The LITSE Algorithm: Theory and Application

by

Curt Christopher Hansen

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

Doctor of Philosophy

in

Biostatistics

and the

Designated Emphasis

in

Computational Data Science and Engineering

in the

Graduate Division

of the

University of California, BERKELEY

Committee in charge:

Professor Alan E. Hubbard, Chair
Professor Sandrine Dudoit
Assistant Professor Bryan R. Greenhouse
Professor Deborah A. Nolan

Fall 2015

The LITSE Algorithm: Theory and Application

Abstract

The LITSE Algorithm: Theory and Application

by

Curt Christopher Hansen

Doctor of Philosophy in Biostatistics

University of California, BERKELEY

Professor Alan E. Hubbard, Chair

In this dissertation, we present a novel method – the Learning with Iteration and Tree-based Search Estimation algorithm – for the estimation of the malarial haplotype composition in one or more individuals and the corresponding haplotype population frequencies, focusing in particular on the case where individuals have been infected by more than one strain. This estimation must take place in the presence of pooled readings of the genetic composition of the parasites present.

The approach consists of the combination of a parameterized tree-based combinatorial search and a refinement phase incorporating the Expectation Maximization algorithm. The EM algorithm is particularly attractive as it is structured to be applied to situations involving both observed and unobserved information.

A test of an implementation of the algorithm on simulated data demonstrates its effectiveness in accurately estimating the haplotype compositions, both prior to and following the refinement. Its effectiveness established, the algorithm is then applied to a set of laboratory-produced malarial strain data.

In addition, the algorithm has also been made available to other researchers through a dedicated website allowing submissions and the downloading of results.

While the current research focused on the application of the method to malarial parasites, the method is general enough to be applied to cases of infection by other organisms.

Finally, the dissertation presents several suggestions for future work in enhancing the algorithm both computationally and statistically and extending its scope to related research topics.

Contents

List of Figures

List of Tables

List of Examples

List of Algorithms

Acknowledgments

There are many people I would like to thank for their help throughout my graduate studies, and in particular in my current research effort.

First, I would like to thank Alan Hubbard and Bryan Greenhouse for their many thoughtful comments and suggestions throughout my work on this topic. I had the pleasure to work with Bryan on a more rudimentary approach to the same problem for a class project several years back, and have the satisfaction of knowing that the LITSE algorithm will be used "in the real world".

Of all the Dissertation Committee members, I have known Deborah Nolan and Sandrine Dudoit the longest and thank them for serving on my current committee and their instruction in the courses I have taken with them.

Thanks as well to Rasmus Nielsen and Art Reingold for serving on my Qualification Committee. Rasmus in particular offered advice on choosing a topic as my plans evolved.

Finally, special thanks are due as well to John Strain from the Department of Mathematics for his tips on combinatorial optimization, Chris Paciorek and Ryan Lovett in the Department of Statistics for their help in debugging `R` and `C++`errors and quirks and other "features" that I managed to introduce in my code, Steve Evans from the Department of Statistics for his guidance regarding compositional data research and analysis, and Sharon Norris for being a superb departmental manager.

Chapter 1

Introduction

In this dissertation, we present a new method – the Learning with Iteration and Tree-based Search Estimation (LITSE) algorithm – for estimating the haplotype phasing of *pooled* samples of genetically distinct strains. By "pooled", we refer to the cases where an individual has been infected with multiple strains yet the genetic data collected for these strains is consolidated across strains in such a way that it is unclear what strains gave rise to these data.

While the focus is on the *Plasmodium* genus underlying malarial infections, the method is not specific to that organism. It is envisioned that the algorithm may be used in any haplotype phasing setting for any organism where the genetic data are collected in a pooled fashion.

Our aim has been to develop an accurate and computationally efficient statistical method for estimating the malarial haplotype composition in several, preferably many, individuals and inferring the relevant parasite population parameters. The primary parameter of interest is the population frequency of different strains in a region at a particular point in time.

This brief introduction motivates the development of such an algorithm, discusses in brief some of the biological considerations of malarial infections, presents the problem as a statistical one, and then introduces the main sections of the dissertation.

