman Group*, 12, 1996.

C. L. Farrar. The influence of colony populations on honey production. *Journal of Agricultural Research*, 54(12):945–954, 1937.

Anne-Michelle Faux, Gregor Gorjanc, R. Chris Gaynor, Mara Battagin, Stefan M. Edwards, David L. Wilson, Sarah J. Hearne, Serap Gonen, and John M. Hickey. AlphaSim: Software for Breeding Program Simulation. *The Plant Genome*, 9(3):plantgenome2016.02.0013, 2016. ISSN 1940-3372. doi: 10.3835/plantgenome2016.02.0013. URL https://onlinelibrary.wiley.com/doi/abs/10.3835/plantgenome2016.02.0013.

Rui Feng, Yinghua Wu, Gun Ho Jang, Jose M Ordovas, and Donna Arnett. A powerful test of parent-of-origin effects for quantitative traits using haplotypes. *PLoS One*, 6 (12):e28909, 2011. Publisher: Public Library of Science San Francisco, USA.

RL Fernando and M Grossman. Genetic evaluation with autosomal and X-chromosomal inheritance. *Theoretical and applied genetics*, 80:75–80, 1990. Publisher: Springer.

Jesús Fernández and Jörn Bennewitz. Chapter 2. Defining genetic diversity based on genomic tools. Brill, March 2017. doi: 10.3920/978-90-8686-850-6_2. URL https://brill.com/edcollchap/book/9789086868506/BP000003.xml.

Ronald A Fisher. XV.—The correlation between relatives on the supposition of Mendelian inheritance. *Earth and Environmental Science Transactions of the Royal Society of Edinburgh*, 52(2):399–433, 1918. Publisher: Royal Society of Edinburgh Scotland Foundation.

Sarah P. Flanagan and Adam G. Jones. The future of parentage analysis: From microsatellites to SNPs and beyond. *Molecular Ecology*, 28(3):544–567, 2019. ISSN 1365-294X. doi: 10.1111/mec.14988. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/mec.14988.

Stanley E. Flanders. Control of Sex in the Honeybee. *The Scientific Monthly*, 71(4): 237–240, 1950. ISSN 0096-3771. URL https://www.jstor.org/stable/20024.

Foley, DeFries, Barford, Bonan, and Chapin. Global Consequences of Land Use \textbar Science, 2005. URL https://www.science.org/doi/full/10.1126/science.1111772?casa_token=BU7fG8HmUxkAAAAA%3Axgv42SR7tP3hwyvGdRBp65F4fOkzeW1NxtlMPoUJCZq7NHYgQr7GDXMvjKoDPStT3qLgjU5lfpg38A.

Giovanni Forcina and Jennifer A. Leonard. Tools for Monitoring Genetic Diversity in Mammals: Past, Present, and Future. In Jorge Ortega and Jesus E. Maldonado, editors, *Conservation Genetics in Mammals: Integrative Research Using Novel Approaches*, pages 13–27. Springer International Publishing, Cham, 2020. ISBN 978-3-030-33334-8. doi: 10.1007/978-3-030-33334-8_2. URL https://doi.org/10.1007/978-3-030-33334-8_2.

Pierre Franck, Lionel Garnery, Michel Solignac, and Jean-Marie Cornuet. The Origin of West European Subspecies of Honeybees (apis Mellifera): New Insights from Microsalite and Mitochondrial Data. *Evolution*, 52(4):1119–1134, 1998. ISSN 1558-5646. doi: 10.1111/j.1558-5646.1998.tb01839.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1558-5646.1998.tb01839.x.

Angela P Fuentes-Pardo and Daniel E Ruzzante. Whole-genome sequencing approaches for conservation biology: Advantages, limitations and practical recommendations. *Molecular ecology*, 26(20):5369–5406, 2017.

Nicola Gallai, Jean-Michel Salles, Josef Settele, and Bernard E. Vaissière. Economic valuation of the vulnerability of world agriculture confronted with pollinator decline. *Ecological Economics*, 68(3):810–821, January 2009. ISSN 0921-8009. doi: 10.1016/j.ecolecon.2008.06.014. URL https://www.sciencedirect.com/science/article/pii/S0921800908002942.

L. Garnery, J.-M. Cornuet, and M. Solignac. Evolutionary history of the honey bee *Apis* mellifera inferred from mitochondrial DNA analysis. *Molecular Ecology*, 1(3):145–154, October 1992. ISSN 0962-1083, 1365-294X. doi: 10.1111/j.1365-294X.1992.tb00170.x. URL https://onlinelibrary.wiley.com/doi/10.1111/j.1365-294X.1992.tb00170.x.

R Chris Gaynor, Gregor Gorjanc, and John M Hickey. AlphaSimR: an R package for breeding program simulations. *G3 Genes\textbarGenomes\textbarGenetics*, 11(2):jkaa017, April 2021. ISSN 2160-1836. doi: 10.1093/g3journal/jkaa017. URL https://academic.oup.com/g3journal/article/doi/10.1093/g3journal/jkaa017/6025179.

Dominique P. Germain and Iulia E. Jurca-Simina. Principles of Human Genetics and Mendelian Inheritance. In Alessandro P. Burlina, editor, *Neurometabolic Hereditary Diseases of Adults*, pages 1–28. Springer International Publishing, Cham, 2018. ISBN 978-3-319-76148-0. doi: 10.1007/978-3-319-76148-0_1. URL https://doi.org/10.1007/978-3-319-76148-0_1.

Thomas L. Gillard and Benjamin P. Oldroyd. Controlled reproduction in the honey bee (*Apis* mellifera) via artificial insemination. In Russell Jurenka, editor, *Advances in Insect Physiology*, volume 59, pages 1–42. Academic Press, January 2020. doi: 10.1016/bs.aiip.2020.08.001. URL https://www.sciencedirect.com/science/article/pii/S0065280620300199.

Roe Goodman. *Introduction to stochastic models*. Courier Corporation, 2006.

Jérôme Goudet, Tomas Kay, and Bruce S Weir. How to estimate kinship. *Molecular ecology*, 27(20):4121–4135, 2018.

Henri Goulet, John T. Huber, and Canada Agriculture Canada Research Branch. *Hymenoptera of the world: an identification guide to families*. Number 1894 in Publication (Canada. Agriculture Canada). Research Branch, Agriculture Canada, Ottawa], 1993. ISBN 978-0-660-14933-2.

gov.uk. Bee importation - Defra in the media, February 2021. URL https://deframedia.blog.gov.uk/2021/02/08/bee-importation/.

S. Graham, M. R. Myerscough, J. C. Jones, and B. P. Oldroyd. Modelling the role of intracolonial genetic diversity on regulation of brood temperature in honey bee (Apis mellifera L.) colonies. *Insectes Sociaux*, 53(2):226–232, May 2006. ISSN 1420-9098. doi: 10.1007/s00040-005-0862-5. URL https://doi.org/10.1007/s00040-005-0862-5.

L. F. Groeneveld, L. A. Kirkerud, B. Dahle, M. Sunding, M. Flobakk, M. Kjos, D. Henriques, M. A. Pinto, and P. Berg. Conservation of the dark bee (*Apis mellifera mellifera*): Estimating C-lineage introgression in Nordic breeding stocks. *Acta Agriculturae Scandinavica, Section A — Animal Science*, 69(3):157–168, July 2020. ISSN 0906-4702, 1651-1972. doi: 10.1080/09064702.2020.1770327. URL https://www.tandfonline.com/doi/full/10.1080/09064702.2020.1770327.

M. Grossman and E. J. Eisen. Inbreeding, coancestry, and covariance between relatives for X-chromosomal loci. *Journal of Heredity*, 80(2):137–142, 1989.

M Grossman and RL Fernando. Covariance between relatives for X-chromosomal loci in a population in disequilibrium. *Theoretical and applied genetics*, 77:311–319, 1989. Publisher: Springer.

Benjamin C Haller and Philipp W Messer. SLiM 3: Forward Genetic Simulations Beyond the Wright–Fisher Model. *Molecular Biology and Evolution*, 36(3):632–637, March 2019. ISSN 0737-4038, 1537-1719. doi: 10.1093/molbev/msy228. URL https://academic.oup.com/mbe/article/36/3/632/5229931.

