

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Probabilistic Graphical Inference of Pedigrees

Permalink

<https://escholarship.org/uc/item/8cw1r5j5>

Author

Ng, Thomas

Publication Date

2021

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-ShareAlike License, available at <https://creativecommons.org/licenses/by-sa/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

PROBABILISTIC GRAPHICAL INFERENCE OF PEDIGREES

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

BIOMOLECULAR ENGINEERING & BIOINFORMATICS

by

Thomas C. Ng

March 2021

The Dissertation of Thomas C. Ng
is approved:

Professor Josh Stuart, Chair

Professor Eric C. Anderson

Professor Russell Corbett-Detig

Professor Lise Getoor

Quentin Williams
Acting Vice Provost and Dean of Graduate Studies

Copyright © by

Thomas C. Ng

2021

Table of Contents

List of Figures	v
List of Tables	vii
Abstract	viii
Dedication	xi
Acknowledgments	xii
1 Background	1
1.1 The Importance Of Pedigrees	1
1.2 Relationship Inference: From Parentage Inference To Multi-generational Pedigree Inference	4
1.3 Likelihood Calculations on Pedigrees	6
1.4 The Pedigree Factor Graph	8
1.5 Overview of Dissertation Chapters	11
2 Learning Acyclic Multigenerational Pedigrees	13
2.1 Motivation	15
2.2 Methods	16
2.2.1 Overview of Computational Method	16
2.2.2 General Pedigree Notation	17
2.2.3 Pedigree Factor Graph	21
2.2.4 The Sum-Product Algorithm on General Factor Graphs	24
2.2.5 Message Propagation and Joint Likelihood Calculation for Pedigree Factor Graphs	28
2.2.6 The Pedigree Prior Model, $P(\mathcal{P})$	33
2.2.7 Sampling from the Space of Pedigree Factor Graphs	38
2.2.8 Convergence and Mixing in MCMC	49

2.2.9	Data and Software Implementation	57
2.2.10	Evaluating the performance of pedFac	59
2.3	Simulations and Results	63
2.3.1	Scenario 1: Inference of Parentage and Full-sibling groups in a Chinook Salmon Pedigree	63
2.3.2	Scenario 2: Parentage And Sibship Inference of Two Generation Acyclic Pedigrees with Polygamous Mating	71
2.3.3	Scenario 3: Inference of Multi-Generation Acyclic Pedigrees	79
2.4	Discussion	87
2.4.1	Multigenerational pedigrees	87
2.4.2	The assumption that markers are independent	89
2.4.3	Mixing of the MCMC Chain	91
2.4.4	Limitations	92
3	Learning Multigenerational Pedigrees that Contain Cycles	94
3.1	The Problem with Loops in Pedigree Inference, and Evaluation of Possible Solutions	97
3.2	Loops in Pedigree Factor Graphs	104
3.2.1	Types of Loops	104
3.2.2	The Reduced Cycle Basis of a Loopy Pedigree	107
3.3	Conditioning upon the Genotypes of Loop-Breakers to Render a Cyclic Pedigree Factor Graph Acyclic	115
3.3.1	Likelihood Calculation Conditioning on Loop Breakers	119
3.3.2	Pedigree Reconfiguration Samplers that Introduce Loops into Pedigrees	120
3.3.3	Relocating Loop-breakers and Updating Their Fixed Genotype Values	130
3.4	Results	138
3.4.1	Grandparentage Inference of cyclic Pedigrees	138
3.5	Discussion	142
3.5.1	Mixing strategies	143
3.5.2	A Pairwise Sampler for Loop-Breaker Shuffling	144
4	Visually guided curation of microhaplotypes from short read sequence data	147
4.1	Summary	147
4.2	Introduction	148
4.3	Microhaplotypes	155
4.4	Extracting microhaplotypes from sequence reads	156
4.5	Visualization of Microhaplotypes	157

List of Figures

2.1	An example of a two-generation pedigree	19
2.2	A two-generation pedigree factor graph	21
2.3	First message propagation step through an example two-generation pedigree factor graph	28
2.4	A snapshot of the intermediate steps during the sum-product algorithm	32
2.5	A pedigree example of sex inference problem	35
2.6	An example of transforming birth years into generation levels.	37
2.7	An example of interchangeable pedigrees given a monogamous and unobserved parent pair within a multigeneration pedigree	51
2.8	Various “swapping” proposal moves performed on a multigenerational pedigree	53
2.8	Various “swapping” proposal moves performed on a multigenerational pedigree (cont.)	54
2.8	Various “swapping” proposal moves performed on a multigenerational pedigree (cont.)	55
2.9	Examples of swapping and substitution moves	58
2.10	Parentage assignment for the most recent cohort of the empirical Chinook salmon pedigree genotyped for 95 SNPs	65
2.11	Accuracy of inference of full-sibling pairs in the monogamous case study	69
2.12	Performance in inference of full sibling groups	72
2.13	Simulated pedigree with 29 connected components used to explore inference for polygamous scenarios	74
2.14	Accuracy of inference of half-sibling pairs in the polygamous case study	75
2.15	Simulated five-generation acyclic monogamous pedigree under incomplete sampling.	80
2.16	Result of multigeneration pedigree inference with incomplete sampling of individuals, and 200 markers.	81

2.17	Simulated three-generation acyclic monogamous pedigrees with only the females sampled	84
2.18	Accuracy of inference of grandmothers in the three-generation case study	85
2.19	pedFac’s log posterior chain under 100 sweeps for scenario 2.	90
3.1	A simple example of “message gridlock” in a pedigree with a cycle	99
3.2	Three pedigrees, each containing a single inbreeding loop	105
3.3	Marriage chains and loops	107
3.4	Disjointed and intersecting cycle bases in pedigrees	108
3.5	Circle and arc representation of loops in pedigrees	111
3.6	An instance of how an <i>rcb</i> is identified through the modified DFS algorithm	114
3.7	Consequences of the choice of loop-breakers	118
3.8	Conditioning upon a fixed genotype value of a variable node: the creation of <i>clones</i>	121
3.9	Legend for the graphical shorthands used in describing the joint likelihood calculation scheme	123
3.10	Removing the conditioning status of variable nodes under edge removal occurring in the initial “pruning” stage to propose a new configuration.	127
3.11	The “fresh-start” parentage sampler	131
3.12	The “spouse-shuffler” parentage sampler-I	132
3.13	The “spouse-shuffler” parentage sampler-II	133
3.14	The “spouse-shuffler” parentage sampler-III	134
3.15	The “family-expansion pack” sampler	135
3.16	Three-generation cyclic pedigree of 335 individuals with various mating patterns	139
3.17	Grandparentage assignment for 101 individuals in a simulated cyclic pedigree.	141
3.18	An illustrated instance of the pairwise loop-breaker shuffler.	146
4.1	Screenshots of microhaplotype summary plots from <code>microhaplot</code> ’s shiny app	158

List of Tables

2.1	Probabilities derived from Mendel’s laws for all possible values.	31
2.2	Comparison of parentage precision (1 - FDR) rates and associated ROC AUC scores from 1157 individuals in four varying sample-fraction scenarios of a two-generation monogamous pedigree.	66
2.3	Error rates, ROC-AUC scores and runtime for inferred full sibling pair from the monogamous Chinook salmon pedigree of scenario 1.	67
2.4	Error rates and ROC-AUC scores for full-sibling group (rather than pair) inferences from scenario 1	70
2.5	Comparison of parentage precision (1 - FDR) rates and associated ROC AUC scores from 428 individuals in four varying sample-fraction scenarios of a two-generation polygamous pedigree	76
2.6	Error rates, ROC-AUC scores and runtime for inferred full and half sibling pair from the polygamous pedigree of scenario 2	77
2.7	Error rates and ROC-AUC scores for full-sibling group (rather than pair) inferences from scenario 2	78
2.8	Comparison of grand-parentage precision (1 - FDR) rates with varying number of physically linked SNPs.	83
3.1	Loops occurring in pedigrees from natural populations	96
3.2	Data structures used in detecting cycles and the <i>rcb</i> of a pedigree factor graph.	115
3.3	Comparison of the fraction of correctly assigned grandparents in five sample fraction scenarios of a three-generation cyclic pedigree.	140

Abstract

Probabilistic Graphical Inference of Pedigrees

by

Thomas C. Ng

Inference of pedigrees from genetic data is a fundamental problem in the field of population genetics with many applications, such as studying the inheritance of traits in natural populations, or characterizing the transmission of genetic disorders. Current methods for reconstructing pedigrees can be divided into two categories. The first category includes approaches that infer pedigrees composed of, at most, two generations. The best performing of these approaches—termed parentage or sibship inference procedures—make direct use of the pedigree likelihood given observed genetic data. Methods in the the second category endeavor to infer multi-generational pedigrees; however, to date, these have only been implemented with restrictive assumptions (such as complete sampling) or by employing approximations to the pedigree likelihood (such as composite likelihoods, or *ad hoc* approaches).

In this dissertation, I develop a novel representation of pedigrees as a type of factor graph that allows for the inference of multigenerational pedigrees within a proper, probabilistic framework. The factor-graph representation allows the rapid calculation of the pedigree likelihood under a variety of rearrangements, which provides an efficient mechanism for Metropolis-Hastings simulation of a Markov chain through the space of possible

pedigrees. My software implementing this, pedFac, produces a sample of pedigrees from their posterior distribution, which allows for fully Bayesian, multigenerational pedigree inference.

I show that pedFac performs as well as other state-of-the-art software for inferring two-generation pedigrees, but it also provides a far superior estimate of uncertainty. PedFac is also successful in inferring multigenerational pedigrees when sampling is incomplete. This means that pedFac can reconstruct multiple, true links in pedigrees through unobserved/unsampled individuals, a task not performed well by any other software available today.

PedFac relies on the sum-product algorithm to calculate the full, joint likelihood of a pedigree factor graph. The sum-product algorithm only delivers the exact joint likelihood when the pedigree has no loops. For pedigrees that do have loops, I develop a “conditioning” approach, that permits the likelihood calculation of cyclic pedigrees by conditioning on sampled genotype values over a set of loop breakers. I show that this conditioning approach allows pedFac to successfully sample the pedigree space, whether it be cyclic or acyclic.

Finally, I present work relevant to identifying and scoring genetic markers that could be used as input to pedFac. A decade ago, most SNPs used in molecular ecology were typed singly on specialized chips using a variant of quantitative PCR; however, today short-read technologies are commonly employed to generate genetic data. To genotype

a small number of SNPs or short focal regions in many individuals, molecular ecologists, now, routinely sequence amplicons (short, PCR-amplified regions) on next-generation sequencing machines. These data can be analyzed not just in terms of the SNPs present, but as very short haplotypic variants termed microhaplotypes. I present the R package ‘microhaplot’ to assist in the extraction and curation of these microhaplotypes from short-read amplicon sequence data, and I briefly consider strategies for reducing microhaplotypes to a biallelic representation that can be used as input to pedFac.

This thesis is dedicated to my late mother, Yolanda Cheung, for your boundless
love for me, for family and for science.

Acknowledgments

This thesis would not have been possible without the generous support and mentorship of my advisor, Eric Anderson. His early idea of representing pedigrees as a probabilistic graphical model—namely as a pedigree factor graph—set the groundwork for this project. These foundational ideas have continually inspired me and led me to critically deal with unexpected challenges of other domains (for example the feedback vertex set problem). Without a doubt, it is a privilege and a treat to work on this exciting megapuzzle every day and, more importantly, to work with someone as inspirational and infectiously passionate about research as Eric is. I am deeply grateful to Eric for his support and guidance, the writing workshops he provided, the open and constructive feedback, and the financial support to complete this research. Eric, I cannot thank you enough.

I want to thank the current and former Southwest fisheries MEGA lab group members and friends. They have been kind, joyful, and knowledgeable. This inviting community has truly made the lab a second home for me. I will always treasure the memories of frequent interactive out-of-the-box lab presentations, fantastic baked goods made by Ellen Campbell, the nerdy exchange in cubicle-land, rounds of Boggle games with Cassie Columbus and the gals over the lunch break, and a memorable scavenger hunt led by Elena Correa. I want to thank Anthony Clemento, Libby Gilbert, Hayley Nuetzel, Devon Pearse, and Kerry Reid for providing insights and experience in using pedigree

inference software and the applications derived from inferred results. Most importantly, the lab would not be as prolific and productive without the lead director John Carlos Garza. I am grateful for his sharp insights, impeccable scientific standards, and encouragement through the development of this project. He has been instrumental in keeping me afloat with funding and connections to other exciting marine genetic projects.

I want to give special thanks to the current and past members of my thesis advisory committee: Russ Corbett, Lise Getoor, Ed Green, Beth Shapiro, and Josh Stuart for your time, feedback, and perspectives that were much needed to expand my work.

I want to give thanks to colleagues and friends, all of whom have motivated me to push on forward, especially during difficult times, by being exemplary scholars and wonderful human beings: Eldridge Alcantara, Prateek Arora, Sienna Ballou, Adrian Bivol, Robin Buck, Brian Chow, Kylie Graim, Kayla Giang, Brian Lin, Andrew Liu, Akshar Lohith, Charlie Markello, Sulakshana Mukherjee, Yulia Newton, Tom Pham, Kristen Ruegg, Dominik Safranek, Nedda Saremi, Robert Shelansky, Jerry Steele, Max Tabachnik and many more.

Sincere thanks to the Bio-molecular Engineering grad community and faculty and staff from the Bio-molecular Engineering Department, Statistical Science Department, and Theater Art and Dance Department.

Lastly, I want to thank my parents and siblings for their unconditional trust, understanding, and mostly everything.

Chapter 1

Background

1.1 The Importance Of Pedigrees

Pedigrees are familiar to most people as a record of familial lineage connecting offspring in one generation to their parents in the previous generation, and those parents to their parents in a further previous generation, and so forth. While the true pedigree of an individual extends back in time indefinitely, in practice, many pedigrees will be recorded only for a finite number of generations. Ancestors found at the top of the pedigree are termed founders and are typically assumed to be “unrelated.” With that assumption, the pedigree provides everything needed to compute the expected degree of relatedness, (e.g., genome sharing), between any individuals (or subsets of individuals) from the pedigree. For this reason, pedigrees have played a central role in the development of the fields of population genetics (Wright 1922; Thompson 2000; Aykanat et al. 2014;

Nielsen et al. 2001; Meagher and Thompson 1986), and quantitative genetics (Lynch et al. 1998; Almasy and Blangero 1998), and have found multiple applications in a number of different fields.

In human-genetic or animal-management studies, the pedigree of a sample of individuals is usually known from birth records, breeding experiment designs, or observations of mating. For example, pedigrees are carefully recorded and used for selecting candidates in domestic breeding programs of cattle (Goddard and Hayes 2009), sheep (Goyache et al. 2003), and crocodiles (Isberg et al. 2004). In medical genetics, there is a long history of combining known pedigrees with genotype and phenotype data to analyze segregation patterns of genetic diseases and to conduct linkage mapping of genetic variants that contribute to disease susceptibility (Goddard and Hayes 2009; Abecasis et al. 2000; Almasy and Blangero 1998). In animal and plant studies, known pedigrees with phenotype data are used to calculate genetic variance components and to estimate trait heritabilities. The addition of genetic marker data in this context permits quantitative trait locus (QTL) mapping and genomic selection (Goddard and Hayes 2009).

In natural populations it is usually not possible to directly observe mating events, so the pedigrees connecting members from such populations are typically unknown. However, since the 1980s, genetic markers have provided a useful tool to infer pedigrees within wild populations. These inferred pedigrees from wild populations can be used for the study of quantitative genetics (Lynch et al. 1998), demography (Kruuk and Hill

2008; Pemberton 2008; S. 2003; Creel and Rosenblatt 2013), mating behavior (Ford et al. 2011), migration rate (S. 2003; Vøllestad et al. 2012), and dispersal distance (Meagher and Thompson 1987; Jones 2003), to name a few applications.

The use of genetic markers to study the ecology of natural populations is termed “molecular ecology.” As late as 2003, some molecular ecologists were predicting that microsatellite markers would remain the “marker of choice” for studying relationships in populations (Glaubitz et al. 2003). However, shortly thereafter, the utility of SNPs for pedigree reconstruction was recognized (Anderson and Garza 2006), and today the use of small panels of 100 to 500 SNPs is becoming more commonplace for inferring pedigrees in wild populations (Abadía-Cardoso et al. 2013; Béréños et al. 2014). While human studies today typically use over one million SNPs to estimate genomic relatedness (Weir et al. 2006), and whole genome sequencing is becoming commonplace in molecular ecology, there remain numerous applications in molecular ecology, and fisheries and wildlife management, where pedigrees are inferred from hundreds to thousands of SNPs. The application of next generation sequencing to these types of problems is greatly expanding the scope of such genetic data sets obtained via targeted sequencing (Meek and Larson 2019). Such data sets, composed of hundreds to a few thousand SNPs, is the scale of data targeted by my work.

1.2 Relationship Inference: From Parentage Inference To Multi-generational Pedigree Inference

Pedigree inference involves identifying the familial relationships between the members of a population. While inferring a pedigree gives information of the joint relationship between everyone in the pedigree, most previous work on relationship inference has focused on identifying specific relationship categories such as half- and full-sibling groups (Wang 2004; Thomas and Hill 2002; Wang and Santure 2009), parents and offspring (Meagher and Thompson 1987; Jones and Wang 2010; Anderson and Garza 2006; Wang and Santure 2009), grandparents and grandchildren (Christie et al. 2011), or cousins (Kirkpatrick et al. 2011; Pemberton et al. 2010). Pedigree inference can be approached in two different ways: one can start with a candidate pedigree and make incremental changes over time—the top-down approach—or, initially find parent-offspring pairs and eventually combine those solutions coherently—the bottom-up approach. Regardless of which direction one takes in pedigree inference, all of the observed genotype information and the inferred relationships should be considered jointly since each member in a pedigree is biologically—and hence probabilistically—connected.

Most pedigree inference studies today proceed by first identifying all parent-offspring pairs as a series of separate problems, and then these are built into an inferred pedigree. Parentage assignment is done by comparing the genotype of an offspring to a set of

candidate parent genotypes and then assigning the offspring to a parent on the basis of likelihood or compliance to Mendel's laws of inheritance (Jones and Ardren 2003; Marshall et al. 1998; Meagher and Thompson 1987; Jones and Wang 2010; Anderson and Garza 2006; Wang and Santure 2009). Several parentage inference software packages handle parentage assignment of a single individual adequately; however, only one program, COLONY (Wang and Santure 2009), over the last decade, has been available to pursue parentage inference jointly and in the presence of genotyping error, thus using all the genetic data on everyone to weigh the support for each parentage assignment. Joint use of the data in this context is desirable because it allows better handling of genotyping errors and would permit the inference of relationship between, for example grandparents and grandchildren, even if the intervening parents are missing from the sample. Though COLONY provides a good start to joint use of data for pedigree inference, it is limited to only two generations of data and is based on a maximum likelihood approach that does not provide an accurate measure of uncertainty (Anderson and Ng 2016).

A recent package named sequoia (Huisman 2017) is reported to rapidly reconstruct multigenerational pedigrees with high assignment rate and low error rate given at least 200 independent biallelic markers. Unlike COLONY, sequoia does not consider the joint likelihood over all individuals. Instead, it reconstructs the pedigree sequentially through a pairwise likelihood approach by comparing likelihoods of focal pairs over all

first, second or third order relationships.

Today’s methods for inferring multi-generational pedigrees work by searching for the pedigree with the highest likelihood, either via simulated annealing (Riester et al. 2009; Almudevar 2003), or mixed integer programming (Sheehan et al. 2014). All of these methods make restrictive assumptions. First they assume no genotyping error. Second, they assume a complete sample—essentially assuming that none of the sampled individuals are related through any unsampled individuals. These assumptions are often violated in molecular ecology. In my work, these assumptions are relaxed by explicitly modeling unobserved genotypes and sampling over the space of pedigrees in proportion to their posterior probability via Markov chain Monte Carlo (MCMC) to provide a Bayesian inference of the pedigree. Moreover, all present multi-generation pedigree inference methods work poorly on complex pedigrees, as found in many animal and plant populations that experience inbreeding and high levels of polygamy. A major aspect of this dissertation is the development of a rigorous system for MCMC sampling over the space of such complex or loopy pedigrees in the inference process.

1.3 Likelihood Calculations on Pedigrees

When sampling different pedigree structures from the space of multi-generational pedigrees, it is important to have an efficient method to calculate the probability of the observed genetic data given the pedigree, i.e., the likelihood of the pedigree. Without

an efficient algorithmic approach, the computational cost for calculating the likelihood goes up exponentially as the size of the pedigree grows. There are two basic methods for calculating pedigree likelihoods: the Elston-Stewart algorithm (Elston and Stewart 1971) and the Lander-Green algorithm (Lander and Green 1987). Both of these methods are a specialized form of Baum's algorithm (Baum et al. 1972) or the sum-product algorithm (Kschischang et al. 2001) in which calculations are done on a local unit, marginalizing over all possible states, and then the resulting information gets passed on to neighboring units.

The Elston-Stewart algorithm was the first general algorithm for rapid likelihood pedigree calculation. Each individual's genotype is considered a latent variable that depends on the observed individual's phenotype. The algorithm proceeds sequentially by "peeling" the outer layer of the pedigree, computing the joint marginalized value of the peeled outer branches, and reassigning those values at the peeled site or the pivot points. This algorithm is designed for large pedigrees with only one or a few genetic markers. By contrast, the Lander-Green algorithm relies on a first-order Hidden Markov model across genetic markers that are physically linked upon the same chromosome, taking as latent data an inheritance vector indicating the source of every meiotic transmission in the pedigree. The Lander-Green algorithm works well for a large number of physically linked genetic markers but is only feasible on pedigrees with a small number of individuals. Following the creation of these two methods, many improvements and extensions have

been made; for example, handling of pedigrees with loops (Cannings and Thompson 1977) and a rapid updating scheme (Lange et al. 1983) for the Elston-Stewart algorithm, and memory reduction on storing the transition matrix in the Lander-Green algorithm (Kruglyak et al. 1996).

Both the Elston-Stewart and the Lander-Green algorithms have been applied exclusively in cases where a pedigree is already known. They have not been employed for inferring pedigrees, outside of their use in identifying pedigree errors in medical genetic studies (Sobel et al. 2002). My work extends the Elston-Stewart algorithm, generalized as a sum-product algorithm on factor graphs, allowing it to be used efficiently in the context of pedigree inference with unlinked markers.

1.4 The Pedigree Factor Graph

Pedigrees are often depicted as acyclic, directed graphs (Lauritzen and Sheehan 2003b; Lange and Elston 1974). However, since the complete-data likelihood function of a pedigree is a product of functions, a pedigree can also be depicted as a special type of factor graph called a “pedigree factor graph” (Anderson and Ng 2016). A factor graph is a bipartite graph consisting of variable nodes, factor nodes, and connected edges; each factor node is connected to at least one of the variable nodes and the factor node represents a function that depends on the connected variable nodes (Kschischang et al. 2001; Koller and Friedman 2009). In the context of a pedigree, a variable node denotes

the true genotypic state of an individual, and each of the variable nodes is connected to at least two different types of factor nodes. Following the notations and nomenclature as defined in Anderson and Ng (2016), the factor nodes can be divided into three categories: 1) m -nodes, the marriage nodes, 2) p -nodes, priors (i.e., relative genotype frequencies derived from population allele frequencies) on the genotypes carried by the founders, and 3) g -nodes, observed genotype data. A founder variable node is always connected to a single p -node and a single g -node and for each offspring it contributes to the pedigree, it connects to an m -node shared by its mating partner, the offspring, and all the offspring's full siblings. For a non-founder in the pedigree, the variable node is always connected to its corresponding g -node, one m -node connecting to its parents, and possibly more m -nodes connecting to its offspring and mating partners. The connection between g -nodes and individual variable nodes conveniently allows users to define the genotyping error model for their datasets. In this framework, individual variables, by their nature, are not directly observed; therefore, they can easily be used to represent unsampled individuals, which makes them useful in cases where parents of observed individuals are missing but their relatives are observed.

Although the pedigree factor graph is less compact than a directed graph, the advantages of framing a pedigree as a factor graph include 1) visual transparency in outlining the dependencies between the observed genotype data and prior assumptions, 2) support for efficient algorithms to compute marginal and joint probabilities, and 3) the flexibil-

ity, if desired, to include additional factor nodes that express individual quantitative or categorical data such as phenotype and penetrance.

When a pedigree is represented as a factor graph, the calculation of joint and marginal genotype probabilities can be expressed in terms of the Sum-product algorithm on factor graphs (Kschischang et al. 2001). The sum-product algorithm calculates exact marginal probabilities for all variable nodes when the factor graph is acyclic. It proceeds by passing messages from nodes, along edges, to neighboring nodes. These outgoing messages from a node depend on the incoming messages to that node upon the other edges adjacent to the node. Full details in the context of the pedigree factor graph appear in the next chapter. After one message has been sent in each direction along every edge, the algorithm is complete, and the marginal probabilities of the variable nodes may be calculated as a function of the messages incoming to a given variable node or of the two messages being sent in opposite directions along a single edge connected to the variable node. Additionally, the joint probability of all the genetic data associated with a pedigree factor graph can be calculated as a function of the two messages being sent along any edge of the graph. This latter property makes it possible to rapidly evaluate the likelihood for proposed changes to a pedigree, allowing efficient MCMC sampling over the space of pedigrees in accordance with their posterior probability given genetic data.

1.5 Overview of Dissertation Chapters

In the next two chapters, I present a novel Bayesian inference approach for learning acyclic (Chapter 2) and cyclic (Chapter 3) pedigree factor graphs by sampling pedigrees, via MCMC, from their posterior distribution. This is implemented in a software called pedFac. In the fourth chapter, I present an R package ‘microhaplot’ to assist in the extraction and curation of short haplotypic markers that could be used as input to pedFac.

In “Learning Acyclic Multigenerational Pedigrees” (Chapter 2), I first introduce the notation and formulation for representing a pedigree as a factor graph and then show how this facilitates calculation of the joint likelihood of acyclic pedigrees through the sum-product algorithm. Subsequently, I present a Metropolis-Hastings (Hastings 1970) sampling approach that exploits the modular property of factor graphs to rapidly compute the requisite proposal distributions and acceptance probabilities. I also describe a selection of prior models and mixing strategies to overcome some technical challenges. Lastly, to demonstrate its validity, we compare pedFac’s acyclic sampler to COLONY, FRANz and sequoia in scenarios with two-generation and multi-generation acyclic pedigrees.

By the biological nature of propagation and reproduction, most pedigrees involving large numbers of individuals and multiple generations contain loops. I discuss and

offer a solution, the “conditioning” approach, in “Learning Multigenerational Pedigrees that Contain Cycles” (Chapter 3). I first summarize the different types of loops that can occur in pedigrees. Then I introduce the idea of a “reduced cycle basis” of a factor graph, and show how it is useful for identifying individuals that are potential loop breakers in a cyclic pedigree. Then, I describe an MCMC sampling scheme that involves temporarily assigning values to the latent genotypes of these loop breakers to break the loops, consequently enabling sampling over loopy pedigree structures. This sampling scheme, implemented in pedFac, is tested in the problem of grandparentage inference in a three-generation pedigree case study with varying fractions of missing individuals in the middle generation.

Finally, improper filtering and quality control in the production of genotypic data can unnecessarily lengthen the runtime of pedigree inference software and potentially invite spurious false assignments during the pedigree reconstruction process. To address this need, I present the R package ‘microhaplot’ that extracts and curates genetic markers from short-read amplicon sequence data in “Visually guided curation of microhaplotypes from short read sequence data” (chapter 4). I wrote this R package to ensure quality inputs to inference programs like pedFac. Since its inception it has been used to manage the genotyping of tens of thousands of genetic samples across multiple laboratories.

Chapter 2

Learning Acyclic Multigenerational Pedigrees

Recently, Anderson and Ng (2016) introduced a representation for pedigrees that provides an efficient way to calculate the joint probability of all observed genotypes at biallelic markers upon a pedigree. This framework explicitly includes latent genotypes and captures the statistical dependencies among observed genotypes in a factor graph. It also allows implementation of the sum-product algorithm (Kschischang et al. 2001) to break the full likelihood calculation into smaller, manageable parts. Additionally, the message-passing operations of the sum-product algorithm provide a framework for storing intermediate quantities that make it fast and easy to compute the marginal genotype probabilities of both observed and unobserved genotypes, and to efficiently

recalculate the likelihood of new pedigree configurations proposed by making small changes to the current configuration. All of these operations can be computed in linear time when the putative pedigree contains no loops (i.e., when the factor graph is acyclic).

While Anderson and Ng (2016) introduced the pedigree factor graph framework and described how it could be used to make Bayesian inference of pedigrees from genetic data, the implementation of the method for pedigree inference was rather limited. In particular, the authors implemented only a simple case involving two generations, with parents monogamous but unobserved, i.e., a problem in the inference of full-sibling groups. Anderson and Ng (2016)'s implementation was shown to outperform existing software, COLONY (Wang 2004; Jones and Wang 2010), particularly with reference to assessing uncertainty in the sibling-group assignments. However, Anderson and Ng (2016) left the implementation of a factor-graph-based approach for multigenerational pedigree inference to the future.

In this chapter we provide a complete development of the ideas sketched in Anderson and Ng (2016) for a Markov chain Monte Carlo (MCMC) method to sample acyclic, multigenerational pedigrees from their posterior probability given genotype data from a sample of related individuals. These novel developments are implemented in C and called from an R software program called pedFac.

On simulated data, pedFac provides accurate estimates of acyclic pedigrees in a two-generation context (i.e., parents and offspring), comparing favorably against existing

software packages. Additionally, pedFac can infer multigenerational pedigrees, even when some individuals are missing from the intermediate generations.

2.1 Motivation

The need for a new pedigree inference approach is driven by the failure of currently-available pedigree reconstruction software to accurately infer the relationship of unobserved individuals to other members in a multigenerational pedigree. Currently, the two most advanced pedigree inference programs that are widely used in molecular ecology are COLONY (Wang 2004; Wang and Santure 2009) and FRANz (Riester et al. 2009). With sufficient genetic data, these programs make accurate parentage assignments when most of the parental candidates are observed. Furthermore, when a high number of parents is missing from the dataset, COLONY and FRANz may correctly avoid assigning an individual to any parents if its true parents are absent from the data set. However, if individuals share unobserved parents in the data set, both programs are limited in their capacity to detect these relationships. The model underlying FRANz simply does not allow unobserved individuals to be related to more than one individual in the pedigree. COLONY's model allows for multiple full and half-siblings to be inferred as children of the same unobserved mother and/or father, but COLONY can only be applied to two generations at a time, making it incapable of inferring multigenerational pedigrees.

A more recent software package called sequoia (Huisman 2017), has been developed

that might fill the gap between COLONY and FRANz. Its pedigree inference procedure is based on first finding parent-offspring relationships, then finding sibling relationships, and finally finding grandparents of sibling groups and also avuncular (aunt-uncle) relationships. Each of these steps are performed as somewhat separate, isolated exercises, but the results from all of them are woven back together into a single inferred pedigree. Such an approach should, in theory, allow the inference of links in a multigenerational pedigree through unobserved individuals. Additionally, the paper describing sequoia reports that it is competitive with COLONY on many two-generation inference problems (like parentage and sibship inference).

2.2 Methods

2.2.1 Overview of Computational Method

During every sweep of its MCMC algorithm, pedFac cycles over each individual in the pedigree, starting from the most recent individual to the oldest, and proposes new parental assignments for that focal individual. Each newly proposed parent of that individual can be an individual that already exists in the pedigree (with or without observed genotype data), or can be a newly instantiated individual without observed genotype data. Rather than considering all individuals in the previous generation as potential parental candidates of the focal individual, pedFac allows the option of con-

sidering only a small set of probable parental candidates. In this chapter, the sampling space is restricted to acyclic pedigrees so that the exact probability of the genotype data on the pedigree can be calculated efficiently. The joint probability of all observed genotype data on each newly proposed pedigree is computed using a sum-product algorithm. These probabilities are used in a Metropolis-Hastings framework (in conjunction with the joint genotype probability given the current pedigree configuration) to sample an updated configuration from amongst the proposals. In the following sections we provide more explicit details of this process.

2.2.2 General Pedigree Notation

The notation adopted here follows closely that used in Anderson and Ng (2016). We define the pedigree \mathcal{P} as a network with a number N_o of observed/sampled individuals and N_u unobserved/unsampled individuals. When we say unobserved, we mean that genotype data are not available from the individual, typically because the individual was neither sampled from nor observed in the population.

In theory, if enough unobserved individuals are included, reaching back a sufficient number of generations to include the most recent common ancestors of all observed individuals, then the true pedigree will consist of a single connected component. In practice, however, even with a large amount of genetic data it is difficult to accurately infer relationships through more than a few generations of unobserved pedigree mem-

bers. Thus, in our inference procedure we consider only pedigrees of a finite number T of generations made up of N_f founders and N_{nf} non-founder individuals in which all terminal descendants are observed. The latter condition is enforced by requiring that all individuals at the “bottom” nodes of any proposed \mathcal{P} are observed. A consequence of this condition is that, in any pedigree that is proposed or visited during the course of MCMC, all unobserved individuals in the pedigree must have offspring in the pedigree as well.

Since the pedigree includes, at most, T generations, \mathcal{P} will typically consist of some number $C \geq 1$ of separate, connected components, where $\mathcal{P} = \{\mathcal{P}^{(1)}, \dots, \mathcal{P}^{(C)}\}$.

A key ingredient of the MCMC sampling procedure is the joint probability of the observed genotypes given the pedigree, $p(Y|\mathcal{P})$. Let genotype data be available from N_m biallelic markers assumed to be physically unlinked and not in linkage disequilibrium. Under these assumptions, the probability of genotypes at all N_m markers is the product over markers of the probability at each. Accordingly, we describe the calculations for a single marker, providing an illustration with the simple pedigree containing six individuals below (Figure 2.1) depicted as a marriage node graph (Thompson et al. 1978). In this two-generation pedigree, there are five individuals with observed genotypes (nodes filled gray) and one individual with no observed genotype (the unfilled node). Let $x_{i,\ell}$ denote the true (though never observed without error) latent genotype at marker ℓ of individual $i \in \{1, \dots, 6\}$. We assume, throughout, that all markers are biallelic, with

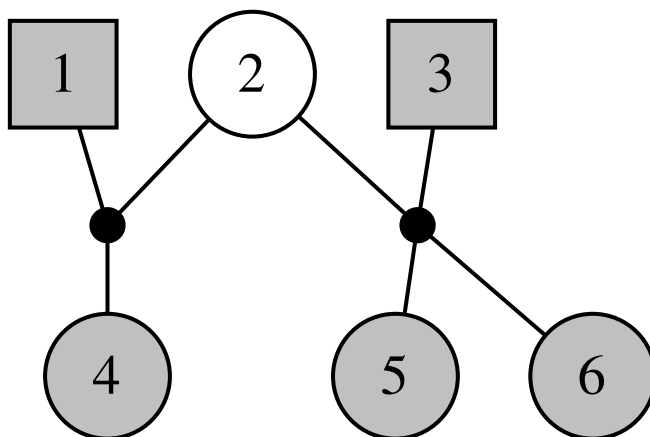


Figure 2.1: This two-generation pedigree is a singly connected component made up of six individuals, with four males and two females. All of them except individual 2 are observed, as denoted by the shading of their nodes. In this scenario, individual 5 and individual 6 are full siblings while individuals 5 and 6 are half siblings of individual 4.

one allele labeled ‘1’ and the other labeled ‘0’. Accordingly, in a diploid organism, the possible values of $x_{i,\ell}$ (i.e., the genotype values) are 0, 1, and 2, each being the number of ‘1’ alleles carried by the individual. Let $y_{i,\ell}$ denote the observed (with error) genotype at marker ℓ in individual $i \in \{1, 3, 4, 5, 6\}$. In each case, the observed genotype $y_{i,\ell}$ is a random variable that depends on the true genotype, $x_{i,\ell}$, and a model for genotyping error parameterized by ϵ . The founders (1, 2, and 3) of the pedigree are assumed to be unrelated to one another, such that the prior probability of each founder’s true genotype at marker ℓ can be obtained from the population allele frequencies θ_ℓ and the assumption of Hardy-Weinberg equilibrium.

The likelihood of \mathcal{P} given θ and ϵ (omitted in following expressions) and the observed

genetic data, is the joint probability of the genetic data given \mathcal{P} , θ , and ϵ . This probability is obtained by summing the joint probability of the observed and latent genotype data (collectively, y and x , respectively) over all possible states of the latent genotypes of all the individuals in \mathcal{P} :

$$l(\mathcal{P}) = p(y|\mathcal{P}) = \sum_x p(x, y|\mathcal{P}).$$

The probability $p(x, y|\mathcal{P})$ consists of a product of probability functions that belong to three general classes:

1. $P(x_{i,\ell}|\theta_\ell)$ is the probability of the latent genotype at marker ℓ within individual i (a founder in the pedigree) given the population allele frequencies θ_ℓ .
2. $P(y_{i,\ell}|x_{i,\ell}, \epsilon)$ is the probability of observing genotype $y_{i,\ell}$ in individual i at marker ℓ given that the true, latent genotype is $x_{i,\ell}$. This depends on the genotyping error model and its parameters, ϵ .
3. $P(x_{i,\ell}|x_{\text{ma},\ell}, x_{\text{pa},\ell})$ is the probability, given the laws of Mendelian inheritance, that an individual i inherits the (latent) genotype $x_{i,\ell}$ from its mother and father who carry the (latent) genotypes $x_{\text{ma},\ell}$, and $x_{\text{pa},\ell}$, respectively.

Thus, the joint probability of x and y , given the pedigree, is:

$$\begin{aligned}
 p(x, y | \mathcal{P}) &= \prod_{\ell=1}^{N_m} p(x_{1,\ell} | \theta_\ell) p(x_{2,\ell} | \theta_\ell) p(x_{3,\ell} | \theta_\ell) \\
 &\times p(x_{4,\ell} | x_{1,\ell}, x_{2,\ell}) p(x_{5,\ell} | x_{2,\ell}, x_{3,\ell}) p(x_{6,\ell} | x_{2,\ell}, x_{3,\ell}) \\
 &\times \prod_{i=1,2,3,5,6} p(y_{i,\ell} | x_{i,\ell}, \epsilon)
 \end{aligned}$$

We will encounter these three general classes of probability functions below, as they constitute the three types of functional classes in a factor graph model.

2.2.3 Pedigree Factor Graph

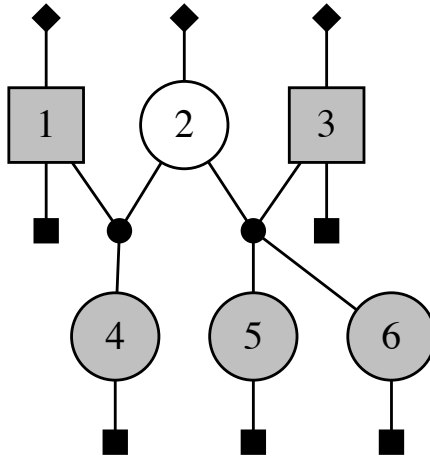


Figure 2.2: A two-generation pedigree factor graph of six individuals. The small, black, filled nodes represent factor nodes, while the numerically labelled nodes are variable nodes. The diamond, square and circle factor nodes represent p -nodes, g -nodes, and m -nodes respectively.

Kschischang et al. (2001) have shown that one can depict a joint probability function

by decomposing it into a network of factors called a factor graph. A factor graph (P, V, E) is a bipartite, undirected graph P made up of edges E and two classes of vertices V : variable nodes and factor nodes. A factor node is a symbolic representation of a function that depends on the variables that the factor node is connected to. In the context of a pedigree, the unobserved true genotypes of the individuals are designated as the variable nodes, which may be connected to three types of factor nodes, which correspond to the three classes of probability functions described in the previous section. The following description of the elements of a pedigree factor graph follows the notation and nomenclature of Anderson and Ng (2016).