1.1 Motivation

The primary purpose of the `LITSE` algorithm is to permit the identification of the transmission networks of malarial strains in a region of interest. By "transmission network", we mean the mechanism by which malaraia transmits from one place to another. A key component of determining such networks is an estimate of the population of malarial strains in specific geographic areas at particular points in time [25].

As will be discussed in later chapters, existing methods have drawbacks that prevent them from handling the data inherent in the tracking of malarial strains.

1.2 Background

This section presents a brief background on malarial infection including the parasite involved, the vector, and some key statistics.

The organism behind malarial infection is the *Plasmodium* genus of parasitic protozoa [5]. In this genus, there are approximately 200 species including *P. falciparum*, *P. malariae*, *P. vivax*, *P. ovale*, and *P. knowlesi*, the primary species underlying infection in humans. The species *Plasmodium falciparum* is of particular interest, as it is the most deadly in humans. Within a species, there exist separate strains, characterized in some cases by different phenotypes such as varied susceptibility to antimalarial drugs, and genotypically by variations in DNA sequence including single nucleotide polymorphisms, insertion/deletions, and length polymorphisms such as microsatellites.

The *Plasmodium falciparum* species gives an indication of the genomes involved. The species has 14 chromosomes and approximately 23.3 million base pairs, comprising roughly 5400 coding genes [14, 29]. This is discussed in more detail in §2.3.

According to the Center for Disease Control (CDC), the vector for malarial disease is the *Anapholes* genus of mosquito [5]. Part of the prevention effort is thus to prevent mosquito bites through protective netting and other means.

Malarial infection is a major disease worldwide and a particular scourge of the developed world, as depicted in Figure 1.1. Again from the CDC, in 2012 alone an estimated

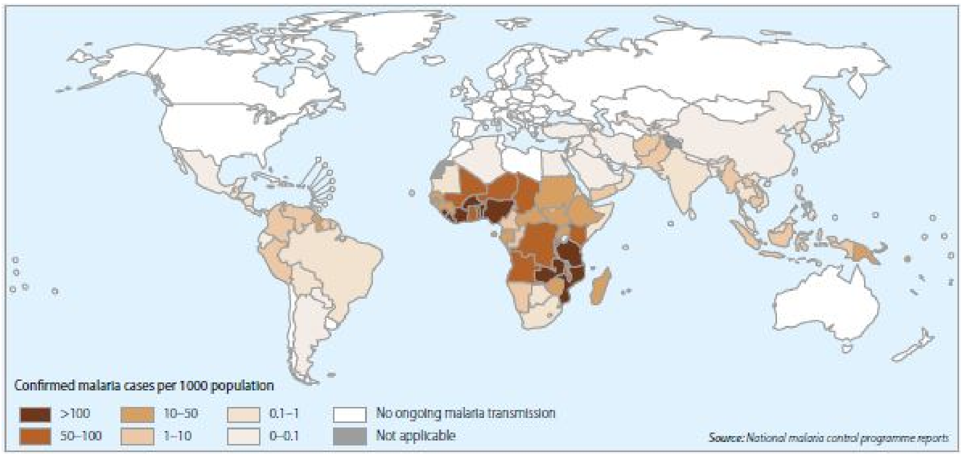

Confirmed malaria cases per 1000 population
>100
50–100
10–50
1–10
0.1–1
0–0.1
No ongoing malaria transmission
Not applicable
Source: National malaria control programme reports

Figure 1.1: Global Distribution of Malarial Cases, 2013. Source: World Health Organization [48]

207 million cases human infections occurred globally, resulting in 627,000 deaths – 85% of which were children under 5. Approximately 3.2 billion people live in endemic areas scattered over 106 countries, with close to 91% cases occurring in Africa [5].

Once it has entered the infected individual's blood stream, the *Plasmodium* parasites travel to the liver, multiply, and then infect the host's blood cells. Symptoms of malarial infection range from relatively mild fever, chills, headache, fatigue, and nausea and vomiting to death.

An individual infected may have been bitten by more than one mosquito. Whether or not this is the case, upon testing the individual will often present with a *Multiplicity of Infection* (MOI) greater than one, where MOI indicates the number of distinct strains within a single individual. This feature complicates the identification of the correct strains in a person since genotypic data will be collected in a pooled fashion.