Matthew Gray Hamilton. Maximum likelihood parentage assignment using quantitative genotypes. *Heredity*, 126(6):884–895, June 2021. ISSN 1365-2540. doi: 10.1038/s41437-021-00421-0. URL https://www.nature.com/articles/s41437-021-00421-0.

Henrik Hansen and Camilla Juul Brødsgaard. American foulbrood: a review of its biology, diagnosis and control. *Bee world*, 80(1):5–23, 1999.

Md Abul Hasnat. Reproductive potential difference of artificially inseminated and naturally mated honey bee queens (apis mellifera l.), 2018.

C. R. Henderson. A Simple Method for Computing the Inverse of a Numerator Relationship Matrix Used in Prediction of Breeding Values. *Biometrics*, 32(1):69–83, 1976. ISSN 0006-341X. doi: 10.2307/2529339. URL https://www.jstor.org/stable/2529339.

Heather J. Henter. Inbreeding depression and haplodiploidy: experimental measures in a parasitoid and comparisons across diploid and haplodiploid insect taxa. *Evolution*, 57(8):1793–1803, 2003.

John Hickey. Alphaimpute: software package for imputing and phasing genotype data. *Edinburgh University*, 2016.

John M Hickey and Gregor Gorjanc. Simulated Data for Genomic Selection and Genome-Wide Association Studies Using a Combination of Coalescent and Gene Drop Methods. *G3 Genes\textbarGenomes\textbarGenetics*, 2(4):425–427, April 2012. ISSN 2160-1836. doi: 10.1534/g3.111.001297. URL https://doi.org/10.1534/g3.111.001297.

Higes, Martín-Hernández, Garrido-Bailón, and González-Porto. Honeybee colony collapse due to Nosema ceranae in professional apiaries - Higes - 2009 - Environmental Microbiology Reports - Wiley Online Library, 2009. URL https://enviromicro-journals.onlinelibrary.wiley.com/doi/full/10.1111/j.1758-2229.2009.00014.x?casa_token=mCCE0KeVeUwAAAAA%3AAecFJWLVOQl8sZK_5tNeTXJ_ZjrmauaqM0cCE7JTcAsKUw7w9csGK1K9TNA8r5LP4zKzz16_Q0QzvQ.

William G Hill and Bruce S Weir. Variation in actual relationship as a consequence of Mendelian sampling and linkage. *Genetics research*, 93(1):47–64, 2011. Publisher: Cambridge University Press.

Sean Hoban, Michael W Bruford, W Chris Funk, Peter Galbusera, M Patrick Griffith, Catherine E Grueber, Myriam Heuertz, Margaret E Hunter, Christina Hvilsom, Belma Kalamujic Stroil, Francine Kershaw, Colin K Khoury, Linda Laikre, Margarida Lopes-Fernandes, Anna J MacDonald, Joachim Mergeay, Mariah Meek, Cinnamon Mittan, Tarek A Mukassabi, David O'Brien, Rob Ogden, Clarisse PALMA-SILVA, Uma Ramakrishnan, Gernot Segelbacher, Robyn E Shaw, Per Sjögren-Gulve, Nevena Veličković, and Cristiano Vernesi. Global Commitments to Conserving and Monitoring Genetic Diversity Are Now Necessary and Feasible. *BioScience*, 71(9): 964–976, September 2021. ISSN 0006-3568. doi: 10.1093/biosci/biab054. URL `https://doi.org/10.1093/biosci/biab054`.

CJ Hogg, B Wright, KM Morris, AV Lee, JA Ivy, CE Grueber, and K Belov. Founder relationships and conservation management: empirical kinships reveal the effect on breeding programmes when founders are assumed to be unrelated. *Animal Conservation*, 22(4):348–361, 2019. Publisher: Wiley Online Library.

Shelley ER Hoover, Christopher I. Keeling, Mark L. Winston, and Keith N. Slessor. The effect of queen pheromones on worker honey bee ovary development. *Naturwissenschaften*, 90(10):477–480, 2003.

Richard Howey, Chrysovalanto Mamasoula, Ana Töpf, Ron Nudel, Judith A Goodship, Bernard D Keavney, and Heather J Cordell. Increased power for detection of parent-of-origin effects via the use of haplotype estimation. *The American Journal of Human Genetics*, 97(3):419–434, 2015. Publisher: Elsevier.

Jisca Huisman. Pedigree reconstruction from SNP data: parentage assignment, sibship clustering and beyond. *Molecular Ecology Resources*, 17(5):1009–1024, 2017. ISSN 1755-0998. doi: 10.1111/1755-0998.12665. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/1755-0998.12665`.

Rustem A. Ilyasov, Myeong-lyeol Lee, Jun-ichi Takahashi, Hyung Wook Kwon, and Alexey G. Nikolenko. A revision of subspecies structure of western honey bee Apis mellifera. *Saudi Journal of Biological Sciences*, 27(12):3615–3621, December 2020. ISSN 1319-562X. doi: 10.1016/j.sjbs.2020.08.001. URL `https://www.sciencedirect.com/science/article/pii/S1319562X20303363`.

Zafar H Israili. Antimicrobial properties of honey. *American journal of therapeutics*, 21(4):304–323, 2014.

Rodolfo Jaffé, Vincent Dietemann, Mike H. Allsopp, Cecilia Costa, Robin M. Crewe, Raffaele Dall'olio, Pilar De La Rúa, Mogbel a. A. El-Niweiri, Ingemar Fries, Nikola Kezic, Michael S. Meusel, Robert J. Paxton, Taher Shaibi, Eckart Stolle, and Robin F.a. Moritz. Estimating the Density of Honeybee Colonies across Their Natural Range to Fill the Gap in Pollinator Decline Censuses. *Conservation Biology*, 24(2):583–593, 2010. ISSN 1523-1739. doi: 10.1111/j.1523-1739.2009.01331.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1523-1739.2009.01331.x`.

J Jamrozik and LR Schaeffer. An equivalent gametic model for animal dominance genetic linear model., 1991.

Elbert R. Jaycox. Honey Bee Queen Pheromones and Worker Foraging Behavior1. *Annals of the Entomological Society of America*, 63(1):222–228, January 1970. ISSN 0013-8746. doi: 10.1093/aesa/63.1.222. URL https://doi.org/10.1093/aesa/63.1.222.

Annette B. Jensen, Kellie A. Palmer, Jacobus J. Boomsma, and Bo V. Pedersen. Varying degrees of Apis mellifera ligustica introgression in protected populations of the black honeybee, Apis mellifera mellifera, in northwest Europe. *Molecular Ecology*, 14(1):93–106, 2005. ISSN 1365-294X. doi: 10.1111/j.1365-294X.2004.02399.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-294X.2004.02399.x.

HA Jensen, F Mayorga, and C Papadimitriou. Reliability sensitivity analysis of stochastic finite element models. *Computer Methods in Applied Mechanics and Engineering*, 296:327–351, 2015.

Reed M. Johnson, Marion D. Ellis, Christopher A. Mullin, and Maryann Frazier. Pesticides and honey bee toxicity – USA. *Apidologie*, 41(3):312–331, May 2010. ISSN 0044-8435, 1297-9678. doi: 10.1051/apido/2010018. URL https://www.apidologie.org/articles/apido/abs/2010/03/m09141/m09141.html.

Julia C Jones, Zhipei G Du, Richard Bernstein, Monique Meyer, Andreas Hoppe, Elmar Schilling, Martin Ableitner, Katrin Juling, Regina Dick, Anja S Strauss, and others. Tool for genomic selection and breeding to evolutionary adaptation: Development of a 100K single nucleotide polymorphism array for the honey bee. *Ecology and Evolution*, 10(13):6246–6256, 2020. Publisher: Wiley Online Library.

Owen R Jones and Jinliang Wang. Colony: a program for parentage and sibship inference from multilocus genotype data. *Molecular ecology resources*, 10(3):551–555, 2010.

Masaki Kamakura. Royalactin induces queen differentiation in honeybees. *Nature*, 473(7348):478–483, May 2011. ISSN 1476-4687. doi: 10.1038/nature10093. URL https://www.nature.com/articles/nature10093.

Terry Ryan Kane and Cynthia M. Faux. *Honey Bee Medicine for the Veterinary Practitioner*. John Wiley & Sons, January 2021. ISBN 978-1-119-58342-4.