In a pedigree factor graph, there are three different types of factor nodes that may be adjacent to the individual-genotype variable nodes: p -nodes, g -nodes and m -nodes. A p -node adjacent to individual i refers to a factor, $h^{(p)}(x_i)$, which is the probability of a founder i 's unobserved genotype given the population allele frequency θ . Thus, for marker ℓ , $h^{(p)}(x_{i,\ell}) \equiv p(x_{i,\ell}|\theta_\ell)$. A g -node adjacent to individual i refers to a factor, $h^{(g)}(x_i, y_i)$, which gives the probability of the observed genotype, y_i , given the underlying true genotype, x_i , and a genotyping error model. Thus, at site ℓ , $h^{(g)}(x_{i,\ell}, y_{i,\ell}) \equiv p(y_{i,\ell}|x_{i,\ell}, \epsilon)$. By default, pedFac uses a simple univariate error model such that the rate

of a genotype mismatch event follows ϵ ; that is,

$$p(y_{i,\ell}|x_{i,\ell}, \epsilon) = \begin{cases} 1 - \epsilon & \text{if } y_{i,\ell} = x_{i,\ell} \\ \epsilon/2 & \text{otherwise} \end{cases} \quad (2.1)$$

The third type of factor node—an m -node—is always shared between three or more variable nodes: a mother, a father, and one or more children. It refers to a factor of the form $h^{(m)}(x_{\text{ma}}, x_{\text{pa}}; x_{\text{kid},1}, \dots, x_{\text{kid},k})$, which represents the probability of the genotypes of the k ($k \geq 1$) children produced by mating between the mother and the father. These probabilities follow directly from the laws of Mendelian inheritance. Figure 2.2 shows the factor graph representation of the pedigree in Figure 2.1 and the joint probability function is expressed as a product of factorized functions as follows:

$$\begin{aligned} p(x, y|\mathcal{P}) &= \prod_{\ell=1}^{N_m} \prod_{i=1,2,3} h^{(p)}(x_{i,\ell}) \prod_{i=1,\dots,6} h^{(g)}(x_{i,\ell}) \\ &\quad \times h^{(m)}(x_{1,\ell}, x_{2,\ell}; x_{4,\ell}) h^{(m)}(x_{2,\ell}, x_{3,\ell}; x_{5,\ell}, x_{6,\ell}) \end{aligned}$$

Technically the m -node function can be further factorized into products of trios as the genotypes of the offspring are conditionally independent given the genotypes of the parents under the law of segregation, e.g., in the pedigree of Figure 2.2),

$$h^{(m)}(x_{2,\ell}, x_{3,\ell}; x_{5,\ell}, x_{6,\ell}) = h^{(m^*)}(x_{2,\ell}, x_{3,\ell}; x_{5,\ell}) h^{(m^*)}(x_{2,\ell}, x_{3,\ell}; x_{6,\ell}).$$

However, we choose to maintain a fully aggregated expression for two main reasons: to prevent the creation of needless cycles in the graph and to preserve a structure similar to the marriage node graph. As described below, however, this additional factorization is still used in the calculation of the sum-product algorithm on a pedigree factor graph.

2.2.4 The Sum-Product Algorithm on General Factor Graphs

The sum-product algorithm is an efficient system that divides the latent variables that must be summed over (to compute a joint probability) small, local groups of variables at which parts of the sum can be taken as manageable local calculation. This process is envisioned as a message-passing scheme operating on a factor graph. The results of local calculations, encoded as messages, are valuable for their use in computing the joint distribution of all observed variables (or the marginal distribution of any single observed variable). However, more importantly, in our case, the messages computed in the sum-product algorithm allow us to rapidly calculate the joint likelihood of new pedigree factor graphs under certain proposed configuration changes. This, in turn, allows us to efficiently sample new pedigree configuration from their posterior distribution, which will provide us a sample from that posterior distribution to use for making inference of the pedigree. In order to describe how the sum-product algorithm is an integral part of this approach to learning pedigree factor graphs, we provide, here, some background on the sum-product algorithm along with the general rules and operations of message

passing on a factor graph.

The sum-product algorithm, also known as belief propagation, is a message-passing algorithm to perform Bayesian inference on a factor graph, as first proposed by Pearl (1982). It applies the principles of dynamic programming to efficiently compute the marginal distribution of each unobserved variable node (i.e., the latent, true genotype of each individual) conditioning on all observed nodes. The algorithm involves an iterative process of passing real-valued functions called messages between adjacent nodes. Upon completion of the algorithm, these message values can also be used to quickly compute the joint likelihood of the factor graph.

In a factor graph, messages are passed from variable node v_i to its adjacent factor node f_j and from factor node f_j to its adjacent variable node v_i . The messages passed from variable nodes to factor nodes $\mu_{v_i \rightarrow f_j}(x_i)$ are computed differently than the messages passed from factor nodes to variable nodes $\mu_{f_j \rightarrow v_i}(x_i)$; however these messages, whether $\mu_{v_i \rightarrow f_j}(x_i)$ or $\mu_{f_j \rightarrow v_i}(x_i)$, share the same form: they all must be non-negative functions of the possible states of the variable x_i . We use $\mathcal{D}(v_i)$ to denote this domain. Recall, since x_i is the latent genotype of a diploid individual, the possible states of x_i at a biallelic genetic marker are 0, 1 or 2, so that $\mathcal{D}(v_i) = \{0, 1, 2\}$. Essentially, in a pedigree factor graph at a single biallelic locus, the messages can be thought of as vectors of length three.

A message from a variable node v_i to a factor node f_j is the product of the messages

sent from the neighboring factor nodes to v_i , excluding f_j :

$$\mu_{v_i \rightarrow f_j}(x_i) = \prod_{f_k \in \eta(v_i) \setminus f_j} \mu_{f_k \rightarrow v_i}(x_i), \quad \forall x_i \in \mathcal{D}(v_i),$$

where $\eta(v_i) \setminus f_j$ denotes the set of all factor nodes adjacent to v_i , excluding f_j .

The calculation in the sum-product algorithm of the message from a factor node f_j to a variable node v_i is more complicated, involving a sum of factor node (function) values, over all possible combinations of the states of the variable nodes adjacent to f_j , excluding v_i , with each term weighted by the product of the various incoming message values. In order to describe this, we introduce some additional notation. First, let $\eta(f_j) \setminus v_i$ denote the set of all neighboring variable nodes adjacent to f_j , excluding v_i . Then let x^* denote a set of values of all the variable nodes in $\eta(f_j) \setminus v_i$. As such, we could expect the function value associated with f_j to be written as $h_j(x^*, x_i)$. Further, let \mathcal{D}^* denote the set of all possible values of x^* (that is, all possible combinations of values at the variable nodes in $\eta(f_j) \setminus v_i$). The message from f_j to v_i is then, in general, calculated for each value, x_i , as follows:

$$\mu_{f_j \rightarrow v_i}(x_i) = \sum_{x^* \in \mathcal{D}^*} h_j(x^*, x_i) \prod_{v_k \in \eta(f_j) \setminus v_i} \mu_{v_k \rightarrow f_j}(x_k), \quad \forall x_i \in \mathcal{D}(v_i). \quad (2.2)$$

If the factor node f_j only depends on one variable node (i.e., if $v_k \in \eta(f_j) \setminus v_i = \emptyset$), then

$$\mu_{f_j \rightarrow v_i}(x_i) = h_j(x_i).$$

As we noted above, the genotypes of the different children of a pair of parents are conditionally independent given the parents' genotypes. While (2.2) shows the general equation for an outgoing message from a factor node, for the marriage factor nodes in a pedigree factor graph, we exploit this additional factorization, as described in the next section.

In both of the message scenarios, the message calculation requires the values of the incoming messages, if any, from the neighboring nodes; therefore, an outgoing message from a node along a specific edge can only be sent so long as incoming messages have been received to the node along all the remaining edges. In an acyclic factor graph, this requirement does not pose a problem, as it is straightforward to verify that each node in an acyclic factor graph will eventually receive the messages needed to send an outgoing message along every edge. This occurs in computational time proportional to twice the length of the longest path of the graph. However, for a factor graph with loops, the standard sum-product algorithm will become stuck waiting for incoming messages that never arrive—a challenge that we address in Chapter 3.

When the sum-product algorithm is applied to a singly-connected pedigree factor graph, it 1) begins with messages being passed from the external factor nodes—the p -nodes adjacent to the founders, and the g -nodes adjacent to all the variable nodes; 2) runs through a finite number of message relaying steps between variable nodes and internal factor nodes (i.e., the m -nodes); and 3) has concluded when messages have been

passed back from all the variable nodes to the adjacent g -node factor nodes, and from the founder variable nodes back to the p -nodes. For the special case of a pedigree factor graph, the next section provides a more in-depth description of the process.

2.2.5 Message Propagation and Joint Likelihood Calculation for Pedigree Factor Graphs

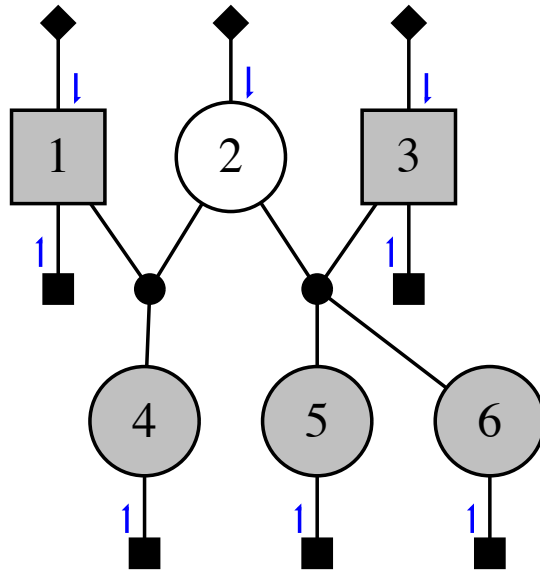


Figure 2.3: First message propagation step through this example two-generation pedigree factor graph. Every external factor node, whether it be a g -node or p -node, is passing an outgoing message (shown as a blue arrow) to its adjacent variable node.

The external nodes of a pedigree factor graph are made up of two types of factor nodes: p -nodes and g -nodes. Each of these external nodes is attached to a single variable node; therefore, the outgoing message from the factor node to its unique variable node

is the factor function itself:

$$\mu_{f_j^{(p)} \rightarrow v_i}(x_i) = h_j^{(p)}(x_i), \forall x_i \in \mathcal{D}(v_i)$$

$$\mu_{f_j^{(g)} \rightarrow v_i}(x_i) = h_j^{(g)}(x_i), \forall x_i \in \mathcal{D}(v_i)$$

Passing these messages from the external factor nodes is the first step of the sum-product algorithm on a factor graph. A visual example of this appears in Figure 2.3.

Each m -node, by contrast, is associated with a joint function of the genotype of the two parents connected to it, and all of their offspring. As noted above, however, this joint probability can be further factorized into a product over trios of functions of each trio, where each trio is defined as the two parents and a particular offspring. This extra factorization can be applied during the sum-product algorithm on a pedigree factor graph to reduce the complexity of the message calculation for the m -nodes. Rather than having to consider all possible genotypic states of the parents and *all* their offspring simultaneously, the possible states of each trio can be considered separately.

For example, if an m -node with multiple progeny is relaying a message to the variable node associated with a particular offspring, the trio-function that involves that offspring and the incoming messages to the parents can be computed separately from the trio-functions of the offspring's siblings. As a consequence, the computation required for passing messages from m -nodes becomes linear, rather than exponential, in the number of offspring, as would be suggested by the factor graph.

In detail, the outgoing message from an m -node, f_j , connected to parental nodes v_p and v_m —father and mother, respectively—to a single one of the progeny, v_i , whilst also considering the information coming in from the remaining offspring connected to f_j , who are denoted by $\eta^{(k)}(f_j) \setminus v_i$, can be computed as follows:

$$\begin{aligned}
\mu_{f_j^{(m)} \rightarrow v_i}(x_i) &= \sum_{x_p^* \in \mathcal{D}(v_p)} \sum_{x_m^* \in \mathcal{D}(v_m)} h^{(m)}(x_p^*, x_m^*, x_i) & (2.3) \\
&\times \mu_{v_p \rightarrow f_j^{(m)}}(x_p^*) \times \mu_{v_m \rightarrow f_j^{(m)}}(x_m^*) \\
&\times \prod_{v_k \in \eta^{(k)}(f_j) \setminus v_i} \sum_{x_k^* \in \mathcal{D}(v_k)} h^{(m)}(x_p^*, x_m^*, x_k^*) \mu_{v_k \rightarrow f_j^{(m)}}(x_k^*),
\end{aligned}$$

for all $x_i \in \mathcal{D}(v_i)$. The domain for any of the variable nodes, i.e., $\mathcal{D}(v_p)$, $\mathcal{D}(v_m)$, $\mathcal{D}(v_k)$, and $\mathcal{D}(v_i)$, contains a total of three discrete states, 0, 1, or 2, representing the possible genotypes of a biallelic marker. The function $h^{(m)}(x_p^*, x_m^*, x_i)$ follows directly from Mendelian segregation, with values listed in Table 2.1.

The factorization applies, as well, to the case of passing a message from the m -node to one of the parental nodes. Here, we describe this specifically for the case of passing a message from an m -node to the paternal node, v_p , connected to it, as in Figure 2.4.

The message in such a case is calculated as follows:

$$\begin{aligned}
\mu_{f_j^{(m)} \rightarrow v_p}(x_p) &= \sum_{x_m^* \in \mathcal{D}(v_m)} \mu_{v_m \rightarrow f_j^{(m)}}(x_m^*) & (2.4) \\
&\times \prod_{v_k \in \eta^{(k)}(f_j)} \sum_{x_k^* \in \mathcal{D}(v_k)} h^{(m)}(x_p, x_m^*, x_k^*) \mu_{v_k \rightarrow f_j^{(m)}}(x_k^*),
\end{aligned}$$

Table 2.1: Probabilities derived from Mendel's laws for all possible values of $h^{(m)}(x_p, x_m, x_i)$, which is listed as h^m in the table header.

x_p	x_m	x_i	$h^{(m)}$	x_p	x_m	x_i	$h^{(m)}$	x_p	x_m	x_i	$h^{(m)}$
0	0	0	1	0	0	1	0	0	0	2	0
0	1	0	$\frac{1}{2}$	0	1	1	$\frac{1}{2}$	0	1	2	0
0	2	0	0	0	2	1	1	0	2	2	0
1	0	0	$\frac{1}{2}$	1	0	1	$\frac{1}{2}$	1	0	2	0
1	1	0	$\frac{1}{4}$	1	1	1	$\frac{1}{2}$	1	1	2	$\frac{1}{4}$
1	2	0	0	1	2	1	$\frac{1}{2}$	1	2	2	$\frac{1}{2}$
2	0	0	0	2	0	1	1	2	0	2	0
2	1	0	0	2	1	1	$\frac{1}{2}$	2	1	2	$\frac{1}{2}$
2	2	0	0	2	2	1	0	2	2	2	1

for all $x_p \in \mathcal{D}(v_p)$.

The calculation for a message that is passed to the mother connected to an m -node is defined similarly, but with obvious modifications:

$$\begin{aligned} \mu_{f_j^{(m)} \rightarrow v_p}(x_m) &= \sum_{x_p^* \in \mathcal{D}(v_p)} \mu_{v_p \rightarrow f_j^{(m)}}(x_p^*) \\ &\times \prod_{v_k \in \eta^{(k)}(f_j)} \sum_{x_k^* \in \mathcal{D}(v_k)} h^{(m)}(x_p^*, x_m, x_k^*) \mu_{v_k \rightarrow f_j^{(m)}}(x_k^*), \end{aligned} \quad (2.5)$$

for all $x_m \in \mathcal{D}(v_m)$.

Once a message has been passed in both directions along all edges in the pedigree factor graph, the joint probability of all the observed data upon the members of the pedigree (i.e., the pedigree likelihood) and the marginal probability of every variable node are available through simple operations involving the dot-product of messages.

The joint probability of all observed data on a singly connected pedigree factor graph

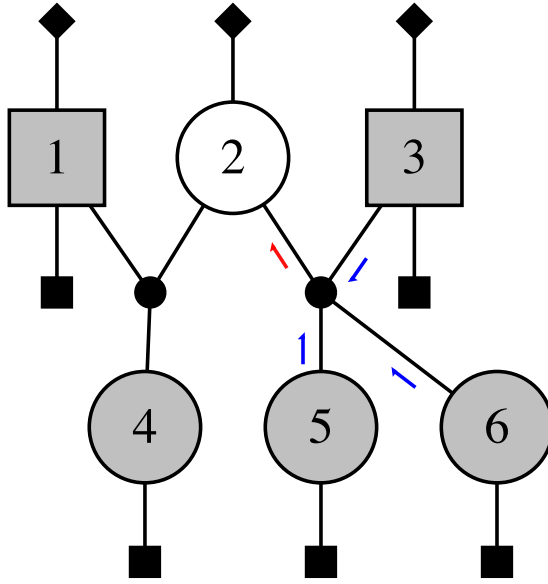


Figure 2.4: A snapshot of the intermediate steps during the sum-product algorithm on a pedigree factor graph. In order for the m -node of parents 2 and 3 to pass a message, $\mu_{f_2^{(m)} \rightarrow v_2}(x_2)$ (in blue), to individual 2's variable node we need to receive messages, $\mu_{v_3 \rightarrow f_2^{(m)}}(x_3)$, $\mu_{v_5 \rightarrow f_2^{(m)}}(x_5)$, and $\mu_{v_6 \rightarrow f_2^{(m)}}(x_6)$ (in red), from the variable nodes of individuals 3, 5 and 6.

\mathcal{P} is available as the dot product of the two messages that have been sent in different directions along any edge of \mathcal{P} :

$$p(y|\mathcal{P}) = \sum_{x_i \in \mathcal{D}(v_i)} \mu_{f_j \rightarrow v_i}(x_i) \mu_{v_i \rightarrow f_j}(x_i),$$

for all adjacent nodes f_j and v_i .

The marginal probability of any variable node v_i is the normalized product of all

incoming messages to that variable node:

$$p(x_i|\mathcal{P}) = \mathcal{N}^{-1} \prod_{f_j \in \eta(v_i)} \mu_{f_j \rightarrow v_i}(x_i), \quad (2.6)$$

$$\text{where } \mathcal{N} = \sum_{x_i \in \mathcal{D}(v_i)} \prod_{f_j \in \eta(v_i)} \mu_{f_j \rightarrow v_i}(x_i).$$

Since the outgoing message along an edge e from any variable node, v_i , in an acyclic factor graph is obtained as the product of incoming messages to v_i along all edges except e , it follows that the marginal probability of v_i can also be computed as the normalized product of the two opposing messages along any edge attached to v_i :

$$p(x_i|\mathcal{P}) = p(y|\mathcal{P})^{-1} \mu_{f_j \rightarrow v_i}(x_i) \mu_{v_i \rightarrow f_j}(x_i), \quad (2.7)$$

for all $x_i \in \mathcal{D}(v_i)$, and any f_j adjacent to v_i .

These general operations on the previously computed messages of a pedigree factor graph serve as essential components for efficiently calculating the joint likelihood of a newly proposed pedigree, as described in Section 2.2.7.

2.2.6 The Pedigree Prior Model, $P(\mathcal{P})$

Previous work has proposed the use of local and global properties of the cliques or nodes in pedigrees to define a prior distribution over pedigrees that is a function of structural, graphical features (Almudevar and LaCombe 2012). By contrast, for molecular ecol-

ogy data it has been customary to define the prior over two-generation pedigrees as a function of the estimated sampling density which clearly influences the probability that a true parent of an individual in the pedigree is included in the sample (Wang and Santure 2009). We work with two possible prior options: an uniform prior where all parental candidates, whether observed or not, have an *a priori* equal chance of being an individual’s parent; and a different prior where the user supplies an expected fraction of the possible parents in the population that were sampled each generation. This value is used as a prior in a beta distribution for the fraction of parents that are unobserved, and it can be updated according to the observed fraction, with weights chosen by the user. In addition to these two priors over the parents of a single generation, we also include a “hard-prior model” (Sheehan and Egeland 2007) to incorporate observed information on age and sex of individuals.

2.2.6.1 Hard Prior Model

Genetic data are often not the only information we can rely on to help infer a pedigree. We can also depend on other observable or secondary information such as sex and age to confine our sampling choices to probable, reasonable, and/or allowable pedigrees.

In general, knowing the sex of an individual patently limits the pool of potential mating partners to those of the opposite sex. Even when an individual’s sex is unknown, we can still enforce the same limitation based on inferred sex, if the sex of the individual’s mating partner is known or can be inferred. Otherwise, an individual of an unknown

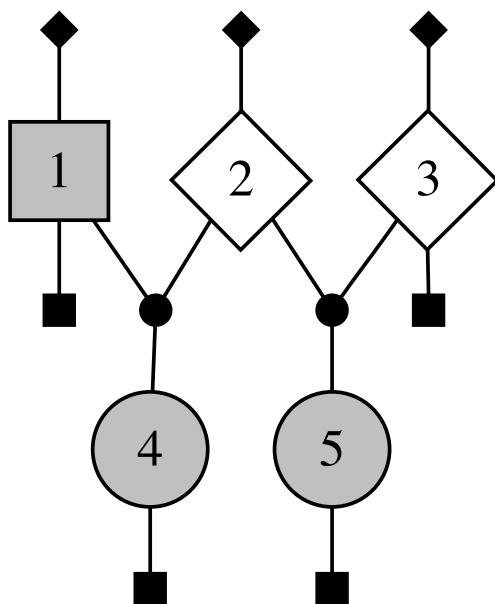


Figure 2.5: All individuals, except x_2 and x_3 , in this two-generation pedigree are observed. Despite x_2 and x_3 being unobserved, we can still easily infer their sexes (x_2 :female and x_3 :male) based on the fact that the sex of their mating partner is known or inferable.

sex will have an equal chance to be male or female.

The availability of individual age information is also useful in reducing the pedigree search space by assigning individuals into generation groups T thereby segregating members of the parental generation when making parent-offspring reassignments. Let $b_i \in (L, E)$ be the birth time of the i^{th} individual. We allow for this to be continuous between the two relative time points E (the earliest time point) and L (the latest time point). Also, let t_{\min}^r and t_{\max}^r be the minimal and maximal ages (in the same units as E and L) at which members of a species are capable of reproduction. We divide time into discrete bins, each of length t_{\min}^r that are roughly analogous to generations, and we

refer to them as “generation groups.” The generation group t_i of the i^{th} individual is defined as $t_i = \lfloor \frac{b_{\max} - b_i}{t_{\min}^r} \rfloor$, where b_{\max} is the most recent birth date of the collection. In order for individual x_j to be considered in the parent generation of individual x_i , the generation group t_j of individual x_j must be in $(t_i + 1, \dots, t_i + \lceil t_{\max}^r / t_{\min}^r \rceil)$ and the birth time difference between the two must be such that, $t_{\min}^r \leq b_j - b_i \leq t_{\max}^r$. Breaking the possible ages of candidate parents into these bins simplifies, somewhat, the treatment of unobserved parents whose ages are not observed, but become constrained once they are assigned offspring in the pedigree. Figure 2.6 shows an example of this discretization into generation groups.

Other *a-priori* information about the reproductive system, such as asexual vs. sexual mating, or a monogamous versus polygamous mating strategy, or semelparity versus iteroparity—as well as quantitative traits of individuals—could be incorporated into the prior model to restrict the space of possible pedigrees. This additional information can help to reduce the pedigree sampling space and allow the sampled pedigree to be consistent with previous observations. However the sampling strategies may need to be modified based on preferential selection for certain sub-network connections or else the sampling may be susceptible to stalling in certain local configurations. At present, ped-Fac accepts age and sex information, if available, and assumes a polygamous, iteroparous population, unless the user chooses to enforce semelparity and/or monogamy.

As stated in the first chapter, if you go back far enough in a population, everyone

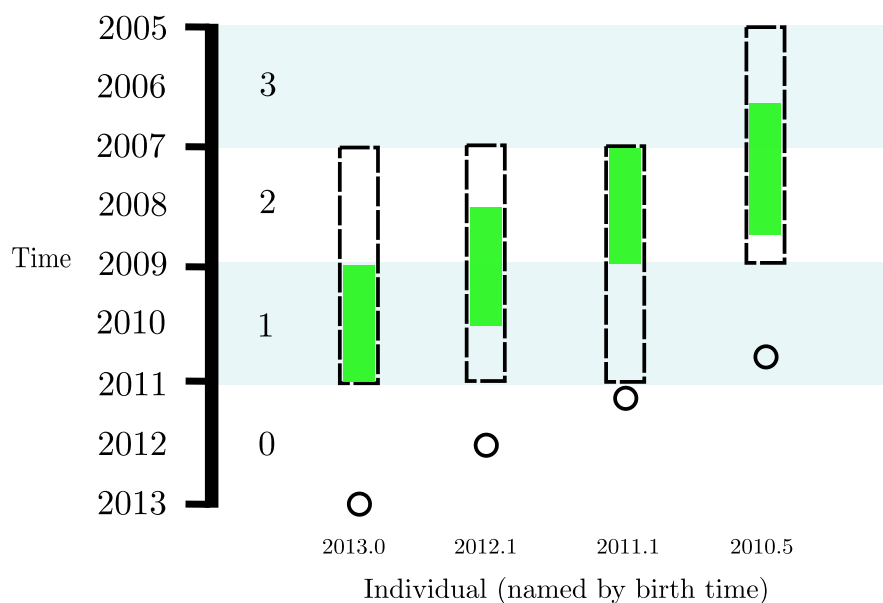


Figure 2.6: An example of transforming birth years into generation groups. Shown are four individuals, born at continuous times 2013.0, 2012.1, 2011.1 and 2010.5, in a population with $t_{\min}^r = 2$. Individuals from a sample of candidate parents with birth times ranging from 2005 to 2013 are placed into four generation-group bins, each of length $t_{\min}^r = 2$. The bins are denoted by the alternating white and gray rectangles, and are numbered from 0 to 3. Height, along the y -axis, of the open circles denotes the birth time of each of the four individuals, and the rectangles above each point show the possible birth years of the parents of each individual. The teal rectangle shows possible birth years of the parents when the exact birth time of each individual is known, while the rectangle with the dashed perimeter shows the range of possible birth years of the parents when the exact birth year of the individual is not known, but its generation group is.

can be connected through a giant pedigree. In most cases, however, data are limited to the most recent generations of that pedigree. If we allowed an unlimited number of unobserved individuals, when sampling over the space of pedigrees, there is a possibility that many generations of unobserved individuals would be inferred, creating very large connected components. However, once an individual is more than a few generations

removed from any observed genetic data, there is very little information available to infer the pedigree. pedFac addresses this problem by imposing a user-specified upper bound on the length of a path traversing exclusively unobserved variable nodes. This is implemented through monitoring of d_i , the minimum number of marriage factor nodes that must be traversed from a variable node v_i to reach any observed variable node. For some examples: if v_i is observed, $d_i = 0$; if v_i is unobserved, but adjacent to a marriage node connected to an observed variable node, then $d_i = 1$, etc. In practice, d_i for every node can be efficiently obtained through a belief-propagation algorithm.

2.2.7 Sampling from the Space of Pedigree Factor Graphs

Our goal, once again, is to evaluate the posterior distribution of the pedigree given the observed genotype, $p(\mathcal{P}|y)$, so as to ascertain the genealogical relationship between individuals. The posterior distribution, $p(\mathcal{P}|y)$, by Bayes' theorem, is the product of the likelihood, $p(y|\mathcal{P})$, and the prior distribution, $p(\mathcal{P})$, divided by a normalization constant that involves a sum over the space of all pedigrees with nonzero likelihood and prior. The task of enumerating all pedigrees in this space is difficult, if not impossible, because the number of possible pedigree configurations grows super-exponentially with the number of observed and unobserved variable nodes (Lauritzen and Sheehan 2003a). As this makes the normalization constant intractable, an alternative is to approximate the posterior distribution by drawing a Monte Carlo sample

from an ergodic—irreducible, aperiodic and recurrent—Markov chain that has, as its stationary distribution, the desired posterior distribution. Drawing such a sample from the posterior is an example of Markov chain Monte Carlo (MCMC).

A general algorithm for devising a Markov chain with the appropriate stationary distribution was provided by Hastings (1970), and is known as the Metropolis-Hastings algorithm. The algorithm proceeds by proposing (typically small) changes to the current state of the Markov chain from a proposal distribution, and then accepting or rejecting such proposed changes on the basis of the Hastings ratio (see below). To sample from the posterior distribution, $p(\mathcal{P}|y)$, by the Metropolis-Hastings algorithm, we propose a new pedigree configuration, \mathcal{P}' , from a proposal distribution, $q(\mathcal{P}'|\mathcal{P})$, which is conditional upon the current configuration state, \mathcal{P} . This new, proposed configuration, \mathcal{P}' , would then be accepted with probability given by the Hastings ratio:

$$\min \left\{ \frac{q(\mathcal{P}|\mathcal{P}') p(y|\mathcal{P}') p(\mathcal{P}')}{q(\mathcal{P}'|\mathcal{P}) p(y|\mathcal{P}) p(\mathcal{P})}, 1 \right\}. \quad (2.8)$$

A special case of the Metropolis-Hastings algorithm which uses the full-conditional distribution as the proposal distribution is called the Gibbs sampler (Geman and Geman 1984). In the Gibbs sampler, the proposals are all accepted with probability 1. In a typical Gibbs sampling scenario, the target distribution is the posterior distribution $P(\theta|y)$, where θ consists of several, say K , components: $\theta = (\theta_1, \dots, \theta_K)$. The full conditional, in this case, is often defined as the conditional distribution of any one

component, i , given the data and the remaining components, which can be written as:

$$p(\theta_i | \dots) = p(\theta_i | y, \theta_{-i}),$$

where θ_{-i} denotes all the components of θ except for the i^{th} component. Within this setup, it is a straightforward exercise to verify that the full-conditional distribution is proportional to the posterior distribution and that, hence, use of the full conditional as the proposal distribution will lead to a Hastings ratio equal to one.

The above perspective does not apply to sampling over the space of pedigrees, because there is not a simple, consistent factorization over different components—since the graphical structure of the model changes when the pedigree is updated, so does the factorization. However, similar reasoning leads to a sampler over pedigree configurations that, like the Gibbs sampler, has an acceptance probability of one. In general, the approach involves proposing a new pedigree state \mathcal{P}' from amongst a restricted set, \mathcal{S} , of possibilities, which must include the current state, \mathcal{P} . Each of the possibilities in \mathcal{S} is proposed with a probability that is proportional to the posterior probability of that new configuration:

$$q(\mathcal{P}' | \mathcal{P}) = \frac{p(\mathcal{P}' | y)}{M} \quad \forall \mathcal{P}' \in \mathcal{S} \quad \text{with} \quad M = \sum_{\mathcal{P}' \in \mathcal{S}} p(\mathcal{P}' | y). \quad (2.9)$$

If the cardinality of \mathcal{S} is not large, then the proposal distribution can be easily computed.

With such a proposal distribution, it is clear that the acceptance probability from the Hastings Ratio is always one, as follows:

$$\begin{aligned} \min \left\{ \frac{q(\mathcal{P}|\mathcal{P}') p(y|\mathcal{P}') p(\mathcal{P}')}{q(\mathcal{P}'|\mathcal{P}) p(y|\mathcal{P}) p(\mathcal{P})}, 1 \right\} &= \min \left\{ \frac{p(\mathcal{P}|y)/M p(y|\mathcal{P}') p(\mathcal{P}')}{p(\mathcal{P}'|y)/M p(y|\mathcal{P}) p(\mathcal{P})}, 1 \right\} \\ &= \min \left\{ \frac{p(\mathcal{P}|y) p(\mathcal{P}'|y) p(y)}{p(\mathcal{P}'|y) p(\mathcal{P}|y) p(y)}, 1 \right\} \\ &= \min \{1, 1\} = 1. \end{aligned}$$

It is worth noting that, since the current state is included in \mathcal{S} , accepting the proposal may still not lead to any changes in the current state. This formulation, however, allows for a proposal distribution that can sample from a large number of candidate parents, and thus may offer better mixing properties than a more naïve, classic, Metropolis-Hastings algorithm.

2.2.7.1 A Proposal Distribution for Updating Pedigrees

When we propose a new pedigree configuration \mathcal{P}' , the suggested change could consist of a simple move, such as adding or removing an edge between a single variable node and an m -node. Or it could involve making multiple moves that, for example, might shuffle the parent-pair assignments for all individuals in the sample. The former approach, developed early on and termed the Metropolis-Hasting sampler for Markov Chain Monte Carlo Model Composition, or MC³, (Madigan and York 1993), was the original and standard approach for learning graphical structure. It emphasizes making small moves

according to the local support of the conditional probability. Unfortunately, this locally-sensitive sampler generally mixes poorly, especially for multivariate distributions with correlated components. The latter approach would be similar to strategies, such as blocking (Goudie and Mukherjee 2016), which involve combining components as a unit and sampling from their conditional probability, thus using a proposal distribution that can be sensitive to both local and global features of the model and data (Eaton and Murphy 2007). Such approaches have been shown to improve mixing and the convergence rate of Markov chains to the posterior; however, care must be taken to not create such large blocks that the full-conditional calculations become intractable.

These published methods that focus on learning general directed acyclic graphs are not directly compatible with pedigree factor graphs. Thus, the proposals we construct combine principal elements from both of the above approaches, but are distinct from either of them. Like MC^3 , our proposals involve relatively small, incremental changes, like the addition/deletion of just a few edges. But, unlike MC^3 , the proposal distribution for adding/deleting edges considers a large number of possible locations within the graph for edges to be inserted, taking a more global view of the graphical structure and providing better mixing.

Specifically, we propose incremental changes to the pedigree \mathcal{P} by reassigning parent pairs of an individual, one at a time. In doing so, the focus on the state of the pedigree is directed toward the parental identities of the edges upstream of each non-founder

member of the pedigree. Such a perspective follows naturally from the the fact that 1) every individual, apart from the founders, in a pedigree must have an edge connected to its parents' m -node and 2) every pedigree \mathcal{P} can be summarised as a tuple of N_{nf} sets of parents—one set of parents for each of the N_{nf} non-founders in the pedigree. For example, we may specify the pedigree \mathcal{P} as the N_{nf} -tuple $[\text{pp}_1^{\mathcal{P}}, \dots, \text{pp}_{N_{\text{nf}}}^{\mathcal{P}}]$, where $\text{pp}_i^{\mathcal{P}}$ denotes the pair of parents of the i^{th} non-founder individual in \mathcal{P} . To highlight the partition between the parents of the i^{th} individual and other individual nodes, we can simplify $\mathcal{P} = [\text{pp}_i^{\mathcal{P}}, \text{pp}_{-i}^{\mathcal{P}}]$, where $\text{pp}_{-i}^{\mathcal{P}}$ denotes a set of parent pairs assigned to all individuals, except the i^{th} .

One proposal distribution implemented in pedFac, proposes a change to the parent pair for a single individual, i . This proposal distribution can be written simply as:

$$q(\text{pp}_i^{\mathcal{P}'} = \text{pp}_i | \text{pp}_{-i}^{\mathcal{P}}, y) = \frac{p([\text{pp}_i, \text{pp}_{-i}^{\mathcal{P}}] | y)}{M}, \quad \text{with } M = \sum_{\text{pp}_i^* \in \mathcal{S}} p([\text{pp}_i^*, \text{pp}_{-i}^{\mathcal{P}}] | y), \quad (2.10)$$

where \mathcal{S} is the set of allowable parent pairs for individual i , or some suitable restriction of that set (see below).

Here, we expand upon how the set \mathcal{S} is envisioned. First, note that pp_i can be regarded as the father and mother of i : $\{x_u^i, x_v^i\}$, where x_u^i and x_v^i are the variable indices of the father and mother of the i^{th} individual. We write X_{dad}^* for the set of possible distinct candidates that could be x_u^i . This includes all males in the sample (i.e., all *observed* individuals), as well as all individuals of unknown sex, in the sample

that are of the right age to have been a father of i . It also includes males and individuals of unknown sex that are not in the sample (i.e., *unobserved* individuals). Thus we write:

$$x_u^i \in X_{\text{dad}}^i = \{X_{\text{male,obs}}^i \cup X_{\text{male,unobs}}^i \cup X_{\text{unk,obs}}^i \cup X_{\text{unk,unobs}}^i\}.$$

The same arguments lead to the set of candidate mothers:

$$x_v^i \in X_{\text{ma}}^i = \{X_{\text{female,obs}}^i \cup X_{\text{female,unobs}}^i \cup X_{\text{unk,obs}}^i \cup X_{\text{unk,unobs}}^i\}.$$

To be included in \mathcal{S} , every $x_u^i \in X_{\text{dad}}^i$ and $x_v^i \in X_{\text{ma}}^i$, proposed as a parent pair pp_i must also satisfy the conditions that:

1. $x_u^i \neq x_v^i$
2. the age gap between i and x_u^i and i and x_v^i must be within the permissible range,
3. for x_u^i and x_v^i that are of unknown sex, a sex can be assigned to the individuals in a way that is compatible within the pedigree with all other individuals of known or implied sex.
4. the resulting pedigree remains acyclic.

Ideally, the domain \mathcal{S} of the proposal distribution would include, as candidate parent pairs for the focal individual, the allowable pairings of all observed and unobserved individuals in the parental generation; however, to reduce the number of possible parent

pairs, and hence the size of \mathcal{S} , the proposal distribution draws only from those pairs that are formed from the n most likely parent candidates of each sex, if known, based on the pairwise parental-versus-unrelated likelihood ratio. It is important that the value of n is not set so low that doing so would exclude the true parent from the pool of candidate pairs. Fortunately the likelihood of that occurring decreases when more genetic markers are used. By default, $n = 10$ is used. Furthermore, the top n candidates amongst *observed* single-parent candidates for all observed individuals can be identified prior to the start of MCMC sampling, which reduces time spent in reassigning parents for observed individuals.

An additional restriction of \mathcal{S} in this parent-pair sampler is available as an option to pedFac. It draws pairs from amongst those formed by combining the single highest-likelihood-ratio parent (call this individual x_{fixed} and all other possible candidates of sex opposite to that of x_{fixed} .

2.2.7.2 Sampling Procedure

The starting point for MCMC sampling over the space of acyclic pedigrees is a state in which each sampled individual is disconnected from every other one. In other words, each sampled individual forms its own connected component, containing a single member, such that the pedigree at this point can be represented as a collection of variable nodes, each with a p -node and a g -node attached to it, but with no m -nodes connecting any members of the sample. Each step of the sampling procedure involves three steps:

1. First, the sum-product algorithm is run on the factor graph corresponding to each connected component so as to compute and store values for all incoming and outgoing messages through every edge, prong and stub (the “imaginary” vertices, as discussed in the next section).
2. Second, we iterate through all individuals (excluding the founders) in the pedigree, starting from the most recent generation to the oldest, and for each, we propose changing the parent pair to which that focal individual is attached. These potential parent pairs can consist of existing observed and unobserved individuals as well as newly created unobserved individuals that are not connected to any other pedigree members. We then use the stored values in the stubs and prongs (depending on whether an m -node already exists between the potential parents pairs), to calculate the likelihood (and, ultimately, by multiplying by the pedigree prior, the unnormalized posterior probability) for each proposed configuration. This set of unnormalized posteriors is then normalized to sum to one, and those probabilities are used to sample the specific change to the pedigree configuration.
3. Finally, if a proposed change connecting the focal individual to a new set of parents is accepted, then, before proceeding to the next focal individual, the sum-product algorithm must be run on the connected component created by the update. This provides updated messages along all the edges in the connected component.