Unsurprisingly given malaria's prevalence and severity, there is a long history of research in the area. Four Noble prizes have been awarded for work related to malaria and the disease is the subject of numerous websites and dedicated journals such as Malaria Journal, Malaria Research and Treatment, and the International Journal of Malaria Research and Reviews. Many relief organizations – both general (for example, Doctors without Borders and the Bill and Melinda Gates Foundation) and specific (for example, NETwork against Malaria and the Malaria Consortium) – exist to promote awareness and raise funds for research and treatment. The interest is such that it has spread beyond the academic and medical communities to the popular press as well; as just one recent example, consider Wang [45].

1.3 Statistical Context

As discussed in §2.5.4, the `LITSE` algorithm differs from the majority of related estimation methods in the following two aspects:

1. It explicitly focuses on cases where the MOI is greater than one within individuals, and
2. It combines a novel tree search approach that identifies candidate solutions with an Expectation Maximization algorithm-based framework to estimate the relevant statistical parameters.

1.4 Outline of Dissertation

The remainder of this dissertation is structured as follows.

Chapter §2 presents the Learning with Iteration and Tree-based Search Estimation (`LITSE`) algorithm including its underlying statistical framework and details of the algorithm. It is the primary chapter in this dissertation and serves as the basis for the remaining two.

Chapter §3 presents an assessment of the algorithm's performance – both in terms of runtime and accuracy– on simulated data. In these simulations, we deliberately alter various "environmental" factors the algorithm may encounter; by environmental, we mean features of the dataset such as number of loci or number of observations, We will see that several of these have a major impact on the algorithm's performance, in particular the noisiness of the observed proportions.

The effectiveness of `LITSE` having been established in the preceding chapters, Chapter §4 demonstrates the application of `LITSE` to real data supplied by Greenhouse Laboratory.

Finally, Chapter §5 presents details on how to apply `LITSE` to new datasets and interpret the results. A dedicated website has been created and documented to this end and an `R` package is under consideration.

Section §**??** outlines the notation standards used in this document.

Chapter 2

The LITSE Algorithm

In this chapter, we present in detail all aspects of the Learning with Iteration and Tree-based Search Estimation (`LITSE`) algorithm for haplotype estimation. This is the main chapter of the dissertation; chapters §4 and §5 discuss the details of an application and the implementation of this algorithm, respectively.

2.1 Chapter Organization

This chapter is organized as follows. First, to put the remaining sections in context, we briefly state the objective of the `LITSE` algorithm. Second, we introduce the relevant genetics of the *Plasmodium falciparum* organism and, third, we then present the data that the `LITSE` algorithm will use and produce. Fourth, using the concepts just introduced, we revisit the objective in more detail and discuss some of the previous work on related problems. Fifth, with the objective in mind, we develop a statistical framework to support the `LITSE` approach. Sixth, we then describe in detail the steps of the `LITSE` algorithm and how the algorithm addresses the objective. Seventh and last, we perform a set of simulations to assess the performance of the `LITSE` algorithm and its sensitivity to important factors that will be encountered in its use.

2.2 Brief Statement of Objective

Our objective can be stated as follows.

> **Objective**: Given observed allele proportions across a number of loci, with one such set of proportions per individual, for one or more individuals,
> 1. *estimate* the underlying haplotype set and the proportion of each haplotype for each individual, and
> 2. *infer* the haplotype population level parameters

The entities used in this objective statement – alleles, loci, and haplotypes – are described in detail later in this chapter.

Any method satisfying this objective will need to address the following two considerations:

1. *accuracy* – compute estimates that are as accurate as possible, and
2. *speed* – perform this estimation as quickly as possible.

Since part of this estimation will be seen to be a combinatorial search problem, there is a fundamental conflict between these two goals: computing accurate estimates will involve considering as many possibilities as possible; but the more exhaustive a search, the longer its duration.

2.3 Relevant Genomics Concepts

This section briefly discusses the genomics of the *Plasmodium*genus and defines the key genomic entities used throughout this dissertation.

2.3.1 Highlights of the *Plasmodium* Genus Genome

The genome of *Plasmodium falciparum* consists of roughly 23 megabases spread over 14 chromosomes [14]. Other species in the genus *Plasmodium* have have a similar genome, in particular the number of chromosomes is the same, though the GC content varies significantly among species [29]. Each species is believed to possess approximately 5000 genes.