Manjit S. Kang, Mónica G. Balzarini, and Jose LL Guerra. Genotype-by-environment interaction. *Genetic analysis of complex traits using SAS*, pages 69–96, 2004. URL https://www.researchgate.net/profile/Ebrahim-Talebi/post/Best-statistical-approach-to-determine-correlation-between-clinical-histological-attachment/5c38f6293843b006755019d7/AS%3A713977482997762%401547236905325/download/epdf.tips_genetic-analysis-of-complex-traits-using-sas.pdf#page=82.

Jaana Kekkonen. Temporal Genetic Monitoring of Declining and Invasive Wildlife Populations: Current State and Future Directions. In Francesco M. Angelici, editor, *Problematic Wildlife: A Cross-Disciplinary Approach*, pages 269–294. Springer

International Publishing, Cham, 2016. ISBN 978-3-319-22246-2. doi: 10.1007/978-3-319-22246-2_13. URL https://doi.org/10.1007/978-3-319-22246-2_13.

Jerome Kelleher, Alison M Etheridge, and Gilean McVean. Efficient Coalescent Simulation and Genealogical Analysis for Large Sample Sizes. *PLOS Computational Biology*, 12(5):e1004842, May 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004842. URL https://dx.plos.org/10.1371/journal.pcbi.1004842.

Robert B Kent. The introduction and diffusion of the african honeybee in south america. *Yearbook of the Association of Pacific Coast Geographers*, 50(1):21–43, 1988.

Warwick E Kerr, Ronald Zucchi, Julio Takeshi Nakadaira, and José Eduardo Butolo. Reproduction in the social bees (Hymenoptera: Apidae). *Journal of the New York Entomological Society*, pages 265–276, 1962. Publisher: JSTOR.

Tristan Kistler, Benjamin Basso, and Florence Phocas. A simulation study of a honeybee breeding scheme accounting for polyandry, direct and maternal effects on colony performance. *Genetics Selection Evolution*, 53:1–16, 2021.

Tristan Kistler, Coline Kouchner, Evert W Brascamp, Charlène Dumas, Fanny Mondet, Alain Vignal, Benjamin Basso, Piter Bijma, and Florence Phocas. Heritability and correlations for honey yield, handling ease, brood quantity, and traits related to resilience in a french honeybee population. *Apidologie*, 55(4):52, 2024.

Alexandra-Maria Klein, Bernard E Vaissière, James H Cane, Ingolf Steffan-Dewenter, Saul A Cunningham, Claire Kremen, and Teja Tscharntke. Importance of pollinators in changing landscapes for world crops. *Proceedings of the Royal Society B: Biological Sciences*, 274(1608):303–313, February 2007. ISSN 0962-8452, 1471-2954. doi: 10.1098/rspb.2006.3721. URL https://royalsocietypublishing.org/doi/10.1098/rspb.2006.3721.

Sarah D. Kocher, Freddie-Jeanne Richard, David R. Tarpy, and Christina M. Grozinger. Queen reproductive state modulates pheromone production and queen-worker interactions in honeybees. *Behavioral Ecology*, 20(5):1007–1014, 2009.

Nikolaus Koeniger, Gudrun Koeniger, Michael Gries, and Salim Tingek. Drone competition at drone congregation areas in four Apis species. *Apidologie*, 36(2):211–221, April 2005. ISSN 0044-8435, 1297-9678. doi: 10.1051/apido:2005011. URL http://dx.doi.org/10.1051/apido:2005011.

Judith Korb, Karen Meusemann, Denise Aumer, Abel Bernadou, Daniel Elsner, Barbara Feldmeyer, Susanne Foitzik, Jürgen Heinze, Romain Libbrecht, and Silu Lin. Comparative transcriptomic analysis of the mechanisms underpinning ageing and fecundity in social insects. *Philosophical Transactions of the Royal Society B*, 376 (1823):20190728, 2021.

Arthur Korte and Ashley Farlow. The advantages and limitations of trait analysis with GWAS: a review. *Plant methods*, 9:1–9, 2013. Publisher: Springer.

Harry H. Laidlaw Jr and Robert E. Page Jr. Polyandry in honey bees (Apis mellifera

L.): sperm utilization and intracolony genetic relationships. *Genetics*, 108(4):985–997, 1984.

Letícia A. de C. Lara, Ivan Pocrnic, Thiago de P. Oliveira, R. Chris Gaynor, and Gregor Gorjanc. Temporal and genomic analysis of additive genetic variance in breeding programmes. *Heredity*, 128(1):21–32, January 2022. ISSN 1365-2540. doi: 10.1038/s41437-021-00485-y. URL https://www.nature.com/articles/s41437-021-00485-y.

G M Lathrop, A B Hooper, J W Huntsman, and R H Ward. Evaluating pedigree data. I. The estimation of pedigree error in the presence of marker mistyping. *American Journal of Human Genetics*, 35(2):241–262, March 1983. ISSN 0002-9297. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1685535/.

M. Elise Lauterbur, Maria Izabel A. Cavassim, Ariella L. Gladstein, Graham Gower, Nathaniel S. Pope, Georgia Tsambos, Jeff Adrion, Saurabh Belsare, Arjun Biddanda, Victoria Caudill, Jean Cury, Ignacio Echevarria, Benjamin C. Haller, Ahmed R. Hasan, Xin Huang, Leonardo Nicola Martin Iasi, Ekaterina Noskova, Jana Obšteter, Vitor Antonio Corrêa Pavinato, Alice Pearson, David Peede, Manolo F. Perez, Murillo F. Rodrigues, Chris C. R. Smith, Jeffrey P. Spence, Anastasia Teterina, Silas Tittes, Per Unneberg, Juan Manuel Vazquez, Ryan K. Waples, Anthony Wilder Wohns, Yan Wong, Franz Baumdicker, Reed A. Cartwright, Gregor Gorjanc, Ryan N. Gutenkunst, Jerome Kelleher, Andrew D. Kern, Aaron P. Ragsdale, Peter L. Ralph, Daniel R. Schrider, and Ilan Gronau. Expanding the stdpopsim species catalog, and lessons learned for realistic genome simulations, October 2022. URL https://www.biorxiv.org/content/10.1101/2022.10.29.514266v1.

Y Le Conte and M Navajas. Changements climatiques : impact sur les populations d'abeilles et leurs maladies. *World Organization for Animal Health (OIE)*, 27(2):pp. 485, August 2008. doi: 10.20506/rst.27.2.1819. URL https://doc.woah.org/dyn/portal/index.xhtml?page=alo&aloId=30769.

Sarah Lechner, Luca Ferretti, Caspar Schöning, Wanja Kinuthia, David Willemsen, and Martin Hasselmann. Nucleotide Variability at Its Limit? Insights into the Number and Evolutionary Dynamics of the Sex-Determining Specificities of the Honey Bee Apis mellifera. *Molecular Biology and Evolution*, 31(2):272–287, February 2014. ISSN 0737-4038. doi: 10.1093/molbev/mst207. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3907057/.

Christina Lehermeier, Simon Teyssèdre, and Chris-Carolin Schön. Genetic Gain Increases by Applying the Usefulness Criterion with Improved Variance Prediction in Selection of Crosses. *Genetics*, 207(4):1651–1661, December 2017. ISSN 1943-2631. doi: 10.1534/genetics.117.300403. URL https://academic.oup.com/genetics/article/207/4/1651/5930776.

Leonard Lepadatu. Risk determination in projects. the advantages and disadvantages of stochastic methods. *Acta Universitatis Danubius. Œconomica*, 5(1):101–110, 2009.

I. M. Lerner. Genetic homeostasis. *Genetic homeostasis.*, 1954. URL https://www.cabdirect.org/cabdirect/abstract/19550100446.

Haoxuan Liu, Xiaohui Zhang, Ju Huang, Jian-Qun Chen, Dacheng Tian, Laurence D Hurst, and Sihai Yang. Causes and consequences of crossing-over evidenced via a high-resolution recombinational landscape of the honey bee. *Genome biology*, 16: 1–20, 2015.

Marco Lodesani and Cecilia Costa. Bee breeding and genetics in Europe. *Bee World*, 84(2):69–85, January 2003. ISSN 0005-772X. doi: 10.1080/0005772X.2003.11099579. URL `https://doi.org/10.1080/0005772X.2003.11099579`.

F Hosseinzadeh Lotfi, N Nematollahi, MH Behzadi, and M Mirbolouki. Ranking decision making units with stochastic data by using coefficient of variation. *Mathematical and Computational Applications*, 15(1):148–155, 2010.