We take this bottom-up approach—starting with the most recent generation and pro-

ceeding up the pedigree—because it makes it straightforward to enforce two fundamental constraints upon the inferred pedigrees: 1) all individuals at the bottom level of each pedigree must be observed, and 2) every non-founder individual must have two parents in the pedigree.

It should be noted that after the first sampling iteration, all individuals, except the founder individuals, will be attached to a parent pair (which may be composed of observed or unobserved individuals). Thus, for subsequent sampling rounds, the edge between the focal variable node and the m -node attaching it to its parents will be severed again prior to rearrangement.

2.2.7.3 Rapid likelihood Calculation For Proposed Pedigree Reconfigurations

The likelihood $p(y|\mathcal{P}')$ for each proposed pedigree is obtained with a straightforward calculation after the sum-product algorithm has been run. Facilitating this calculation requires a very small amount of extra effort during the sum-product algorithm: each variable node and m -node in the factor graph is adorned with an additional edge leading to an “imaginary” node carrying no data. These additional edges are called “prongs” when attached to an m -node and “stubs” when attached to a variable node. During the sum-product algorithm, outgoing messages from the variable nodes and m -nodes are computed and stored along these stubs and prongs, respectively. These messages then get used directly in computing $p(y|\mathcal{P}')$.

For example, if we propose the focal individual to be attached to an existing m -node, the likelihood of this proposed connected component, for each locus, will be the dot-product of the stub of the focal variable node and the prong from the m -node. In the case of attaching a focal individual to a parent pair that is not already attached as parents to an m -node, computing the likelihood of the proposal requires one more sum-product operation to compute the outgoing message along a prong from a newly created m -node attached to the stubs from the two candidate parents, so as to be able to take the dot-product of this prong message with the stub message from the focal individual.

2.2.7.4 Acyclic Pedigree Sampling Space

Under belief propagation, the likelihood calculation of a pedigree factor graph is exact when the pedigree factor graph has no cycles. The likelihood of a cyclic pedigree factor graph will not be exact under traditional loopy belief propagation. For now, we assume that the true pedigree contains no loops and limit ourselves to problems involving no more than a few generations. To maintain an acyclic pedigree, we only accept proposals in which the m node that connects the parent pair is in a different component than the focal individual being assigned to the parents.

2.2.8 Convergence and Mixing in MCMC

2.2.8.1 Irreducibility Proof

It is critical that, under the current proposal method, all pedigrees in the acyclic sampling space are accessible from any starting pedigree so that any pedigrees, even improbable ones, *could* be sampled. By definition, state j is accessible from state i if it takes a finite number of steps to transition from state i to state j with non-zero probability. In a formal context, pedigree \mathcal{P}_j is said to be accessible from \mathcal{P}_i under the transition probability matrix \mathbf{P} , implied by the proposal distribution, with rows and columns indexed by different pedigree configurations, if there exists some integer $n > 0$, such that $\mathbf{P}_{\mathcal{P}_i, \mathcal{P}_j}^n > 0$ is the probability of reaching \mathcal{P}_j from \mathcal{P}_i in n steps.

Under the current proposal system, it is always possible to transition from \mathcal{P}_i to \mathcal{P}_j in $2n_d$ number of steps, where n_d is the number of individuals, x' , with assigned parents differing in \mathcal{P}_i and \mathcal{P}_j . The first n_d steps assign all x' to parent pairs with both unobserved members, and the remaining n_d steps assign x' to the parent pairs found in \mathcal{P}_j .

2.2.8.2 Poor Mixing for Interchangeable Pedigrees

Even though the current sampling method is irreducible as previously shown, the sampling chain may mix poorly through acyclic pedigrees that involve reshuffling the orders of unobserved individuals within a connected component. This poses a challenge to

smoothly accessing these “interchangeable” pedigrees which are often equally probable and involve rearrangements centering on a pair of unobserved parents.

Figures 2.7a and 2.7b illustrate a common scenario of interchangeable or equiprobable pedigrees when a parent pair is monogamous and unobserved. The difference between these two interchangeable pedigrees is the attachment of the grandparent pairs. One condition that would resolve the uncertainty of the parental placement would be the inclusion of an observed mating partner of either of the unobserved parent pairs (i.e., individual 4 or 5). Regardless of whether the unobserved parent pair is truly monogamous, the sampling method should allow for movement between these two equiprobable configurations.

2.2.8.3 Additional Proposal Move: Swapping

To improve mixing of the MCMC sampler for interchangeable pedigrees, we include a “swapping” move to expand the current working domain \mathcal{S} of the proposal distribution. The “swapping” moves entail exchanging one or both members of parent pair pp_i (along with the upstream ancestors) of the focal individual i with one or both members, respectively, of parent pair pp_j (along with the upstream ancestors) of its mating partner j , where $j \in J$ and J is the set of all mating partners of individual i .

Among multiple ways of choosing which parent pairs of the mating partners to be swapped, the “swapping” exchanges can be subdivided into three categories:

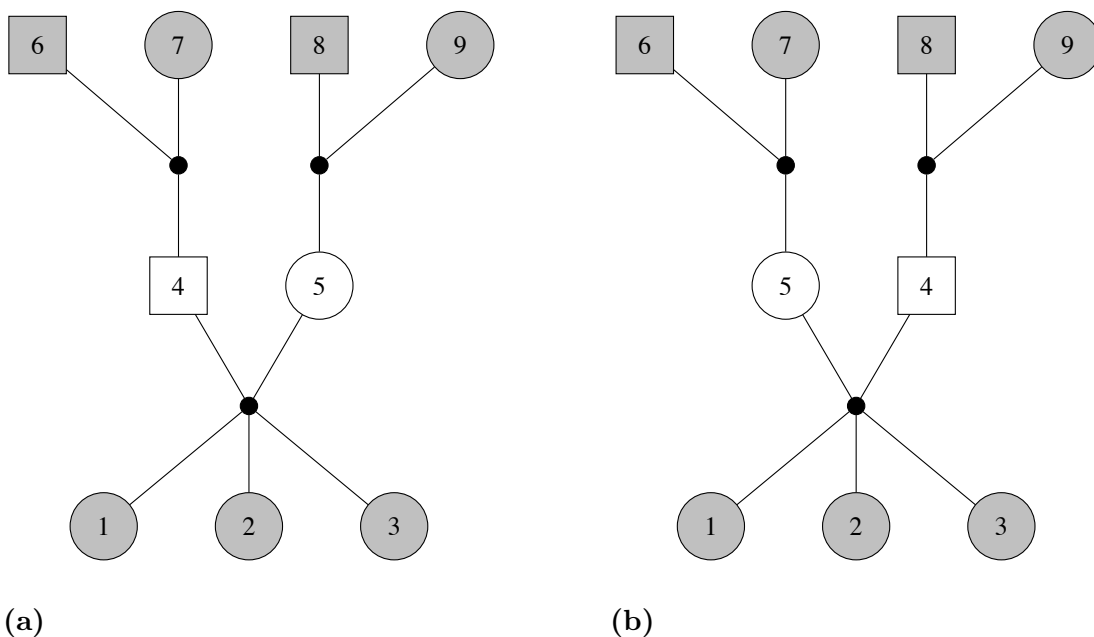


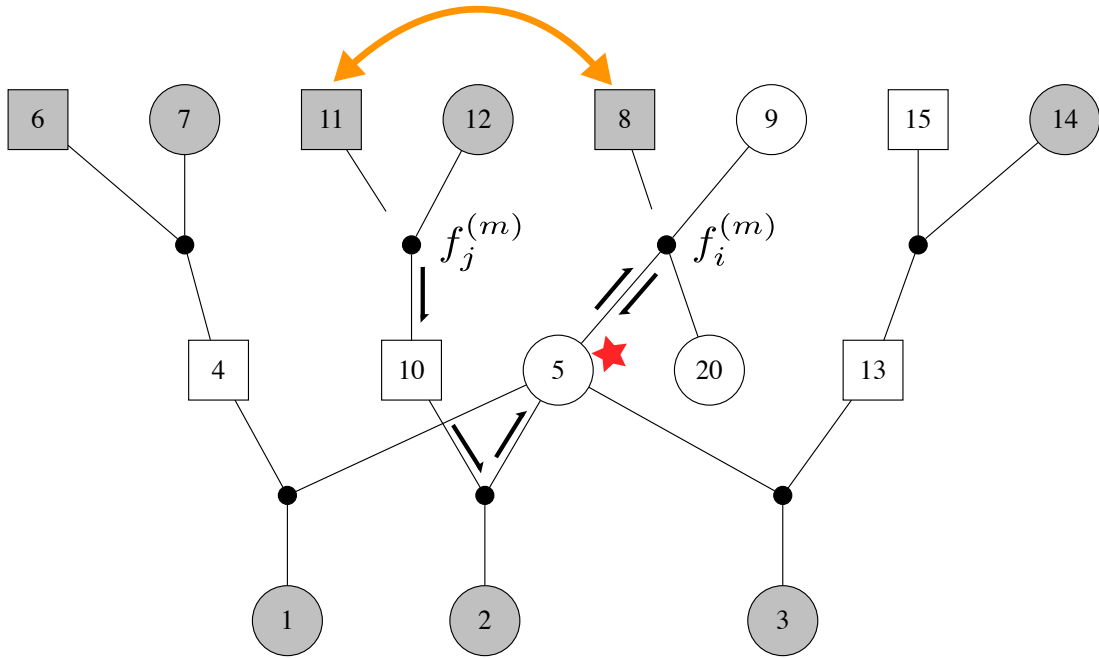
Figure 2.7: An example of interchangeable pedigrees given a monogamous and unobserved parent pair within a multigeneration pedigree. The pedigree, on the left, is interchangeable and equiprobable with the one on the right; it is impossible to identify whether each set of grandparental pairs should be assigned to the paternal or maternal side.

- exchanging a single parent of individual i with a single parent of j , where $j \in J$ (Fig 2.8a),
- exchanging both parents of individual i with both parents of j , where $j \in J$ (Fig 2.8b), or
- exchanging one parent of individual i with one parent member of i 's mating partner j and exchanging the other parent of i with one parent of a different mating partner k , where $j, k \in J$ and $j \neq k$ (Fig 2.8c).

In pedFac, the likelihood of the proposed, swapped pedigree \mathcal{P}' is calculated by passing messages *from* the parental m -node of the mating partner (i.e., j or k) and *to* the parental m -node of the focal individual (i.e., i). (Note that it would also be possible to calculate the likelihood by passing messages in the reverse of the fashion it has been implemented in pedFac). For all three cases, it takes at most eight message passing steps to calculate the likelihood of the new, proposed pedigree under the “swapping” regime because of the modularity of the factor graph framework and the reusability of previously computed messages.

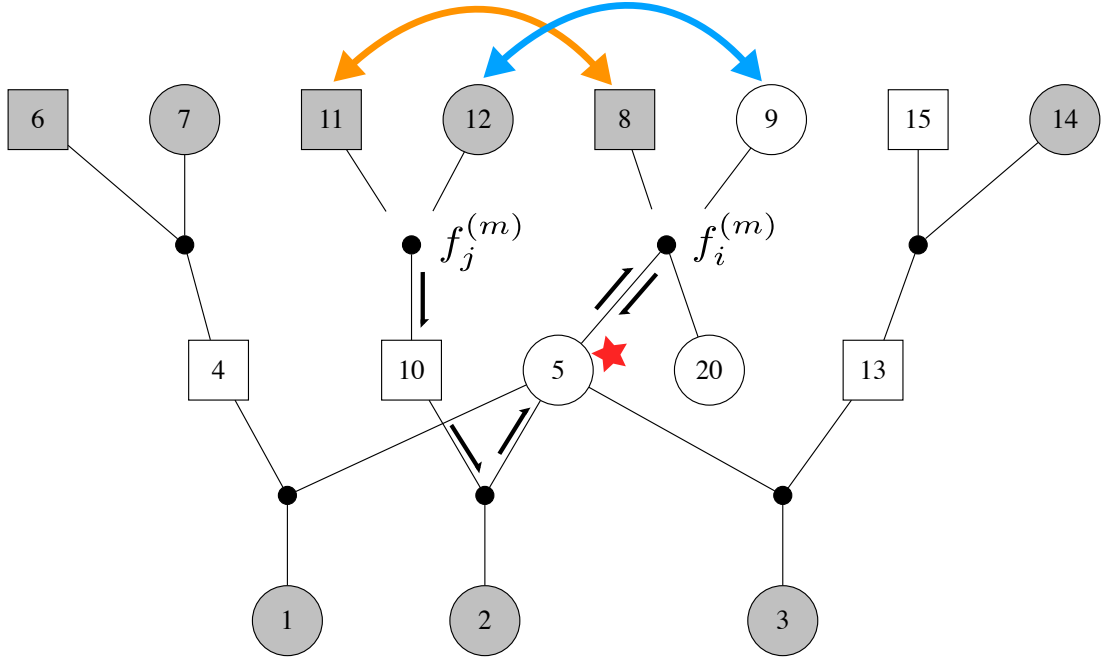
Using Fig. 2.8a as an example, we first take pre-computed outgoing messages from the swapped variable node x_8 and unswapped variable node x_{12} , i.e., $\mu_{v_8 \rightarrow f_i^{(m)}}(x_8)$ and $\mu_{v_{12} \rightarrow f_j^{(m)}}(x_{12})$, to recompute the outgoing message from the parental m -node of mating partner x_{10} , i.e., $\mu_{f_j^{(m)} \rightarrow v_{10}}(x_{10})$. This computed message is used to update and propagate messages along the edges between x_{10} and x_5 and finally as an outgoing message from x_5 to its parental m -node $f_i^{(m)}$. The dot-product of this outgoing message with the message (recomputed using the outgoing message from x_{11}) coming down from $f_i^{(m)}$ to x_5 gives the likelihood of the proposed pedigree.

It should be noted that the proposals in Figures 2.8a–2.8c are not equiprobable with the current configuration (unlike the scenario shown in Fig. 2.7). Thus, swapping moves are not solely useful for such equiprobable situations, but, rather, represent an efficient method for rearranging ancestors in a pedigree with using very few intermediate steps.



(a) The first type of “swapping” proposal involves exchanging one of the parents of the focal individual, (x_5 , annotated with a star) with one of the parents of its mating partner (x_{10} , in the figure). In the example, the newly proposed pedigree \mathcal{P}' is made by rearranging the placement of individual 8 with individual 11. To calculate the likelihood of \mathcal{P}' , we first perform message propagation along the path starting from the parental m -node of x_{10} (i.e., $f_j^{(m)}$) to the parental m -node of the focal individual x_5 (i.e., $f_i^{(m)}$) and take the product of this outgoing message with the newly updated incoming message from $f_i^{(m)}$.

Figure 2.8: Various “swapping” proposal moves performed on a multigenerational pedigree. The red star marks individual 5 as our focal individual. The bidirectional colored arrows denote the proposed swapping actions between the parent pair of individual 5 and those of its mating partner, 10. The small arrows along the factor graph’s edges represent the messages that needed to be updated in order to calculate the likelihood of the proposed pedigree.

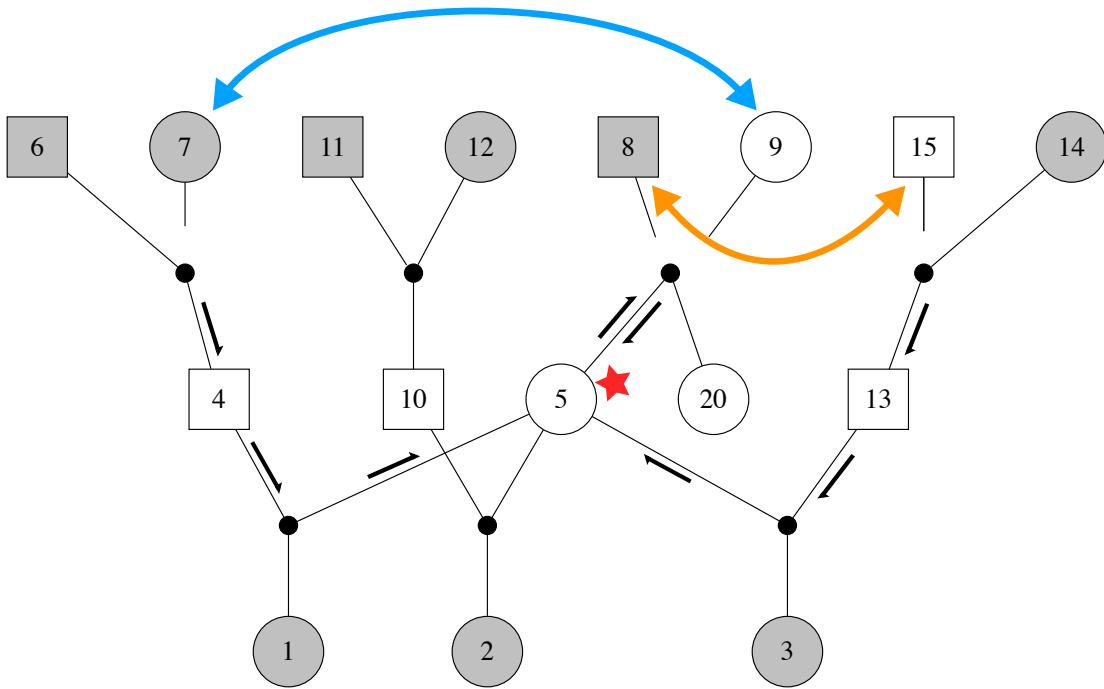


(b) The second type of swapping move involves exchanging both parental components of the focal individual with those of its mating partner. In this pedigree scenario, parents of focal individual x_5 are swapped in place with parents of individual x_{10} . The message update path follows a similar routine to that of (a).

Figure 2.8: Various “swapping” proposal moves performed on a multigenerational pedigree. (cont.)

2.2.8.4 Additional proposal moves: general swapping and substitution

The “swap” move proposed in Section 2.2.8.3 focuses narrowly on swapping a specific set of individuals from the same connected component for the purpose of improving the rate of mixing of the MCMC chain between interchangeable pedigrees. Here we apply similar principles, more generally, to improve mixing among a wider variety of pedigree structures, introducing two fundamental proposal types in pedFac: “swapping” and “substitution.” In the context of pedigree factor graphs, “swapping” involves simply



(c) The third class of swapping move works by swapping both parents of the focal individual with individual parents from two different mating partners. The father of focal individual x_5 is swapped in place with the father of its mating partners, x_{13} , while the mother of x_5 is swapped in place with the mother of x_4 . Outgoing messages from those two mating partners, x_4 and x_{13} , must be updated and passed to x_5 in order to calculate the likelihood of this proposed change to the pedigree.

Figure 2.8: Various “swapping” proposal moves performed on a multigenerational pedigree. (cont.)

exchanging the labels or identities between two variable nodes while leaving the edges and neighbors of each of the two focal individuals intact and unchanged (see example in Figure 2.9a). In contrast, the process of “substitution” (Figure 2.9b), occurs when an individual *and other individuals in its neighborhood* are replaced by the individuals in the neighborhood of another individual. The latter can result in much larger changes within the connected component, than can the former.

We include general swapping and substitution in pedFac’s set of proposal-move types to ensure good mixing of the Markov chain—specifically to overcome bottlenecks resulting from user-specified restrictions on the types of pedigree allowed. For example, if the population of interest is known to follow a monogamous mating pattern, it is desirable and logical to restrict the pedigree sample space to only those with monogamous mating. However, this restricted sampling domain makes it difficult to transition between pedigrees with different parental assignments to the focal individual when all of the candidate parents have already been attached to a mating partner.

Using Figure 2.9a as an example, if individual x_1 and x_3 are the true parents of the focal individual x_{focal} , in order for those parent pairs to join together, given the current configuration, we must first effect the disassociation of the marriage node of x_3 and x_4 . The latter reconfiguration relies on an additional prerequisite: that it is possible and also probable for individuals x_6 and x_7 to be assigned to a different pair of parents. As a result, in the absence of the general swapping and/or substitution moves, this nested state of conditions impedes the mixing of the MCMC chain.

With general swapping and substitution moves in the proposal repertoire, the transition between pedigrees of differing parental assignments is accessible under a single move, and the calculation for the posterior of the proposed reconfiguration remains tractable and efficient through the factor graph representation.

As of now, we are limiting these proposal moves to be performed between individuals

of different connected components for simplicity.

2.2.9 Data and Software Implementation

2.2.9.1 Implementation of pedFac

pedFac is available as an R package and also as a python script on Github. For input, pedFac accepts 1) a file of genotypes, 2) the user's specifications about the population under study and the desired length and type of sampling runs, and 3) an optional file of secondary meta data. pedFac generates intermediate files and passes them to a C program that generates the pedigree factor graph and runs all major algorithms such as the sum-product algorithm and MCMC sampler. pedFac, in turn, outputs all sampled pedigrees in a text file with four columns: sweep number, kid, father, mother, and also summarizes this output (MAP estimate, parentage assignments, sibling groups, etc.)

2.2.9.2 Pedigree simulator

I have also written a pedigree simulator in R and Python, used to generate known multigenerational pedigrees to serve as test datasets. The user can specify the following parameters: the number of generations T , the number of founders N_f , the fraction of males expected in the founder generation $N_{f,m}$, the λ_m parameter of a Poisson distribution that sets the expected number of mating partners and a separate λ_o that sets the expected number of offspring per parent pair. The user can specify the number of

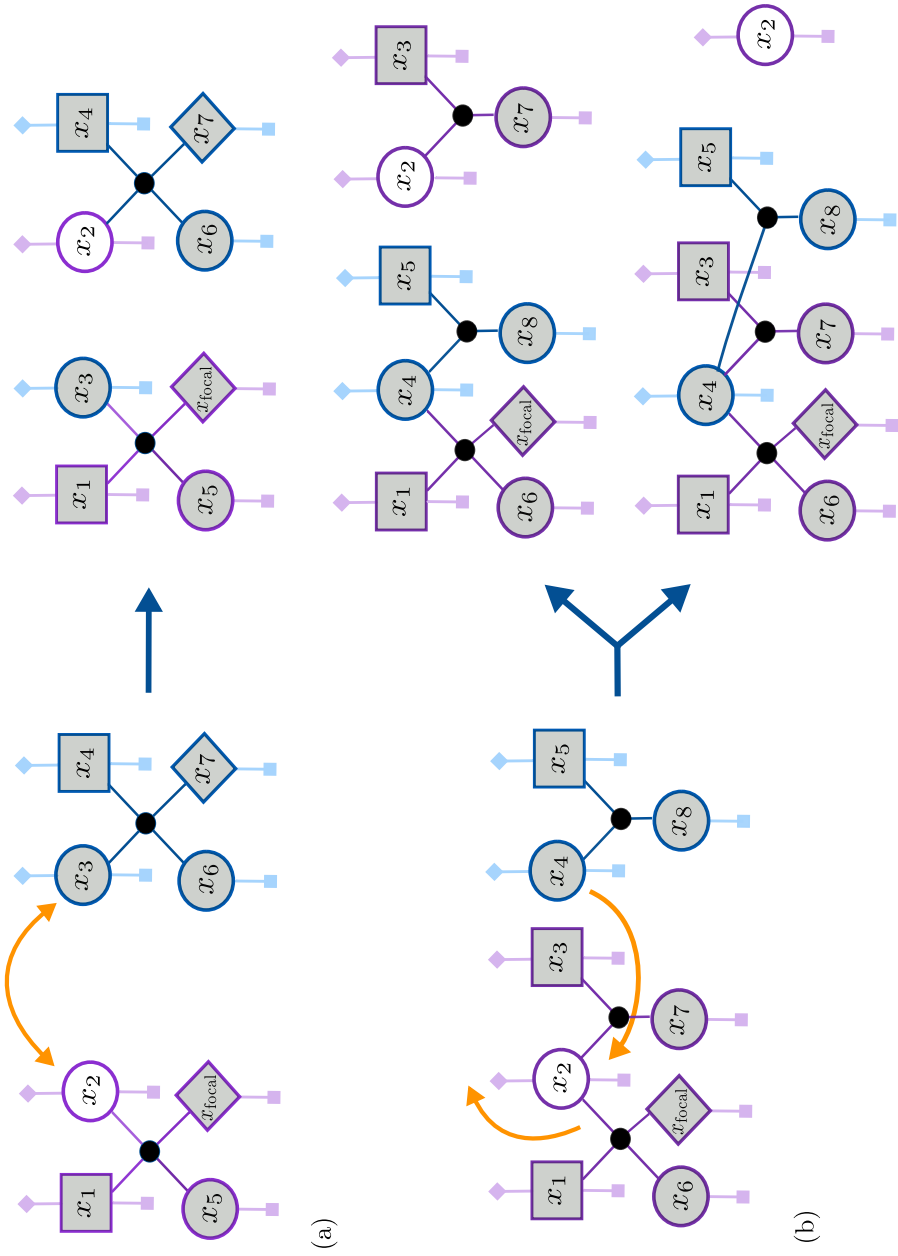


Figure 2.9: Examples of swapping and substitution moves. Each connected component is colored distinctly to highlight the changes under swapping (a) and substitution (b). With swapping, the exchange between x_2 —the mother of the focal individual—and x_3 results in a single possible outcome. However in the case of substitution, multiple outcomes are possible, depending on which of the neighbors of x_2 remain attached to their original connected component.

SNPs N_s , the expected genotyping error rate ϵ , the expected genotype missingness rate, the alpha ρ_α and beta ρ_β parameters of a beta distribution to simulate the population allele frequencies, and the fraction f of individuals in the pedigree that are sampled. The user can also restrict the simulated pedigrees to be acyclic.

2.2.10 Evaluating the performance of pedFac

2.2.10.1 Simulation scenarios

We test pedFac and compare its performance to three other software programs using simulated data from three different scenarios:

1. two-generation monogamous pedigrees
2. two-generation polygamous pedigrees
3. multigenerational monogamous pedigrees.

All of the pedigrees simulated in these scenarios are constrained to be strictly acyclic. Each of these scenarios will be more fully described when they are discussed in the “Simulations and Results” section below.

2.2.10.2 Competing software

We compare the results of pedFac with those obtained by using three other software programs: COLONY v.2.06.5 (Jones and Wang 2010), sequoia v.2.0.7 (Huisman 2017)

and FRANZ v 2.0 (Riester et al. 2009). COLONY is recognized amongst molecular ecologists as the most accurate method available today for sibship reconstruction and parentage inference, and Sequoia has been recently proposed as a faster method with accuracy approaching that of COLONY. FRANZ is known to be less accurate than COLONY; however unlike COLONY, FRANZ allows for the inference of multigenerational pedigrees. Therefore we only use it for comparison in scenario 3.

For all scenarios we set the “maximum sibling iterations” in Sequoia to 20. For COLONY, in all scenarios, we chose the Full-Likelihood (FL) method without sibship scaling and without a sibship prior. COLONY does not allow the user to specify a fixed number of iterations of its algorithm. Rather, the user chooses a combination of run-length (short, medium, long or very long) and precision (low, medium, high, and very high) to set the number of iterations. Even when COLONY was set to a short run length with low precision, it required roughly $5\times$ the runtime required for pedFac to perform 100 sweeps for an average case, so we uniformly chose the short runtime settings for COLONY (short run-length and low precision).

Settings for these programs, and for pedFac, that were chosen specifically for different scenarios are reported with the results.

2.2.10.3 Evaluating performance

Comparing networks, especially those involving many vertices and edges, can be a daunting task. In our case, we need to measure how close the inferred pedigree is to the truth

in units that can be easily interpreted by potential users of pedFac. Instead of reporting the topological differences of pedigrees in an abstract graphical fashion, for scenarios 1 and 2 we take a more pragmatic approach by focusing on how well each program performs in inferring familiar relationships. To assess this, for different focal relationships we report the false discovery rate (FDR) and the false negative rate (FNR). The FDR is defined as the fraction of assigned/inferred relationships (of a particular type) that are incorrect. The FNR is the fraction of true relationships (of a particular type) that are not correctly identified by the software.

In scenarios 1 and 2, we also calculate the receiver-operator-characteristic area under the curve (ROC-AUC) score to evaluate how accurately the “confidence” scores produced by pedFac, COLONY and sequoia can be used to separate out true from false assignments. All of the four programs—pedFac, COLONY, sequoia, and FRANZ—provide a value associated with each inferred relationship that can be used to assess confidence. In the case of pedFac, this is transparently a posterior probability—the frequency with which the relationship is observed over the total number of sweeps, after a burn-in period. We use the maximum *a posteriori* (MAP) estimate for all of our analysis. FRANz also reports a posterior probability obtained from MCMC simulation. The interpretation of the values provided by COLONY and sequoia is more difficult.


COLONY’s objective is to find a pedigree that maximizes the likelihood. Once it has achieved that, the program reports a “probability” for each trio (pair of parents and

a single offspring) in this optimal pedigree. This is not a proper posterior probability, but it is intended to provide a measure of support for the parentage assignments in the trio. COLONY assesses the accuracy of its inferred full-sibling groups, by reporting two numbers: “Prob(Inc)”—how likely it is that the inferred sibship includes one or more non-siblings—and “Prob(Exc)”—how likely it is that the inferred sibship is missing one or more true siblings. For sibship-inference ROC-AUC calculation with COLONY we ranked sibling assignments by “Prob(Inc),” and ties were resolved by “Prob(Exc).” For any sibship pair, we ranked sibling pair based on the output “Probability”.

Sequoia does not estimate the uncertainty of trio assignments, nor of inferred full siblings. Rather, for each inferred parent-pair and offspring trio, it provides an “LLR” score which is the log of the likelihood that the trio of individuals consists of two parents and an offspring minus the log of the likelihood that the members of the trio are unrelated. With no other options given by the program for assessing confidence, we use these LLR scores to rank inferred relationships for ROC-AUC calculation. Additionally, for some plots (such as Figure 2.11, below) we plot the results against the posterior probability. For these cases, with sequoia, we simply use $\exp(\text{LLR}_i) / \max_i \{\exp(\text{LLR}_i)\}$ as a proxy.

2.3 Simulations and Results

2.3.1 Scenario 1: Inference of Parentage and Full-sibling groups in a Chinook Salmon Pedigree

We simulate data from the same Chinook salmon pedigree of 1157 genotyped individuals, all of a single cohort, described in Anderson and Ng (2016), as our first test data set. The pedigree is composed of 432 monogamous families, with a mean sibling size 2.7, and a distribution of full-sibship sizes like: . In the first scenario we include all of the parents of the full sibling groups, but in later scenarios we include a smaller fraction of the parents, doing five scenarios total, with fractions of included parents $f \in \{1, 0.75, 0.5, 0.25, 0\}$. In the last scenario, none of the parents are included in the data set, corresponding to the well-known problem of full sibship inference. We simulate 5 replicate panels of 95 SNPs, each with a 2% chance that they have been genotyped with error and a 0.5% chance that the genotype is missing. For this study, pedFac is run with 100 sweeps under the assumption that mating in the population is monogamous (so that proposals creating half-siblings are not permitted), and a uniform prior is applied to the space of pedigrees—that is, every pedigree configuration is assumed to have equal probability *a priori*.

For the monogamous case study, we assess parentage assignment (for the first four

scenarios when parents are included in the dataset), and the inference of full-siblings in all scenarios. To comply with the nature of the pedigree, we applied the “monogamous mating” option available in pedFac, COLONY and sequoia.

Figure 2.10 includes results for three different settings within pedFac, to emphasize the importance of including the “swap” and “substitute” proposals described in Section 2.2.8.4. The “swap” and “sub” proposal types are particularly beneficial when pedFac is constrained to find monogamous pedigrees, because the monogamy constraint makes it difficult for the correct parent pair to be reached, if one of the parents is already attached to an incorrect mating partner. For inferring parentage, pedFac (with the swapping and subbing steps included) performs as well as COLONY and its performance is far superior to that of sequoia in the first four sampling-fraction scenarios (Table 2.2). For all following results, pedFac is run always with the swapping and subbing proposals included.

We also assess the results of simulation scenario 1 in terms of the accuracy with which full-siblings are identified. COLONY provides the maximum likelihood pedigree, and all full-sibling pairs within that pedigree were considered to be pairs inferred by COLONY. By contrast, since pedFac samples over the space of pedigrees, we retained, for this analysis, MAP-estimated sibling pairs—that is full sibling pairs identified in $> 50\%$ of the sweeps (after burn-in).

Over all five fractions of parental sampling, the performance of pedFac is comparable



Figure 2.10: Parentage assignment for the most recent cohort of the empirical Chinook salmon pedigree genotyped for 95 SNPs. Rows correspond to sampling fraction of adults (i.e. candidate parents) and columns show different software and options (see text). In each panel, individuals are ordered according to a “confidence measure” for the inferred trio (see text). The black curve shows this measure (y -axis) for each individual. The violet and blue, vertical tick marks emanating from the black curve, indicate errors. Violet tick marks going upward denote cases in which the individual’s true parents were sampled, but the individual was still incorrectly assigned to one (0.1 units on the y scale) or two (0.2 units in the y direction) incorrect parents. The blue tick-marks, facing downward, are scaled in the same fashion but denote one or two incorrect parentage assignments for individuals whose true parent(s) are not in the data set.

fraction	software	1 - FDR		ROC-AUC	
		mean w/ sd	range	mean w/ sd	range
1	pedFac	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	... (+swap)	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	... (+swap,+sub)	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	COLONY	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	sequoia	0.98 ± 0.01	(0.97,0.98)	1.00 ± 0.00	(1.00,1.00)
0.75	pedFac	0.96 ± 0.01	(0.95,0.97)	0.59 ± 0.03	(0.56,0.65)
	... (+swap)	1.00 ± 0.00	(0.99,1.00)	0.71 ± 0.27	(0.46,1.00)
	... (+swap,+sub)	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.01	(0.99,1.00)
	COLONY	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	sequoia	0.95 ± 0.01	(0.94,0.96)	0.80 ± 0.01	(0.79,0.81)
0.5	pedFac	0.93 ± 0.01	(0.91,0.95)	0.55 ± 0.03	(0.50,0.59)
	... (+swap)	1.00 ± 0.00	(0.99,1.00)	0.71 ± 0.24	(0.42,1.00)
	... (+swap,+sub)	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	COLONY	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	sequoia	0.95 ± 0.00	(0.95,0.96)	0.64 ± 0.02	(0.62,0.66)
0.25	pedFac	0.95 ± 0.01	(0.94,0.96)	0.54 ± 0.02	(0.52,0.57)
	... (+swap)	1.00 ± 0.00	(1.00,1.00)	0.74 ± 0.26	(0.42,1.00)
	... (+swap,+sub)	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	COLONY	1.00 ± 0.00	(1.00,1.00)	0.90 ± 0.22	(0.50,1.00)
	sequoia	0.96 ± 0.00	(0.96,0.96)	0.54 ± 0.01	(0.52,0.55)

Table 2.2: Comparison of parentage precision (1 - FDR) rates and associated ROC AUC scores from 1157 individuals in four varying sample-fraction scenarios of a two-generation monogamous pedigree. The mean and standard deviation is calculated from five replicates.

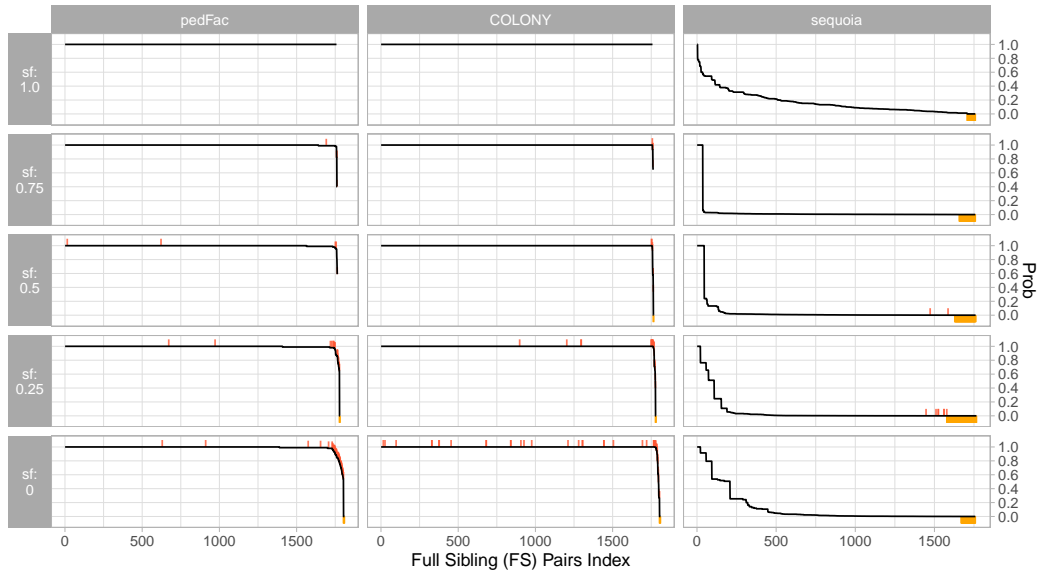
fraction	software	FDR			FNR			ROC-AUC			Runtime (sec)	
		mean w/ sd	range	mean w/ sd	range	mean w/ sd	range	mean w/ sd	range	mean w/ sd	sd	
1.00	pedFac	0.0000 ± 0.0000	(0.000,0.000)	0.0011 ± 0.0025	(0.000,0.006)	1.000 ± 0.000	(1.00,1.00)	411 ± 8				
	COLONY	0.0000 ± 0.0000	(0.000,0.000)	0.0011 ± 0.0025	(0.000,0.006)	1.000 ± 0.000	(1.00,1.00)	502 ± 71				
	sequoia	0.0000 ± 0.0000	(0.000,0.000)	0.0381 ± 0.0096	(0.027,0.048)	1.000 ± 0.000	(1.00,1.00)	119 ± 10				
0.75	pedFac	0.0011 ± 0.0009	(0.000,0.002)	0.0001 ± 0.0003	(0.000,0.001)	0.998 ± 0.004	(0.99,1.00)	391 ± 5				
	COLONY	0.0011 ± 0.0009	(0.000,0.002)	0.0001 ± 0.0003	(0.000,0.001)	1.000 ± 0.000	(1.00,1.00)	1157 ± 233				
	sequoia	0.0001 ± 0.0003	(0.000,0.001)	0.0799 ± 0.0168	(0.057,0.101)	0.962 ± 0.025	(0.94,1.00)	94 ± 9				
0.50	pedFac	0.0046 ± 0.0011	(0.003,0.006)	0.0006 ± 0.0008	(0.000,0.002)	0.969 ± 0.061	(0.86,1.00)	333 ± 12				
	COLONY	0.0049 ± 0.0006	(0.004,0.006)	0.0008 ± 0.0008	(0.000,0.002)	0.989 ± 0.020	(0.95,1.00)	1645 ± 421				
	sequoia	0.0007 ± 0.0008	(0.000,0.002)	0.0666 ± 0.0119	(0.051,0.080)	0.959 ± 0.014	(0.94,0.97)	108 ± 7				
0.25	pedFac	0.0143 ± 0.0043	(0.010,0.020)	0.0022 ± 0.0016	(0.000,0.005)	0.981 ± 0.016	(0.96,1.00)	230 ± 10				
	COLONY	0.0134 ± 0.0028	(0.010,0.017)	0.0013 ± 0.0009	(0.000,0.002)	0.942 ± 0.029	(0.90,0.97)	1982 ± 285				
	sequoia	0.0025 ± 0.0023	(0.001,0.006)	0.0797 ± 0.0183	(0.057,0.106)	0.953 ± 0.016	(0.93,0.97)	165 ± 15				
0.00	pedFac	0.0317 ± 0.0050	(0.027,0.040)	0.0039 ± 0.0011	(0.003,0.006)	0.971 ± 0.017	(0.95,0.99)	87 ± 1				
	COLONY	0.0312 ± 0.0054	(0.025,0.040)	0.0033 ± 0.0017	(0.001,0.006)	0.856 ± 0.041	(0.80,0.91)	2111 ± 217				
	sequoia	0.0016 ± 0.0018	(0.000,0.004)	0.0623 ± 0.0091	(0.051,0.076)	0.967 ± 0.007	(0.96,0.97)	151 ± 4				

Table 2.3: Error rates, ROC-AUC scores and runtime for inferred full sibling pair from the monogamous Chinook salmon pedigree of scenario 1.