The genome is characterized by a number of microsatellites or short tandem repeats, locations scattered across the genome where short sequences of 2-5 base pairs are repeated a variable number of times.

Within a *Plasmodium* species, we can distinguish between different strains, as discussed in §2.3.2.3 below. Each strain is characterized by a particular configuration of microsatellites.

The *Plasmodium falciparum* species is haploid throughout the human stage of its life cycle [46]; that is, it has a single copy of each chromosome[1] This naturally simplifies the identification of malarial haplotypes in a human isolate. However, as will be seen later, this simplification is lost when a human host has more than one malarial strain present.

2.3.2 Genomic Concepts

There are three key genomic concepts underlying the objective of this dissertation. They are most logically presented in the following order – locus, allele, and haplotype – as each concept builds upon its predecessors.

2.3.2.1 Locus

A *locus* is simply a specific location on a genome. Given our focus on microsatellites, the loci of interest are simply the locations on the genome where the chosen microsatellites exist. As such, within the scope of this dissertation, the terms microsatellite and locus are synonymous; for simplicity, we settle on the term “locus”.

Figure 2.1 depicts the structure of a locus, with its varying number of repeats.

The number and location of loci of interest may vary from one study to the next. Furthermore, loci may be given particular names carrying meaning; see §3.A for some examples. To generalize the approach of the `LITSE` algorithm, this dissertation simply labels the loci sequentially; that is, “locus 1”, “locus 2”, and so on.

Figure 2.1 implicitly introduces the concept of an allele, a concept which is discussed next.

[1]This can be contrasted to the situation for humans, for example, where each organism has two copies of each chromosome.

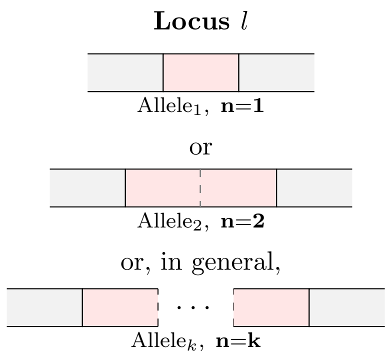


Figure 2.1: Diagram of a Locus. A locus is defined by the occurrence of one or more repeats. Each of the three vertically stacked bars represents a possible configuration for a given locus l. The distinguishing characteristic is the number of times a repeat occurs at the locus of interest.

2.3.2.2 Allele

In the context of this study, an *allele* exists for a particular locus and is simply the number of repeats n at that locus for the organism in question. Figure 2.1 above thus depicts three distinct alleles for the same locus: an allele where the number of repeats equals one, an allele with two repeats, and the general case where the number of repeats equals some integer k. The total number of repeats witnessed will vary by locus.

We make the distinction at each locus between *existing* alleles and *present* alleles. The first term refers to all repeat numbers known to exist at the locus in question among all organisms in the genus, while the second refers to the alleles actually observed in an individual sample. For example, for a particular locus it may be known that there are ten existing alleles, $n = 33, \ldots, 42$, in the parasite population, yet in a given infected individual only two are detected, say $n = 35$ and $n = 38$, as present.

We define as *locus length* the total number of existing alleles at a locus l and denote this as $\ell(l)$. Thus the locus length equals 10 in the example in the previous paragraph.

In the context of this study, even though it is defined in terms of a numerical value, an allele is viewed as a categorical variable. This implies that there is no concept of similarity between alleles. Consider three possible alleles for a locus l: one with 20 repeats, one with 40 repeats, and one with 41 repeats. In this study, all are simply distinct values; there is no sense in which the second and third alleles are “closer” to one another than, say, the first and the third.

2.3.2.3 Haplotype

A *haplotype* is defined as, given a chosen set of loci, the set of alleles, one per locus, representing a particular strain.[2] An allele can be denoted $A_{\text{locus \#,allele \# in locus}}$.