M. L. Makgahlela, I. Strandén, U. S. Nielsen, M. J. Sillanpää, and E. A. Mäntysaari. The estimation of genomic relationships using breedwise allele frequencies among animals in multibreed populations. *Journal of Dairy Science*, 96(8): 5364–5375, August 2013. ISSN 0022-0302. doi: 10.3168/jds.2012-6523. URL `https://www.sciencedirect.com/science/article/pii/S0022030213004359`.

Gustave Malécot. Mathématiques de l'hérédité. *Unknown Journal*, 1948.

Ani Manichaikul, Josyf C. Mychaleckyj, Stephen S. Rich, Kathy Daly, Michèle Sale, and Wei-Min Chen. Robust relationship inference in genome-wide association studies. *Bioinformatics (Oxford, England)*, 26(22):2867–2873, November 2010. ISSN 1367-4811. doi: 10.1093/bioinformatics/btq559.

Gabriel Marais. Biased gene conversion: implications for genome and sex evolution. *TRENDS in Genetics*, 19(6):330–338, 2003.

Heather R. Mattila and Thomas D. Seeley. Genetic Diversity in Honey Bee Colonies Enhances Productivity and Fitness. *Science*, 317(5836):362–364, July 2007. doi: 10.1126/science.1143046. URL `https://www.science.org/doi/full/10.1126/science.1143046`.

Oliver Mayo. A century of Hardy–Weinberg equilibrium. *Twin Research and Human Genetics*, 11(3):249–256, 2008. Publisher: Cambridge University Press.

THE Meuwissen, T Luan, and JA Woolliams. The unified approach to the use of genomic and pedigree information in genomic evaluations revisited. *Journal of Animal Breeding and Genetics*, 128(6):429–439, 2011. Publisher: Wiley Online Library.

Charles Duncan Michener. *The bees of the world*, volume 1. JHU press, 2000.

Irati Miguel, Michel Baylac, Mikel Iriondo, Carmen Manzano, Lionel Garnery, and Andone Estonba. Both geometric morphometric and microsatellite data consistently support the differentiation of the Apis mellifera M evolutionary branch. *Apidologie*, 42(2):150–161, March 2011. ISSN 0044-8435, 1297-9678. doi: 10.1051/apido/2010048. URL `http://link.springer.com/10.1051/apido/2010048`.

Ignacy Misztal, Andres Legarra, and Ignacio Aguilar. Computing procedures for genetic evaluation including phenotypic, full pedigree, and genomic information. *Journal of dairy science*, 92(9):4648–4655, 2009. Publisher: Elsevier.

Jamal Momeni, Melanie Parejo, Rasmus O. Nielsen, Jorge Langa, Iratxe Montes, Laetitia Papoutsis, Leila Farajzadeh, Christian Bendixen, Eliza Căuia, Jean-Daniel Charrière, Mary F. Coffey, Cecilia Costa, Raffaele Dall'Olio, Pilar De la Rúa, M. Maja Drazic, Janja Filipi, Thomas Galea, Miroljub Golubovski, Ales Gregorc, Karina Grigoryan, Fani Hatjina, Rustem Ilyasov, Evgeniya Ivanova, Irakli Janashia, Irfan Kandemir, Aikaterini Karatasou, Meral Kekecoglu, Nikola Kezic, Enikö Sz. Matray, David Mifsud, Rudolf Moosbeckhofer, Alexei G. Nikolenko, Alexandros Papachristoforou, Plamen Petrov, M. Alice Pinto, Aleksandr V. Poskryakov, Aglyam Y. Sharipov, Adrian Siceanu, M. Ihsan Soysal, Aleksandar Uzunov, Marion Zammit-Mangion, Rikke Vingborg, Maria Bouga, Per Kryger, Marina D. Meixner, and Andone Estonba. Authoritative subspecies diagnosis tool for European honey bees based on ancestry informative SNPs. *BMC Genomics*, 22(1):101, February 2021. ISSN 1471-2164. doi: 10.1186/s12864-021-07379-7. URL https://doi.org/10.1186/s12864-021-07379-7.

Robin F. A. Moritz and Edward E. Southwick. Phenotype interactions in group behavior of honey bee workers (Apis mellifera L.). *Behavioral Ecology and Sociobiology*, 21(1):53–57, July 1987. ISSN 1432-0762. doi: 10.1007/BF00324435. URL https://doi.org/10.1007/BF00324435.

Robin F. A. Moritz, Stephan Härtel, and Peter Neumann. Global invasions of the western honeybee (*Apis mellifera*) and the consequences for biodiversity. *Écoscience*, 12(3):289–301, January 2005. ISSN 1195-6860, 2376-7626. doi: 10.2980/i1195-6860-12-3-289.1. URL https://www.tandfonline.com/doi/full/10.2980/i1195-6860-12-3-289.1.

Robin FA Moritz, F Bernhard Kraus, Per Kryger, and Robin M Crewe. The size of wild honeybee populations (apis mellifera) and its implications for the conservation of honeybees. *Journal of Insect Conservation*, 11:391–397, 2007.

Morris, Pfister, Tuljapurkar, and Boggs. LONGEVITY CAN BUFFER PLANT AND ANIMAL POPULATIONS AGAINST CHANGING CLIMATIC VARIABILITY - Morris - 2008 - Ecology - Wiley Online Library, 2008. URL https://esajournals.onlinelibrary.wiley.com/doi/full/10.1890/07-0774.1?casa_token=qmqCyhyX0iMAAAAA%3Aq44QCIdxkTag-tJG_X8Nrj2ltWY43jDFTlzrIDyofp0E-pn1Xktb64Ol3JSF_GR7KAQj5el057PNQQ.

Robert Mroczek, Joanna Niedbalska-Tarnowska, Ajda Moškrič, Kinga Adamczyk-Węglarzy, Agnieszka Łaszkiewicz, and Małgorzata Cebrat. The Potential of Complementary Sex-Determiner Gene Allelic Diversity for Studying the Number of Patrilines within Honeybee Colonies. *Applied Sciences*, 14(1):26, January 2024. ISSN 2076-3417. doi: 10.3390/app14010026. URL https://www.mdpi.com/2076-3417/14/1/26.

W.m. Muir. Comparison of genomic and traditional BLUP-estimated breeding value accuracy and selection response under alternative trait and genomic parameters. *Journal of Animal Breeding and Genetics*, 124(6):342–355, 2007. ISSN 1439-0388. doi: 10.1111/j.1439-0388.2007.00700.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1439-0388.2007.00700.x.

Fiona N. Mumoki and Robin M. Crewe. Pheromone communication in honey bees (*Apis* mellifera). In Gary J. Blomquist and Richard G. Vogt, editors, *Insect Pheromone Biochemistry and Molecular Biology (Second Edition)*, pages 183–204. Academic Press, London, January 2021. ISBN 978-0-12-819628-1. doi: 10.1016/B978-0-12-819628-1.00006-7. URL https://www.sciencedirect.com/science/article/pii/B9780128196281000067.

Irene Muñoz, Maria Alice Pinto, and Pilar De la Rúa. Effects of queen importation on the genetic diversity of Macaronesian island honey bee populations (Apis mellifera Linneaus 1758). *Journal of Apicultural Research*, 53(2):296–302, January 2014. ISSN 0021-8839. doi: 10.3896/IBRA.1.53.2.11. URL https://doi.org/10.3896/IBRA.1.53.2.11.

Barry L Nelson. *Stochastic modeling: analysis & simulation*. Courier Corporation, 2010.

Angelika Neukirch. Dependence of the life span of the honeybee (Apis mellifica) upon flight performance and energy consumption. *Journal of comparative physiology*, 146 (1):35–40, March 1982. ISSN 1432-1351. doi: 10.1007/BF00688714. URL https://doi.org/10.1007/BF00688714.

Peter Neumann, Robin F A Moritz, and Job van Praagh. Queen mating frequency in different types of honey bee mating apiaries. *Journal of Apicultural Research*, 38 (1-2):11–18, January 1999. ISSN 0021-8839. doi: 10.1080/00218839.1999.11100990. URL https://doi.org/10.1080/00218839.1999.11100990.

T. R. New. *Hymenoptera and conservation*. Wiley, Chichester, West Sussex, UK ; Hoboken, NJ, 2012. ISBN 978-1-118-38131-1.