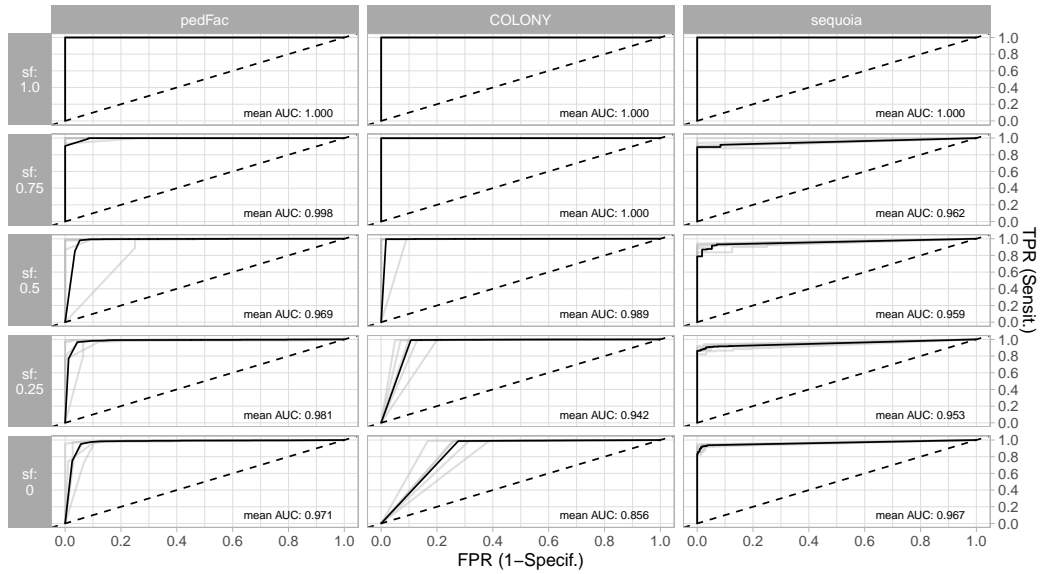
to that of COLONY with respect to false discovery rate and false negative rate, but pedFac outperforms COLONY in ROC-AUC score (Table 2.3). This is a consequence of the fact that the posterior value produced by pedFac, for each pair of siblings, provides a much more accurate estimate of the probability that the pair truly are full-siblings. In contrast, the “probability” scores that COLONY reports do not relate as strongly to the actual rate of mis-characterization of sibling pairs. This is visually apparent in Figure 2.11a, which shows a representative result from one of the five replicate simulations.

When compared to sequoia, pedFac and COLONY have higher false discovery rates but this occurs because sequoia is very conservative in calling full siblings, and consequently has a vastly higher false negative rate than either pedFac or COLONY.

We also examine the accuracy of the entire full sibling groups inferred by pedFac, COLONY and sequoia (rather than summarizing the siblings in pairs). To compose the inferred full sibling groups from pedFac, we sorted the MCMC-sampled full sibling groups in order of decreasing posterior (and decreasing sibling size, when two groups had the same posterior) and then cycled through that sorted list assigning each individual exclusively to the first full-sibling group it belonged to. In the context of full-sibling groups, we define a “false discovery” to be an any inferred sibling group that is not a true full sibling group. Likewise, a false negative is defined as a true full sibling group that does not have an exact match amongst the inferred full sibling groups. Over all five sampling-fraction cases, pedFac performs as well as COLONY at inferring full sibling



(a) Full sibling pairs on the x -axis in descending order of the “confidence metric” (black curve). Red tickmarks above the black curve show inferred pairs that are not full siblings. Orange tick marks below the curve indicate true full sibling pairs that were not identified by the software.



(b) ROC plots of pairs inferred to be full siblings. The average ROC curve (in black) is plotted over the ROC curves for each of the five replicate trials (in grey).

Figure 2.11: Accuracy of inference of full-sibling pairs in the monogamous case study.

fraction	software	FDR			FNR			ROC-AUC		
		mean w/ sd	range		mean w/ sd	range		mean w/ sd	range	
1.00	pedFac	0.0000 ± 0.0000	(0.000,0.000)		0.0000 ± 0.0000	(0.000,0.000)		1.000 ± 0.000	(1.00,1.00)	
	COLONY	0.0009 ± 0.0021	(0.000,0.005)		0.0005 ± 0.0010	(0.000,0.002)		0.900 ± 0.224	(0.50,1.00)	
	sequoia	0.0852 ± 0.0222	(0.065,0.117)		0.0435 ± 0.0116	(0.032,0.060)		0.813 ± 0.049	(0.76,0.87)	
0.75	pedFac	0.0033 ± 0.0039	(0.000,0.009)		0.0051 ± 0.0053	(0.000,0.012)		0.995 ± 0.008	(0.98,1.00)	
	COLONY	0.0046 ± 0.0017	(0.002,0.007)		0.0079 ± 0.0045	(0.002,0.014)		0.900 ± 0.223	(0.50,1.00)	
	sequoia	0.2338 ± 0.0346	(0.193,0.288)		0.1162 ± 0.0176	(0.102,0.146)		0.852 ± 0.028	(0.81,0.88)	
0.50	pedFac	0.0136 ± 0.0019	(0.012,0.016)		0.0241 ± 0.0026	(0.021,0.028)		0.933 ± 0.062	(0.83,0.97)	
	COLONY	0.0178 ± 0.0027	(0.014,0.021)		0.0306 ± 0.0038	(0.025,0.035)		0.912 ± 0.070	(0.81,1.00)	
	sequoia	0.2350 ± 0.0318	(0.195,0.267)		0.1236 ± 0.0219	(0.100,0.150)		0.785 ± 0.014	(0.77,0.81)	
0.25	pedFac	0.0448 ± 0.0119	(0.033,0.064)		0.0792 ± 0.0224	(0.062,0.116)		0.909 ± 0.024	(0.87,0.93)	
	COLONY	0.0437 ± 0.0050	(0.038,0.051)		0.0778 ± 0.0137	(0.062,0.100)		0.868 ± 0.036	(0.83,0.93)	
	sequoia	0.2670 ± 0.0360	(0.231,0.307)		0.1486 ± 0.0240	(0.127,0.176)		0.744 ± 0.016	(0.72,0.76)	
0.00	pedFac	0.1107 ± 0.0109	(0.091,0.117)		0.1894 ± 0.0113	(0.171,0.197)		0.817 ± 0.018	(0.79,0.84)	
	COLONY	0.1044 ± 0.0130	(0.087,0.122)		0.1773 ± 0.0184	(0.153,0.201)		0.795 ± 0.049	(0.74,0.85)	
	sequoia	0.2210 ± 0.0302	(0.196,0.271)		0.1236 ± 0.0242	(0.106,0.164)		0.717 ± 0.011	(0.70,0.73)	

Table 2.4: Error rates and ROC-AUC scores for full-sibling group (rather than pair) inferences from scenario 1

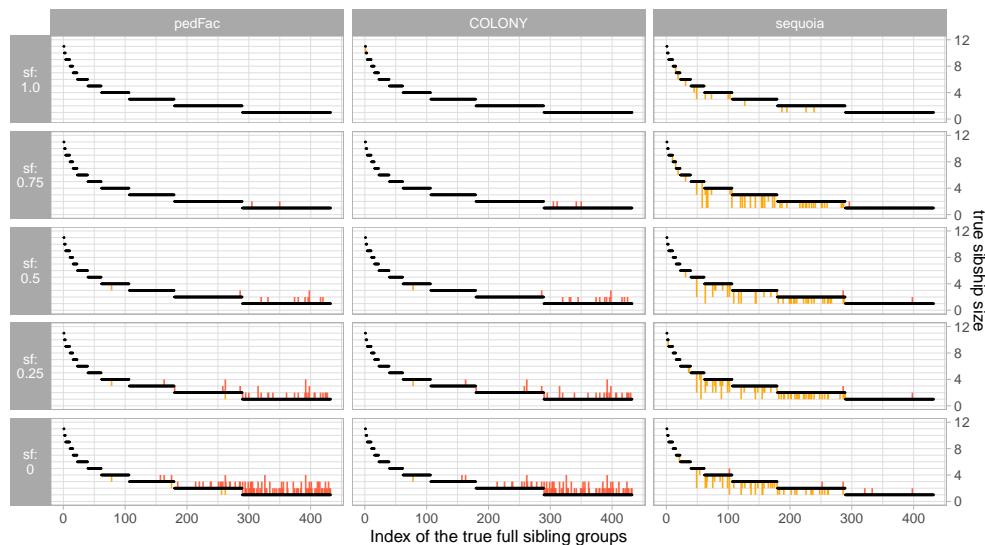
groups, based on the FDR and FNR, and pedFac achieves a higher mean ROC-AUC score than COLONY in all cases (Table 2.4).

As can be seen in Figure 2.12b, pedFac returns more informative posterior probabilities than does COLONY: pedFac’s posterior curve tapers downward with the accumulation of false sibling-group calls. Overall, pedFac and COLONY outperform sequoia by a wide margin.

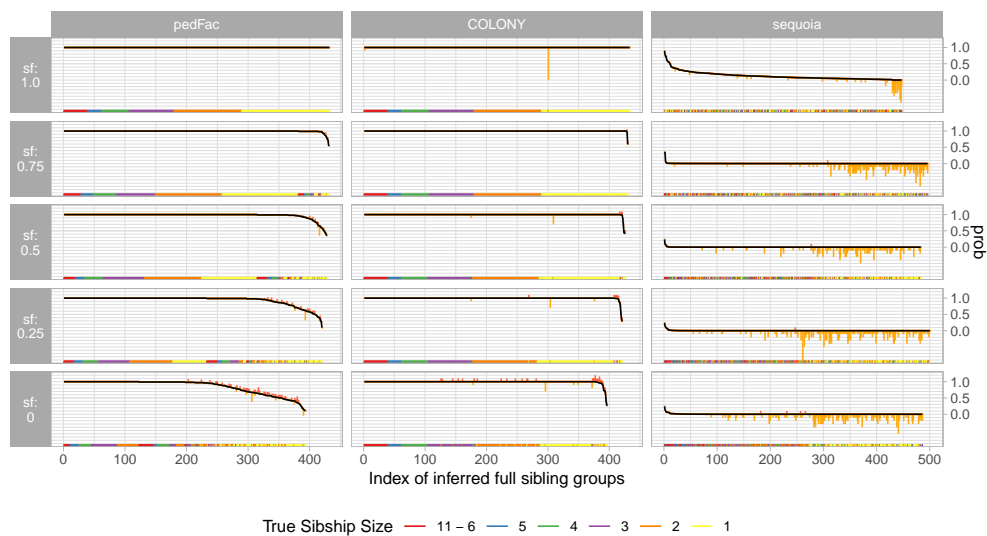
2.3.2 Scenario 2: Parentage And Sibship Inference of Two Generation Acyclic Pedigrees with Polygamous Mating

When males and females have multiple mates, the space of possible pedigrees increases considerably and the pedigree inference problem becomes much harder. Polygamous mating amongst the adults can produce offspring groups of full sibships nested within a potentially wide variety of half-sibling groups.

To simulate polygamous mating, and to assess pedFac’s performance at recovering polygamous pedigrees, we use a single true two generation pedigree, simulated from our pedigree simulator. Over different simulation replicates, genotypes are assigned to individuals according to Mendelian inheritance down the pedigree. The pedigree, itself, is composed of 29 connected components, each of which belong to two separate groups (Figure 2.13). In Group A each mating produces only a single offspring, creating 12 of the connected components; while the adults in Group B all produce more than one



(a) True full sibling groups on the x -axis in descending order of the true sibship size (black curve).



(b) Inferred full sibling groups on the x -axis in descending order of the “confidence metric” (black curve). Colors in the “rug” atop the x -axis denote the sizes of the inferred full sibling groups.

Figure 2.12: Performance in inference of full sibling groups. Red tick marks above the curve indicate the inclusion of one or more members (0.1 unit per member on the y -scale) incorrectly inferred full sibling(s) in the true full sibling group. Orange tickmarks below the black curve show one or more members (0.1 unit per member on the y -scale) from the true full sibling group that are missing from the inferred full sibling group.

offspring and belong to 17 distinct connected components. There are approximately 200 founders in this pedigree, with roughly half males and half females. For each connected component, at least one of the male and female founders display polygamous mating.

Similar to the simulation setup for the monogamous case study, we simulate five replicate panels of 100 SNPs, each with a 2% chance of being genotyped with error and a 0.5% chance of being a missing genotype, for five different parental sampling-fraction scenarios with $f \in \{1, 0.75, 0.5, 0.25, 0\}$. The number of sweeps and all software settings remained the same as the previous run, with the exception of altering the mating system specification (i.e., no software was run in its “monogamous” mode). Confidence scores for sorting COLONY’s inferred half-sibling pairs for ROC-AUC calculation were taken from the “Probability” column of COLONY’s “half-sib dyad” output.

In the category of parental assignment, pedFac performs as well as COLONY in precision and provides the most informative confidence metric among the three software packages (Table 2.5). With respect to inferring whether pairs are full siblings or half siblings, pedFac has the lowest FDR among the three but yields a higher FNR than COLONY, and a lower FNR than sequoia (Table 2.6). Sequoia does not perform well in terms of error rates and the ROC-AUC. Additionally, when all candidate parents are absent from the data set ($f = 0$), sequoia is unable to identify any half-sibling pairs.

Upon close examination of the pedFac results like those shown in Figure 2.14a, we see that, as the sampling fraction of the parent generation decreases, the number of half-

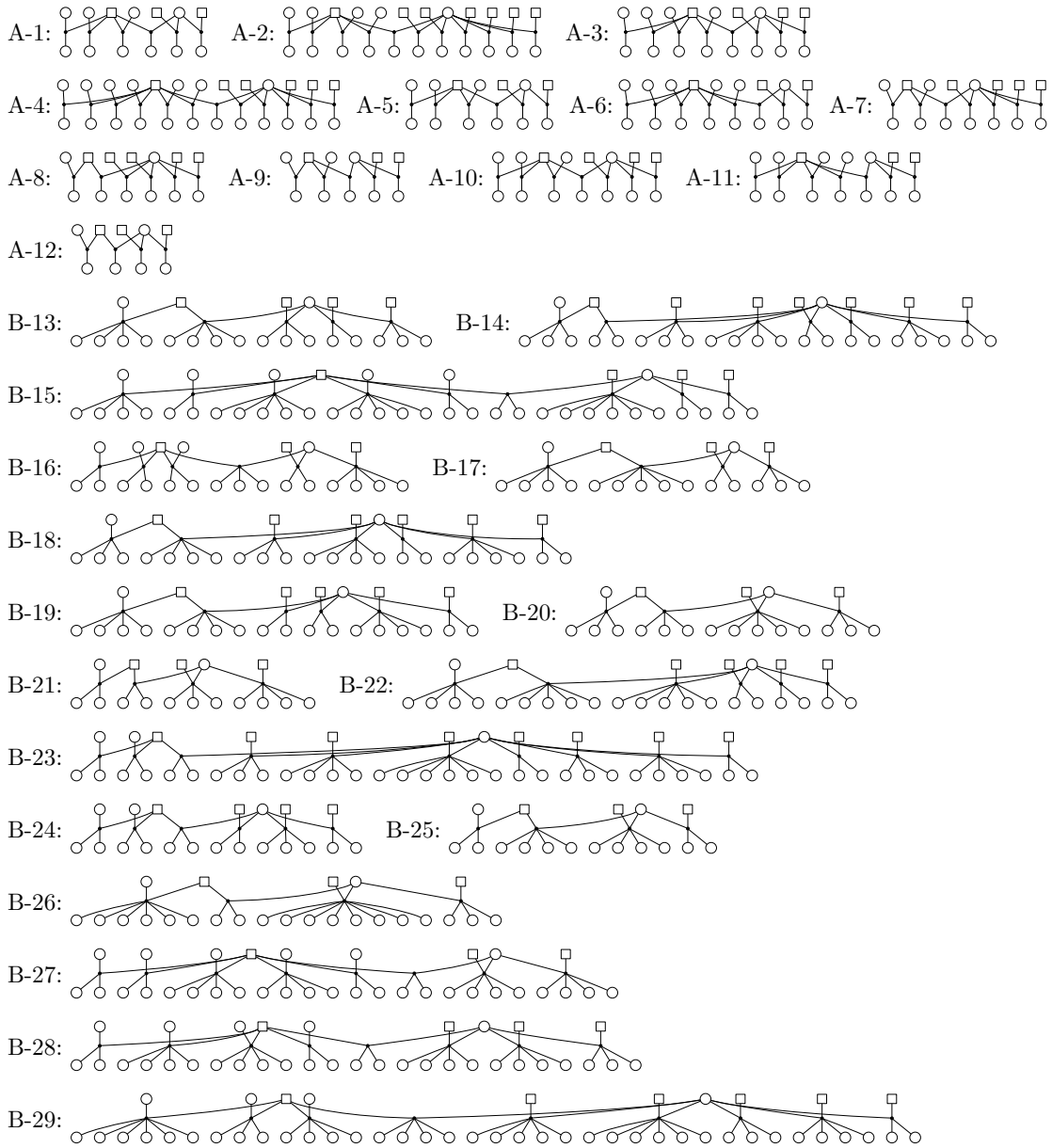
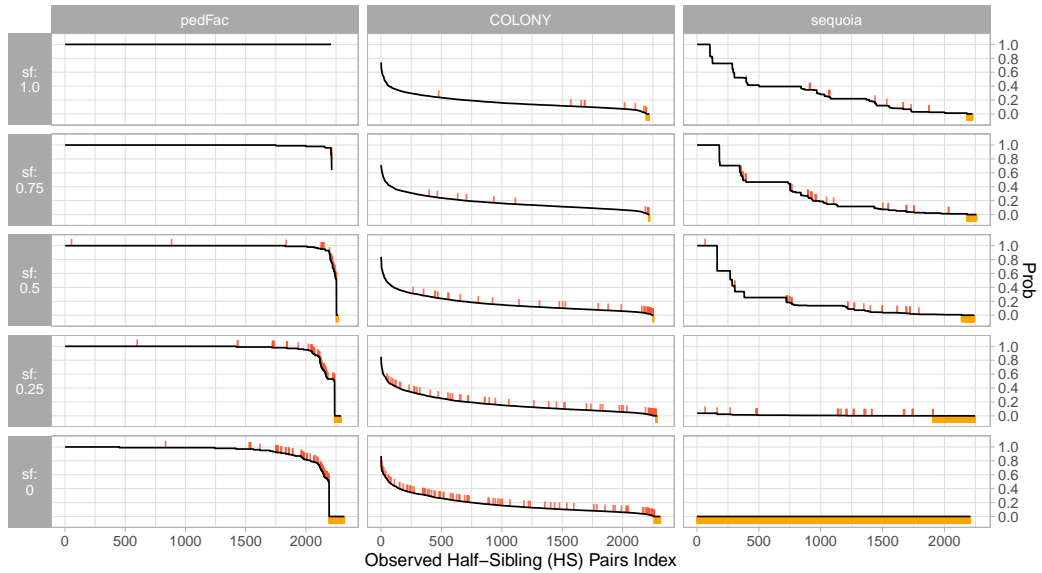
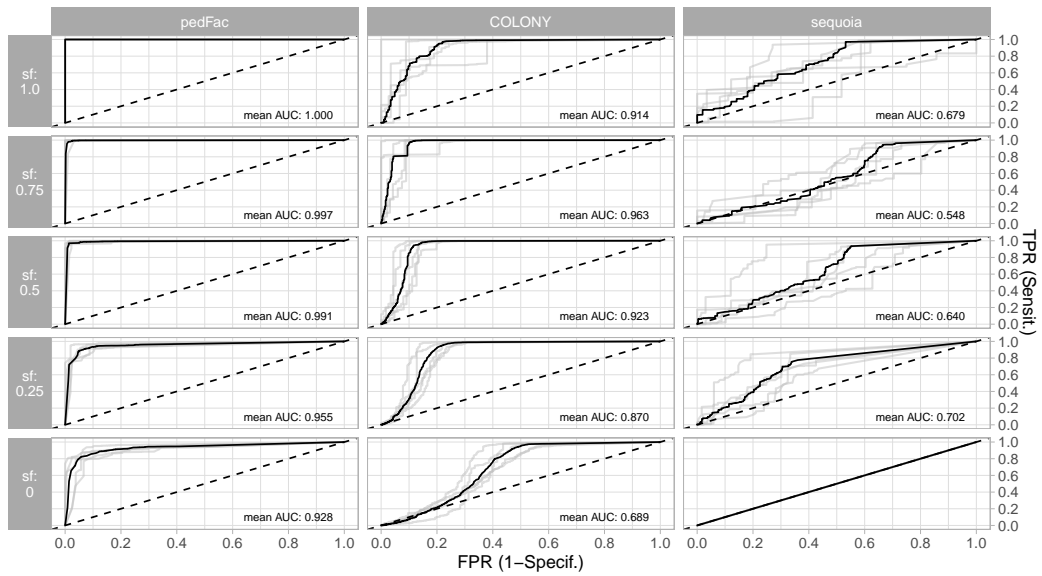


Figure 2.13: Simulated pedigree with 29 connected components used to explore inference for polygamous scenarios. Components from group A and group B are named accordingly.



(a) Half sibling pairs on the x -axis in descending order of the “confidence metric” (black curve). Red tickmarks above the black curve show inferred pairs that are not half siblings. Orange tick marks below the curve indicate true half sibling pairs that were not identified by the software.



(b) ROC plots of pairs inferred to be half siblings. The average ROC curve (in black) is plotted over the ROC curves for each of the five replicate trials (in grey).

Figure 2.14: Accuracy of inference of half-sibling pairs in the polygamous case study.

fraction	software	1 - FDR		ROC-AUC	
		mean w/ sd	range	mean w/ sd	range
1	pedFac	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	COLONY	0.99 ± 0.01	(0.98,1.00)	0.61 ± 0.22	(0.50,1.00)
	sequoia	0.98 ± 0.01	(0.97,0.99)	1.00 ± 0.00	(1.00,1.00)
0.75	pedFac	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	COLONY	1.00 ± 0.00	(1.00,1.00)	0.95 ± 0.11	(0.75,1.00)
	sequoia	0.96 ± 0.01	(0.95,0.97)	0.79 ± 0.01	(0.77,0.81)
0.5	pedFac	1.00 ± 0.00	(1.00,1.00)	1.00 ± 0.00	(1.00,1.00)
	COLONY	1.00 ± 0.00	(1.00,1.00)	0.80 ± 0.27	(0.50,1.00)
	sequoia	0.95 ± 0.01	(0.94,0.96)	0.66 ± 0.03	(0.62,0.70)
0.25	pedFac	1.00 ± 0.00	(0.99,1.00)	0.97 ± 0.08	(0.83,1.00)
	COLONY	1.00 ± 0.00	(1.00,1.00)	0.95 ± 0.11	(0.75,1.00)
	sequoia	0.97 ± 0.01	(0.96,0.97)	0.54 ± 0.01	(0.52,0.56)

Table 2.5: Comparison of parentage precision (1 - FDR) rates and associated ROC AUC scores from 428 individuals in four varying sample-fraction scenarios of a two-generation polygamous pedigree. The mean and standard deviation is calculated from five replicates.

sibling false negatives—true half-sibling pairs that have gone completely undetected by pedFac—increases at a rate slightly higher than that of COLONY. This pattern is likely a consequence of poor mixing in pedFac, and can likely be resolved by introducing additional proposal move types that are conducive to sampling alternative polygamous pedigrees. This topic is addressed in greater detail in the discussion section that follows. It is worth noting that the poor mixing in this chain has also led to a negative impact on pedFac’s inference of full-sibling groups: throughout all five sampling fraction cases, pedFac returns a higher FDR, a mixed FNR result and higher ROC-AUC score compared to that of COLONY; though pedFac outperforms sequoia in all categories (Table 2.7).

fraction	software	Assigned Full-Sibling Pair			Assigned Half-Sibling Pair			Runtime (sec)	
		FDR(mean,sd)	FNR(mean,sd)	AUC(mean,sd)	FDR(mean,sd)	FNR(mean,sd)	AUC(mean,sd)	mean	w/ sd
1.00	pedFac	0.0000 ± 0.0000	0.0000 ± 0.0000	1.000 ± 0.000	0.0000 ± 0.0000	0.0000 ± 0.0000	1.000 ± 0.000	60 ± 1	
	COLONY	0.0000 ± 0.0000	0.0000 ± 0.0000	1.000 ± 0.000	0.0084 ± 0.0092	0.0080 ± 0.0048	0.914 ± 0.057	1089 ± 117	
	sequoia	0.0000 ± 0.0000	0.0419 ± 0.0179	1.000 ± 0.000	0.0103 ± 0.0024	0.0282 ± 0.0232	0.679 ± 0.127	10 ± 1	
0.75	pedFac	0.0009 ± 0.0020	0.0075 ± 0.0025	0.996 ± 0.007	0.0046 ± 0.0028	0.0024 ± 0.0034	0.997 ± 0.004	78 ± 11	
	COLONY	0.0009 ± 0.0012	0.0013 ± 0.0030	0.946 ± 0.121	0.0075 ± 0.0065	0.0010 ± 0.0013	0.963 ± 0.025	1883 ± 313	
	sequoia	0.0009 ± 0.0020	0.0446 ± 0.0180	0.991 ± 0.014	0.0283 ± 0.0038	0.0376 ± 0.0160	0.548 ± 0.101	15 ± 1	
0.50	pedFac	0.0022 ± 0.0038	0.0088 ± 0.0049	0.994 ± 0.005	0.0117 ± 0.0076	0.0055 ± 0.0044	0.991 ± 0.006	279 ± 92	
	COLONY	0.0053 ± 0.0033	0.0031 ± 0.0030	0.948 ± 0.033	0.0142 ± 0.0047	0.0024 ± 0.0010	0.923 ± 0.029	2621 ± 478	
	sequoia	0.0065 ± 0.0039	0.0596 ± 0.0271	0.852 ± 0.162	0.0316 ± 0.0122	0.0646 ± 0.0137	0.640 ± 0.141	18 ± 4	
0.25	pedFac	0.0063 ± 0.0062	0.0221 ± 0.0083	0.984 ± 0.008	0.0265 ± 0.0071	0.0382 ± 0.0216	0.955 ± 0.019	664 ± 62	
	COLONY	0.0145 ± 0.0062	0.0093 ± 0.0083	0.945 ± 0.041	0.0376 ± 0.0096	0.0079 ± 0.0033	0.870 ± 0.027	3049 ± 76	
	sequoia	0.0081 ± 0.0037	0.0764 ± 0.0349	0.897 ± 0.108	0.0433 ± 0.0153	0.2239 ± 0.0757	0.702 ± 0.093	53 ± 14	
0.00	pedFac	0.0108 ± 0.0059	0.0318 ± 0.0135	0.963 ± 0.032	0.0348 ± 0.0101	0.0524 ± 0.0199	0.928 ± 0.025	959 ± 40	
	COLONY	0.0185 ± 0.0082	0.0172 ± 0.0094	0.915 ± 0.041	0.0495 ± 0.0119	0.0206 ± 0.0091	0.689 ± 0.038	2956 ± 567	
	sequoia	0.0094 ± 0.0065	0.1038 ± 0.0409	0.914 ± 0.050	-	-	-	29 ± 0	

Table 2.6: Error rates, ROC-AUC scores and runtime for inferred full and half sibling pair from the polygamous pedigree of scenario 2.

fraction	software	FDR			FNR			ROC-AUC		
		mean w/ sd	range	mean w/ sd	range	mean w/ sd	range	mean w/ sd	range	
1.00	pedFac	0.0000 ± 0.0000	(0.000,0.000)	0.0000 ± 0.0000	(0.000,0.000)	1.000 ± 0.000	(1.00,1.00)			
	COLONY	0.0000 ± 0.0000	(0.000,0.000)	0.0000 ± 0.0000	(0.000,0.000)	1.000 ± 0.000	(1.00,1.00)			
	sequoia	0.0681 ± 0.0214	(0.050,0.098)	0.0349 ± 0.0111	(0.026,0.051)	0.796 ± 0.117	(0.63,0.94)			
0.75	pedFac	0.0204 ± 0.0102	(0.010,0.031)	0.0133 ± 0.0106	(0.005,0.031)	0.993 ± 0.004	(0.99,1.00)			
	COLONY	0.0041 ± 0.0043	(0.000,0.010)	0.0051 ± 0.0051	(0.000,0.010)	0.878 ± 0.211	(0.51,1.00)			
	sequoia	0.0899 ± 0.0248	(0.055,0.116)	0.0482 ± 0.0134	(0.026,0.062)	0.818 ± 0.055	(0.72,0.86)			
0.50	pedFac	0.0335 ± 0.0182	(0.010,0.061)	0.0226 ± 0.0176	(0.005,0.051)	0.872 ± 0.100	(0.75,0.98)			
	COLONY	0.0206 ± 0.0062	(0.015,0.031)	0.0256 ± 0.0126	(0.010,0.041)	0.661 ± 0.281	(0.24,0.98)			
	sequoia	0.1569 ± 0.0418	(0.099,0.210)	0.0923 ± 0.0181	(0.067,0.113)	0.755 ± 0.047	(0.68,0.80)			
0.25	pedFac	0.0660 ± 0.0299	(0.030,0.112)	0.0544 ± 0.0351	(0.015,0.108)	0.884 ± 0.088	(0.74,0.95)			
	COLONY	0.0559 ± 0.0147	(0.036,0.076)	0.0667 ± 0.0202	(0.041,0.087)	0.649 ± 0.120	(0.52,0.78)			
	sequoia	0.1754 ± 0.0432	(0.132,0.244)	0.1077 ± 0.0301	(0.082,0.159)	0.710 ± 0.024	(0.68,0.73)			
0.00	pedFac	0.0910 ± 0.0245	(0.073,0.132)	0.0779 ± 0.0284	(0.046,0.123)	0.831 ± 0.059	(0.74,0.90)			
	COLONY	0.0781 ± 0.0124	(0.066,0.093)	0.0913 ± 0.0225	(0.056,0.113)	0.632 ± 0.093	(0.51,0.75)			
	sequoia	0.2305 ± 0.0476	(0.186,0.294)	0.1426 ± 0.0295	(0.113,0.174)	0.648 ± 0.027	(0.62,0.69)			

Table 2.7: Error rates and ROC-AUC scores for full-sibling group (rather than pair) inferences from scenario 2

2.3.3 Scenario 3: Inference of Multi-Generation Acyclic Pedigrees

So far, we have only presented inference problems involving two-generation pedigrees. In a population where every individual is sampled, inferring a multi-generation pedigree can be reduced to a simple problem of inferring a series of two-generation pedigrees. However, when sampling of the population is not complete, the approach of reconstructing a multi-generational pedigree as a series of two-generation inference problems is not feasible for the simple reason that it requires more than two generations to link an offspring through an unobserved parent to an observed grandparent or another ancestor further back in time.

To show pedFac's performance in inferring relationships through unsampled individuals, we devise two testing scenarios based on multi-generational acyclic pedigrees. In the first scenario, we started with a simulated, five-generation pedigree with 83 members, and manually selected 21 individuals to be considered as unsampled, leaving a sample of 62 individuals from the pedigree (Figure 2.15). We then generate a panel of 200 SNPs markers based on the pedigree. In the second scenario, we simulated a larger three-generation, acyclic pedigree of multiple disconnected components, and 522 individuals in total (Figure 2.17). We mimicked a sampling scenario in which only females can be sampled (such as when returning to land to lay eggs, e.g., sea turtles), and males are never sampled. Additionally, we used the software program MENDEL (Lange et al. 2013) to generate genetic markers from a model that incorporates physical linkage upon

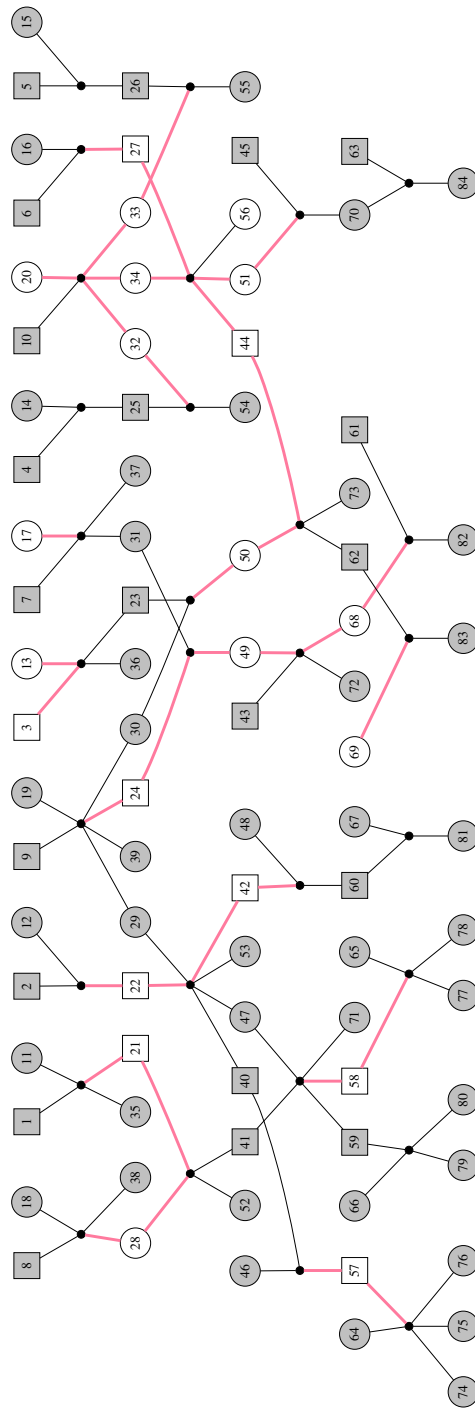
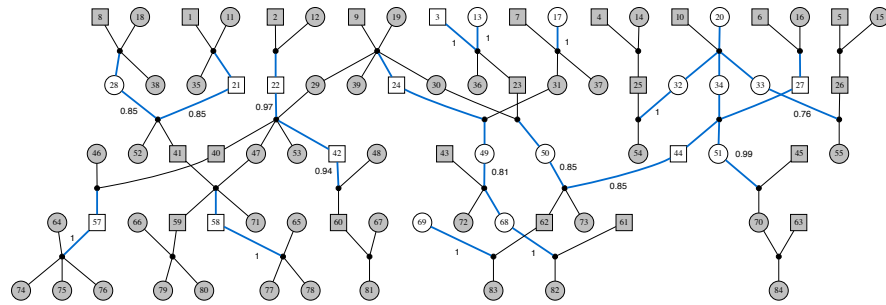
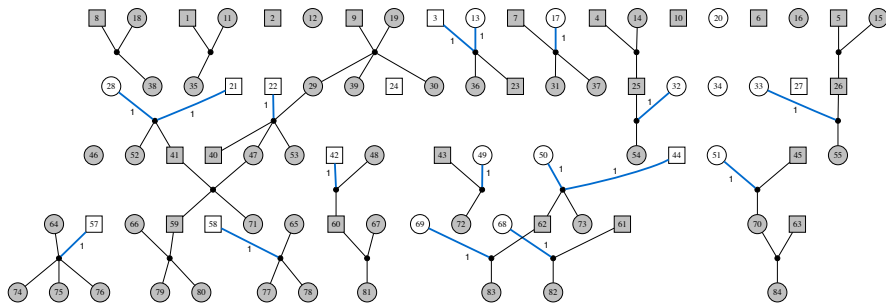


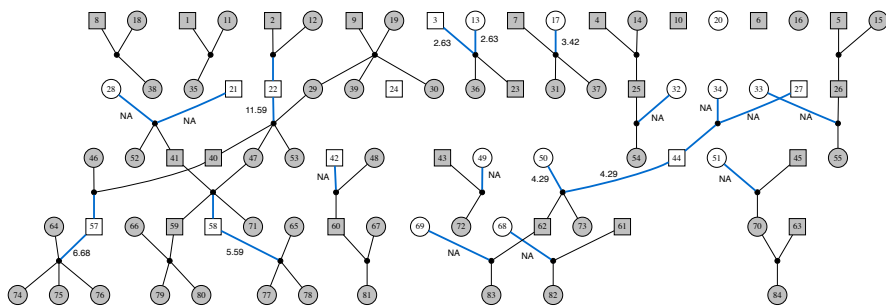
Figure 2.15: Simulated five-generation acyclic monogamous pedigree under incomplete sampling. This reference pedigree is used to test whether pedFac or any pedigree reconstruction software can infer the edges (in pink) through unobserved/un-sampled individuals, given 200 SNP markers. The average size of unobserved cliques increases from left to right.



(a) pedFac result from 100 sweeps with no burn-in. Listed edge weights are posterior probabilities. The pedigree pedFac inferred is resolved as a single connected component.



(b) FRANz result. Edge weights are posterior probabilities from FRANz's MCMC simulation routine. The inferred pedigree is composed of 20 separate, connected components.



(c) Sequoia result. Edge weights are reported log likelihood pair ratios from sequoia's parentage result. The reported pedigree is composed of 16 separate, connected components.

Figure 2.16: Result of multigeneration pedigree inference with incomplete sampling of individuals, and 200 markers. Any paths involving unsampled individuals that are correctly inferred by the respective software are highlighted in blue. Metrics provided by the software for these inferred edges appear as weights adjacent to edges.

chromosomes. We simulated two types of genome: one consisting of 34 autosomal chromosomes, with sizes comparable to those of Chinook salmon; and another consisting of a much smaller number, three, of autosomes, such as might be found in fruit flies. We simulate 5 replicate trials for a panel of 100, 200, 400, and 800 biallelic markers to assess whether performance can be improved through the increasing the number of markers, even if they are physically linked in the genome. In both cases, pedigrees were simulated under an assumption of monogamous mating, since polygamous mating leads to a far higher incidence of cycles within the pedigree (and we are restricting ourselves in this chapter to acyclic pedigrees).

The pedigree of the first multi-generational scenario was analyzed using pedFac (with monogamy enforced), FRANz, and sequoia. Because sequoia performed better than FRANz (though not nearly as well as pedFac) in this first scenario, we did not apply FRANz to the second scenario.

For both of the scenarios, pedFac outperforms sequoia and FRANz in reconstructing multi-generational pedigrees when individuals in the pedigrees are not fully sampled. In the first scenario with 200 independent SNPs, 100 sweeps with no burn-in, the pedigree reconstructed as the MAP estimate of first and second degree relationships from pedFac's sample from the posterior distribution is identical to the true pedigree (Figure 2.16a). PedFac not only accurately inferred all parent-offspring trios but also inferred all edges to or through 21 unsampled individuals. In the result from FRANz, by contrast,

		1 - FDR (mean \pm sd)			
		N=3		N=34	
software	N ^o SNPs	maternal	paternal	maternal	paternal
pedFac	100	0.94 \pm 0.01	0.81 \pm 0.02	0.99 \pm 0.01	0.88 \pm 0.03
	200	0.99 \pm 0.01	0.94 \pm 0.03	1.00 \pm 0.01	0.96 \pm 0.01
	400	1.00 \pm 0.00	0.97 \pm 0.01	1.00 \pm 0.00	0.99 \pm 0.01
	800	0.99 \pm 0.01	0.97 \pm 0.01	1.00 \pm 0.00	1.00 \pm 0.00
sequoia	100	0.91 \pm 0.03	0.56 \pm 0.01	0.93 \pm 0.02	0.57 \pm 0.02
	200	0.96 \pm 0.01	0.57 \pm 0.02	0.99 \pm 0.01	0.58 \pm 0.01
	400	0.98 \pm 0.01	0.57 \pm 0.01	1.00 \pm 0.00	0.59 \pm 0.01
	800	0.98 \pm 0.01	0.58 \pm 0.01	0.99 \pm 0.01	0.60 \pm 0.01

Table 2.8: Comparison of grand-parentage precision (1 - FDR) rates with varying number of SNPs with positions that are uniformly distributed amongst either three chromosomes or 34 chromosomes. Data were simulated from the three-generation monogamous pedigree. The females were observed and all males were unobserved, so identifying paternal grandparents requires inference of edges through unsampled individuals, and is much harder than identifying maternal grandparents. The precision is separated into two categories of identifying grandparents through the maternal or the paternal link. The mean and standard deviation is calculated from five replicates.

no parental relationship was inferred beyond a single unobserved individual, resulting in 20 separate, connected components as opposed to the true pedigree, which is a single connected component. Sequoia was able to infer 6 more edges than FRANz, some of which are connected through unobserved individuals. Of those 6 additional edges, 3 of them are noted to have an ‘NA’ log likelihood ratio.