Example 2.1 (Haplotypes). Assume 10 loci. Then the following are two possible haplotypes.

$$h_1 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline A_{1,33} & A_{2,12} & A_{3,10} & A_{4,9} & A_{5,14} & A_{6,35} & A_{7,67} & A_{8,43} & A_{9,23} & A_{10,4} \\ \hline\end{array}$$

$$h_2 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline A_{1,33} & A_{2,6} & A_{3,5} & A_{4,56} & A_{5,45} & A_{6,3} & A_{7,34} & A_{8,74} & A_{9,86} & A_{10,7} \\ \hline\end{array}$$

Each haplotype is distinguished by a unique set of alleles.

□

Thus a haplotype can be viewed as a genetic "fingerprint" of a parasitic strain.

Let $\ell(l)$ denote the locus length[3] of locus l with N_l loci in total. Then the number of possible haplotypes is,

$$N_{\text{haps}} := \prod_{l=1}^{N_l} \ell(l). \tag{2.1}$$

Example 2.2 (Number of Possible Haplotypes). Assume that there are three loci of interest and thus $Nl = 3$. Assume further that the loci have 4, 5, and 7 possible alleles, respectively. Then, regardless of what may be observed in a particular set of individuals, there are

$$N_{\text{haps}} = \prod_{l=1}^{3} \ell(l) = 4 \cdot 5 \cdot 7 = 140$$

different haplotypes possible.[4]

□

Finally, the following assumptions are made regarding haplotypes and their population:

- There is a population of parasite haplotypes in each geographic region of interest (e.g., region in Tanzania).

[2]Frequently used synonym for haplotype in this context are "strain" and "clone".
[3]Or simply, "length".
[4]The vast majority of which we would not see in a single observation.

- This haplotype population is present in parasite-carrying mosquitoes.
- Each *possible* haplotype has a population frequency, usually zero (meaning it does not exist).
- Individual mosquitoes may carry multiple haplotypes.
- The probability of infection with a particular haplotype is a increasing function of that haplotype's population frequency. An infected individual will more likely be infected with a common strain than a rare one.

2.3.2.4 Multiplicity of Infection

The *multiplicity of infection*, or MOI, is the number of unique haplotypes in an infected individual. This is not to be confused with the total number of *Plasmodium falciparum*organisms in an infected individual.

Example 2.3 (MOI). The concept of MOI is straightforward. Nevertheless, consider the following example. Assume an individual has the following set of unique haplotypes:

Loc_1	Loc_2	Loc_3	Loc_4	Loc_5	Loc_6	Loc_7
$A_{1,99}$	$A_{2,120}$	$A_{3,123}$	$A_{4,126}$	$A_{5,132}$	$A_{6,150}$	$A_{1,147}$
$A_{1,99}$	$A_{2,120}$	$A_{3,123}$	$A_{4,126}$	$A_{5,132}$	$A_{6,150}$	$A_{1,150}$

While the two haplotypes are nearly identical, they differ at the last locus. Thus the MOI in this case equals two.

$\square$

It follows that the MOI is simply a positive integer with no theoretical upper bound. However, in reality the MOI in infected individuals has been found to typically be in the range of 1 to 5 [25].

2.4 Data

This section describes the two primary data objects related to the research question and then presents the mechanism by which data are collected. Briefly, the `LITSE` algorithm uses as input a set of observed allele proportions per individual and produces as output a set of haplotypes proposed as the origin of the observed allele proportions. Each input observation may or not be complete, as defined below.

The role of these entities is discussed further in §2.5.

2.4.1 Key Data Components

This section describes the data objects fundamental to the research problem.

2.4.1.1 Chosen Loci

Before any data are collected or analyzed, a choice will have been made regarding which loci on the malarial genome will be used as markers to identify strains. The criteria used to choose loci is outside the scope of the current research question and this choice is treated as given. As will be seen later, the performance of the `LITSE` algorithm is sensitive to this choice, particularly in terms of run time; however, the algorithm is flexible enough to handle a number of loci significantly greater than similar methods.

An important characteristic of a chosen locus is its "noisiness", formally measured by the standard deviation in the recorded allele proportions at that locus around the true allele proportions. This noisiness will vary from locus to locus, as described in §2.4.2, and is explicitly modeled by the `LITSE` algorithm.

2.4.1.2 Allele Proportions

The primary input is the set of allele proportions, one per locus, for all malarial strains found in a single infected individual. Each individual will have one or more alleles represented at each locus, which are pooled from all strains infecting that individual (however, as will be seen below, this information may be missing for one or more loci).