Nicola Gallai, Jean-Michel Salles, Josef Settele, and Bernard E. Vaissière. Economic valuation of the vulnerability of world agriculture confronted with pollinator decline \textbar Elsevier Enhanced Reader. URL https://reader.elsevier.com/reader/sd/pii/S0921800908002942?token=5CCD5396DEE311E33F026E71357220CA6CECD23673B61F72AC0672E1047FD433D28B7593DD9AEF8923D6originRegion=eu-west-1&originCreation=20210504110532.

Anders Nielsen, Trond Reitan, Andreas W Rinvoll, and Anne K Brysting. Effects of competition and climate on a crop pollinator community. *Agriculture, Ecosystems & Environment*, 246:253–260, 2017.

NIHBS. The Native Irish Honey Bee Society – Apis mellifera mellifera, 2012. URL https://nihbs.org/.

Magnus Nordborg and Simon Tavaré. Linkage disequilibrium, haplotype evolution, and the coalescent. *Trends Genet.*, 18:83–90, 2002.

O'Brien and Marsh. Vertebrate pests of beekeeping, 1990. URL https://escholarship.org/uc/item/40v8v51b.

J Obšteter, A Marinč, J Prešern, D Wragg, and G Gorjanc. Inferring whole-genome tree sequences and population and demographic parameters of the Western honeybee.

Rotterdam, The Netherlands, 2022. Wageningen Academic Publishers. URL https://www.wageningenacademic.com/pb-assets/wagen/WCGALP2022/53_013.pdf.

Jana Obšteter, Laura K. Strachan, Jernej Bubnič, Janez Prešern, and Gregor Gorjanc. SIMplyBee: an R package to simulate honeybee populations and breeding programs. *Genetics Selection Evolution*, 55(1):31, May 2023. ISSN 1297-9686. doi: 10.1186/s12711-023-00798-y. URL https://doi.org/10.1186/s12711-023-00798-y.

Benjamin P. Oldroyd, Morag J. Clifton, Kerrie Parker, Siriwat Wongsiri, Thomas E. Rinderer, and Ross H. Crozier. Evolution of Mating Behavior in the Genus Apis and an Estimate of Mating Frequency in Apis cerana (Hymenoptera: Apidae). *Annals of the Entomological Society of America*, 91(5):700–709, September 1998. ISSN 0013-8746. doi: 10.1093/aesa/91.5.700. URL https://doi.org/10.1093/aesa/91.5.700.

Randy Oliver. The Honey Bee Queen. *Honey Bee Medicine for the Veterinary Practitioner*, pages 55–71, 2021. Publisher: Wiley Online Library.

Krzysztof Olszewski, Grzegorz Borsuk, Jerzy Paleolog, and Aneta Strachecka. Evaluation of economic traits in buckfast bees in comparison with the hybrids of european black bees and caucasian bees. *Annales Universitatis Mariae Curie-Skłodowska. Sectio EE: Zootechnica*, 30(2), 2012.

Peter R Oxley, Pantip Hinhumpatch, Rosalyn Gloag, and Benjamin P Oldroyd. Genetic evaluation of a novel system for controlled mating of the honeybee, Apis mellifera. *Journal of heredity*, 101(3):334–338, 2010. Publisher: Oxford University Press.

Robert E Page and Christine Y. S Peng. Aging and development in social insects with emphasis on the honey bee, *Apis* mellifera L. *Experimental Gerontology*, 36(4):695–711, April 2001. ISSN 0531-5565. doi: 10.1016/S0531-5565(00)00236-9. URL https://www.sciencedirect.com/science/article/pii/S0531556500002369.

Delphine Panziera, Fabrice Requier, Panuwan Chantawannakul, Christian Walter Werner Pirk, and Tjeerd Blacquiere. The diversity decline in wild and managed honey bee populations urges for an integrated conservation approach. March 2022. ISSN 2296-701X (online). doi: 10.3389/fevo.2022.767950. URL https://repository.up.ac.za/handle/2263/86561.

Melanie Parejo, Egoitz Galartza, Jamal Momeni, June Gorrochategui-Ortega, Leila Farajzadeh, Jakob Wegener, Kaspar Bienefeld, Iratxe Zarraonaindia, and Andone Estonba. A SNP-based honey bee paternity assignment test for evaluating the effectiveness of mating stations and its application to the Ataun valley, Basque Country, Spain. *bioRxiv*, pages 2024–02, 2024. Publisher: Cold Spring Harbor Laboratory.

Avani Patel, M. Kim Fondrk, Osman Kaftanoglu, Christine Emore, Greg Hunt, Katy Frederick, and Gro V. Amdam. The Making of a Queen: TOR Pathway Is a Key Player in Diphenic Caste Development. *PLoS ONE*, 2(6):e509, June 2007. ISSN 1932-6203. doi: 10.1371/journal.pone.0000509. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1876819/.

J.m Pemberton. Wild pedigrees: the way forward. *Proceedings of the Royal Society*

B: Biological Sciences, 275(1635):613–621, January 2008. doi: 10.1098/rspb.2007.1531. URL https://royalsocietypublishing.org/doi/full/10.1098/rspb.2007.1531.

Ralph S. Peters, Lars Krogmann, Christoph Mayer, Alexander Donath, Simon Gunkel, Karen Meusemann, Alexey Kozlov, Lars Podsiadlowski, Malte Petersen, Robert Lanfear, Patricia A. Diez, John Heraty, Karl M. Kjer, Seraina Klopfstein, Rudolf Meier, Carlo Polidori, Thomas Schmitt, Shanlin Liu, Xin Zhou, Torsten Wappler, Jes Rust, Bernhard Misof, and Oliver Niehuis. Evolutionary History of the Hymenoptera. *Current biology*, 27(7):1013–1018, 2017. ISSN 0960-9822. doi: 10.1016/j.cub.2017.01.027.

Mark Pinsky and Samuel Karlin. *An Introduction to Stochastic Modeling*. Academic Press, November 2010. ISBN 978-0-12-381417-3.

Manuel Plate, Richard Bernstein, Andreas Hoppe, and Kaspar Bienefeld. The importance of controlled mating in honeybee breeding. *Genetics Selection Evolution*, 51(1):74, December 2019. ISSN 1297-9686. doi: 10.1186/s12711-019-0518-y. URL https://gsejournal.biomedcentral.com/articles/10.1186/s12711-019-0518-y.

Torsten Pook, Martin Schlather, and Henner Simianer. MoBPS - Modular Breeding Program Simulator. *G3 Genes\textbarGenomes\textbarGenetics*, 10(6):1915–1918, June 2020. ISSN 2160-1836. doi: 10.1534/g3.120.401193. URL https://doi.org/10.1534/g3.120.401193.

H. K. Poole. The Wall Structure of the Honey Bee1 Spermatheca with Comments about Its Function2. *Annals of the Entomological Society of America*, 63(6):1625–1628, November 1970. ISSN 0013-8746. doi: 10.1093/aesa/63.6.1625. URL https://doi.org/10.1093/aesa/63.6.1625.

Simon G. Potts, Jacobus C. Biesmeijer, Claire Kremen, Peter Neumann, Oliver Schweiger, and William E. Kunin. Global pollinator declines: trends, impacts and drivers. *Trends in Ecology & Evolution*, 25(6):345–353, June 2010. ISSN 0169-5347. doi: 10.1016/j.tree.2010.01.007. URL https://www.cell.com/trends/ecology-evolution/abstract/S0169-5347(10)00036-4.

JE Pryce, ME Goddard, HW Raadsma, and BJ Hayes. Deterministic models of breeding scheme designs that incorporate genomic selection. *Journal of Dairy Science*, 93(11):5455–5466, 2010.

Shaun Purcell, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, Manuel A. R. Ferreira, David Bender, Julian Maller, Pamela Sklar, Paul I. W. de Bakker, Mark J. Daly, and Pak C. Sham. PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. *The American Journal of Human Genetics*, 81(3):559–575, September 2007. ISSN 0002-9297, 1537-6605. doi: 10.1086/519795. URL https://www.cell.com/ajhg/abstract/S0002-9297(07)61352-4.

Ratnieks and Carreck. Clarity on Honey Bee Collapse? \textbar Science. 2010. URL https://www.science.org/doi/full/10.1126/science.1185563?casa_token=f1SnODe6st8AAAAA%3A8T3PuNTrcr2JqiXn-Wq55-YfFuD5-5-AQdYwhLV25nrMXi5EPSc8XkKAhcuDfR8qouORitLQsYSDng.