As for the second scenario, we evaluate the performance of pedFac and sequoia by how accurate they can infer the four grandparents of each individual in the final generation of a three generation acyclic pedigree in which no males are sampled. Across all replicates and chromosomal linkage scenarios, pedFac performs exceptionally well

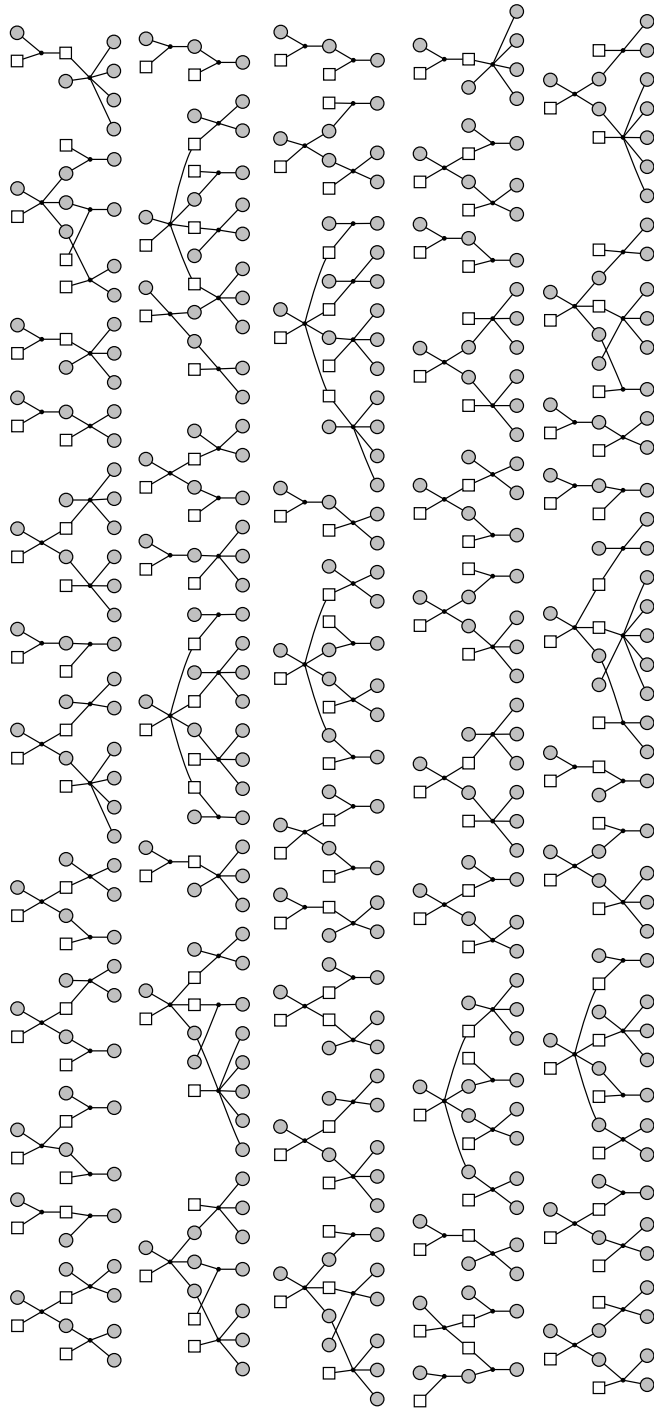


Figure 2.17: Simulated three-generation acyclic monogamous pedigree with only the females sampled. The pedigree is comprises 50 separate connected components. Founders occur throughout the earliest and the middle generation.

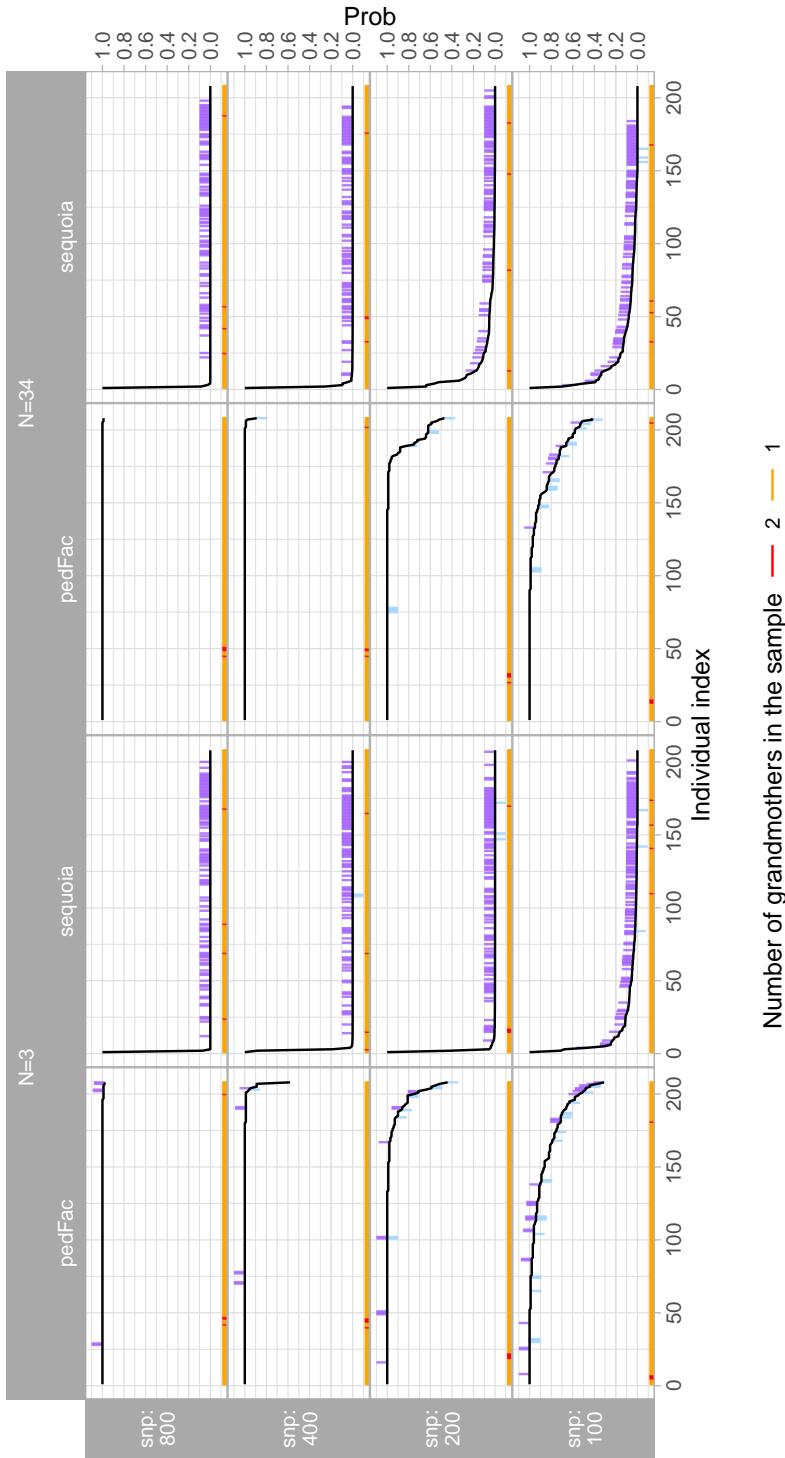


Figure 2.18: Accuracy of inference of grandmothers in the three-generation case study. Rows correspond to total number of markers and columns show different software and the total number of chromosomes across with the markers are placed. Individuals on the x -axis are in descending order of the “confidence measure” for the inferred paternal and maternal grandmother. Violet tick marks going upward denote cases in which the individual’s true grandmother(s) were sampled, but the individual was still incorrectly assigned to one (0.1 units on the y scale) or two (0.2 units in the y direction) incorrect grandmothers. The blue tick-marks, facing downward, are scaled in the same fashion but denote one or two incorrect grandmother assignments for individuals whose true grandmother(s) are not in the data set. The bottom bar along the x -axis is colored according to the number of the individual’s true grandmothers that were sampled.

in inferring the grandparents (Table 2.8). It outperforms sequoia by a large margin in correctly assigning both maternal and paternal grandparents. This success highlights pedFac’s strength in inferring relationships through unobserved individuals—in this particular case, all the males in the pedigree.

We normally expect that for a sensitive robust model, the precision rate improves logarithmically when additional information is given. However, this is not the case for sequoia in inferring correct paternal grandparent pairs. Adding more markers in the sample benefits pedFac much more than it does sequoia for this second-degree relationship inference problem. Additional markers allow pedFac to achieve nearly perfect paternal grandparent assignment when ≥ 400 physically linked SNPs are used. By contrast, sequoia’s precision appears to reach a plateau at about 0.60 correct assignment.

PedFac’s posterior output serves as a better estimate of uncertainty than sequoia’s LLR, supported by the observation that most of the false assignments gravitate toward lower posterior value as seen in one of the five replicates (Figure 2.18). This holds true more for the case of markers spread across 34 chromosomes than for the case with only three chromosomes. Both pedFac and sequoia are built on an assumption that the genetic markers are independent, and are not physically linked. Having such correlated markers reduces the amount of inferential power in PedFac, and, as expected, appears to inappropriately inflate its measure of certainty of some inferences; however, the effect of physical linkage is not dramatic.

2.4 Discussion

Across all cases of various sampling fractions in two-generation, acyclic pedigrees, pedFac performs on an equal footing with COLONY and outperforms sequoia in parentage assignment, sibling-group inference, and full- and half- sibling pair assignment. Additionally, pedFac provides a far better estimate of uncertainty than either COLONY, sequoia or FRANz, and pedFac requires significantly shorter run times than COLONY. pedFac can accurately infer multigenerational acyclic pedigrees under incomplete sampling, a feat that FRANz and sequoia are not capable of accurately doing.

2.4.1 Multigenerational pedigrees

Prior to pedFac and sequoia, it might have been possible to reconstruct multigenerational pedigrees using COLONY by separating the individuals into a set of two-generation bins as input and knitting all of the output into a single output pedigree. However, this approach is limited by its ability to only assign parents to individuals who are observed, similar to the sort of performance observed from FRANz in Figure 2.16b. Hence, an elaboration of a COLONY-based multigenerational strategy could involve assigning putative genotype value for these unobserved individuals (based on COLONY's marginal genotype inferences for unobserved parents) and returning them back to the software for more runs. This overall strategy would require a substantial

amount of overhead work and would rely heavily on imputed genotypes through which it would be very difficult to propagate uncertainty. If the genotype values were not correctly imputed, or even worse, the unobserved individual is a product of a past run's false assignment, this approach would further amplify the error and the rate of false discovery. Hence, to attain a proper estimate of the genotypes of unobserved individuals for relationship inference, it is necessary to have an efficient framework, i.e., pedFac that incorporates the genotype uncertainty of these unobserved individuals in the pedigree, as well as of the observed nodes in the joint likelihood calculation of pedigrees under proposed structural changes.

Sequoia, on the other hand, does not attempt to estimate the genotypes of the unobserved individuals to conduct relationship inference. Instead it bypasses the need for such consideration by focusing on finding pairwise relationships of first to third degree—a total of seven alternative hypotheses—between two observed individuals. We observe that sequoia mostly accepts higher-order relationships between focal pairs only when the relationship link is also partially supported by accepted/identified lower-degree relationships. This is observed in two instances: the inability to infer any half-sibling pairs when all parents are missing from the sample (Figure 2.14a), and the poor grandparentage inference on the paternal side when all males are missing from the sample (Table 2.8). Across all scenarios, sequoia accepts assignment conservatively due to its heuristic, pairwise, log-likelihood approach, and for such inferences, the relatively small number of

markers provided in our simulations creates a difficult challenge for sequoia to identify and infer higher-order relationships. According to its manual, the log pair-wise model is also sensitive to rate of genotyping error. The genotype error rate of 2% we imposed in the simulated data set is much higher than the rate of 0.1% found in (Huisman 2017), but also more in line with observed genotyping error rates in many laboratories.

2.4.2 The assumption that markers are independent

As the joint likelihood of a pedigree is computed in pedFac as the product over SNPs, it is apparent that an underlying assumption is that the markers are independently segregating and that they are not in linkage disequilibrium. By using the software MENDEL, we were able to simulate genetic marker data from two different model genomes: one that is similar to the Chinook salmon genome in 34 chromosomes, and another in only 3 chromosomes. We show that even when the markers are physically linked, pedFac's accuracy can be increased by adding additional linked markers to the data set. However estimates of uncertainty obtained from the high linkage scenario (3 chromosomes) are worse than in the low linkage (34 chromosomes) scenario, which itself produces less accurate estimates of uncertainty than the assumed case of no independent assortment.

Physically linked markers could be handled comprehensively by constructing a first order Hidden Markov model of inheritance vectors along each chromosome as outlined

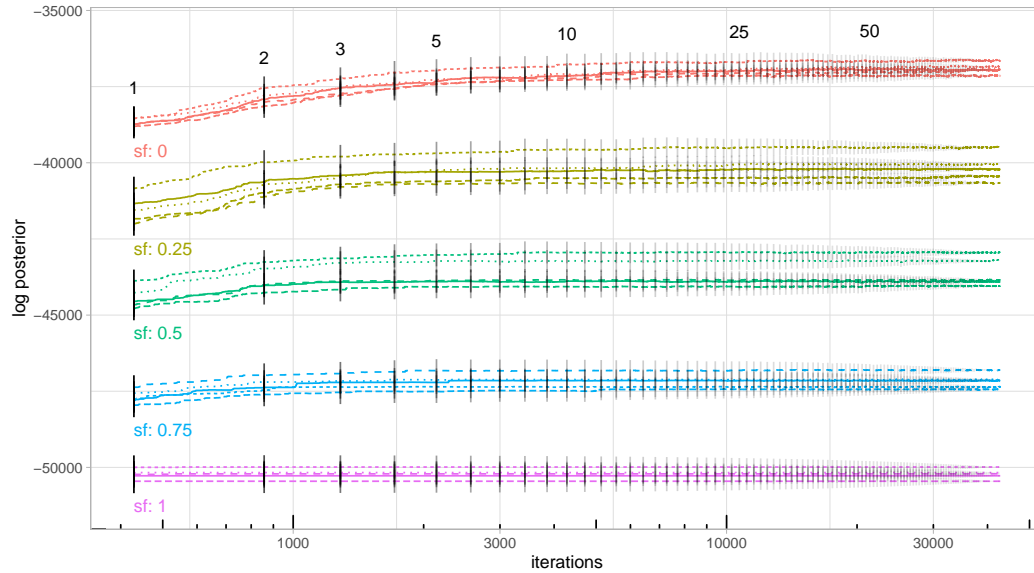


Figure 2.19: pedFac’s log posterior chain under 100 sweeps for scenario 2. The log posterior curve (drawn along the log-based x -axis) begins with the log posterior value at the end of the first sweep and ends with the log posterior value at the end of the 100-th sweep. Each log posterior curve is colored according to the sampling fraction study it belongs to and each line type corresponds to one of the five replicate trials. Each black vertical line on the log posterior curve demarcates the start and end of a sweep cycle, with some labeled with a n -th sweep number. An iteration on the x -axis refers to a proposal to update a particular focal individual. In each sweep, every individual in the pedigree is taken, in turn, as the focal individual.

in the Lander-Green algorithm to learn the derived parental origin. (Lander and Green 1987). In the context of pedigree inference, this would be extraordinarily costly in terms of computation, given that the true pedigree is unknown, and, thus, all individuals' connections are subject to change, and unobserved individuals may be introduced. A more ad-hoc approach would involve altering the transmission probability only for highly selected pairs of markers with substantially low recombination rate.

Fortunately, however, it appears that the true occurrence of physical linkage does not badly degrade pedFac's performance. In the context of physical linkage, pedFac's existing algorithm can be interpreted as a sort of composite likelihood approximation.

2.4.3 Mixing of the MCMC Chain

All the studies presented so far were run with 100 sweeps. We show here, by monitoring the log of the pedigree posterior, that by the 20-*th* sweep for all of the runs for scenario 2, the MCMC appears to have reached a state of convergence (Figure 2.19). We also observe that as the fraction of observed parental individuals increases, a smaller number of sweeps is required for the chain to reach a state from the stationary distribution. This accords with our reasoning that a decrease in the fraction of sampled individuals increases uncertainty and hence the time required for pedFac to estimate the genotype distributions of the unobserved individuals.

Overall, there's no fixed formula for defining the necessary number of sweeps as

long as we meet two conditions: the convergence of the MCMC chain and the desired precision of the posterior estimate. As discussed earlier, for some scenarios, we could have chosen to shorten the number of sweeps to obtain comparably short run times as sequoia. However, doing so would likely decrease our precision in the posterior estimate.

Regardless of the length of run chosen, as with any MCMC exercise, in practical use of pedFac, we recommend performing multiple runs from different random seeds and comparing the output to ensure the chain is reliably converging.

2.4.4 Limitations

All of the case studies we have presented require the true pedigree to be acyclic because the acyclic sampler, by its very design (and name), does not allow the creation of loops when proposing novel pedigree rearrangements. The main reason for this restriction is that sum-product algorithm, even in its loopy, iterative version, is not guaranteed to converge to the likelihood if the pedigree is cyclic.

Through the process of creating simulated, acyclic multigenerational pedigrees for testing pedFac, it was quite apparent that it is difficult to avoid creating any loops while keeping the number of individuals and connected components in a pedigree small. In general, the higher the fraction of population that is sampled, the more likely one is to encounter loops in the true pedigree of the sample. The same is true when sampling is conducted over many generations: as additional generations are added to a multigener-

ational pedigree, it becomes increasingly likely that cycles will be formed. Accordingly, in order for pedFac to be practically useful for multigenerational pedigree inference, it is imperative to address loops in pedigree. The next chapter describes the challenge and a solution to it.

Chapter 3

Learning Multigenerational Pedigrees that Contain Cycles

The previous chapter presented an MCMC scheme for sampling over the space of acyclic pedigrees proportional to their posterior probability, given genetic data. This approach is appropriate for pedigree reconstruction in cases where cycles (or “loops”) are absent in the true pedigree that is being inferred. Such acyclic pedigrees will be encountered when the scope of analysis is limited to genetic data from a small number of generations (like two or three) from organisms displaying low levels of polygyny and polyandry. However, when analyzing data from three or more generations—especially in populations of promiscuous organisms—loops are almost certain to occur in the true pedigree, and will be particularly frequent in well-sampled, small populations, such as

those of conservation concern (Table 3.1).

When the true pedigree is cyclic, the acyclic pedigree sampler will fail to pair true parents that lie along a cyclic path, because the acyclic sampler, by definition, disallows proposals that form cyclic pedigrees. Furthermore, the error will not be restricted to a single misassigned parent pair, as that single error can induce a chain of incorrect pedigree edges throughout other members within the same connected component. Thus, in order to perform inference of multigenerational pedigrees in all but the simplest scenarios, it is imperative to be able to sample over the space of acyclic *and* cyclic pedigrees. This chapter addresses that need.

Before delving into the details of our method for sampling cyclic pedigrees, we briefly discuss why loops pose a problem for pedigree inference, and we explain why existing approaches to calculating approximate probabilities on factor graphs with loops (such as loopy belief propagation) are not applicable to learning graphical pedigree structure. We then introduce and categorize the two main types of simple loops that occur in pedigrees, and note that in most large pedigrees these simple loops can combine into much more complex cycles. In order to efficiently represent and manipulate such complex loops, we introduce the notion of a *reduced cycle basis*, a minimal set of linearly independent paths that forms the cycle basis (i.e., they span all the cycles in the pedigree), and use this system to identify individuals that are potential *loop breakers* in a cyclic pedigree. Finally, we describe an MCMC sampling scheme (within a Metropolis-

species name	source	N° of gen	N° of individuals			N° of loop types			
			total	frac. in loop	inbreeding	marriage	complex	total	
Great tit (<i>P. major</i>)	(Firth et al. 2015)	12	1771	0.50%	1	0	2	3	
Chinook salmon (<i>O. tshawytscha</i>)	(Clemento 2014)	5	14458	28%	12	2	1872	1886	
Song sparrow (<i>M. melodia</i>)	(Reid et al. 2019)	13	3344	17%	25	3	717	745	
Red deer (<i>C. elaphus</i>)	(Gauzere et al. 2020)	9	6382	19%	10	0	2448	2458	
Banded Mongoose (<i>M. mungo</i>)	(Wells et al. 2018)	10	2169	19%	4	10	521	535	
Florida scrub-jay (<i>A. coerulescens</i>)	(Chen et al. 2019)	11	6358	14%	1	5	488	494	

Table 3.1: Loops occurring in pedigrees inferred or observed from a variety of natural populations. of gen: number of generations; total: total number of individuals in the pedigrees; frac. in loop: fraction of all individuals participating in loops. Inbreeding, marriage, and complex are types of loops in pedigrees as defined in Section 3.2. Loops were identified via depth-first search, and exact numbers of different loop types can vary depending on search order. However fraction of individuals in loops is invariant to search order.

Hastings framework) that involves temporarily assigning values to the latent genotypes of these loop breakers, which effectively breaks the loops in the pedigree, allowing the exact sum-product algorithm to be applied, ultimately enabling sampling over loopy pedigree structures.

To evaluate the improvement this new algorithm offers, we compare results achieved with the cyclic pedigree sampler to those obtained using the acyclic pedigree sampler on simulated pedigrees with varying loop complexity.

3.1 The Problem with Loops in Pedigree Inference, and Evaluation of Possible Solutions

In the previous chapter, we showed that the sum-product algorithm efficiently calculates the joint probability of the observed genotypes of all the individuals connected in an acyclic pedigree. This joint probability is the likelihood of the acyclic pedigree, and hence can be used to sample over the space of pedigrees, proportional to their posterior probability. When, the pedigree factor graph is acyclic, the sum-product algorithm is guaranteed to run to completion, and it will, in the process, compute two messages along every edge; each message is computed only once, and the two messages on an edge travel in opposite directions from one another. However, when the pedigree factor graph has a cycle in it, the sum-product algorithm is unable to complete its message-

passing protocol because an outgoing message can be passed along an edge from a node only upon the condition that all the incoming messages along the remaining edges to the node have been received. Thus, a cycle in the graph always creates a “gridlock” situation where incoming messages to a node cannot be received until outgoing messages have been sent. This causes the algorithm to “get stuck” before all the messages can be sent (Figure 3.1).

In many applications involving cyclic factor graphs, the approach known as the Pearl Polytree algorithm, or loopy belief propagation, LBP, (Pearl 1988) is a popular method to approximate the marginal probabilities of the variable nodes. Rather than initiating message passing from the external factor nodes (as in Figure 3.1), in LBP the two messages along each edge in the factor graph are assumed to always be present and are assigned arbitrary initial values at the outset of the algorithm. This eliminates the “gridlock,” because every factor and variable node is now receiving incoming messages from every adjacent edge, so they are all in a position to send outgoing messages. After initialization, the LBP algorithm runs simply by iteratively updating the messages passed in and out of each node using the standard sum-product operations, with an additional step of normalizing the messages at each iteration. It is hoped (but not guaranteed) that, over the course of these iterations, the messages converge to values that allow the calculation of approximate marginal probabilities for the variable nodes in the factor graph.

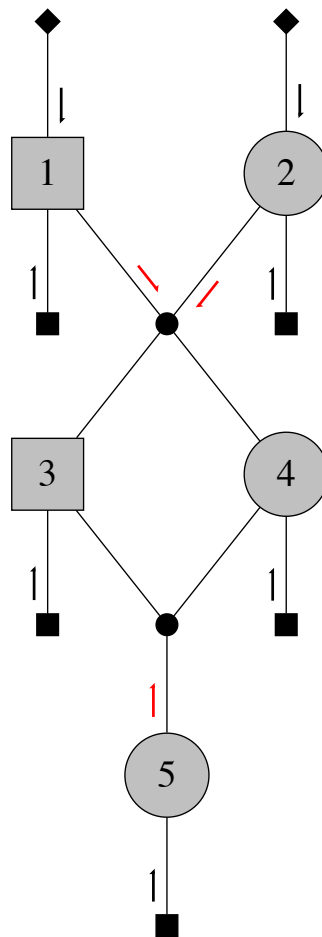


Figure 3.1: A simple example of “message gridlock” in a pedigree with a cycle. During the first step of the sum-product algorithm on this cyclic pedigree, messages (black arrows in the figure) from the p -nodes and g -nodes are passed to the attached variable nodes. In the second step, these outermost variable nodes (individuals 1, 2, and 5) are able to pass messages (depicted as red arrows) out from themselves; however, neither of the m -nodes that receive those messages can send any further messages, because each awaits incoming messages (which never arrive) on two edges.

LBP has enjoyed tremendous success in some instances, such as error correcting codes (Berrou et al. 1993) and computer vision problems (Freeman et al. 2000); however, the performance of LBP in novel problems is unpredictable, and can be quite poor (Murphy et al. 1999; Mooij 2007). Accordingly, the improvement of LBP remains an active area of research (Tatikonda and Jordan 2013; Mooij and Kappen 2012), spawning methods such as “unwrapped” loopy networks (Weiss 1997) and the imposition of consistency constraints to improve accuracy (Mooij 2007).

Ultimately, however, even if LBP upon a pedigree factor graph could be tuned to give reliable approximations, this would not be enough to make it useful for sampling over the space of pedigree configurations. LBP provides estimates of the marginal probabilities of the variable nodes, but sampling over pedigree structures requires the calculation of the joint probability of all the variable nodes upon the network. This is readily available from the sum-product algorithm on an acyclic pedigree, but, because the messages in LBP are locally normalized, LBP cannot deliver the required joint probability. Accordingly, we cannot use it for inferring pedigrees.

Other methods, besides LBP, to approximate or compute the joint probability of a cyclic pedigree, fall into three general categories: clustering, stochastic simulation, and conditioning (Pearl 1988). Clustering involves judiciously combining a group of variables (which are involved in a cycle) together, into a single compound variable, such that a graph expressed in terms of the compound variable is a singly connected network

or “junction tree” (Koller and Friedman 2009; Meiri et al. 1991). Although combining variables in this fashion provides a way to calculate, exactly, the joint likelihood of a cyclic pedigree, the computation to do so increases exponentially with the number of individuals grouped into the compound variable, making this practically infeasible for all but the smallest, simplest pedigree loops. Furthermore, choosing which variable nodes to merge so as to minimize the resulting computation needed is an NP-Hard problem (Cooper 1990). Finally, the graphical object created by combining variables in a pedigree factor graph is no longer a pedigree factor graph, and, in fact, can be difficult to recognize as a pedigree. Consequently, using a clustering approach would require that we abandon the pedigree factor graph representation and all the advantages (for example efficient reuse of messages to calculate the likelihood of pedigree modifications) it confers. Thus, although clustering and the junction-tree algorithm have been used for computations on known pedigrees, with fixed structure (Slooten 2011), we find limited prospects for applying such approaches to reconstructing unknown pedigrees.

Another approach to evaluating probabilities in complex (i.e., loopy) belief networks is stochastic simulation, in which realized values for the nodes at the top of the network are simulated from the priors, and then values for their daughters (and their daughters) are successively simulated and probabilities are estimated by the frequency of occurrence of different events. Such a Monte Carlo approach has been used to estimate probabilities of inbreeding and coefficients of relationship within a given pedigree since

the early 1900's (Wright and McPhee 1925). More recently such a simulation approach has come to be known as the "gene dropping" method (MacCluer et al. 1986), as it involves segregating simulated genes down through the pedigree. This works well for evaluating genotype probabilities for only a few pedigree members at a time. Unfortunately, trying to evaluate the joint probability of all the genotyped members of a pedigree using this simulation method is not practical, because the probability of any (joint) configuration is so small as to never occur during simulation. Furthermore, to sample over the space of pedigrees requires calculating the likelihood of a large number of possible (re)configurations of the pedigree, and this cannot be done rapidly enough via simulation. Accordingly, stochastic simulation is not an option for the inference of different pedigree structures.

A third possible approach to evaluate probabilities within a cyclic pedigree network is through "conditioning" which involves iteratively conditioning upon the possible genotypes carried by sets of individuals participating in loops, and evaluating the joint genotype probability as a weighted sum over all those different conditioning values. An early application of such an approach to pedigrees was the *peeling* algorithm of Cannings et al. (1978). This method works well on pedigrees with one, or a few, simple loops but is not practical for pedigrees with multiple or complex loops, since the computational cost of summing over all possible genotype values increases exponentially with each loop-breaking individual required. Peeling was not proposed within a

pedigree-factor-graph framework, nor has it been used outside of applications to known and fixed pedigrees; however, it is straightforward to see that the act of conditioning upon one or more loop-breakers can transform a cyclic pedigree factor graph into an acyclic one. Ultimately, out of the three methods of clustering, stochastic simulation, and conditioning, we choose conditioning to break cycles and compute the likelihood of cyclic pedigrees (conditioned on the loop breakers) as it allows us to preserve the factor graph representation, thereby enabling efficient re-use of precomputed messages from the sum-product algorithm for evaluating a wide range of proposed pedigree reconfigurations.

In order to simulate from the posterior distribution of pedigree structures, it is not sufficient to merely condition upon one set of genotypes within a single, fixed set of loop breakers. Rather, the individuals chosen to be conditioned upon—and their latent genotypes—must be, themselves, continuously resampled in accordance with their posterior probability. In other words, the loop-breaking individuals and their latent genotypes are sampled as part of the MCMC chain that samples over the space of pedigrees. Doing so requires keeping careful account of cycles within the current and proposed pedigree structures in the MCMC chain over pedigree factor graphs. We discuss our approach for doing so in Section 3.2, before describing the “conditioning” method in the remainder of the chapter.

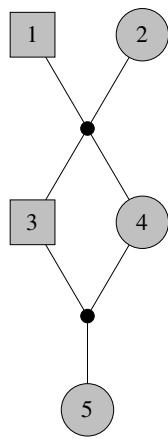
3.2 Loops in Pedigree Factor Graphs

3.2.1 Types of Loops

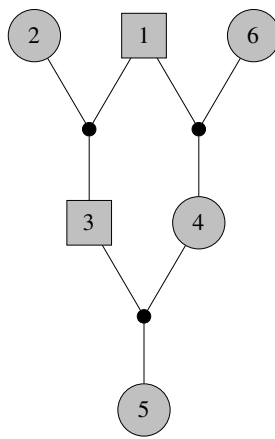
In the field of graph theory, a *cycle* is defined as a path that departs a vertex, and eventually returns to that initial vertex, possibly after passing through other vertices. By contrast, a *loop*, in graph-theory parlance, is typically defined as a cycle that has a path length of one. In other words, a loop leaves a vertex and returns to it before intersecting any other vertices. Within a valid pedigree it is not possible to have a loop as defined in the graph-theory sense, and, in fact, in the genetics literature, the term “loop” has been used to refer to any type of cycle in a pedigree (for example, Cannings et al. 1978). Therefore, in the context of pedigrees and pedigree factor graphs, we adopt the term “loop” in the population-genetic sense, using it interchangeably with “cycle.”

In a pedigree factor graph, in the absence of self-fertilization, it is evident that the length (number of edges) of any cycle must be an even number ≥ 4 , with the cyclic paths running through marriage factor nodes and individual variable nodes (alternating between those two types of nodes). All cycles in a pedigree factor graph can be categorized into three types: 1) inbreeding loops, 2) marriage loops, and 3) complex loops, which contain features from both inbreeding loops and marriage loops.

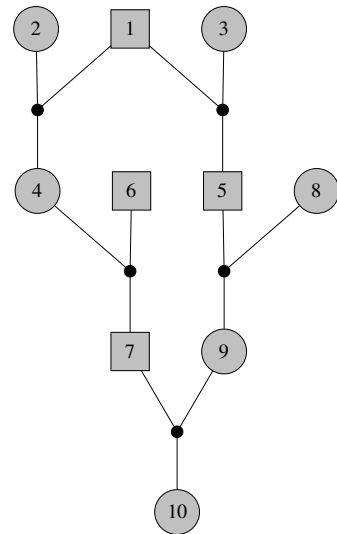
An inbreeding loop, as the name suggests, is created from reproduction between individuals that are closely related, such that their offspring are *inbred*. To be inbred,



(a)



(b)



(c)

Figure 3.2: Three pedigrees, each containing a single inbreeding loop. In (a), full-sibling individuals 3 and 4 produce an offspring, 5, creating the smallest inbreeding loop possible, a loop of size 4. The offspring has a $1/4$ chance that both of the gene copies it carries are descended from a single gene copy amongst its grandparents (1 and 2). An inbreeding loop can also be formed by the marriage of half-siblings, as in (b), where the variable nodes in the loop include not just the parents (3 and 4) of the inbred individual, but also their shared parent (individual 1). (c) Larger inbreeding loops can also be found in pedigrees that span multiple generations when related individuals produce offspring.

here, means to have a chance of receiving gene copies from one's mother and father that are *identical by descent*, meaning that both gene copies have descended from the same gene copy possessed by an ancestor in the pedigree. A simple example of an inbreeding loop is the marriage of two full siblings (Figure 3.2a). This inbreeding loop contains two full sibling individuals and the two marriage nodes above and below them. The marriage of half-siblings also forms an inbreeding loop, which additionally includes the shared parent of the half-sibling pairs (Figure 3.2b). In fact, the production of offspring by any two related individuals produces an inbreeding loop in the pedigree (e.g., Figure 3.2c).

Another loop category is the marriage loop—a closed marriage chain which runs through marriage factor nodes and the polygamous parents they are connected to, typically running through the edges of alternating male and female parents. An example of an open (non-loop-forming) chain could occur as follows: male A and female B have an offspring W, but also male A and female C have an offspring X, and then female C and male D have an offspring Y (Figure 3.3a). Given such an open chain, if male D also has an offspring Z with female C, then a loop is formed (Figure 3.3b). Notice that although the marriage loop creates a cycle in the graph, none of the individuals in Figure 3.3b is inbred.

While marriage loops and inbreeding loops are the two essential types of loops in a pedigree, it should be noted that a variety of *complex loops*, combining features of both

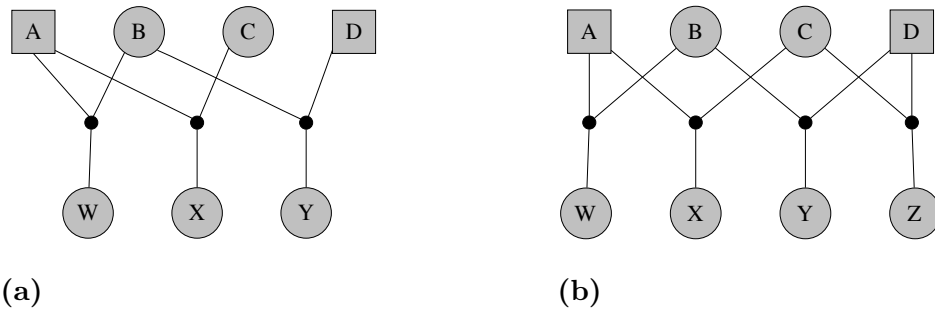


Figure 3.3: Marriage chains and loops. (a) A pedigree with an open marriage chain. (b) Addition of the mating event between female C and male D, producing offspring Z, creates a marriage loop by closing the marriage chain in a.

marriage chains and inbreeding loops, can occur in large pedigrees. They are found to be the predominant type among the three in pedigrees of natural populations (Table 3.1).

3.2.2 The Reduced Cycle Basis of a Loopy Pedigree

Across the spectrum from simple to complex loops in a pedigree, all the cycles can be decomposed into a minimal set of simple cycles known as the *undirected cycle basis*, B (Chartrand et al. 2010). Each member of B is a single, simple cycle, and every cycle in the pedigree factor graph can be obtained as the union of some members of B . The undirected cycle basis is a cornerstone from graph theory for the analysis of cyclic graphs; however it does not always, without further refinement, provide a straightforward and efficient means of identifying loop-breaking individuals (variable nodes) in a pedigree factor graph. We can use the pedigrees in Figure 3.4 to demonstrate this point.

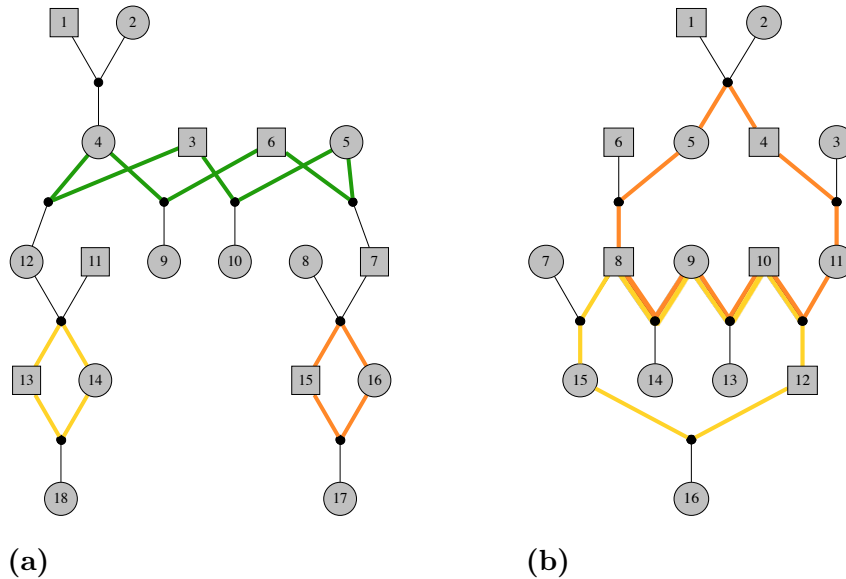


Figure 3.4: Disjointed and intersecting cycle bases in pedigrees. (a) A pedigree with three simple cycles consisting of a marriage loop (outlined in green) involving $\{3, 4, 5, 6\}$ and two full-sibling inbreeding loops (outlined in orange and yellow) involving the sibling pairs $\{13, 14\}$ and $\{15, 16\}$, respectively. Since all three cycles are fully disjoint, choosing, as a loop breaker, any single variable node from each cycle of the undirected cycle basis B would transform this pedigree into an acyclic one. (b) When the pedigree has intersecting cycles (marked in yellow and orange), selecting loop breakers requires additional attention because simply choosing one loop breaker from each cycle may not suffice. For example choosing individual 9 from the orange loop and individual 10 from the yellow loop does not break all the loops in the pedigree, because 9 and 10 are part of a non-terminal intersecting path between the two cycles.

In the relatively simple pedigree in Figure 3.4a, there are three cycles (green, yellow, and orange), that are disjoint from one another and are all chordless. As such, B for this pedigree is merely the set of these three simple cycles. In this case, selecting a single individual to be a loop breaker from each of the three cycles would break all the cycles in the pedigree. Thus, in this trivial case, the undirected cycle basis provides a useful guide for identifying loop breakers.

In a more complex pedigree (Figure 3.4b) the cycles can still be decomposed into the undirected cycle basis B (consisting, for example, of the orange and yellow simple cycles in the figure); but since all the cycles in the graph are neither disjoint nor chordless, the cycles in B do not provide a particularly useful guide for identifying loop breakers. In this case, choosing a single variable node from each element of B to be a loop breaker does not necessarily eliminate all the cycles in the pedigree. Specifically, selecting as loop breakers the two nodes (9 and 10) on the nonterminal intersecting path between the elements of B will not release all the cycles; however selecting just the single variable node 8 (which occurs at the terminus of an intersecting path) would release *all* the cycles in the pedigree.