We will later make a distinction between proportions that are observed versus those that are actual. For now, we focus on properties common to both types.

For a given individual, there are two equivalent formats for presenting this allele information:

1. For each locus, simply list the identifier of each allele along with its proportion.

2. For each locus, present the locus's allele proportions in a vector, with each entry non-negative and the sum of the entries equaling one. The i^{th} entry in this vector represents the proportion of the i^{th} *possible* allele at that locus. By "possible", we mean known to exist in the population of all strains. The number of possible alleles will vary by locus.

The following example demonstrates these formats.

Example 2.4 (Allele Proportions). Consider the scenario with three loci with possible alleles $A_{1,30}, A_{1,33}, A_{1,36}, A_{1,48}, A_{1,51}, A_{1,57}$, $A_{2,30}, A_{2,33}, A_{2,36}$, and $A_{3,60}, A_{3,63}, A_{3,66}$. Assume an individual has proportions of 0.3 and 0.7 for alleles $A_{1,33}, A_{1,51}$, 0.2 and 0.8 for alleles $A_{2,30}, A_{2,36}$, and 1.0 for $A_{3,66}$, respectively.

Then this information presented using the first format is,

Locus 1	Locus 2	Locus 3
$A_{l,33} : 0.3, A_{l,51} : 0.7$	$A_{2,30} : 0.2, A_{2,36} : 0.8$	$A_{3,66} : 1.0$

And the same information presented in the second format is,

$$\left\{\begin{pmatrix} p(A_{l,30}) = 0 \\ p(A_{l,33}) = 0.3 \\ p(A_{l,36}) = 0 \\ p(A_{l,48}) = 0 \\ p(A_{l,51}) = 0.7 \\ p(A_{l,57}) = 0 \end{pmatrix}, \begin{pmatrix} p(A_{l,30}) = 0.2 \\ p(A_{l,33}) = 0 \\ p(A_{l,36}) = 0.8 \end{pmatrix}, \begin{pmatrix} p(A_{l,30}) = 0 \\ p(A_{l,36}) = 1.0 \end{pmatrix}\right\}$$

$\square$

In general, under the second the set of allele proportions across all loci takes the following form,

$$\left\{ \begin{pmatrix} p(A_{1,a_{11}}) \\ p(A_{1,a_{12}}) \\ \vdots \\ p(A_{1,a_{1,\ell(1)}}) \end{pmatrix}, \begin{pmatrix} p(A_{2,a_{21}}) \\ p(A_{2,a_{22}}) \\ \vdots \\ p(A_{2,a_{1,\ell(2)}}) \end{pmatrix}, \ldots, \begin{pmatrix} p(A_{N_l,a_{N_l,1}}) \\ p(A_{N_l,a_{N_l 2}}) \\ \vdots \\ p(A_{N_l,a_{N_l,\ell(N_l)}}) \end{pmatrix} \right\}$$

where $a_{i,j}$ is the identifier (e.g., 33) for the j^{th} possible allele for the i^{th} locus.

While the first format is more compact, the second format is more suited for many of the computations in the underlying statistical model to be presented.

Next, we make the distinction below between "true" and "observed" allele proportions. Each class takes the same vector form presented above.

2.4.1.2.1 True Allele Proportions

These are the allele proportions that would be observed in the absence of any measurement error, as discussed in §2.4.2. The true allele proportions for an individual are implied by two factors:

1. the haplotypes the individual is infected with and
2. the proportion of each haplotype.

This relationship is discussed in greater detail in §2.6.1.4.

Example 2.5 (True Allele Proportions). Consider again the haplotypes h_1, h_2 in Example 2.1. Assume that an individual is infected with these two haplotypes, and no others, with proportions 0.7 and 0.3, respectively. Note that there are 10 loci being tracked, and consider the first two only.