Francis L. W. Ratnieks and Laurent Keller. Queen control of egg fertilization in the honey bee. *Behavioral Ecology and Sociobiology*, 44(1):57–61, October 1998. ISSN 1432-0762. doi: 10.1007/s002650050514. URL https://doi.org/10.1007/s002650050514.

Silvia C. Remolina and Kimberly A. Hughes. Evolution and mechanisms of long life and high fertility in queen honey bees. *AGE*, 30(2):177–185, September 2008. ISSN 0161-9152, 1574-4647. doi: 10.1007/s11357-008-9061-4. URL http://link.springer.com/10.1007/s11357-008-9061-4.

John-Kaarli Rentof, Caleigh Chamberlain, and Nicholas M Sard. Evaluating factors that affect full-and half-sibling inferences via genetic pedigree reconstruction without parental genotypes. *Journal of Great Lakes Research*, 50(2):102282, 2024.

Fabrice Requier, Lionel Garnery, Patrick L Kohl, Henry K Njovu, Christian WW Pirk, Robin M Crewe, and Ingolf Steffan-Dewenter. The conservation of native honey bees is crucial. *Trends in ecology & evolution*, 34(9):789–798, 2019a.

Fabrice Requier, Quentin Rome, Guillaume Chiron, Damien Decante, Solène Marion, Michel Menard, Franck Muller, Claire Villemant, and Mickaël Henry. Predation of the invasive asian hornet affects foraging activity and survival probability of honey bees in western europe. *Journal of pest science*, 92:567–578, 2019b.

Brian D. Ripley. *Stochastic Simulation*. John Wiley & Sons, September 2009. ISBN 978-0-470-31738-9.

WILLIAM C. Roberts. Multiple mating of queen bees proved by progeny and flight tests. *Gleanings in Bee Culture*, 72(6):225–59, 1944.

P Rosenkranz, P Aumeier, and B Ziegelmann. Biology and control of Varroa destructor - ScienceDirect, 2010. URL https://www.sciencedirect.com/science/article/pii/S0022201109001906.

T'ai H Roulston and Karen Goodell. The role of resources and risks in regulating wild bee populations. *Annual review of entomology*, 56(1):293–312, 2011.

Olav Rueppell, Osman Kaftanouglu, and Robert E. Page. Honey bee (*Apis* mellifera) workers live longer in small than in large colonies. *Experimental Gerontology*, 44 (6):447–452, June 2009. ISSN 0531-5565. doi: 10.1016/j.exger.2009.04.003. URL https://www.sciencedirect.com/science/article/pii/S0531556509000801.

Olav Rueppell, Ryan Kuster, Katelyn Miller, Bertrand Fouks, Sara Rubio Correa, Juan Collazo, Mananya Phaincharoen, Salim Tingek, and Nikolaus Koeniger. A new metazoan recombination rate record and consistently high recombination rates in the honey bee genus apis accompanied by frequent inversions but not translocations. *Genome biology and evolution*, 8(12):3653–3660, 2016.

Friedrich Ruttner. *Biogeography and taxonomy of honeybees*. Springer Science & Business Media, 1988a.

Friedrich Ruttner. Morphometric Analysis and Classification. In *Biogeography and Taxonomy of Honeybees*, pages 66–78. Springer Berlin Heidel-

berg, Berlin, Heidelberg, 1988b. ISBN 978-3-642-72651-4 978-3-642-72649-1. doi: 10.1007/978-3-642-72649-1_6. URL http://link.springer.com/10.1007/978-3-642-72649-1_6.

Friedrich Ruttner and Gudrun Koeniger. Die Füllung der Spermatheka der Bienenkönigin. *Zeitschrift für vergleichende Physiologie*, 72(4):411–422, December 1971. ISSN 1432-1351. doi: 10.1007/BF00300712. URL https://doi.org/10.1007/BF00300712.

Dylan K Ryals, Amos C Buschkoetter, J Krispn Given, and Brock A Harpur. Individual and social heterosis act independently in honey bee (apis mellifera) colonies. *Journal of Heredity*, page esae043, 2024.

Heli Salmela, Gyan P Harwood, Daniel Münch, Christine G Elsik, Elías Herrero-Galán, Maria K Vartiainen, and Gro V Amdam. Nuclear translocation of vitellogenin in the honey bee (apis mellifera). *Apidologie*, 53(1):13, 2022.

Mehdi Sargolzaei and Flavio S Schenkel. QMSim: a large-scale genome simulator for livestock. *Bioinformatics*, 25(5):680–681, March 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp045. URL https://doi.org/10.1093/bioinformatics/btp045.

Nathan M. Schiff and Walter S. Sheppard. Genetic Analysis of Commercial Honey Bees (Hymenoptera: Apidae) from the Southeastern United States. *Journal of Economic Entomology*, 88(5):1216–1220, October 1995. ISSN 0022-0493. doi: 10.1093/jee/88.5.1216. URL https://doi.org/10.1093/jee/88.5.1216.

Michael K. Schwartz, Gordon Luikart, and Robin S. Waples. Genetic monitoring as a promising tool for conservation and management. *Trends in Ecology & Evolution*, 22(1):25–33, January 2007. ISSN 0169-5347. doi: 10.1016/j.tree.2006.08.009. URL https://www.cell.com/trends/ecology-evolution/abstract/S0169-5347(06)00272-2.

Thomas D. Seeley. Life history strategy of the honey bee, Apis mellifera. *Oecologia*, 32(1):109–118, January 1978. ISSN 1432-1939. doi: 10.1007/BF00344695. URL https://doi.org/10.1007/BF00344695.

Thomas D. Seeley and Susannah C. Buhrman. Group decision making in swarms of honey bees. *Behavioral Ecology and Sociobiology*, 45(1):19–31, January 1999. ISSN 1432-0762. doi: 10.1007/s002650050536. URL https://doi.org/10.1007/s002650050536.

Thomas D Seeley and David R Tarpy. Queen promiscuity lowers disease within honeybee colonies. *Proceedings of the Royal Society B: Biological Sciences*, 274 (1606):67–72, September 2006. doi: 10.1098/rspb.2006.3702. URL https://royalsocietypublishing.org/doi/full/10.1098/rspb.2006.3702.

Maria L. Selle, Ingelin Steinsland, Owen Powell, John M. Hickey, and Gregor Gorjanc. Spatial modelling improves genetic evaluation in smallholder breeding programs. *Genetics Selection Evolution*, 52(1):69, November 2020. ISSN 1297-9686. doi: 10.1186/s12711-020-00588-w. URL https://doi.org/10.1186/s12711-020-00588-w.

Shabnam Parichehreh, Gholamhosein Tahmasbi, Alimorad Sarafrazi, Naser Tajabadi, Samaneh Solhjuyi Fard, and Hamed Rezaei. Predicting distribution modeling of Apis florea F. in the world. August 2022. Place: Turkey.

Walter S Sheppard and Marina D Meixner. Apis mellifera pomonella, a new honey bee subspecies from Central Asia. *Apidologie*, 34(4):367–375, 2003. Publisher: EDP Sciences.

Walter S. Sheppard, Thomas E. Rinderer, Julio A. Mazzoli, J. Anthony Stelzer, and Hachiro Shimanuki. Gene flow between African- and European-derived honey bee populations in Argentina. *Nature*, 349(6312):782–784, February 1991. ISSN 1476-4687. doi: 10.1038/349782a0. URL https://www.nature.com/articles/349782a0.

Ana Silva, Jorge de Brito, Pedro Lima Gaspar, Ana Silva, Jorge de Brito, and Pedro Lima Gaspar. Deterministic models. *Methodologies for Service Life Prediction of Buildings: With a Focus on Façade Claddings*, pages 67–162, 2016.

Michael Simone-Finstrom, Megan Walz, and David R. Tarpy. Genetic diversity confers colony-level benefits due to individual immunity. *Biology Letters*, 12(3):20151007, March 2016. doi: 10.1098/rsbl.2015.1007. URL https://royalsocietypublishing.org/doi/full/10.1098/rsbl.2015.1007.

G.P. Slater and B.A. Harpur. Using genomics to predict drone quality: why are there so many 'dud' male honey bees? Rotterdam, The Netherlands, July 2022. URL https://www.wageningenacademic.com/pb-assets/wagen/WCGALP2022/53_005.pdf.