From the foregoing, it should be clear that the undirected cycle basis, B , by dint of the fact that its elements can share paths (i.e., the parts that are *both* yellow and orange in Figure 3.4b), is not a representation that is well-suited to rapidly identifying the loop breakers required to render a pedigree acyclic. For such a purpose, we develop a novel

basis representation for the cycles in the pedigree. We call this the reduced cycle basis or *rcb*.

The *rcb* is a minimal set of contiguous or non-contiguous, *non-overlapping* paths that satisfy the conditions that a) selecting any single variable node as a loop-breaker from each *rcb* member guarantees that all the loops will be “broken”, and b) the number of members in the *rcb* is the same as the number of elements in the cycle basis, B . The members of an *rcb* are paths which may or may not be, themselves, cycles; however, each cycle in the pedigree can still be obtained by the union of a subset of paths within the *rcb*. Furthermore, any two members (paths) within an *rcb* will intersect one another at, at most, two nodes, and no edges are included in the intersection of any (or all) members of the *rcb* (this is what is meant by “non-overlapping”).

To illustrate the relationship between the undirected cycle basis B and the reduced cycle basis (*rcb*) of a pedigree, it can be helpful to focus only on the structure of loops within the pedigree, as depicted using a simple “circle and arc” diagram (Fig 3.5). In such a diagram, the nodes and edges are implicit, but not explicitly drawn, in the paths of the loops. In the undirected cycle basis B , all simple cycles can be represented as intersecting or separate circles (Fig 3.5a), while the paths which are members of an *rcb* will either be circles or arcs, which may be composed either of contiguous or discontinuous paths (as will be described later). Revisiting the example pedigree with the complex loop in Fig 3.4b, the two simple cycles within B could be drawn as two

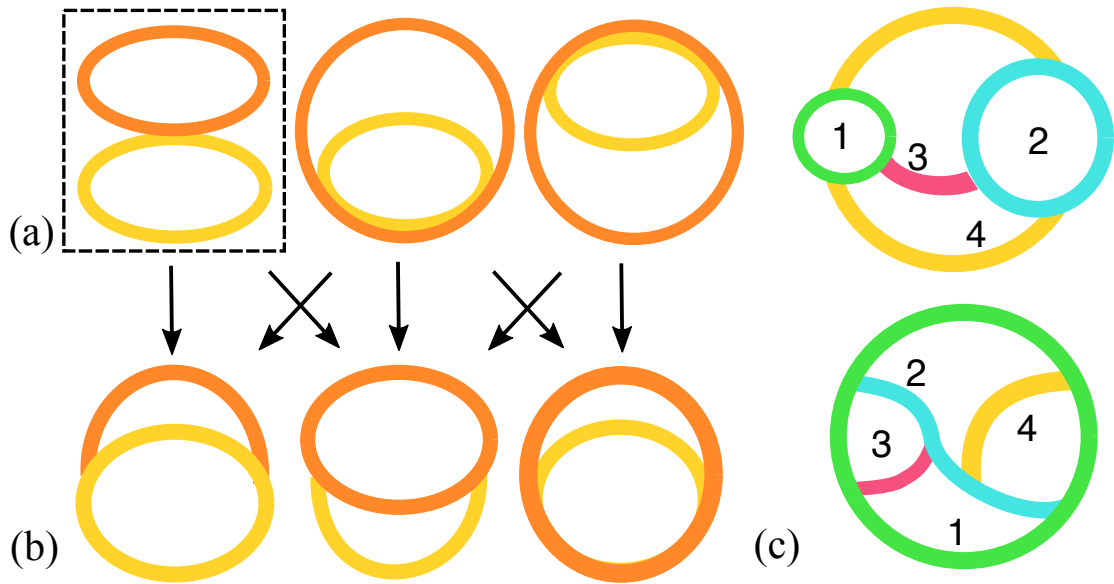


Figure 3.5: Circle and arc representation of loops in pedigrees. (a) Depiction of the simple cycles from the undirected cycle basis B in the pedigree of Figure 3.4b. The circle-and-arc diagram boxed by the dashed line corresponds to the cycles highlighted in Figure 3.4b, while the other two diagrams show the other two ways of describing the cycle basis, B . Note that the partial overlay of the two circles represents the shared path between those two cycles (in yellow and orange). (b) The simple cycles of the cycle basis shown in (a) can be expressed in three possible ways as an *rcb*. In each case, one of the *rcb* members is a simple cycle and the other is a non-overlapping path that includes part of the remaining simple cycle in B . (c) In a more complex case with nested loops, there exist many ways of decomposing the loops into an *rcb*, only two of which are depicted in the figure. In all cases, however, the number of members in an *rcb* is always the same and is equal to the number of simple cycles in the pedigree. In both diagrams, *rcb*-member 1 is colored green, member 2 is turquoise, member 3 is red, and member 4 is gold. (Note: the sizes of the circles or the lengths of the arcs may not directly reflect the relative scale between the actual pedigree loops.)

circles conjoined through a common path (Fig 3.5a). The cycle basis we defined in the pedigree of Fig 3.4b is one of the three possible forms for the undirected cycle basis. There are also multiple—three in this case—ways to partition these cycles into an *rcb* (Fig 3.5b). Yet in all three ways, one of the simple cycles retains its form as a circle in the *rcb* while the other is truncated to an arc, and both members of the *rcb* share two intersection nodes across all forms.

In less trivial cases (i.e., a pedigree with multiple complex, nested loops), such as that depicted in Fig 3.5c, the number of possible *rcbs* increases as the number of loops in the pedigree factor graph increases. However, regardless of which *rcb* representation is chosen, if a single variable node from each member of the *rcb* is selected as a loop breaker, the pedigree is rendered acyclic.

A common way to find simple cycles for the undirected cycle basis B of a graph is through the depth-first search (DFS) algorithm (Tarjan 1972). The same is true for a pedigree factor graph, which we will denote $\mathcal{P}(\mathcal{N}, \mathcal{E})$, where \mathcal{N} refers to the sets of vertices, or “nodes,” which include both the variable nodes and the factor nodes, and \mathcal{E} refers to the edges. DFS is a recursive process that travels between adjacent nodes with the goal of moving outwardly from the starting node as far as possible before backtracking. On a factor graph the algorithm begins with an arbitrary selected variable node v_i and proceeds in the next step to a neighboring factor node $f_k \in \eta(v_i)$. If that node has any neighbors other than v_i , the subsequent step proceeds to one of

those variable nodes, (i.e., $v_k \in \eta(f_k) \setminus v_i$), and so on. If there are no “new” nodes to visit from the current node, the algorithm backtracks to the previous node and proceeds from there to an unvisited node (if available). The algorithm also backtracks when the search path encounters a previously visited node, as will only occur within a cyclic graph.

By storing information about the nodes visited and edges traversed while exploring the graph using DFS, we can identify the cycles within a cycle basis B , and choose how to break those into segments so as to identify and store a reduced cycle basis for the pedigree factor graph. An explicit definition of this modified-DFS algorithm for identifying a reduced cycle basis appears in Algorithm 1 on page 116. See Table 3.2 for definitions of variables and notation in Algorithm 1. Before delving into the algorithm details, we recommend visiting Figure 3.6, which provides a step-by-step description of how the DFS algorithm would traverse the graph at the top part of Figure 3.5c.

Verbally, the algorithm works as follows: after the algorithm has reached the ℓ^{th} distinct node it has visited, whose identity is denoted by n_ℓ , it proposes searching forward to a new node, n_{next} . If n_{next} happens to be one of the previously visited nodes $n_i \in \{n_0, \dots, n_\ell\}$, a simple cycle consisting of nodes $\{n_i, \dots, n_\ell, n_i\}$ has been detected (lines 7–11 in Algorithm 1). Deriving the *rcb* member from this newly detected simple cycle is done by retaining the edges of the simple cycle that are not associated with any other previously identified *rcb* members (refer to lines 23–30 in Algorithm 1) and merging adjacent edges (i.e., any edges that are connected to a common node) into contiguous

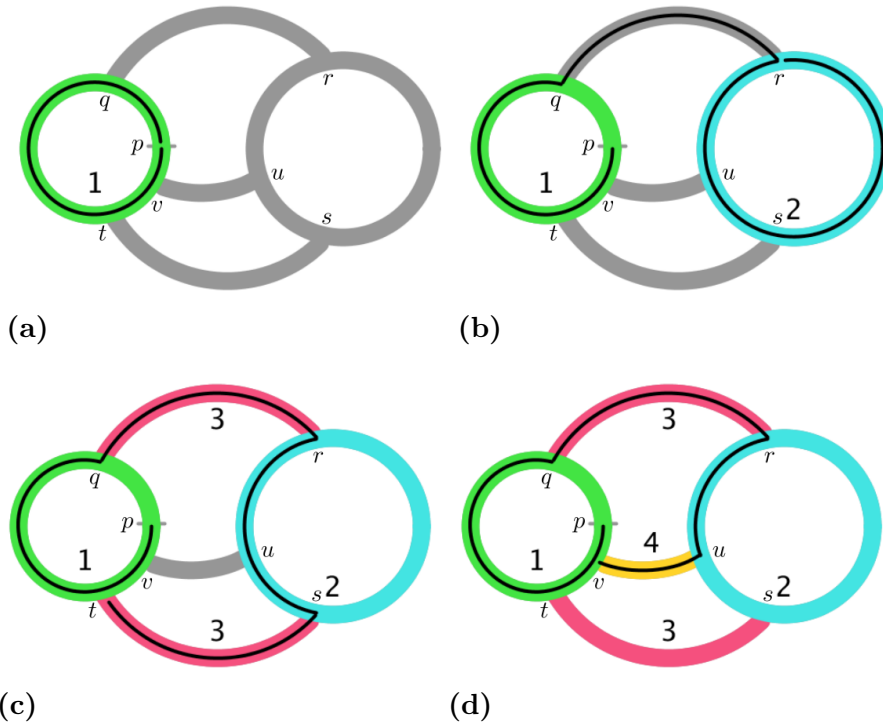


Figure 3.6: An instance of how an *rcb* is identified through the modified DFS algorithm. (a) The DFS algorithm starts at p on the right side of the green circle. It proceeds clockwise until it encounters its starting point, thus finding its first cycle and first *rcb* member (the green circle, 1). (b) After backtracking to q along circle 1, the DFS algorithm proceeds forward along the gray path, reaching r then proceeding anticlockwise around the circle, 2, eventually re-encountering the node at r , triggering the identification of the second *rcb* member, the turquoise circle, 2. (c) The algorithm backtracks to s and then proceeds forward along the branch to t , where it re-encounters a previously visited node. The third *rcb* member identified at this juncture is found by starting with the cycle visited forward from t (excluding backtracked portions), $t \rightarrow q \rightarrow r \rightarrow u \rightarrow s \rightarrow t$, and retaining only those portions that are not already included in an *rcb*; i.e., the green and turquoise portions are removed and the *rcb* member is given by the two red segments, 3. (d) The algorithm backtracks from t , through s to u , and then moves forward along the branch from u to v where it again encounters a previously visited node. The fourth *rcb* member here is the gold arc, 4, which includes the only edges along the active visitation cycle $v \rightarrow t \rightarrow q \rightarrow r \rightarrow u \rightarrow v$ that are not already included in *rcb* members. Finally, at this point, the algorithm backtracks from v along the thin black line all the way to p , finding no other unvisited paths, thus concluding the algorithm. Note that the *rcb* found here is only one of many instances. Different *rcbs* could be found by choosing different starting locations or different turning “directions” at branches.

Notations	Description
$N_\ell = [n_1, n_2, \dots, n_\ell]$	an ordered list of length l of the nodes that have been visited in DFS.
$E_m = [e_1, e_2, \dots, e_m]$	an ordered list of length m of edges that have been visited in DFS.
\mathcal{H}	a set of binary flags, one for each edge and node in the pedigree, each indicating whether the edge or node has been visited.
\mathcal{I}_e	a set of binary flags for each edge in pedigree indicating whether the edge is associated with an <i>rcb</i>

Table 3.2: Data structures used in detecting cycles and the *rcb* of a pedigree factor graph. These variables appear in the listing of Algorithm 1. N_ℓ and E_m can be seen in Figure 3.6, depicted as the thin, solid, black line.

paths, while non-adjacent edges lead to non-contiguous segments that are parts of the *rcb* member. We want to highlight that the terminal nodes of these contiguous paths may play an important role in reducing the total number of loop-breakers required to render a graph acyclic. We will discuss the utility of these nodes to the selection of “high-value” loop breakers in the next section.

3.3 Conditioning upon the Genotypes of Loop-Breakers to Render a Cyclic Pedigree Factor Graph Acyclic

The reduced cycle basis provides an extremely useful framework to determine a set of loop breakers to transform a cyclic pedigree into an acyclic one. Specifically, as long as a variable node from each *rcb* member is assigned as a loop breaker, the pedigree, in turn,

Algorithm 1: Identifying simple cycles and an *rcb* in pedigree factor graph.

input : $\mathcal{P}(\mathcal{N}, \mathcal{E})$ and starting variable node v_i
output: An undirected cycle basis B and a reduced cycle basis R

// Initialization:

- 1 set all edges and nodes marked as not visited in \mathcal{H} ;
- 2 set all edges marked as not associated with *rcb* in \mathcal{I}_e ;
- 3 $n_1 \leftarrow v_i, \ell \leftarrow 1, m \leftarrow 0$;
- 4 v_i is marked as visited in \mathcal{H} ;
- 5 $\text{DFS}(\mathcal{P}, N_\ell, E_m, \mathcal{H})$;
- 6 **Function** $\text{DFS}(\mathcal{P}, N_\ell, E_m, \mathcal{H})$:
 - // visiting each neighboring node n of n_ℓ via adjacent edge e*
 - for** each $\{n, e\} \in \eta(n_\ell)$ **do**
 - 7 **if** edge e is not marked as visited in \mathcal{H} **then**
 - 8 $\ell \leftarrow \ell + 1, m \leftarrow m + 1$;
 - 9 $n_\ell \leftarrow n, e_m \leftarrow e$;
 - 10 edge e is marked as visited ;
 - 11 **if** node n is marked as visited in \mathcal{H} **then** *// a loop is found*
 - 12 $\text{ReportLoops}(\mathcal{P}, N_\ell, E_m, \mathcal{I}_e)$;
 - 13 **else**
 - 14 node n is marked as visited in \mathcal{H} ;
 - 15 $\text{DFS}(\mathcal{P}, N_\ell, E_m, \mathcal{H})$;
 - 16 **end**
 - 17 $\ell \leftarrow \ell - 1, m \leftarrow m - 1$ *// backtracking*;
 - 18 **end**
 - 19 **end**
- 20 **Function** $\text{ReportLoops}(\mathcal{P}, N_\ell, E_m, \mathcal{I}_e)$:
 - 21 $N_r = \{\}, E_r = \{\}$; *// A place-holder for nodes and edges for $r \in R$*
 - 22 find index i where $n_i = n$, where $n_i \in N_\ell \setminus n_\ell$;
 - 23 **for** each $e \in \{e_i, \dots, e_m\}$ **do**
 - 24 **if** edge e is not marked as associated with any *rcb* in \mathcal{I}_e **then**
 - 25 let N be the pair of nodes associated with edge e ;
 - 26 $N_r \leftarrow N_r \cup N$;
 - 27 $E_r \leftarrow E_r \cup e$;
 - 28 edge e is marked as associated with an *rcb* in \mathcal{I}_e ;
 - 29 **end**
 - 30 **end**
 - 31 report $\{n_i, \dots, n_{\ell-1}\}, \{e_i, \dots, e_m\}$ as $b \in B$
 - 32 report N_r, E_r as $r \in R$

will be rendered loop-free. Some of these variable nodes may be shared across multiple *rcb* members, and when such a variable node is selected as a loop breaker, it counts as a loop breaker drawn from each of the *rcb* members in which it occurs. Consequently, choosing loop breakers that are shared between multiple *rcb* members reduces the total number of variable nodes that must be selected as loop breakers to render the pedigree acyclic. (Fig 3.7b and Fig 3.7c).

Choosing a set of vertices as loop breakers to make a graph loop-free is, in fact, a well-studied problem in graph theory that dates back to the 1960s with applications in combinatorial circuit design (Younger 1963; Lempel and Cederbaum 1966). This proven NP-complete problem (Yannakakis 1978) is known as the *Feedback Set Problem* (FSP) (Festa et al. 1999) or the *Loop Cutset Problem* (Becker et al. 2000) depending on the discipline and the type of graph being studied. Recently, the FSP has gained attention for its appearance in constraint satisfaction and Bayesian inference problems. Notably, novel recursive algorithms, which are variants of the Davis-Putnam-style exponential-time backtracking algorithm, have been used to reduce the search time of the FSP in undirected graphs of n vertices from 2^n to t^n , where $1.86 < t < 2$ (Festa et al. 1999; Fomin et al. 2008).

Although our interest in identifying loop-breakers in a cyclic pedigree factor graph is aligned with the objective of finding a minimal feedback vertex set, our goal is not necessarily to find the absolutely minimal number of nodes to be chosen as loop-breakers.

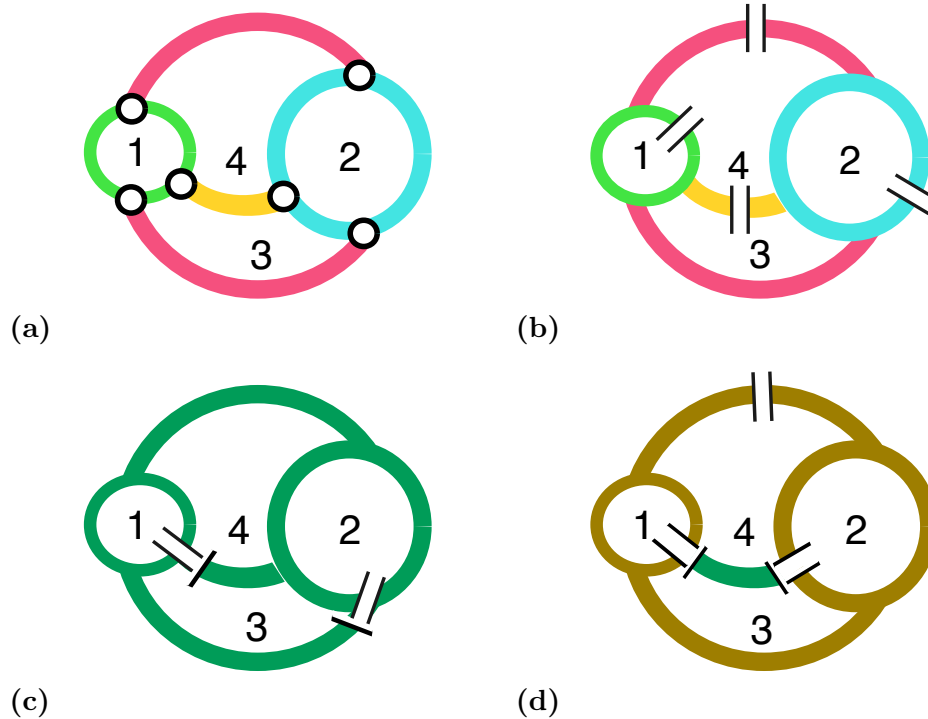


Figure 3.7: Consequences of the choice of loop-breakers. In (a), segments of the four different colors correspond to the four *rcb* members, while the outlined black circles represent nodes shared between pairs of those *rcb* members. Selecting any node as a loop breaker from every colored segment is guaranteed to disrupt all cycles (see hash marks in b, c, and d indicating such nodes). However, selecting the highlighted nodes (outlined black circles) as loop breakers reduces the total number of loop-breakers needed to render the graph acyclic. Panel (c) illustrates one such solution that achieves the smallest possible number of loop-breakers. Depending on which nodes are chosen and conditioned upon as loop-breakers, the resulting loop-broken pedigree factor graph may consist of a variable number of *conditionally separated components (cscs)* e.g., the loop breakers chosen for (c) lead to different *cscs*, than the choice of loop breakers in (d); different *cscs* shown in different colors. Note that some nodes can be associated with multiple *cscs*, and some *cscs* do not appear in the circle-and-arc diagrams that, by design, emphasize the internal elements of the pedigree factor graph.

Rather our goals are focused on proposing changes that reassign the current loop-breaking node(s) in a pedigree factor graph to other nodes, in a manner that allows the overall number of loop-breakers to remain small. Doing so is achieved in the context of the Metropolis-Hastings algorithm using the *rcb* to guide the proposals (see Section 3.3.3). In the process, the number of loop breakers can be reduced whenever loop breakers are chosen from the variable nodes found at the intersections between members of the *rcb* of the pedigree factor graph.

3.3.1 Likelihood Calculation Conditioning on Loop Breakers

Given a set of loop breakers that is adequate to break all the cycles in the graph, we can easily obtain the joint pedigree likelihood, *conditional on* a choice of genotype values for the loop breakers, by using the Sum-Product algorithm. By summing these joint conditional likelihoods over all possible choices of the genotypes for the loop breakers we can compute the marginal joint pedigree likelihood. As detailed below, these quantities figure into the calculations for accepting or rejecting proposed changes to the set of loop breakers and their genotypes that we shall condition upon (or stop conditioning upon) via the Metropolis-Hastings algorithm.

The act of fixing the genotype of the loop-breakers (i.e., conditioning upon those values of the genotypes) disrupts the cyclic paths in the pedigree, thereby avoiding the problem of “message gridlock” during message propagation (Fig 3.7). In the context

of a factor graph, when we condition upon the genotype of a variable node, we, in essence, create multiple *clones* of that variable node—one clone for each factor node the individual is adjacent to. The outgoing messages from each of these clones are solely determined by the fixed genotype value of the loop breaker. For example, if the loop-breaker’s genotype is fixed at a value of 0 the outgoing message is $(1, 0, 0)$, signifying a probability of 1 that its genotype is 0 and a probability of 0 that the genotype is either 1, or 2. Furthermore, each clone of the conditioned node is connected to only one of the factor nodes adjacent to the loop breaker in the original pedigree (Figure 3.8).

We should note that when we condition upon any variable node that is connected to other pedigree members, we will be breaking up the node’s connected component into multiple conditionally separated components, or *cscs* (Fig 3.7c and Fig 3.7d). All of these *cscs* must be accounted for, as the overall joint pedigree likelihood conditional on the loop breakers involves a product over these *cscs*, and, further, when we stop conditioning upon a loop breaker, all the clones must be “consolidated” back into their original variable node.

3.3.2 Pedigree Reconfiguration Samplers that Introduce Loops into Pedigrees

In the previous two sections we have described generally how to identify potential loop breakers and condition upon them; however in the course of sampling over pedigree

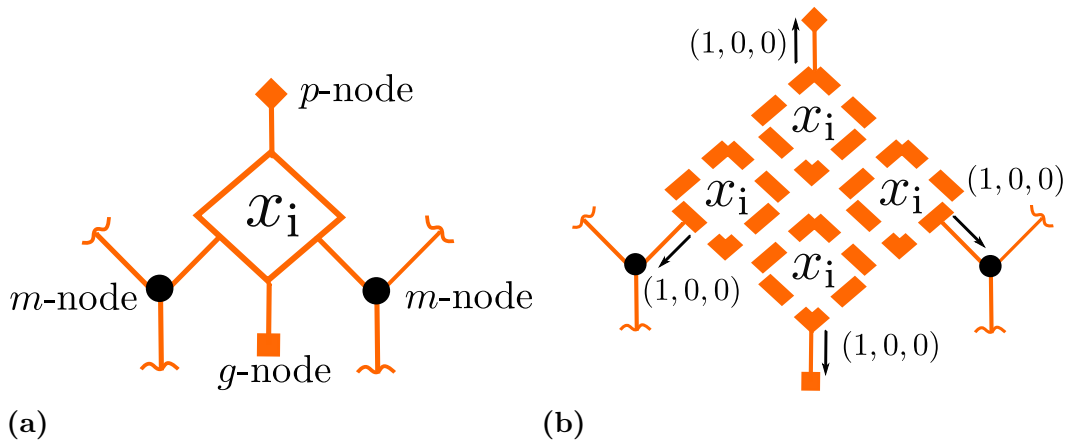








Figure 3.8: Conditioning upon a fixed genotype value of a variable node: the creation of *clones*. (a) Initial setup: for this example, assume an acyclic pedigree containing an individual x_i involved in two mating events. The individual variable node x_i , here, is attached to four factor nodes: a p -node, a g -node and two m -nodes. (b) When we set the value of the genotype of x_i —in this case to the value of 0—and condition upon that fixed value, the result can be interpreted as a factor graph in which x_i has been expanded into four clonal copies, each one connected only to a single one of the four factor nodes originally adjacent to x_i in (a). The expansion into clones, here, creates four conditionally separated components (*cscs*). Outgoing messages from the clones of x_i are determined entirely by the fixed value of the genotype. Since the latent genotype is assumed to have the value of 0 with full certainty, the messages reflect that: $(1, 0, 0)$. Note that the joint likelihood of the pedigree conditional on individual node $x_i = 0$ is the product over clones of the dot product of the incoming message to the clone (not shown) with the outgoing message from the clone. The incoming messages to each clone are computed in a straightforward manner using the Sum-Product algorithm applied to each clone’s *csc*.

structures, loops will occur as a consequence of *proposed changes* in which it is proposed that an individual becomes attached to a node that is already part of its connected component via an existing path. Thus, the identification and treatment of loop breakers must be an integral part of these proposed configurations. In the following section, we will describe a variety of proposal types that allow loops to be formed and then simultaneously broken by conditioning upon a loop breaker (or a set of loop breakers) that is itself selected as part of the proposal. These samplers, taken together and in conjunction with proposals to rearrange loop breakers and their fixed genotype values, allow for MCMC sampling over the space of all possible pedigrees—even those with loops. From this point forward, the illustrations depicting the different types of samplers follow the shorthand notations established in Figure 3.9.

Similar to the acyclic pedigree sampler, the new configurations are designed around proposing new parents, x'_{pa} , and x'_{ma} of a focal individual that we label x_{focal} , with current parents x_{pa} , and x_{ma} . In this section we treat such proposals that may introduce a new loop into the pedigree. To estimate the pedigree likelihood under such loop-forming proposals, we apply the previously mentioned conditioning method on a set of one or more candidate loop-breakers. The choice of which individuals should be chosen as candidate loop breakers varies widely depending on the conditioning status and the *csc* membership of x_{focal} , and of its proposed family members (e.g., parents and/or full-sibling(s) that x_{focal} is proposed to be associated with). In this section, we break down

variable node that is...	message value that...
 not conditioned upon	 requires update
 conditioned upon	 does not require update
 either conditioned or not conditioned upon	
 proposed to be conditioned upon	

shorthand graphical notations (in orange):

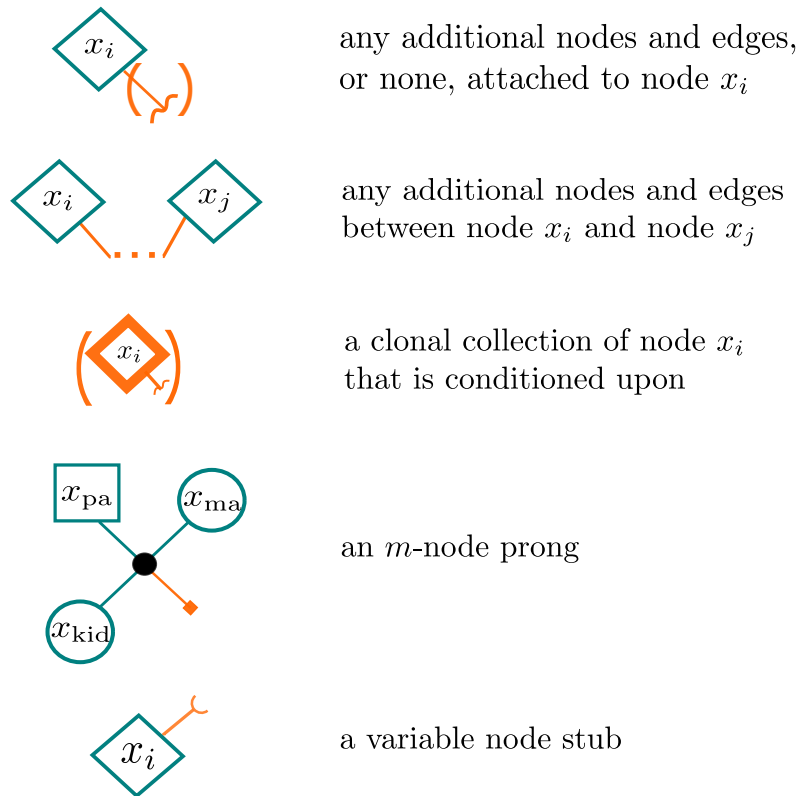


Figure 3.9: Legend for the graphical shorthands used in describing the joint likelihood calculation scheme for pedigree proposals that create loops, and thus require conditioning upon variable nodes. In the figures that follow, the parentheses denoting replication of the inner contents will be colored black, and other elements (such as nodes, edges, stubs, and prongs) will be colored according to the conditionally separated components (*cscs*) to which they belong, with black denoting that the element could belong to any *csc* (i.e, the likelihood calculation is invariant to the *csc* affiliation of such, black elements).

different categories of loop-creating proposals, and identify the individuals we choose as loop breakers in the myriad possible cases.

3.3.2.1 The Initial “Pruning” Stage

Every proposal, regardless of which category it belongs to, begins with an initial step of pruning x_{focal} from its current parents. The specific task of pruning x_{focal} from its original parents x_{pa} and x_{ma} is straightforward in an acyclic pedigree, but within a loopy pedigree the process can be more complicated as we may wish to consider reverting the conditioning status of one or more loop-breakers (Fig 3.10) found upon cyclic paths affected by the pruning. Pruning x_{focal} from its parents is equivalent to removing an edge e_{focal} between the variable node x_{focal} and the factor node, $f_i^{(m)}$, between x_{focal} and its parents. Removal of e_{focal} will break a cycle that runs through it. Therefore, any loop breakers that are currently conditioned upon, in order to break that specific cycle, may no longer be required for breaking the loop after pruning. In such cases, better mixing of our sampler should be achieved by reverting any such loop-breakers from their conditioned state to an unconditioned state after pruning, but before proposing reconfigurations to the pedigree.

If a node x_a serves as a loop breaker, is part of the reduced cycle basis member that includes the edge e_{focal} , and is not included in any other *rcb* members, then x_a is clearly no longer required to maintain a role as a loop-breaker after removing the edge e_{focal} (Figs 3.10a and 3.10b). If, on the other hand, x_a is included in the *rcb* member

that includes e_{focal} *and also* belongs to another *rcb* member, then x_a will likely still be required as a loop breaker, even after removing e_{focal} . The only exception to this occurs when only a single offspring—in this case x_{focal} —descends from marriage node $f_i^{(m)}$, and x_a , in addition to being included in the *rcb* member that includes e_{focal} , belongs only to *rcb* members that also include $f_i^{(m)}$ (Fig 3.10c). In such a case, removing e_{focal} from $f_i^{(m)}$ leaves $f_i^{(m)}$ without any remaining children, demanding that $f_i^{(m)}$, itself, gets removed along with e_{focal} . When this occurs, any cycles that run through the edges leading through $f_i^{(m)}$ will be broken by its removal.

Here, we describe the conditions in the previous paragraph more formally. Let $M_{\text{rcb}}(n)$ denote the set of *rcb* members that include node n (n can be either a variable node or a factor node). We denote by $S(x_a, f_i^{(m)})$ the set of all *rcb* members that share *both* variable node x_a and factor node $f_i^{(m)}$. That is, $S(x_a, f_i^{(m)}) = M_{\text{rcb}}(x_a) \cap M_{\text{rcb}}(f_i^{(m)})$. As in Algorithm 1 we use $r \in R$ to denote an *rcb* member in an *rcb* R , while E_r and V_r denote the edges and variable nodes included in that *rcb* member. Assume for simplicity of discussion that each *rcb* member has, within it, only a single loop breaker currently conditioned upon (the extension to extra, currently “unrequired” loop breakers is trivial). For any *rcb* member $r \in S(x_{\text{focal}}, f_i^{(m)})$, either 0 or 1 current loop breakers $x_a \in V_r$ will be unconditioned as a result of pruning, and across *all* $r \in S(x_{\text{focal}}, f_i^{(m)})$ and $x_a \in V_r$, the maximum total number of variable nodes serving as loop breakers along an $r \in S(x_{\text{focal}}, f_i^{(m)})$ that will be unconditioned upon is $T(f_i^{(m)})$,

if and only if:

1. $M_{rcb}(x_a) \subseteq M_{rcb}(f_i^{(m)})$, and
2. $0 < \|S(x_a, f_i^{(m)})\| \leq T(f_i^{(m)})$.

$\|S(x_a, f_i^{(m)})\|$ denotes the cardinality of $S(x_a, f_i^{(m)})$, and $T(f_i^{(m)})$ is a function that depends on the number of children immediately descending from the marriage node $f_i^{(m)}$ and is defined as follows:

$$T(f_i^{(m)}) = \begin{cases} 1 & \text{if } f_i^{(m)} \text{ has } > 1 \text{ offspring} \\ 2 & \text{otherwise.} \end{cases}$$

If x_{focal} is the only offspring of $f_i^{(m)}$, we also perform the same unconditioning on any loop breaker that shares the same *rcb* membership as any edges adjacent to $f_i^{(m)}$, because, pruning in this case, also removes $f_i^{(m)}$ and all the edges adjacent to it. At this point, we have described unconditioning loop breakers that share the same *rcb* membership of $f_i^{(m)}$ or x_{focal} . In some cases, it might also be possible to uncondition a loop breaker that is not found in the same *rcb* member as $f_i^{(m)}$ or x_{focal} but is found in the same cycle basis member of $f_i^{(m)}$ or x_{focal} ; however we do not pursue such an aggressive unconditioning approach here, choosing instead to focus only on those loop breakers sharing *rcb* membership with $f_i^{(m)}$ or x_{focal} .

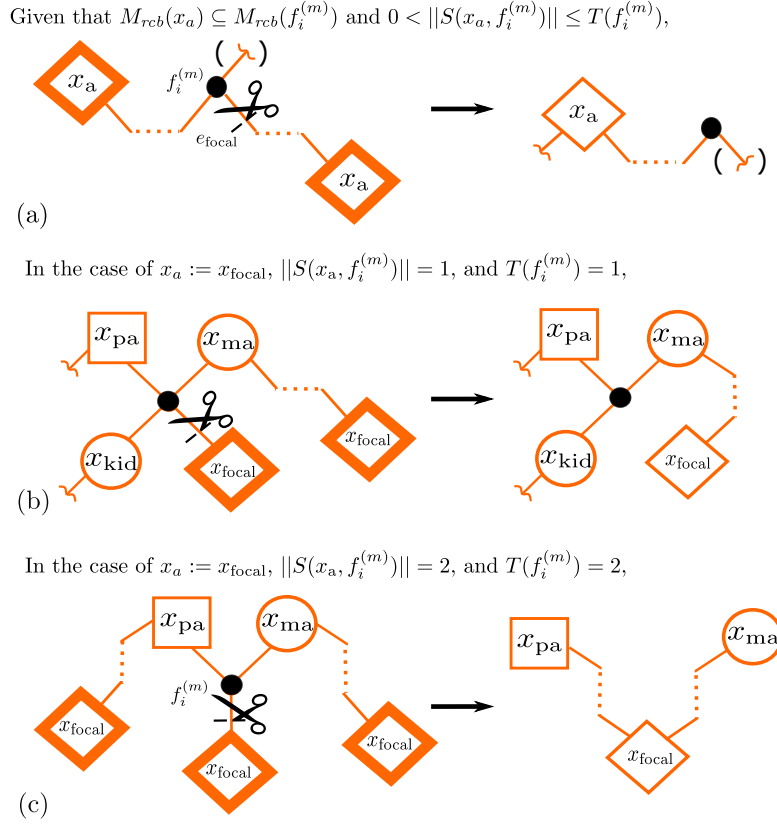


Figure 3.10: Removing the conditioning status of variable nodes under edge removal occurring in the initial “pruning” stage to propose a new configuration. Prior to assigning x_{focal} to proposed parent pairs, x_{focal} is required to be pruned from its parents through removal of edge e_{focal} . When edge e_{focal} is removed, a loop breaker which is included upon the same rcb member as e_{focal} may no longer be required to break a loop and therefore may no longer need to be conditioned upon. Panel (a) presents a formal description of the requirements needed to “uncondition” a loop breaker x_a (see text for description and definition of notation). The remaining panels highlight two examples of the principles in (a). (b) x_{focal} has at least one full sibling and is a loop breaker upon the cyclic path running through x_{focal} , $f_i^{(m)}$, x_{ma} , and eventually back to x_{focal} . Since x_{focal} is not included in any rcb members other than the one involving $f_i^{(m)}$ and x_{ma} , the removal of e_{focal} leads to x_{focal} being no longer needed as a loop breaker. (c) x_{focal} is the sole child in a family that is included in two rcb members (one through x_{ma} and the other through x_{pa}). Here, x_{focal} is in more than one rcb member but can still be unconditioned upon because $f_i^{(m)}$ is included in both of the rcb members x_{focal} is included in, and the removal of edge e_{focal} in a family with only a single offspring also leads to the removal of $f_i^{(m)}$.

3.3.2.2 The Pedigree Reconfiguration Step

Once the focal individual is peeled off from its initial parent, we propose a number of choices in which the focal individual is either attached to a new set of parents or is reattached to the parents from which it was recently pruned off. The choices we propose could conceivably include an exhaustive combination of all possible parent pairs; however, the computational overhead in doing so would be prohibitive, because the loop breakers required for different *pairs* of proposed parents might be different, such that multiple runs of the Sum-Product algorithm would be required to compute the Hastings ratio for many different proposed pairs of new parents. Instead, we simplify the proposed moves by proposing changes only to a single parent at a time while keeping the assignment of the other parent fixed in place. Limiting the set of proposed choices in this manner eases the computational workload involved in calculating the joint pedigree likelihood, especially when a new loop is created by any of the proposed moves.

In the following, we present three classes of parentage samplers: the “fresh-start” parentage sampler (Fig 3.11), the “spouse-shuffler” parentage samplers (Figs 3.12, 3.13, and 3.14), and the “family-expansion pack” sampler (Fig 3.15). Each of the samplers is described in detail in the figure captions. The “fresh-start” sampler and the “spouse-shuffler” samplers propose attaching the focal individual, x_{focal} , to a parent pair consisting of x_{fixed} —a parent whose identity is fixed across all proposed changes—and x_i , a candidate parent whose identity varies over a set of candidates. The parent, x_{fixed} ,

whose identity is fixed in place is often, but not always, one member of the parent pair that x_{focal} was recently pruned from. In the case of the “fresh-start” sampler, x_{fixed} is taken to be an individual, x_{unobs} , that is not in the genetic sample and is not currently attached to any other individuals. The “spouse-shuffler” type of parentage samplers are limited to the cases when x_{fixed} and x_i do not share any children (apart from x_{focal} , before pruning). By contrast, in the “family-expansion pack” sampler, it is proposed that x_{focal} be added to one of a set of candidate marriage nodes that already occur in the pedigree.

Although we present each sampler as if it is a distinct proposal type, there are cases wherein the same rearrangement may occur from different samplers. In other words, these classes do not lead to mutually exclusive sets of parental rearrangements. For example, if x_{focal} does currently have any assigned parents, there is no distinction between applying the “fresh-start” sampler and “spouse-shuffler” sampler-I on x_{focal} . Nonetheless, we find that categorizing the samplers in the manner that we have is helpful for understanding their operation.

If a proposed configuration creates a new cycle in the proposed pedigree factor graph, it is necessary to designate a loop-breaker (or two) to break that newly formed cycle. Instead of randomly choosing any of the members along the newly formed loop as loop-breakers, we selectively choose either x_{focal} , or x_{fixed} , or both, to condition upon as loop breakers to streamline the likelihood calculation for all possible x_i . For every

proposed individual x_i (and for every proposed marriage node in the family-expansion pack sampler) there are multiple possible proposed values of the new loop breakers' conditioned-upon genotypes that are additionally proposed. That is, for every proposed reconfiguration which forms a new loop, there are also either three proposed genotype values (or nine, if *both* x_{focal} and x_{fixed} are to serve as the new, additional loop breakers) for the new loop breaking genotype(s). Thus, when a new one of the proposed loop-forming reconfigurations is accepted, not only does it involve accepting the proposed topology of the pedigree, but also the conditioned-upon genotype values of the new loop breakers (either x_{focal} , x_{fixed} , or both).

3.3.3 Relocating Loop-breakers and Updating Their Fixed Genotype Values

When we accept a change in pedigree configuration that adds more new loops to the pedigree, we will have conditioned upon the genotype value of one or more new loop breakers in the process. We don't actually know the true, latent genotypes of these loop breakers, and if we were to leave them in their conditioned-upon state and proceed with MCMC, the sample of pedigree configurations obtained would not be from the posterior distribution, but rather from the posterior distribution conditional on the simulated value of the loop breakers' latent genotypes. Accordingly, the conditioned-upon genotype values of the loop breakers must also be sampled, in a fashion that

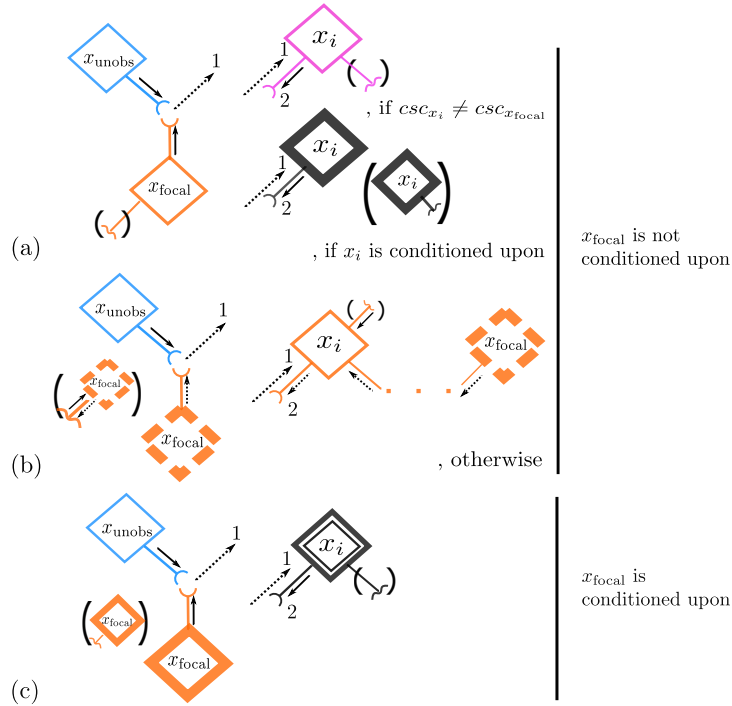


Figure 3.11: The “fresh-start” parentage sampler. Assume x_{focal} is detached from any parents. Parent pairs to which x_{focal} may be proposed to be attached will always include at least one unobserved member, x_{unobs} , that is unconnected to any others and has no observed genotype. The second member, x_i , of the pair may be observed and comes from an appropriate parental candidate group for x_{focal} . Since all proposed configurations will include individuals x_{unobs} and x_{focal} , we first calculate the outgoing messages along the stubs of x_{unobs} and x_{focal} and combine them as in (Eqn 2.4) to send a message toward x_i (arrow 1). This message is computed only once, as it is the same for all x_i . The joint pedigree likelihood is the dot product of the message of arrow 1 with the outgoing message along the stub from x_i (arrow 2). Normally when x_i and x_{focal} are from different *cscs*, or x_i or x_{focal} are already conditioned upon (i.e., are already assigned as loop-breakers), as shown in cases (a) and (c), then the only message that requires updating is that of arrow 1. However, if x_i and x_{focal} are not conditioned upon, and they belong to the same *csc*, as shown in case (b), the proposed connection between x_i and x_{focal} will add a new loop to the pedigree thus requiring that a new loop-breaker be designated. In such cases, we make the simple choice of designating x_{focal} as the loop-breaker and then, for each of the three possible genotype values of x_{focal} that we might condition upon, we compute updated messages along arrow 1 and along the outgoing prong of all x_i s in the same *csc* as x_{focal} . These messages are used in the calculation to jointly accept or reject the proposed pedigree configuration *and* the genotype value of the proposed loop breaker.

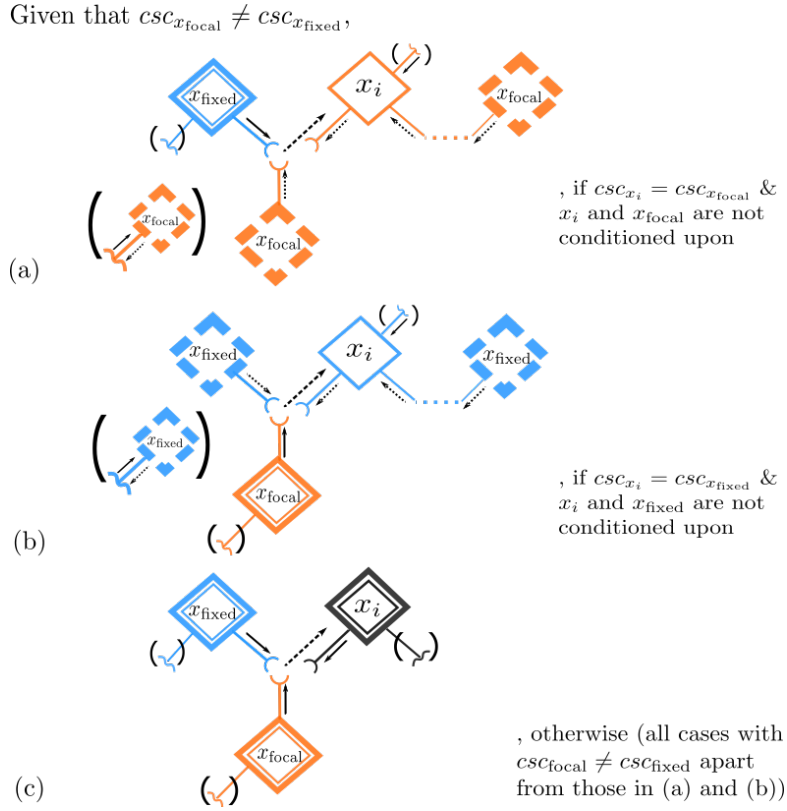


Figure 3.12: The “spouse-shuffler” parentage sampler-I: the three proposed trio members belong to two or more distinct csc s, with $csc_{x_{focal}} \neq csc_{x_{fixed}}$. Assume x_{focal} , has been temporarily pruned from its two parents in the current pedigree, one of whom is x_{fixed} . This “spouse-shuffler” move proposes x_i (from an appropriate parental candidate group) as a new mating partner (which may be the same as in the previous iteration) for x_{fixed} . In this sampler, x_i and x_{fixed} must not currently have any other children, otherwise the possible updates are covered by the “family-expansion pack” sampler of Figure 3.15. Similar to the “fresh-start” sampler, we first compute and combine the outgoing messages along the stubs of x_{fixed} and x_{focal} to send a message toward x_i . The dot product of this incoming message and the outgoing message along the stub of x_i gives the joint pedigree likelihood of each proposed configuration. When no loops are created by the formation of the proposed family, as in case (c), the amount of work in message updates is minimal. However, if two of the trio members share the same csc and are not yet conditioned upon, we must condition upon the genotype of x_{focal} in (a) or x_{fixed} in (b) in order to compute the joint pedigree likelihood.

Given that $csc_{x_{\text{focal}}} = csc_{x_{\text{fixed}}} \neq csc_{x_i}$,

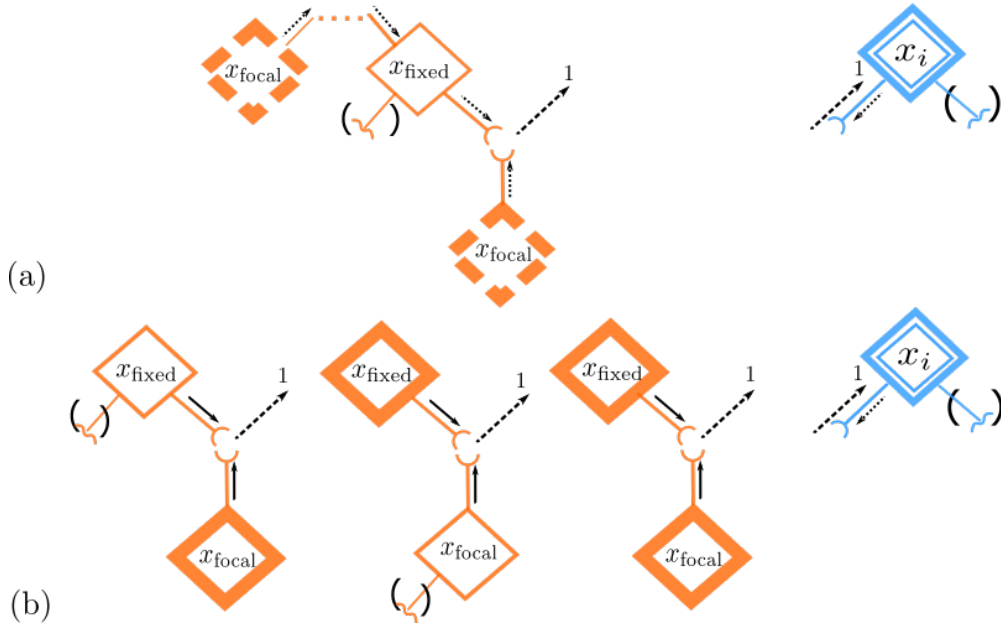


Figure 3.13: The “spouse-shuffler” parentage sampler-II: the three proposed trio members belong to two distinct *cscs*, with $csc_{x_{\text{focal}}} = csc_{x_{\text{fixed}}} \neq csc_{x_i}$. The proposal in this sampler is initiated in the same fashion as that in Fig. 3.12; however, here, x_{fixed} and x_{focal} share the same *csc* while x_i is on a different *csc*. When x_{fixed} and x_{focal} are not yet conditioned upon and share the same *csc*, as shown in (a), we must choose one of the two individuals to be a loop-breaker to prevent the formation of a loop. In this case, we designate x_{focal} as the loop-breaker. As a result, three versions of the outgoing message along arrow 1 must be computed at each locus—one for each of the three possible genotype values at the locus. The genotype value of x_{focal} to condition upon also constitutes part of the proposal which is accepted or rejected according to the Hastings Ratio. (b) When at least one of the variables nodes, x_{focal} and/or x_{fixed} , is already conditioned upon, the computation for arrow 1 is straightforward as the existing conditioning upon those nodes prevents the formation of any new loops that must be broken.

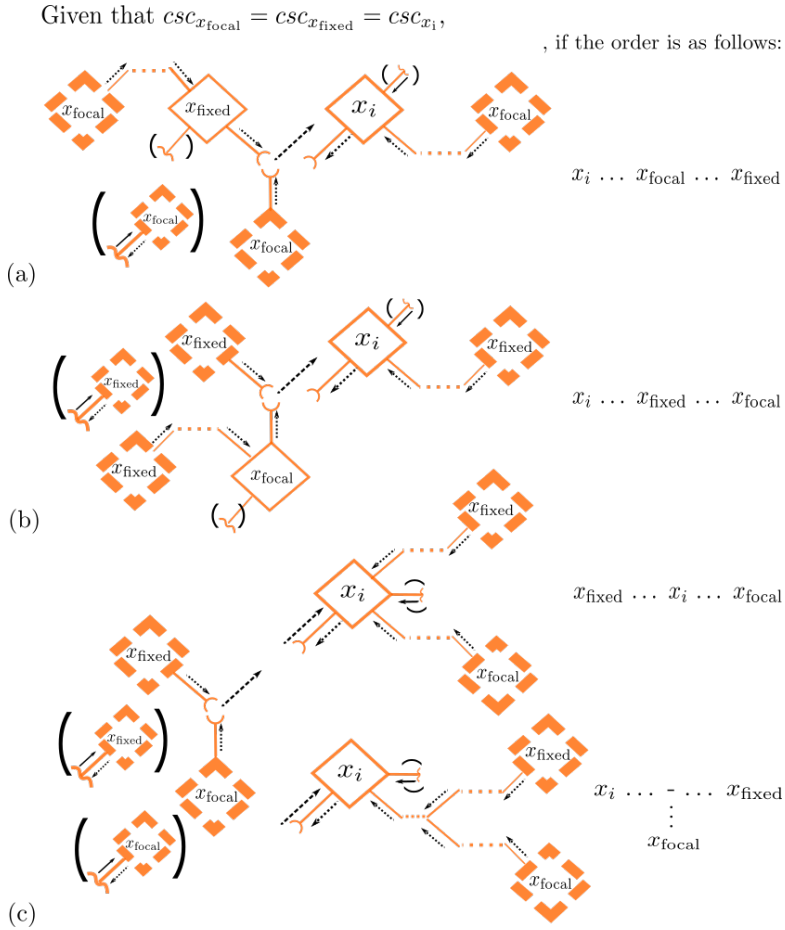


Figure 3.14: The “spouse-shuffler” parentage sampler-III: the three proposed trio members belong to the same csc . The proposal in this sampler is initiated in the same fashion as that in Fig. 3.12; however, in this case, all three of x_{focal} , x_{fixed} , and x_i belong to the same csc . The choice of which node or nodes to be chosen as loop-breakers is sensitive to the relative arrangement of x_i , x_{focal} , and x_{fixed} within the current csc to which they all belong as shown in (a) through (c). We are only illustrating the case when all three nodes are not yet conditioned upon. In the cases where one or more individuals have already been conditioned upon, the message update is still similar to that shown, albeit without messages passing through the additional conditioned-upon individuals.

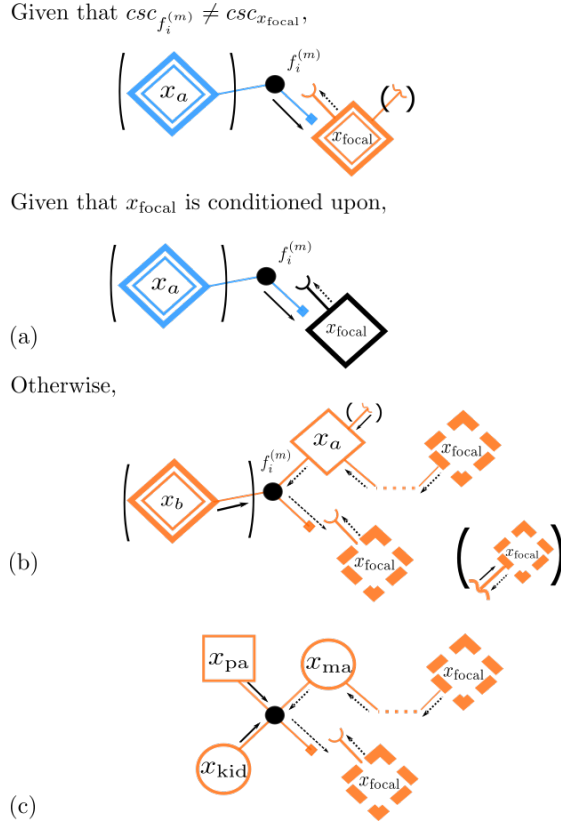


Figure 3.15: The “family-expansion pack” sampler: addition of x_{focal} to an existing m -node. Assume x_{focal} is detached from any parents. This sampler proposes x_{focal} as a full-sibling to any offspring attached to an existing m -node $f_i^{(m)}$. Thus, under this “family-expansion pack” sampler, it is possible for x_{focal} to be reattached to its previously assigned parent pair when that pair has other offspring. The csc affiliations of $f_i^{(m)}$ and x_{focal} determine several different cases. (a) When the factor node $f_i^{(m)}$ and variable node x_{focal} are from different csc s, or if x_{focal} is conditioned upon, the joint pedigree likelihood is simply the dot product of the outgoing messages along the prong from $f_i^{(m)}$ and the stub from x_{focal} . (b) When $f_i^{(m)}$ and x_{focal} are from the same csc and x_{focal} is not conditioned upon, attaching x_{focal} to $f_i^{(m)}$ will form a cyclic path that runs from x_a , a variable node adjacent to $f_i^{(m)}$, to x_{focal} , and back to x_a , necessitating that x_{focal} be conditioned upon to break the loop. x_a could be a parent or an offspring attached to $f_i^{(m)}$. An example where x_a is a parent x_{ma} appears in (c).

preserves detailed balance, in the course of the MCMC. Furthermore, the loop-forming pedigree-configuration proposals detailed above may introduce more loop breakers than necessary, because the choice of which variable node in a loop is assigned as a loop breaker is made purely for efficiency and convenience in calculating the joint likelihood of each proposal. As a consequence, we develop several MCMC move types that allow for the conditioning status of individual variable nodes to be changed. Such moves, taken together, permit the positions of, the number of, and the genotype values of loop breakers to be updated over the course of the MCMC, independently of the pedigree re-configuration steps, ultimately allowing a sample of loopy pedigrees from their posterior distribution.

The MCMC move type for updating the status of loop breakers and their genotypes is a simple, single-site sampler that proposes a change to only a single variable node at a time. It is a simple Gibbs sampler that samples the state for a single variable node from the full conditional distribution of its possible values. There are only four such values: either the variable node is not conditioned upon (if permissible, see below), or it is conditioned upon with one of three possible genotype values. Specifically, the single-site Gibbs sampler for loop-breaker status proceeds as follows, for any variable node v (which may currently be either conditioned upon or not) in the cycle basis:

1. Compute the joint pedigree likelihood of the four following cases:

- v is conditioned upon with a genotype value of 0

- v is conditioned upon with a genotype value of 1
- v is conditioned upon with a genotype value of 2
- v is not conditioned upon—a case that is allowable if and only if there is already at least one other loop breaker currently conditioned upon in every *rcb* member in which v is included.

2. Normalize those probabilities and sample one of the outcomes according to those normalized probabilities.

The joint pedigree likelihood for each of the cases can be easily computed through the use of the pre-computed messages along the edges of factor graph. If the variable node v is initially not conditioned upon, the joint pedigree likelihood with v being conditioned upon with a genotype value of g is proportional to the product over the dot products of incoming messages to variable node v with the outgoing messages from v , specified in accordance with fixed genotype g (i.e., if $g = 1$ the outgoing messages are all $(0, 1, 0)$). If, on the other hand, the variable node v is already conditioned upon as a loop breaker, the joint pedigree likelihood for different values of g is found as the product over clones of v of the dot products of incoming messages and outgoing messages given g . Fig. 3.8 provides a good starting point to visualize the latter calculation. Finally, if v is decommissioned from its status as loop breaker, the joint pedigree likelihood is proportional to a simple sum. Let $L(\mathcal{P}|v = g)$ be the joint pedigree likelihood conditioned on the genotype of v being g . If v is decommissioned as a loop breaker the

joint pedigree likelihood in that case is $\sum_{g \in \{0,1,2\}} L(\mathcal{P} | v = g)$.

3.4 Results

3.4.1 Grandparentage Inference of cyclic Pedigrees

To evaluate the ability of pedFac’s cyclic sampler to infer cyclic, multigenerational pedigrees, we simulate a three generation pedigree with the pedigree simulator described in Section 2.2.9.2. Setting the number of founders, N_f , to 200 and using all other default parameters, the simulator returns a three generation pedigree composed of 335 individuals, approximately one third of which lie along a cyclic path (Fig 3.16). This pedigree has 33 simple cycles, all of them being of the complex loop type.

To create a series of grandparentage inference problems, we create 5 different scenarios each with a different sampling fraction, $f \in \{1, 0.75, 0.5, 0.25, 0\}$, for the second/middle generation. This partially sampled (or completely missing) intermediate generation scenario is inspired by a situation commonly encountered in the assessment of salmon hatchery supplementation programmes. In this situation, offspring produced at the hatchery are released, but do not return back to the hatchery to spawn (Christie et al. 2011). Rather, they spawn in the wild (where they are neither sampled nor genotyped) and produce offspring who themselves return back to the hatchery to be sampled and genotyped. Through grandparentage analysis, it is possible to confirm whether any fish

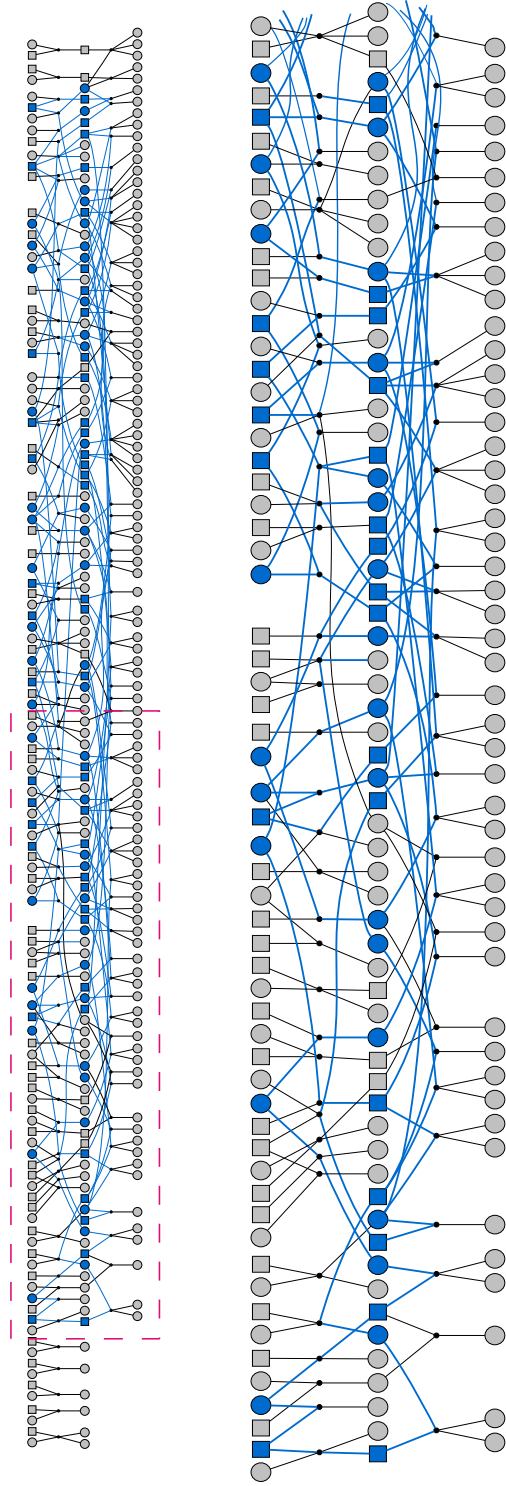


Figure 3.16: Three-generation cyclic pedigree of 335 individuals with various mating patterns in full view (top) and selected closeup view (bottom). Approximately one third of the individuals (31 & 64 in the 1st and 2nd generation) are involved in loops. Individual nodes and edges highlighted in blue are part of the 33 simple cycles (all complex-type pedigree loops) in this pedigree.

fraction	fraction of correctly assigned grandparents					
	pedFac (cyclic sampler)		pedFac (acyclic sampler)		sequoia	
	mean \pm sd	range	mean \pm sd	range	mean \pm sd	range
1	1.00 \pm 0.00	(1.00,1.00)	0.72 \pm 0.00	(0.71,0.72)	0.99 \pm 0.01	(0.97,1.00)
0.75	1.00 \pm 0.00	(1.00,1.00)	0.69 \pm 0.03	(0.65,0.72)	0.94 \pm 0.03	(0.89,0.97)
0.5	0.98 \pm 0.01	(0.98,1.00)	0.71 \pm 0.05	(0.66,0.76)	0.73 \pm 0.08	(0.61,0.84)
0.25	0.97 \pm 0.00	(0.97,0.98)	0.72 \pm 0.03	(0.68,0.74)	0.54 \pm 0.06	(0.47,0.60)
0	0.96 \pm 0.01	(0.96,0.98)	0.77 \pm 0.01	(0.76,0.78)	0.03 \pm 0.04	(0.00,0.10)

Table 3.3: Comparison of the fraction of correctly assigned grandparents for 101 individuals in five sample-fraction scenarios of a three-generation cyclic pedigree. The mean and standard deviation is calculated from five replicates.

returning to the hatchery are descendants of the original release cohort, thereby making it possible to evaluate the long term impact of hatcheries on the resident, wild-spawning population.

We used the program MENDEL to generate 200 SNPs from 34 Chinook Salmon autosomes and we simulated the same genotyping error rate and the rate of missingness as described in the previous chapter. pedFac’s cyclic and acyclic samplers were run for 25 sweeps with no burn-in, allowing, at most, 10 candidates to be chosen as x_{fixed} or as any pairwise parental candidates. For sequoia, we selected the default, “full” mating system to match the structure of the true pedigree and no ‘maxSibIter’ specification was applied.

Across all five middle-generation sampling fractions, pedFac’s cyclic sampler was able to correctly assign a majority of the grandparent pairs and it outperforms its acyclic sampler and sequoia, particularly when the entire parental generation is missing (Table

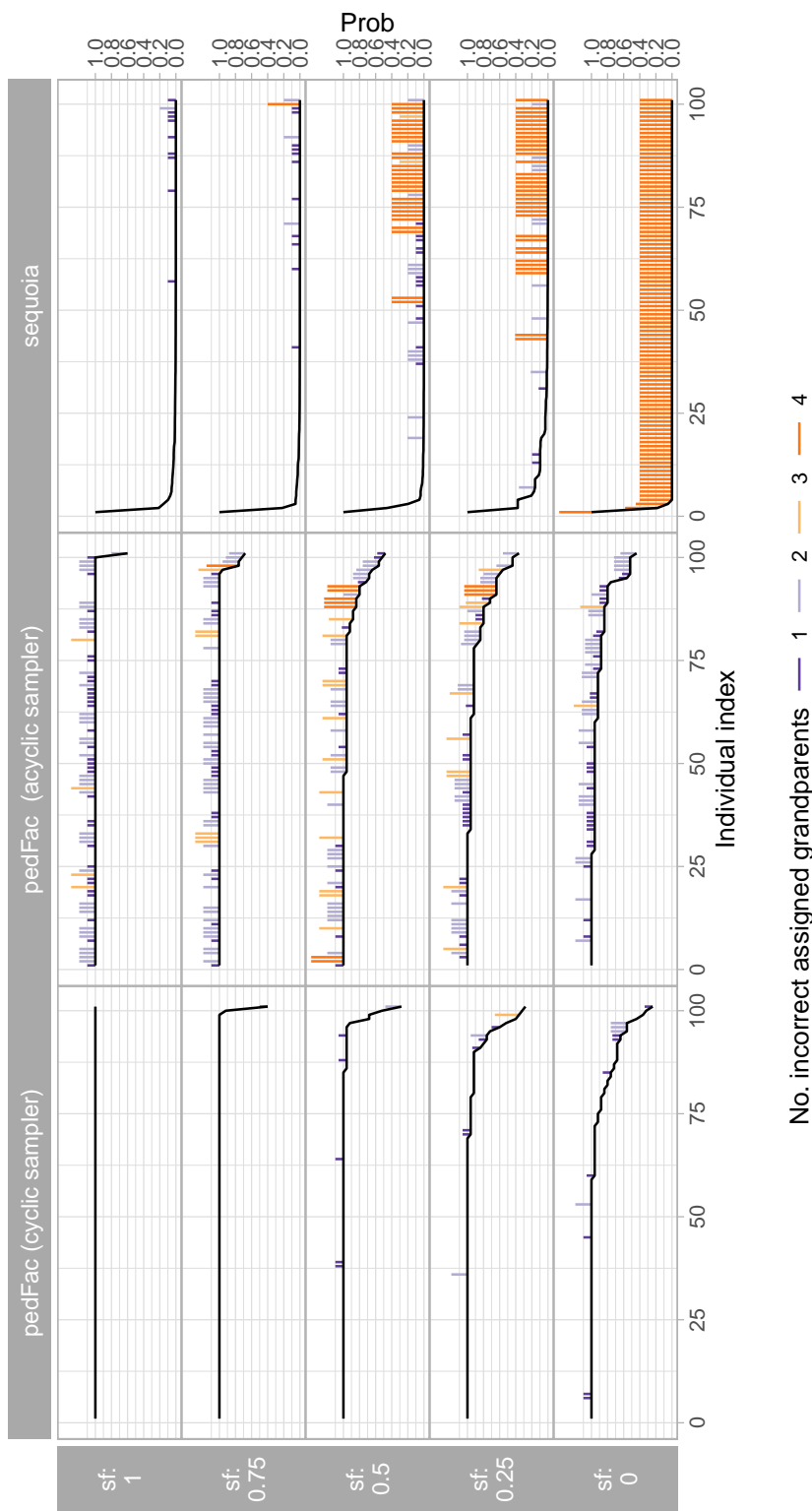


Figure 3.17: Grandparentage assignment for 101 individuals in a simulated cyclic pedigree. Observed individuals in the pedigree were genotyped at 200 simulated SNPs distributed uniformly across 34 chromosomes. Rows correspond to the sampling fraction of intermediate adults (i.e., parents in the second generation) and columns show different software and options (see text). In each panel, individuals are ordered according to a “confidence measure” for the inferred grandparent pairs (see text). The black curve shows this measure (y-axis) for each individual. The four vertical tick marks emanating from the black curve indicate the number of mis-assigned grandparents.

3.3). pedFac’s cyclic sampler mostly misses only one out of the four grandparents of an individual while the acyclic sampler, due to its inability to form loops, misses mostly half of the assigned grandparents (Fig 3.17). The fraction of correct assignment by the acyclic sampler does not change much with the different sampling rates and is likely responding to the fact that roughly 30% of the individuals in the pedigree participate in loops.

For sequoia, the fraction of correctly assigned grandparents plummets when more than half of the parental generation is unsampled. For this particular case study, sequoia has the tendency to either identify all, half or none of the four grandparents correctly.

3.5 Discussion

PedFac’s cyclic sampler can reconstruct multigenerational pedigrees with complex loops and mating structure and incomplete sampling. It outperforms its alternative acyclic sampler and sequoia in grandparent assignment, a challenging task with important applications in marine and freshwater ecology that includes applications such as identifying stock origin (Letcher and King 2001) and estimating cohort replacement rate (Sard et al. 2016).

The success of the pedFac’s cyclic sampler in grandparentage inference validates the use of the conditioning method to compute the joint likelihood of cyclic pedigrees, allowing us to propose and accept cyclic or acyclic pedigree structures, unlike the acyclic

sampler.

Even with the substantial advances the cyclic sampler has made for pedFac’s capabilities in grandparentage inference and learning cyclic pedigrees, there remains room for improvement. Key areas for improvement include providing a more robust uncertainty estimate from the posterior and reducing overall run times.

3.5.1 Mixing strategies

Since the proposal distribution for the cyclic sampler allows the formation of complex cyclic structure, the problem of “spatial bottlenecking” is much less pressing for the cyclic sampler than for the acyclic sampler. Due to this reason, the current cyclic sampler does not include any special type of mixing moves as developed for the acyclic sampler in section 2.2.8.3. Nonetheless, the cyclic sampler may still benefit from the inclusion of “swap” and “sub” types of moves. This is particularly apparent in the grandparentage inference problem where there are several instances where a single grandparent is misassigned with a posterior value of 1 (Fig 3.17). Such cases of incorrect inference with high estimated certainty are characteristic of poor mixing that could be improved with additional, specialized move types (Fig 2.10)

Thus, in order to further improve the mixing rate of the Metropolis-Hasting sampling chain, we can apply similar mixing strategies such as substitution and swapping moves, but with extra attention on keeping the proposed pedigree to be conditionally acyclic.

Such advancements and improvements are left for future work on pedFac.

3.5.2 A Pairwise Sampler for Loop-Breaker Shuffling

Section 3.3.3 describes a single-site update system for sampling the genotype values and loop-breaker status of variable nodes (individuals) within a loopy pedigree. Such single site updates may not mix as efficiently as desired. For that reason, we describe a pairwise loop-breaker shuffling proposal that makes larger moves. This is not implemented, but the ideas are sketched out here for completeness.

1. Select an *rcb* member r , $r \in R$ to focus on.
2. Select a loop breaker $v_i \in v_r$, where $I_{\text{breaker}}(v_i) = 1$ to focus on.
3. Sample a site (even if the node is already a loop breaker) to where this loop breaker can be relocated with probability proportional to the number of *rcb* members each site is included in. (i.e. probability $\propto M_{\text{rcb}}(v)$). Thus, the proposal distribution for relocating a focal loop breaker of variable node x_i to node x_j in this new pedigree configuration state \mathcal{P}' is specified as:

$$q(\mathcal{P}'|\mathcal{P}) = \frac{\|M_{\text{rcb}}(x_j)\|}{\sum_{k=1}^{\|V\|} \|M_{\text{rcb}}(x_k)\|} \quad (3.1)$$

The acceptance probability can be expressed as:

$$\min \left\{ \frac{q(\mathcal{P}|\mathcal{P}') p(y|\mathcal{P}') p(\mathcal{P}')}{q(\mathcal{P}'|\mathcal{P}) p(y|\mathcal{P}) p(\mathcal{P})}, 1 \right\}. \quad (3.2)$$

4. Simulate the fixed value for that proposed new site to be conditioned upon from its marginal (given the current loop breaker's genotype value, for convenience). If it is already a loop breaker, then choose its current value with probability 1.
5. Compute the likelihood l_{new} of the new configuration with the "old" loop breaker (i.e., the one that we "focused on" in (2) in the list) being conditioned upon (if the "old" loop breaker belongs to a separate rcb member) or unconditioned upon, otherwise, with, in both cases the new loop breaker being conditioned upon.
6. Compute the Hastings Ratio. To do so, we must compute the probability of proposing the reverse move, and also we must know the current joint likelihood (i.e., the likelihood before step 1).

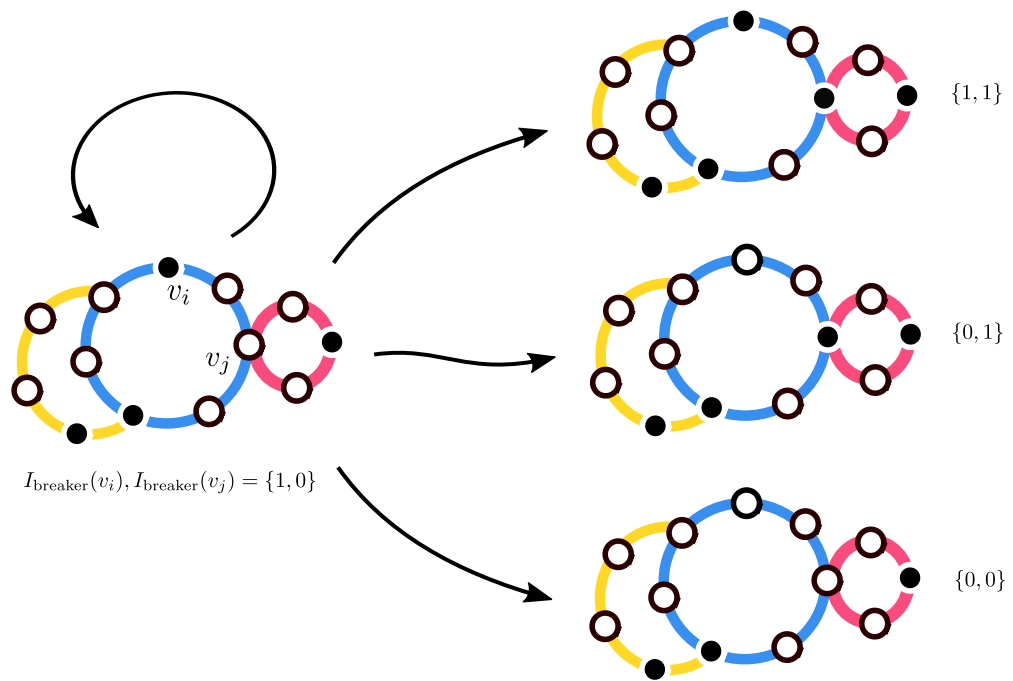


Figure 3.18: An illustrated instance of the pairwise loop-breaker shuffler.

Chapter 4

Visually guided curation of microhaplotypes from short read sequence data

4.1 Summary

Although next-generation sequencing (NGS) technologies are heralded for their ability to deliver sequence data across large parts of the genome, they are also increasingly used for routine genotyping of many individuals at a reduced subset of the genome. By targeting only a small number of loci (100 to a few thousand), using amplicon sequencing or capture arrays, hundreds to thousands of individuals can be genotyped in a single

NGS run at a low, per-individual cost. Initially, such efforts focused on genotyping only one single nucleotide polymorphism (SNP) from each sequenced region; however recent studies have documented the benefits of using microhaplotypes—the haplotypes formed by multiple SNPs on each sequencing read (Baetscher et al. 2017; McKinney et al. 2017). We present the R package `microhaplot`, that implements a workflow for extracting, vetting, and scoring microhaplotypes from sequenced amplicons. Starting from aligned sequence reads and a list of SNP positions, `microhaplot` automates the task of extracting microhaplotypes from each read. Following that, a rich, visual interface gives the user a rapid means of assessing data quality across individuals and loci and selecting suitable filtering criteria. A variety of filtering options are available to aid in the calling of genotypes, and specific data summaries and visualizations provide a means to rapidly understand the consequences of different filtering options, as well as to identify errors such as sample contamination or “index hopping” (Sinha et al. 2017). The R package `microhaplot` has already served as an indispensable tool for developing microhaplotype panels on multiple species.

4.2 Introduction

In just a few years, next-generation sequencing (NGS) technologies have revolutionized the study of evolution and ecology in both model and non-model organisms, and have become established as standard tools in molecular ecology. These technologies are now

capable of generating a huge volume of DNA sequence at relatively low cost, and can be used to perform a variety of functions, such as sequencing for *de novo* assembly of the genome of non-model species (for example, in Ruegg et al. 2018), evaluation of gene expression levels (Barshis et al. 2013), and genotyping via low-coverage, whole-genome, resequencing of individuals at hundreds of thousands to millions of SNPs (Foote et al. 2016; Thompson et al. 2020).

Using NGS to accurately determine the sequence of an individual at a region of the genome requires that sufficient sequencing reads of that region be obtained, so that the actual sequence can be distinguished from sequencing errors, and so that the genotypes of individuals with ploidy greater than one can be accurately inferred. This sets up a fundamental tradeoff, encountered in the design of NGS experiments, between the depth of sequence coverage at each site in the genome and the number of sites in the genome that are sequenced. If only a small number of genomic regions are targeted, then a sequencing run can obtain high read depth from many individuals. However, if a large number of genomic regions (up to the entire genome) is targeted, then it might be possible to obtain useful sequence from only a moderate number of individuals, and only at low depth.

How this tradeoff is negotiated depends heavily on the goals of any particular sequencing study. For example, obtaining wide coverage of the genome by resequencing the whole genome of a relatively few individuals is appropriate when seeking regions

of the genome associated with trait differences (Toews et al. 2016), or when assessing inbreeding within small populations by the distribution of runs of homozygosity (Kardos et al. 2018). On the other hand, a number of questions in molecular ecology can be answered without full, genomic-scale data, but require samples of many individuals. RAD-Seq and related methods (Andrews et al. 2016) which target genomic regions flanking enzyme recognition sites are commonly employed for reducing the representation of the genome down to 10s or 100s of thousands of regions; however some questions can be addressed with an even smaller number of carefully chosen markers. For example, researchers interested in identifying close relatives (like parent-offspring) in a sample of 10s of thousands of individuals (Bravington et al. 2016; Baetscher et al. 2019), or the population of origin of a large collection of individuals (Bradbury et al. 2018), may choose to use NGS to economically obtain sequences at a tiny portion of the genome from a very large number of individuals.