Kristine M. Smith, Elizabeth H. Loh, Melinda K. Rostal, Carlos M. Zambrana-Torrelio, Luciana Mendiola, and Peter Daszak. Pathogens, Pests, and Economics: Drivers of Honey Bee Colony Declines and Losses. *EcoHealth*, 10(4):434–445, December 2013. ISSN 1612-9210. doi: 10.1007/s10393-013-0870-2. URL https://doi.org/10.1007/s10393-013-0870-2.

S P Smith and F R Allaire. Efficient selection rules to increase non-linear merit: application in mate selection (*). *Gen.Sel.Evo*, 1985.

SNHBS. Promoting the conservation, maintenance, breeding and study of the native Scottish honey bee, September 2017. URL https://www.snhbs.scot/.

Doug Speed and David J. Balding. Relatedness in the post-genomic era: is it still useful? *Nature Reviews Genetics*, 16(1):33–44, January 2015. ISSN 1471-0064. doi: 10.1038/nrg3821. URL https://www.nature.com/articles/nrg3821.

Doug Speed, Gibran Hemani, Michael R. Johnson, and David J. Balding. Improved Heritability Estimation from Genome-wide SNPs. *The American Journal of Human Genetics*, 91(6):1011–1021, December 2012. ISSN 00029297. doi: 10.1016/j.ajhg.2012.10.010. URL https://linkinghub.elsevier.com/retrieve/pii/S0002929712005332.

A. L. St. Clair, G. Zhang, A. G. Dolezal, M. E. O'Neal, and A. L. Toth. Diversified Farming in a Monoculture Landscape: Effects on Honey Bee Health and

Wild Bee Communities, 2020. URL https://academic.oup.com/ee/article/49/3/753/5816328.

Nathalie Steinhauer, Kelly Kulhanek, Karina Antúnez, Hannelie Human, Panuwan Chantawannakul, Marie-Pierre Chauzat, and Dennis vanEngelsdorp. Drivers of colony losses. *Current Opinion in Insect Science*, 26:142–148, April 2018. ISSN 2214-5745. doi: 10.1016/j.cois.2018.02.004. URL https://www.sciencedirect.com/science/article/pii/S2214574517302080.

Jane C. Stout and Carolina L. Morales. Ecological impacts of invasive alien species on bees. *Apidologie*, 40(3):388–409, May 2009. ISSN 0044-8435, 1297-9678. doi: 10.1051/apido/2009023. URL http://dx.doi.org/10.1051/apido/2009023.

Alfio Strano, Teodora Stillitano, Anna Irene De Luca, Giacomo Falcone, and Giovanni Gulisano. Profitability Analysis of Small-Scale Beekeeping Firms by Using Life Cycle Costing (LCC) Methodology. *American Journal of Agricultural and Biological Sciences*, 10(3):116–127, March 2015. ISSN 1557-4989. doi: 10.3844/ajabssp.2015.116.127. URL http://thescipub.com/abstract/10.3844/ajabssp.2015.116.127.

Simona Sušnik, Peter Kozmus, Janez Poklukar, and Vladimir Meglic. Molecular characterisation of indigenous apis mellifera carnica in slovenia. *Apidologie*, 35(6):623–636, 2004.

S. Taber. Bee behavior: determining resistance to brood diseases. *American Bee Journal*, 122(6):422–425, 1982. ISSN 0002-7626. URL https://scholar.google.com/scholar_lookup?title=Bee+behavior%3A+determining+resistance+to+brood+diseases+%5BAmerican+foulbrood%2C+genetic+aspects%5D&author=Taber%2C+S.&publication_year=1982.

David R. Tarpy. Genetic diversity within honeybee colonies prevents severe infections and promotes colony growth. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1510):99–103, January 2003. doi: 10.1098/rspb.2002.2199. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspb.2002.2199.

David R. Tarpy, Dennis vanEngelsdorp, and Jeffrey S. Pettis. Genetic diversity affects colony survivorship in commercial honey bee colonies. *Naturwissenschaften*, 100(8):723–728, August 2013. ISSN 1432-1904. doi: 10.1007/s00114-013-1065-y. URL https://doi.org/10.1007/s00114-013-1065-y.

David R. Tarpy, Joel R. Caren, and Deborah A. Delaney. Meta-analysis of genetic diversity and intercolony relatedness among reproductives in commercial honey bee populations. *Frontiers in Insect Science*, 3, January 2023. ISSN 2673-8600. doi: 10.3389/finsc.2023.1112898. URL https://www.frontiersin.org/articles/10.3389/finsc.2023.1112898.

Beti Thompson, Gloria D. Coronado, Julia E. Grossman, Klaus Puschel Md, Cam C. Solomon, Ilda Islas, Cynthia L. Curl, Jeffry H. Shirai, John C. Kissel, and Richard A. Fenske. Pesticide Take-Home Pathway among Children of Agricultural Workers: Study Design, Methods, and Baseline Findings. *Journal of Oc-*

cupational and Environmental Medicine, 45(1):42, January 2003. ISSN 1076-2752. URL https://journals.lww.com/joem/abstract/2003/01000/pesticide_take_home_pathway_among_children_of.12.aspx.

EA Thompson. Likelihood and parsimony: Comparison of criteria and solutions. *Cladistics*, 2(1):43–52, 1986. Publisher: Wiley Online Library.

B. Tier and J. Sölkner. Analysing gametic variation with an animal model. *Theoretical and Applied Genetics*, 85(6):868–872, February 1993. ISSN 1432-2242. doi: 10.1007/BF00225031. URL https://doi.org/10.1007/BF00225031.

A. Uzunov, E. W. Brascamp, and R. Büchler. The Basic Concept of Honey Bee Breeding Programs. *Bee World*, 94(3):84–87, July 2017. ISSN 0005-772X, 2376-7618. doi: 10.1080/0005772X.2017.1345427. URL https://www.tandfonline.com/doi/full/10.1080/0005772X.2017.1345427.

JA Van Arendonk, Bruce Tier, and Brian P Kinghorn. Use of multiple genetic markers in prediction of breeding values. *Genetics*, 137(1):319–329, 1994. Publisher: Oxford University Press.

Paul M. VanRaden. Efficient methods to compute genomic predictions. *Journal of dairy science*, 91(11):4414–4423, 2008.

PM VanRaden. Genomic measures of relationship and inbreeding. *INTERBULL bulletin*, (37):33–33, 2007.

Julio G. Velazco, Marcos Malosetti, Colleen H. Hunt, Emma S. Mace, David R. Jordan, and Fred A. van Eeuwijk. Combining pedigree and genomic information to improve prediction quality: an example in sorghum. *Theoretical and Applied Genetics*, 132 (7):2055–2067, July 2019. ISSN 1432-2242. doi: 10.1007/s00122-019-03337-w. URL https://doi.org/10.1007/s00122-019-03337-w.

José Marcelo Soriano Viana. Dominance, epistasis, heritabilities and expected genetic gains. *Genetics and Molecular Biology*, 28:67–74, 2005. Publisher: SciELO Brasil.

Victoria Viert, Jakob Wegener, and Kaspar Bienefeld. Europe's first gene bank for honey bees. *Bee World*, 98(4):110–114, 2021.

Peter M. Visscher, Sarah E. Medland, Manuel A. R. Ferreira, Katherine I. Morley, Gu Zhu, Belinda K. Cornes, Grant W. Montgomery, and Nicholas G. Martin. Assumption-free estimation of heritability from genome-wide identity-by-descent sharing between full siblings. *PLoS genetics*, 2(3):e41, 2006.

Andreas Wallberg, Fan Han, Gustaf Wellhagen, Bjørn Dahle, Masakado Kawata, Nizar Haddad, Zilá Luz Paulino Simões, Mike H Allsopp, Irfan Kandemir, Pilar De la Rúa, Christian W Pirk, and Matthew T Webster. A worldwide survey of genome sequence variation provides insight into the evolutionary history of the honeybee Apis mellifera. *Nature Genetics*, 46(10):1081–1088, October 2014. ISSN 1061-4036, 1546-1718. doi: 10.1038/ng.3077. URL http://www.nature.com/articles/ng.3077.

Jinliang Wang. Monitoring and managing genetic variation in group breeding populations without individual pedigrees. *Conservation Genetics*, 5(6):813–825, Novem-

ber 2004a. ISSN 1572-9737. doi: 10.1007/s10592-004-1982-6. URL https://doi.org/10.1007/s10592-004-1982-6.

Jinliang Wang. Sibship Reconstruction From Genetic Data With Typing Errors. *Genetics*, 166(4):1963–1979, April 2004b. ISSN 1943-2631. doi: 10.1093/genetics/166.4.1963. URL https://doi.org/10.1093/genetics/166.4.1963.

Jinliang Wang. Pedigrees or markers: Which are better in estimating relatedness and inbreeding coefficient? *Theoretical Population Biology*, 107:4–13, February 2016. ISSN 0040-5809. doi: 10.1016/j.tpb.2015.08.006. URL https://www.sciencedirect.com/science/article/pii/S0040580915000842.

Jinliang Wang. Pedigree reconstruction from poor quality genotype data. *Heredity*, 122(6):719–728, June 2019. ISSN 1365-2540. doi: 10.1038/s41437-018-0178-7. URL https://www.nature.com/articles/s41437-018-0178-7.

Zhiheng Wang and Roger Ghanem. Stochastic modeling and statistical calibration with model error and scarce data. *Computer Methods in Applied Mechanics and Engineering*, 416:116339, 2023.

Zhuoshi Wang, Harmen Doekes, and Piter Bijma. Towards genetic improvement of social behaviours in livestock using large-scale sensor data: data simulation and genetic analysis. *Genetics Selection Evolution*, 55(1):67, 2023.

Zilong Wang, Zhiyong Liu, Xiaobo Wu, Weiyu Yan, and Zhijiang Zeng. Polymorphism analysis of csd gene in six apis mellifera subspecies. *Molecular Biology Reports*, 39: 3067–3071, 2012.

Bruce S. Weir, Amy D. Anderson, and Amanda B. Hepler. Genetic relatedness analysis: modern data and new challenges. *Nature Reviews Genetics*, 7(10):771–780, October 2006. ISSN 1471-0064. doi: 10.1038/nrg1960. URL https://www.nature.com/articles/nrg1960.

Christian R. Werner, R. Chris Gaynor, Daniel J. Sargent, Alessandra Lillo, Gregor Gorjanc, and John M. Hickey. Genomic selection strategies for clonally propagated crops. preprint, Genetics, June 2020. URL http://biorxiv.org/lookup/doi/10.1101/2020.06.15.152017.

Rick Wertenbroek, Robin J Hofmeister, Ioannis Xenarios, Yann Thoma, and Olivier Delaneau. Improving population scale statistical phasing with whole-genome sequencing data. *PLoS genetics*, 20(7):e1011092, 2024. Publisher: Public Library of Science San Francisco, CA USA.

Andrew Whalen and John M Hickey. AlphaImpute2: fast and accurate pedigree and population based imputation for hundreds of thousands of individuals in livestock populations. *BioRxiv*, pages 2020–09, 2020. Publisher: Cold Spring Harbor Laboratory.

Andrew Whalen, Roger Ros-Freixedes, David L. Wilson, Gregor Gorjanc, and John M. Hickey. Hybrid peeling for fast and accurate calling, phasing, and imputation with sequence data of any coverage in pedigrees. *Genetics Selection Evolution*, 50(1):67,

December 2018. ISSN 1297-9686. doi: 10.1186/s12711-018-0438-2. URL https://gsejournal.biomedcentral.com/articles/10.1186/s12711-018-0438-2.

Andrew Whalen, Gregor Gorjanc, and John M. Hickey. Parentage assignment with genotyping-by-sequencing data. *Journal of Animal Breeding and Genetics*, 136 (2):102–112, 2019. ISSN 1439-0388. doi: 10.1111/jbg.12370. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/jbg.12370.

Yvonne C J Wientjes, Piter Bijma, Jérémie Vandenplas, and Mario P L Calus. Multi-population Genomic Relationships for Estimating Current Genetic Variances Within and Genetic Correlations Between Populations. *Genetics*, 207(2):503–515, October 2017. ISSN 1943-2631. doi: 10.1534/genetics.117.300152. URL https://doi.org/10.1534/genetics.117.300152.

Edward O Wilson. The evolution of caste systems in social insects. *Proceedings of the American Philosophical Society*, 123(4):204–210, 1979. Publisher: JSTOR.

Victoria A Wojcik, Lora A Morandin, Laurie Davies Adams, and Kelly E Rourke. Floral resource competition between honey bees and wild bees: is there clear evidence and can we guide management and conservation? *Environmental entomology*, 47(4): 822–833, 2018.

Mathew E. Wolak. nadiv: an R package to create relatedness matrices for estimating non-additive genetic variances in animal models. *Methods in Ecology and Evolution*, 2012.

John Woolliams and Miguel Toro. What is genetic diversity? Brill, October 2007. doi: 10.3920/9789086865925_004. URL https://brill.com/edcollchap/book/9789086865925/BP000004.xml.

Theresa C. Wossler, Georgina E. Jones, Michael H. Allsopp, and Randall Hepburn. Virgin Queen Mandibular Gland Signals of Apis mellifera capensis Change with Age and Affect Honeybee Worker Responses. *Journal of Chemical Ecology*, 32(5): 1043–1056, May 2006. ISSN 1573-1561. doi: 10.1007/s10886-006-9053-8. URL https://doi.org/10.1007/s10886-006-9053-8.

J. Woyke. Drone Larvae from Fertilized Eggs of the Honeybee. *Journal of Apicultural Research*, 2(1):19–24, January 1963. ISSN 0021-8839. doi: 10.1080/00218839.1963.11100052. URL https://doi.org/10.1080/00218839.1963.11100052.

J Woyke. Effect of Sex Allele Homo-Heterozygosity on Honeybee Colony Populations and on their Honey Production I. Favourable Development Conditions and Unrestricted Queens. *Journal of Apicultural Research*, 19(1):51–63, January 1980. ISSN 0021-8839. doi: 10.1080/00218839.1980.11099997. URL https://doi.org/10.1080/00218839.1980.11099997.

Jerzy Woyke. Natural and Artificial Insemination of Queen Honeybees. *Bee World*, 43 (1):21–25, March 1962. ISSN 0005-772X, 2376-7618. doi: 10.1080/0005772X.1962.11096922. URL https://www.tandfonline.com/doi/full/10.1080/0005772X.1962.11096922.

Sewall Wright. Systems of Mating. I. the Biometric Relations between Parent and Offspring. *Genetics*, 6(2):111–123, March 1921. ISSN 0016-6731. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1200501/.

Jian Yang, Beben Benyamin, Brian P. McEvoy, Scott Gordon, Anjali K. Henders, Dale R. Nyholt, Pamela A. Madden, Andrew C. Heath, Nicholas G. Martin, Grant W. Montgomery, Michael E. Goddard, and Peter M. Visscher. Common SNPs explain a large proportion of the heritability for human height. *Nature Genetics*, 42(7):565–569, July 2010. ISSN 1546-1718. doi: 10.1038/ng.608. URL https://www.nature.com/articles/ng.608.

Sihai Yang, Long Wang, Ju Huang, Xiaohui Zhang, Yang Yuan, Jian-Qun Chen, Laurence D. Hurst, and Dacheng Tian. Parent–progeny sequencing indicates higher mutation rates in heterozygotes. *Nature*, 523(7561):463–467, July 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14649. URL http://www.nature.com/articles/nature14649.

Joanna Zareba, Pawel Blazej, Agnieszka Laszkiewicz, Lukasz Sniezewski, Michal Majkowski, Sylwia Janik, and Malgorzata Cebrat. Uneven distribution of complementary sex determiner (csd) alleles in Apis mellifera population. *Scientific reports*, 7 (1):2317, 2017.

Amro Zayed. Bee genetics and conservation. *Apidologie*, 40(3):237–262, 2009. ISSN 0044-8435.

Amro Zayed, Laurence Packer, Jennifer C. Grixti, Luisa Ruz, Robin E. Owen, and Haroldo Toro. Increased genetic differentiation in a specialist versus a generalist bee: implications for conservation. *Conservation Genetics*, 6(6):1017–1026, February 2006. ISSN 1566-0621, 1572-9737. doi: 10.1007/s10592-005-9094-5. URL http://link.springer.com/10.1007/s10592-005-9094-5.