This type of Highly Targeted Sequencing (HTS), in which hundreds to thousands of individuals can be sequenced at hundreds to a few thousand genomic regions, has been approached primarily in two different ways (reviewed in Meek and Larson 2019). Hybridization-capture methods such as RAPTURE (Ali et al. 2016) or DArTcap (Feutry et al. 2020) reduce genome complexity by targeting regions that hybridize to a set of probes. Amplicon sequencing, on the other hand, uses PCR primers that target a number of small segments of the genome. The amplified products from a PCR reaction

are then sequenced on a high throughput instrument. This approach first appeared with the name GT-Seq (Campbell et al. 2015).

In the original application of GT-Seq, and many uses after that (e.g., Schmidt et al. 2020, Sjodin et al. 2020), only a single SNP per PCR-amplicon was used. Even if multiple variants occurred within an amplicon, the original bioinformatic workflow for GT-Seq discarded all of them except the focal SNP for the amplicon—in the process, potentially discarding a substantial amount of informative genetic variation. While the original GT-Seq bioinformatic workflow could have been amended to target additional SNPs within each amplicon, the workflow was configured such that this would also necessitate treating each SNP as a separate genetic marker. Such a treatment is undesirable, especially in the context of analysis programs that assume markers are not physically linked—an assumption that is dramatically violated for two SNPs within a single, short amplicon.

An alternative approach to analyzing amplicon sequencing data is to specify the entire short genomic region of the amplicon as a genetic marker whose alleles are designated by the actual DNA sequence within that region. Using first-generation sequencing technologies such as Sanger cycle sequencing (Sanger et al. 1977) in diploid (and higher ploidy) genomes, it was difficult or impossible to treat nuclear sequences as alleles, because the haplotypic phase of heterozygous sites was not known. Essentially, sequencing output from such technologies yielded a composite signal that was a mixture of the se-

quences in the region found on both the maternal and paternal chromosomes of an individual. In contrast, in most next generation sequencing technologies (like Illumina sequencing by synthesis), a sequencing read is not a mixture of the sequences from two homologous chromosomes in an individual, but rather directly derives from the DNA sequence upon only one of the two homologous chromosomes. Therefore, the sequence upon each read from next generation sequencing can be treated as a very short haplotype whose sequence specifies an allele at the marker. Such short haplotypes of variants found upon a single sequencing read were termed “microhaplotypes” by Baetscher et al. (2017), who noted that they provide dramatically greater power for relationship inference than merely using a single SNP from each amplicon. Use of multi-SNP haplotypes from next generation sequencing has also been shown to provide improved resolution for genetic stock identification (McKinney et al. 2017).

Surprisingly, although there has been a considerable amount of work in the area of “read-backed phasing” to improve variant detection in whole genome sequencing data (for example GATK’s HaplotypeCaller: McKenna et al. 2010; Poplin et al. 2018), and an enormous effort in statistical methods for phasing long-range haplotypes (Browning and Browning 2011), most mainstream bioinformatic packages do not offer “out-of-the-box” options for identifying microhaplotypes in amplicon sequencing data. Only relatively recently have molecular ecologists started developing methods for haplotyping RAD loci that take account of the reads upon which each variant allele is

found. Portnoy et al. (2015) introduce a package called `rad_haplotyper` (https://github.com/chollenbeck/rad_haplotyper) and describe its utility for filtering out paralogous loci. More recently, the software STACKS 2 offers an option of calling haplotypes on single or paired-end RAD read data using a graph-based algorithm to report a set of reliable haplotypes (Rochette et al. 2019). Neither of these approaches, however, are well-suited to amplicon sequencing data.

Several features distinguish amplicon sequencing data from the sequence data typically found in RAD-generated libraries. First, variation in PCR efficiency can lead to dramatically different read depths at different amplicons. In RAD data, substantial variation in read depths between loci is often an indicator of paralogous or otherwise duplicated regions. This is not necessarily the case with amplicon sequencing data; even in the absence of duplication, some amplicons may merely produce many times more reads than others. Such high read depths can prove troublesome for software tailored to consistent-depth, whole genome sequencing data. Second, the dynamics of the multiplex PCR process used for amplicon sequencing are poorly understood and amplicons can interact in surprising ways. This can be especially true in the presence of contamination between samples. Finally, panels developed for amplicon sequencing often comprise far fewer targeted regions than RADseq or whole genome sequencing experiments—few enough, in many cases, that the individual amplicons/markers can be individually inspected for quality control.

Over the last five years in our lab we have done amplicon sequencing to score haplotypes on tens of thousands of individuals across many species; mostly fish (Baetscher et al. 2017; Reid et al. 2020; Thompson et al. 2020), but also mammals (in preparation) and insects (Batz et al. 2020). In the process of this effort, we developed an R package called ‘microhaplot’ (<https://github.com/ngthomas/microhaplot>) that implements a bioinformatic pipeline for extracting the bases found at variant positions within amplicon sequences and summarizing that variation into recorded reads of microhaplotype alleles. Information about these reads/alleles is then processed within R and summarized in a rich variety of plots and tables using an interactive Shiny interface (Chang et al. 2017). This interface allows the user to rapidly scan multiple features of different amplicons as well as visualize patterns at individual loci, permitting rapid and efficient curation of microhaplotype panels, identification of aberrant individuals and genotypes, and the microhaplotype genotyping of many individuals.

In the following, we start with a more formal definition of what we mean by a “microhaplotype,” we describe the microhaplot workflow for extracting microhaplotypes from amplicons sequenced on an NGS machine, and then give a brief description of microhaplot’s interactive interface. Microhaplot has proven indispensable in our laboratory workflows and we foresee it being widely useful as other labs adopt microhaplotype genotyping from amplicon sequences.

4.3 Microhaplotypes

For the purposes here, we define a microhaplotype as a condensed string of DNA bases found at a set of aligned positions, known to harbor variants, on a single sequencing read. The value at each variant is named using a single, uppercase letter. The possible values that microhaplotypes use are the four nucleotide bases (A, C, G, T), a deletion relative to the reference sequence (denoted by an X), and a non-scored base on a read, (denoted by an N). For example, if a read spans 5 sites with known single nucleotide variant positions, a microhaplotype might be scored as, “TTCGA”. In this simplest case, a microhaplotype allele is a string resulting from concatenating all of the nucleotides on a read that map to the known-variant positions on a reference sequence.

More formally, we describe a microhaplotype as follows: let read sequence r be a string of nucleotides N_r with aligned positions P_r . Given P_v - a set of positions of variants, relative to the reference sequence, a microhaplotype of sequence r is a string of nucleotides $n_m \in N_r$ with aligned positions P_v , and in the same order that those positions occur in P_v . Therefore, in order for microhaplotypes to be constructed, we assume the presence of two components: a reference sequence to which the amplicon read have been aligned, and a list of variant positions.

4.4 Extracting microhaplotypes from sequence reads

The R package `microhaplot` is not a method for variant detection. Therefore the desired variants (those at positions P_v for each amplicon) must be identified prior to running `microhaplot`.

A typical workflow is as follows:

- The initial inputs are: 1) reads from amplicon sequences, 2) a reference that includes sequences to which the amplicons are expected to align.
- The first step is mapping the sequence reads to the reference. This is accomplished with standard software such as `bwa` (Li 2013) or `bowtie` (Langmead and Salzberg 2012).
- Subsequently, variants at particular positions should be identified. Either those positions may be known from previous sequencing, or they must be identified amongst the recently aligned reads, using, for example `bcftools` (CITE), FreeBayes (Garrison and Marth 2012), or GATK Haplotypecaller (McKenna et al. 2010). Regardless of how these variants, and the positions they occupy, are identified, they should be stored in a VCF file (though it should be noted that the individual information is not used; only the CHROM and POS columns are used).
- Taking as input the SAM files (one for each sequenced individual), and the VCF

file, the function `prepHaplotFiles()` calls a PERL script to extract aligned position information from each read in each SAM file, and identify the microhaplotype of each read. These microhaplotypes are saved into a data frame and written to disk for later use, including visualization with the Shiny app.

4.5 Visualization of Microhaplotypes

Given data frames of previously-processed microhaplotype tables, the microhaplot Shiny app generates plot summaries of filtered microhaplotypes using simple read-depth based filtering criteria. These plots include genotype-biplots (Figure 4.1a), read depth and fraction of callable haplotypes (Figure 4.1b), Hardy-Weinberg equilibrium plots (Figure 4.1c), and more. The examples shown of the different plots were made using the microhaplot Shiny app on a subset of amplicon data from kelp rockfish (Baetscher et al. 2017).

Full details on how to use and interpret such plots and summaries are provided in the `microhaplot` package vignettes, which are most easily read under the “Articles” tab at <https://ngthomas.github.io/microhaplot/>. The R package `microhaplot` is available for download or remote installation from its github repo: <https://github.com/ngthomas/microhaplot>.

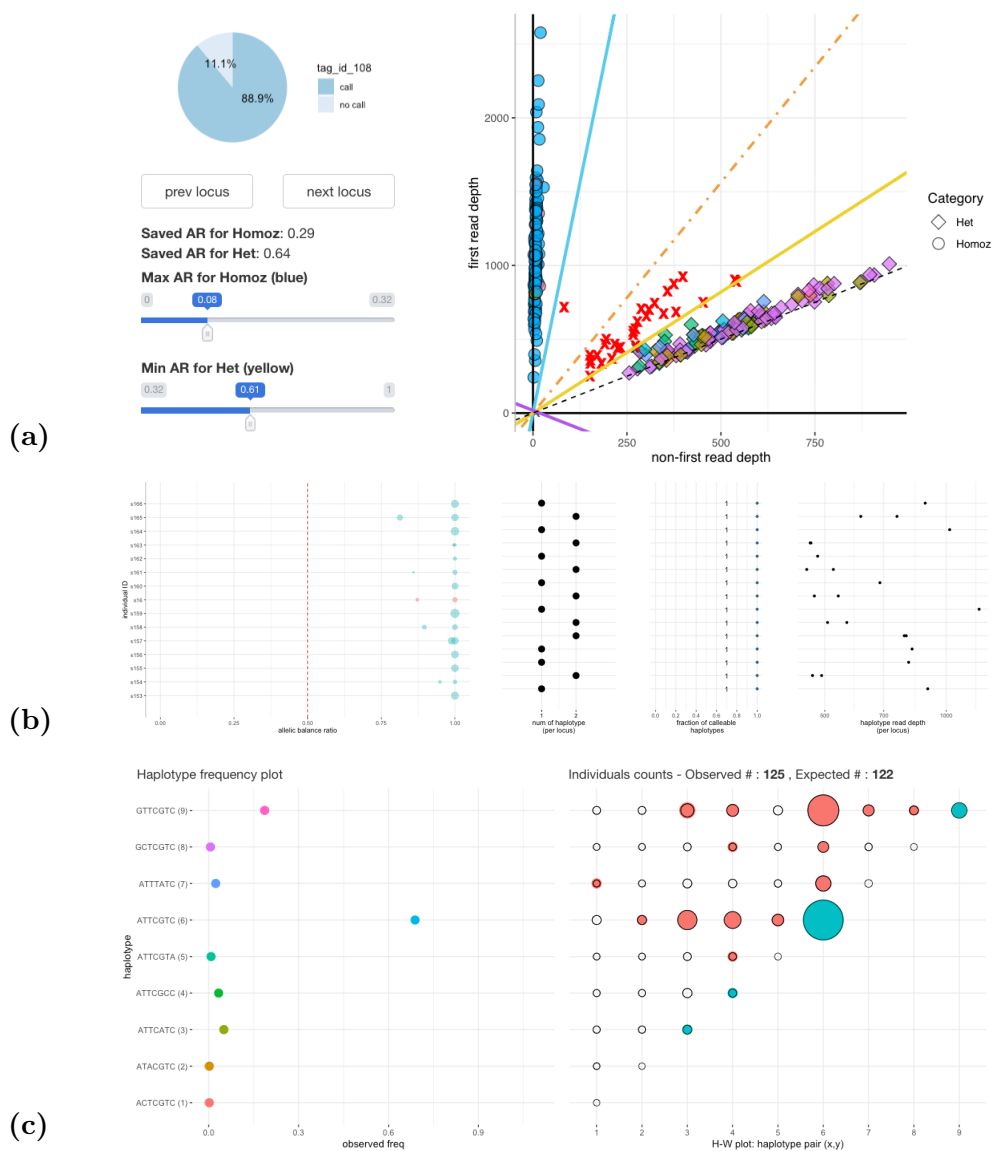


Figure 4.1: Screenshots of microhaplotype summary plots from `microhaplot`'s shiny app. (a) The “genotype-biplot” output in refining genotype call. (b) `microhaplot`'s summary of haplotype read depth and callable fraction of haplotypes. (c) Frequency of observed genotypes and values expected under Hardy-Weinberg equilibrium, summarized by `microhaplot`. The text in the upper right refers to the currently selected genotype, represented by the large blue-green circle.

Bibliography

- A. Abadía-Cardoso, E. C. Anderson, D. E. Pearse, and J. Carlos Garza. Large-scale parentage analysis reveals reproductive patterns and heritability of spawn timing in a hatchery population of steelhead (*oncorhynchus mykiss*). *Molecular ecology*, 22(18):4733–4746, 2013.
- G. R. Abecasis, W. O. Cookson, and L. R. Cardon. Pedigree tests of transmission disequilibrium. *European Journal of Human Genetics*, 8(7):545–551, 2000.
- O. A. Ali, S. M. O'Rourke, S. J. Amish, M. H. Meek, G. Luikart, C. Jeffres, and M. R. Miller. RAD Capture (Rapture): Flexible and Efficient Sequence-Based Genotyping. *Genetics*, 202(2):389–400, Feb. 2016. ISSN 0016-6731, 1943-2631. doi: 10.1534/genetics.115.183665.
- L. Almasy and J. Blangero. Multipoint quantitative-trait linkage analysis in general pedigrees. *The American Journal of Human Genetics*, 62(5):1198–1211, 1998.

- A. Almudevar. A simulated annealing algorithm for maximum likelihood pedigree reconstruction. *Theoretical population biology*, 63(2):63–75, 2003.
- A. Almudevar and J. LaCombe. On the choice of prior density for the bayesian analysis of pedigree structure. *Theoretical Population Biology*, 81(2):131 – 143, 2012. ISSN 0040-5809. doi: <https://doi.org/10.1016/j.tpb.2011.12.003>. URL <http://www.sciencedirect.com/science/article/pii/S0040580911001067>.
- E. C. Anderson and J. C. Garza. The power of single-nucleotide polymorphisms for large-scale parentage inference. *Genetics*, 172(4):2567–2582, 2006.
- E. C. Anderson and T. C. Ng. Bayesian pedigree inference with small numbers of single nucleotide polymorphisms via a factor-graph representation. *Theoretical population biology*, 107:39–51, 2016.
- K. R. Andrews, J. M. Good, M. R. Miller, G. Luikart, and P. A. Hohenlohe. Harnessing the power of RADseq for ecological and evolutionary genomics. *Nature Reviews Genetics*, 17(2):81–92, Feb. 2016. ISSN 1471-0064. doi: 10.1038/nrg.2015.28.
- T. Aykanat, S. Johnston, D. Cotter, T. Cross, R. Poole, P. Prodohl, T. Reed, G. Rogan, P. McGinnity, and C. Primmer. Molecular pedigree reconstruction and estimation of evolutionary parameters in a wild atlantic salmon river system with incomplete sampling: a power analysis. *BMC Evolutionary Biology*, 14(1):68, 2014. ISSN 1471-2148.

doi: 10.1186/1471-2148-14-68. URL <http://www.biomedcentral.com/1471-2148/14/68>.

D. S. Baetscher, A. J. Clemento, T. C. Ng, E. C. Anderson, and J. C. Garza. Micro-haplotypes provide increased power from short-read dna sequences for relationship inference. *Molecular ecology resources*, 2017.

D. S. Baetscher, E. C. Anderson, E. A. Gilbert-Horvath, D. P. Malone, E. T. Saarman, M. H. Carr, and J. C. Garza. Dispersal of a nearshore marine fish connects marine reserves and adjacent fished areas along an open coast. *Molecular Ecology*, 0(ja), 2019. ISSN 1365-294X. doi: 10.1111/mec.15044.

D. J. Barshis, J. T. Ladner, T. A. Oliver, F. O. Seneca, N. Traylor-Knowles, and S. R. Palumbi. Genomic basis for coral resilience to climate change. *Proceedings of the National Academy of Sciences*, 110(4):1387–1392, Jan. 2013. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1210224110.

Z. A. Batz, A. J. Clemento, J. Fitzenwanker, T. J. Ring, J. C. Garza, and P. A. Armbruster. Rapid adaptive evolution of the diapause program during range expansion of an invasive mosquito. *Evolution*, 2020.

L. E. Baum et al. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8, 1972.

- A. Becker, R. Bar-Yehuda, and D. Geiger. Randomized algorithms for the loop cutset problem. *Journal of Artificial Intelligence Research*, 12:219–234, 2000.
- C. Béréanos, P. A. Ellis, J. G. Pilkington, and J. M. Pemberton. Estimating quantitative genetic parameters in wild populations: a comparison of pedigree and genomic approaches. *Molecular ecology*, 23(14):3434–3451, 2014.
- C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Proceedings of ICC '93 - IEEE International Conference on Communications*, volume 2, pages 1064–1070 vol.2, 1993.
- I. R. Bradbury, B. F. Wringe, B. Watson, I. Paterson, J. Horne, R. Beiko, S. J. Lehnert, M. Clément, E. C. Anderson, N. W. Jeffery, S. Duffy, E. Sylvester, M. Robertson, and P. Bentzen. Genotyping-by-sequencing of genome-wide microsatellite loci reveals fine-scale harvest composition in a coastal Atlantic salmon fishery. *Evolutionary Applications*, 11(6):918–930, 2018. ISSN 1752-4571. doi: 10.1111/eva.12606.
- M. V. Bravington, P. M. Grewe, and C. R. Davies. Absolute abundance of southern bluefin tuna estimated by close-kin mark-recapture. *Nature Communications*, 7:13162, Nov. 2016. ISSN 2041-1723. doi: 10.1038/ncomms13162.
- S. R. Browning and B. L. Browning. Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714, 2011.

- N. R. Campbell, S. A. Harmon, and S. R. Narum. Genotyping-in-Thousands by sequencing (GT-seq): A cost effective SNP genotyping method based on custom amplicon sequencing. *Molecular Ecology Resources*, 15(4):855–867, 2015. ISSN 1755-0998. doi: 10.1111/1755-0998.12357.
- C. Cannings and E. A. Thompson. Ascertainment in the sequential sampling of pedigrees. *Clinical Genetics*, 12(4):208–212, 1977. ISSN 1399-0004. doi: 10.1111/j.1399-0004.1977.tb00928.x. URL <http://dx.doi.org/10.1111/j.1399-0004.1977.tb00928.x>.
- C. Cannings, E. Thompson, and M. Skolnick. Probability functions on complex pedigrees. *Advances in Applied Probability*, pages 26–61, 1978.
- W. Chang, J. Cheng, J. Allaire, Y. Xie, J. McPherson, et al. Shiny: web application framework for r. *R package version*, 1(5), 2017.
- G. Chartrand, L. Lesniak, and P. Zhang. *Graphs & Digraphs, Fifth Edition*. Chapman & Hall/CRC, 5th edition, 2010. ISBN 1439826277.
- N. Chen, I. Juric, E. J. Cosgrove, R. Bowman, J. W. Fitzpatrick, S. J. Schoech, A. G. Clark, and G. Coop. Allele frequency dynamics in a pedigreed natural population. *Proceedings of the National Academy of Sciences*, 116(6):2158–2164, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1813852116. URL <https://www.pnas.org/content/116/6/2158>.

- M. R. Christie, M. L. Marine, and M. S. Blouin. Who are the missing parents? grand-parentage analysis identifies multiple sources of gene flow into a wild population. *Molecular Ecology*, 20(6):1263–1276, 2011.
- A. J. Clemento. *Creation and Utilization of Novel Genetic Methods For Studying and Improving Management of Chinook Salmon Populations*. PhD thesis, University of California, Santa Cruz, 2014.
- G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990.
- S. Creel and E. Rosenblatt. Using pedigree reconstruction to estimate population size: genotypes are more than individually unique marks. *Ecology and evolution*, 3(5):1294–1304, 2013.
- D. Eaton and K. Murphy. Bayesian structure learning using dynamic programming and mcmc. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, UAI’07, pages 101–108, Arlington, Virginia, United States, 2007. AUAI Press. ISBN 0-9749039-3-0. URL <http://dl.acm.org/citation.cfm?id=3020488>. 3020501.
- R. C. Elston and J. Stewart. A general model for the genetic analysis of pedigree data. *Hum Hered*, 21(6):523–542, 1971.

- P. Festa, P. M. Pardalos, and M. G. Resende. Feedback set problems. In *Handbook of combinatorial optimization*, pages 209–258. Springer, 1999.
- P. Feutry, F. Devloo-Delva, A. Tran Lu Y, S. Mona, R. M. Gunasekera, G. Johnson, R. D. Pillans, D. Jaccoud, A. Kilian, D. L. Morgan, et al. One panel to rule them all: Dartcap genotyping for population structure, historical demography, and kinship analyses, and its application to a threatened shark. *Molecular Ecology Resources*, 2020.
- J. A. Firth, J. D. Hadfield, A. W. Santure, J. Slate, and B. C. Sheldon. The influence of nonrandom extra-pair paternity on heritability estimates derived from wild pedigrees. *Evolution*, 69(5):1336–1344, 2015. doi: <https://doi.org/10.1111/evo.12649>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/evo.12649>.
- F. V. Fomin, S. Gaspers, A. V. Pyatkin, and I. Razgon. On the minimum feedback vertex set problem: Exact and enumeration algorithms. *Algorithmica*, 52(2):293–307, 2008.
- A. D. Foote, N. Vijay, M. C. Ávila-Arcos, R. W. Baird, J. W. Durban, M. Fumagalli, R. A. Gibbs, M. B. Hanson, T. S. Korneliussen, M. D. Martin, K. M. Robertson, V. C. Sousa, F. G. Vieira, T. Vinař, P. Wade, K. C. Worley, L. Excoffier, P. A. Morin, M. T. P. Gilbert, and J. B. W. Wolf. Genome-culture coevolution promotes

- rapid divergence of killer whale ecotypes. *Nature Communications*, 7:11693, May 2016. ISSN 2041-1723. doi: 10.1038/ncomms11693.
- M. J. Ford, M. B. Hanson, J. A. Hempelmann, K. L. Ayres, C. K. Emmons, G. S. Schorr, R. W. Baird, K. C. Balcomb, S. K. Wasser, K. M. Parsons, et al. Inferred paternity and male reproductive success in a killer whale (*Orcinus orca*) population. *Journal of Heredity*, 102(5):537–553, 2011.
- W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International journal of computer vision*, 40(1):25–47, 2000.
- E. Garrison and G. Marth. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907*, 2012.
- J. Gauzere, J. M. Pemberton, S. Morris, A. Morris, L. E. B. Kruuk, and C. A. Walling. The genetic architecture of maternal effects across ontogeny in the red deer. *Evolution*, 74(7):1378–1391, 2020. doi: <https://doi.org/10.1111/evo.14000>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/evo.14000>.
- S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, Nov 1984. doi: 10.1109/TPAMI.1984.4767596.
- J. C. Glaubitz, O. E. Rhodes, and J. A. DeWoody. Prospects for inferring pairwise

- relationships with single nucleotide polymorphisms. *Molecular Ecology*, 12(4):1039–1047, 2003.
- M. E. Goddard and B. J. Hayes. Mapping genes for complex traits in domestic animals and their use in breeding programmes. *Nature Reviews Genetics*, 10(6):381–391, 2009.
- R. J. B. Goudie and S. Mukherjee. A gibbs sampler for learning dags. *J. Mach. Learn. Res.*, 17(1):1032–1070, Jan. 2016. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=2946645.2946675>.
- F. Goyache, J. P. Gutiérrez, I. Fernández, E. Gómez, I. Álvarez, J. Díez, and L. Royo. Using pedigree information to monitor genetic variability of endangered populations: the xalda sheep breed of asturias as an example. *Journal of Animal Breeding and Genetics*, 120(2):95–105, 2003.
- W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. URL <http://www.jstor.org/stable/2334940>.
- J. Huisman. Pedigree reconstruction from snp data: parentage assignment, sibship clustering and beyond. *Molecular Ecology Resources*, 17(5):1009–1024, 2017. doi: <https://doi.org/10.1111/1755-0998.12665>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/1755-0998.12665>.

- S. Isberg, Y. Chen, S. Barker, and M. C. Analysis of Microsatellites and Parentage Testing in Saltwater Crocodiles. *Journal of Heredity*, 5(95):445–449, 2004.
- A. G. Jones and W. R. Ardren. Methods of parentage analysis in natural population. *Molecular ecology*, 10(12):2511–2523, 2003.
- B. Jones. Maximum likelihood inference for seed and pollen dispersal distributions. *Journal of agricultural, biological, and environmental statistics*, 8(2):170–183, 2003.
- O. R. Jones and J. Wang. Colony: a program for parentage and sibship inference from multilocus genotype data. *Molecular Ecology Resources*, 10(3):551–555, 2010.
- M. Kardos, M. Åkesson, T. Fountain, O. Flagstad, O. Liberg, P. Olason, H. Sand, P. Wabakken, C. Wikenros, and H. Ellegren. Genomic consequences of intensive inbreeding in an isolated wolf population. *Nature Ecology & Evolution*, 2(1):124, Jan. 2018. ISSN 2397-334X. doi: 10.1038/s41559-017-0375-4.
- B. Kirkpatrick, S. C. Li, R. M. Karp, and E. Halperin. Pedigree reconstruction using identity by descent. *Journal of Computational Biology*, 18(11):1481–1493, 2011.
- D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- L. Kruglyak, M. J. Daly, M. P. Reeve-Daly, and E. S. Lander. Parametric and nonpara-

- metric linkage analysis: a unified multipoint approach. *American journal of human genetics*, 58(6):1347, 1996.
- L. Kruuk and W. Hill. Introduction. evolutionary dynamics of wild populations: the use of long-term pedigree data. *Proc Biol Sci*, 275(1635):593–6, 2008.
- F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.
- E. S. Lander and P. Green. Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Sciences*, 84(8):2363–2367, 1987.
- K. Lange and R. Elston. Extensions to pedigree analysis i. likelihood calculations for simple and complex pedigrees. *Human heredity*, 25(2):95–105, 1974.
- K. Lange, M. Boehnke, and J. M. Opitz. Extensions to pedigree analysis. iv. covariance components models for multivariate traits. *American journal of medical genetics*, 14(3):513–524, 1983.
- K. Lange, J. C. Papp, J. S. Sinsheimer, R. Sripracha, H. Zhou, and E. M. Sobel. Mendel: the swiss army knife of genetic analysis programs. *Bioinformatics*, 29(12):1568–1570, 2013.
- B. Langmead and S. Salzberg. Langmead b, salzberg sl.. fast gapped-read alignment

- with bowtie 2. *nat methods* 9: 357-359. *Nature methods*, 9:357–9, 03 2012. doi: 10.1038/nmeth.1923.
- S. L. Lauritzen and N. A. Sheehan. Graphical models for genetic analyses. *Statist. Sci.*, 18(4):489–514, 11 2003a. doi: 10.1214/ss/1081443232. URL <https://doi.org/10.1214/ss/1081443232>.
- S. L. Lauritzen and N. A. Sheehan. Graphical models for genetic analyses. *Statistical Science*, pages 489–514, 2003b.
- A. Lempel and I. Cederbaum. Minimum feedback arc and vertex sets of a directed graph. *IEEE Transactions on circuit theory*, 13(4):399–403, 1966.
- B. H. Letcher and T. L. King. Parentage and grandparentage assignment with known and unknown matings: application to connecticut river atlantic salmon restoration. *Canadian Journal of Fisheries and Aquatic Sciences*, 58(9):1812–1821, 2001. doi: 10.1139/f01-125. URL <https://doi.org/10.1139/f01-125>.
- H. Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem, 2013.
- M. Lynch, B. Walsh, et al. *Genetics and analysis of quantitative traits*, volume 1. Sinauer Sunderland, 1998.

- J. W. MacCluer, J. L. VandeBerg, B. Read, and O. A. Ryder. Pedigree analysis by computer simulation. *Zoo biology*, 5(2):147–160, 1986.
- D. Madigan and J. York. Bayesian graphical models for discrete data, 1993.
- T. Marshall, J. Slate, L. Kruuk, and J. Pemberton. Statistical confidence for likelihood-based paternity inference in natural populations. *Molecular Ecology*, 7(5):639–655, 1998. ISSN 0962-1083.
- A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytzky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, et al. The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303, 2010.
- G. J. McKinney, J. E. Seeb, and L. W. Seeb. Managing mixed-stock fisheries: genotyping multi-snp haplotypes increases power for genetic stock identification. *Canadian journal of fisheries and aquatic sciences*, 74(4):429–434, 2017.
- T. R. Meagher and E. Thompson. The relationship between single parent and parent pair genetic likelihoods in genealogy reconstruction. *Theoretical Population Biology*, 29(1):87–106, Feb. 1986. doi: 10.1016/0040-5809(86)90006-7. URL [http://dx.doi.org/10.1016/0040-5809\(86\)90006-7](http://dx.doi.org/10.1016/0040-5809(86)90006-7).
- T. R. Meagher and E. Thompson. Analysis of parentage for naturally established seedlings of *Chamaelirium luteum* (Liliaceae). *Ecology*, pages 803–812, 1987.

- M. H. Meek and W. A. Larson. The future is now: Amplicon sequencing and sequence capture usher in the conservation genomics era. *Molecular Ecology Resources*, 19(4): 795–803, 2019.
- I. Meiri, R. Dechter, and J. Pearl. *Tree decomposition with applications to constraint processing*. University of California (Los Angeles). Computer Science Department, 1991.
- J. M. Mooij. Loop corrections for approximate inference on factor graphs. *Journal of Machine Learning Research*, pages 1113–1143, 2007. URL <http://www.jmlr.org/papers/volume8/mooij07a/mooij07a.pdf>.
- J. M. Mooij and H. J. Kappen. Sufficient conditions for convergence of loopy belief propagation. *CoRR*, abs/1207.1405, 2012. URL <http://arxiv.org/abs/1207.1405>.
- K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI’99, pages 467–475, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-614-9. URL <http://dl.acm.org/citation.cfm?id=2073796.2073849>.
- R. Nielsen, D. K. Mattila, P. J. Clapham, and P. J. Palsboll. Statistical approaches to paternity analysis in natural populations and applications to the North Atlantic humpback whale. *Genetics*, 157(4):1673–1682, 2001.

- J. Pearl. Reverend bayes on inference engines: a distributed hierarchical approach. In *in Proceedings of the National Conference on Artificial Intelligence*, pages 133–136, 1982.
- J. Pearl. Chapter 4 - belief updating by network propagation. In J. Pearl, editor, *Probabilistic Reasoning in Intelligent Systems*, pages 143 – 237. Morgan Kaufmann, San Francisco (CA), 1988. ISBN 978-0-08-051489-5. doi: <https://doi.org/10.1016/B978-0-08-051489-5.50010-2>. URL <http://www.sciencedirect.com/science/article/pii/B9780080514895500102>.
- J. Pemberton. Wild pedigrees: the way forward. *Proceedings of the Royal Society Biological Sciences*, 275(1635):613–621, 2008. ISSN 0962-8452.
- T. J. Pemberton, C. Wang, J. Z. Li, and N. A. Rosenberg. Inference of unexpected genetic relatedness among individuals in hapmap phase iii. *The American Journal of Human Genetics*, 87(4):457–464, 2010.
- R. Poplin, V. Ruano-Rubio, M. A. DePristo, T. J. Fennell, M. O. Carneiro, G. A. Van der Auwera, D. E. Kling, L. D. Gauthier, A. Levy-Moonshine, D. Roazen, K. Shakir, J. Thibault, S. Chandran, C. Whelan, M. Lek, S. Gabriel, M. J. Daly, B. Neale, D. G. MacArthur, and E. Banks. Scaling accurate genetic variant discovery to tens of thousands of samples. *bioRxiv*, 2018. doi: 10.1101/201178. URL <https://www.biorxiv.org/content/early/2018/07/24/201178>.

- D. Portnoy, J. Puritz, C. Hollenbeck, J. Gelsleichter, D. Chapman, and J. Gold. Selection and sex-biased dispersal in a coastal shark: The influence of philopatry on adaptive variation. *Molecular Ecology*, 24(23):5877–5885, 2015.
- J. M. Reid, P. Nietlisbach, M. E. Wolak, L. F. Keller, and P. Arcese. Individuals’ expected genetic contributions to future generations, reproductive value, and short-term metrics of fitness in free-living song sparrows (*melospiza melodia*). *Evolution Letters*, 3(3):271–285, 2019. doi: <https://doi.org/10.1002/evl3.118>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/evl3.118>.
- K. Reid, J. Carlos Garza, S. R. Gephard, A. Caccone, D. M. Post, and E. P. Palkovacs. Restoration-mediated secondary contact leads to introgression of alewife ecotypes separated by a colonial-era dam. *Evolutionary applications*, 13(4):652–664, 2020.
- M. Riester, P. F. Stadler, and K. Klemm. Franz: reconstruction of wild multi-generation pedigrees. *Bioinformatics*, 25(16):2134–2139, 2009.
- N. C. Rochette, A. G. Rivera-Colón, and J. M. Catchen. Stacks 2: Analytical methods for paired-end sequencing improve radseq-based population genomics. *bioRxiv*, 2019. doi: 10.1101/615385. URL <https://www.biorxiv.org/content/early/2019/04/22/615385>.
- K. Ruegg, R. A. Bay, E. C. Anderson, J. F. Saracco, R. J. Harrigan, M. Whitfield, E. H. Paxton, and T. B. Smith. Ecological genomics predicts climate vulnerability in

- an endangered southwestern songbird. *Ecology Letters*, 21(7):1085–1096, 2018. ISSN 1461-0248. doi: 10.1111/ele.12977.
- B. M. S. DNA-based methods for pedigree reconstruction and kinship analysis in natural populations. *Trends in Ecology and Evolution*, 18(10):503–511, 2003.
- F. Sanger, S. Nicklen, and A. R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467, 1977.
- N. M. Sard, D. P. Jacobson, and M. A. Banks. Grandparentage assignments identify unexpected adfluvial life history tactic contributing offspring to a reintroduced population. *Ecology and Evolution*, 6(19):6773–6783, 2016. doi: <https://doi.org/10.1002/ece3.2378>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/ece3.2378>.
- D. A. Schmidt, P. Govindarajulu, K. W. Larsen, and M. A. Russello. Genotyping-in-thousands by sequencing reveals marked population structure in western rattlesnakes to inform conservation status. *Ecology and Evolution*, 10(14):7157–7172, 2020.
- N. A. Sheehan and T. Egeland. Structured incorporation of prior information in relationship identification problems. *Annals of Human Genetics*, 71(4):501–518, 2007.
- N. A. Sheehan, M. Bartlett, and J. Cussens. Improved maximum likelihood reconstruction of complex multi-generational pedigrees. *Theoretical population biology*, 97:11–19, 2014.

- R. Sinha, G. Stanley, G. S. Gulati, C. Ezran, K. J. Travaglini, E. Wei, C. K. F. Chan, A. N. Nabhan, T. Su, R. M. Morganti, et al. Index switching causes “spreading-of-signal” among multiplexed samples in illumina hiseq 4000 dna sequencing. *BioRxiv*, page 125724, 2017.
- B. M. Sjodin, R. L. Irvine, and M. A. Russello. Rapidrat: Development, validation and application of a genotyping-by-sequencing panel for rapid biosecurity and invasive species management. *PloS one*, 15(6):e0234694, 2020.
- K. Slooten. Validation of dna-based identification software by computation of pedigree likelihood ratios. *Forensic Science International: Genetics*, 5(4):308–315, 2011.
- E. Sobel, J. C. Papp, and K. Lange. Detection and integration of genotyping errors in statistical genetics. *The American Journal of Human Genetics*, 70(2):496–508, 2002.
- R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.
- S. Tatikonda and M. I. Jordan. Loopy belief propogation and gibbs measures. *CoRR*, abs/1301.0605, 2013. URL <http://arxiv.org/abs/1301.0605>.
- S. C. Thomas and W. G. Hill. Sibship reconstruction in hierarchical population structures using markov chain monte carlo techniques. *Genetical research*, 79(03):227–234, 2002.

- E. A. Thompson. Statistical inference from genetic data on pedigrees. *NSF-CBMS Regional Conference Series in Probability and Statistics*, 6:pp. i–xiv+1–169, 2000. ISSN 19355920. URL <http://www.jstor.org/stable/4153187>.
- E. A. Thompson, C. Cannings, and M. H. Skolnick. Ancestral inference: I. the problem and the method. *Annals of Human Genetics*, 42(1):95–108, 1978. doi: 10.1111/j.1469-1809.1978.tb00934.x. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1978.tb00934.x>.
- N. F. Thompson, E. C. Anderson, A. J. Clemento, M. A. Campbell, D. E. Pearse, J. W. Hearsey, A. P. Kinziger, and J. C. Garza. A complex phenotype in salmon controlled by a simple change in migratory timing. *Science*, 370(6516):609–613, 2020.
- D. P. L. Toews, S. A. Taylor, R. Vallender, A. Brelsford, B. G. Butcher, P. W. Messer, and I. J. Lovette. Plumage Genes and Little Else Distinguish the Genomes of Hybridizing Warblers. *Current Biology*, 26(17):2313–2318, Sept. 2016. ISSN 0960-9822. doi: 10.1016/j.cub.2016.06.034.
- L. A. Vøllestad, D. Serbezov, A. Bass, L. Bernatchez, E. M. Olsen, A. Taugbøl, and J. Grant. Small-scale dispersal and population structure in stream-living brown trout (*salmo trutta*) inferred by mark–recapture, pedigree reconstruction, and population genetics. *Canadian Journal of Fisheries and Aquatic Sciences*, 69(9):1513–1524, 2012.

- J. Wang. Sibship reconstruction from genetic data with typing errors. *Genetics*, 166(4):1963–1979, 2004.
- J. Wang and A. W. Santure. Parentage and sibship inference from multilocus genotype data under polygamy. *Genetics*, 181(4):1579–1594, 2009.
- B. S. Weir, A. D. Anderson, and A. B. Hepler. Genetic relatedness analysis: modern data and new challenges. *Nature Reviews Genetics*, 7(10):771–780, 2006.
- Y. Weiss. Belief propagation and revision in networks with loops. Technical report, Massachusetts Institute of Technology, USA, 1997.
- D. A. Wells, M. A. Cant, H. J. Nichols, and J. I. Hoffman. A high-quality pedigree and genetic markers both reveal inbreeding depression for quality but not survival in a cooperative mammal. *Molecular Ecology*, 27(9):2271–2288, 2018. doi: <https://doi.org/10.1111/mec.14570>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/mec.14570>.
- S. Wright. Coefficients of inbreeding and relationship. *The American Naturalist*, 56(645): pp. 330–338, 1922. ISSN 00030147. URL <http://www.jstor.org/stable/2456273>.
- S. Wright and H. C. McPhee. *An approximate method of calculating coefficients of inbreeding and relationship from livestock pedigrees*. Journal of agricultural research, 1925.

M. Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 253–264, New York, NY, USA, 1978. Association for Computing Machinery. ISBN 9781450374378. doi: 10.1145/800133.804355. URL <https://doi.org/10.1145/800133.804355>.

D. Younger. Minimum feedback arc sets for a directed graph. *IEEE Transactions on Circuit Theory*, 10(2):238–245, 1963.