For locus 1, the haplotypes share the same allele – $A_{1,33}$. Thus for this locus, only this allele will have a non-zero proportion, in this case equal to 1. If there are three possible alleles at locus 1, say $A_{1,30}, A_{1,33}, A_{1,36}$, then the true allele proportion vector would be the following,

$$\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}^T .$$

For locus 2, the haplotypes do not share the same allele. Thus for this locus, only the two alleles represented – $A_{2,12}$ and $A_{2,6}$ – will have a non-zero proportion equal to the proportion of the respective haplotype. If there are four possible alleles at locus 2, say $A_{2,3}, A_{2,6}, A_{2,9}, A_{2,12}$, then the true allele proportion vector would be the following,

$$\begin{pmatrix} 0 & 0.3 & 0 & 0.7 \end{pmatrix}^T .$$

$\square$

The computation of the true allele proportions becomes more complicated as the number of haplotypes grows, with the possibility that for any particular locus some but not all haplotypes will share an allele.

2.4.1.2.2 Observed Allele Proportions

The true allele proportions discussed in §2.4.1.2.1 are those that would be observed in the absence of any measurement error from the process described below in §2.4.2. However, as explained in that section, such errors can arise for several possible reasons in the collection process. Possible reasons include (in decreasing order of importance):

1. artifacts, where the locus allele reads of individual parasite organisms are misread (for example, an $A_{1,33}$ is read as a $A_{1,36}$) or their relative proportions are altered during amplification,
2. an uneven distribution of the haplotypes' location in the body, with some demonstrating a greater tendency to be found in the blood versus tissue, or simply an uneven distribution of the haplotypes within the blood itself, with some relatively concentrated in the location from which the blood sample was taken.

The second reason becomes an issue only to the extent to which some haplotypes are favored over others in the collection of a sample.

Observed allele proportion vectors take the same form as true allele proportions vectors and are subject to the same constraints. They are discussed in detail in §2.6.1.6.

At the extreme, the process whereby an individual's locus allele proportions are measured may fail to produce any readings at all for a particular locus. We denote such cases simply as 'missing loci".

Example 2.6 (Missing Loci). Consider the case where there are three loci under review, five observations in total, and the following observed allele proportions, In this

Obs	Locus 1	Locus 2	Locus 3
1	$A_{1,33} : 0.80, A_{1,36} : 0.20$	$A_{2,24} : 0.11, A_{2,27} : 0.89$	$A_{3,66} : 0.45, A_{3,102} : 0.55$
2		$A_{2,24} : 0.56, A_{2,30} : 0.44$	$A_{3,60} : 0.40, A_{3,90} : 0.60$
3		$A_{2,24} : 0.30, A_{2,27} : 0.70$	$A_{3,69} : 0.75, A_{3,102} : 0.25$
4	$A_{1,33} : 0.48, A_{1,36} : 0.52$	$A_{2,21} : 0.61, A_{2,27} : 0.39$	
5	$A_{1,39} : 0.26, A_{1,42} : 0.74$	$A_{2,24} : 0.37, A_{2,27} : 0.63$	$A_{3,36} : 0.54, A_{3,102} : 0.46$

example, proportions are clearly missing at locus 1 for observations 2 and 3 and at locus 3 for observation 4.

□

While limited, the example above illustrates a common occurrence: some loci (in this case, locus 1) are more prone to missing data than others. What is critical is that each locus have at least one observation, preferably many, with a reading for that locus.

A key feature of the `LITSE` algorithm, discussed in §2.7.3, will be to handle such incomplete datasets, essentially borrowing information across observations to impute the missing data for such situations.

2.4.1.3 Haplotype Set Solutions

A haplotype set solution (or just "solution") Sol_n for an observation n is the set of haplotypes and their proportions that gave rise to the allele proportions observed. Each solution takes the following form,

$$Sol_n = \begin{array}{c|ccccc|r}
\text{ID} & \text{Loc}_1 & \text{Loc}_2 & \text{Loc}_3 & \dots & \text{Loc}_{N_l} & \text{Prop} \\
\hline
h_1 & A_{1,\cdot} & A_{2,\cdot} & A_{3,\cdot} & \dots & A_{N_l,\cdot} & p_1 \\
h_2 & A_{1,\cdot} & A_{2,\cdot} & A_{3,\cdot} & \dots & A_{N_l,\cdot} & p_2 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
h_z & A_{1,\cdot} & A_{2,\cdot} & A_{3,\cdot} & \dots & A_{N_l,\cdot} & p_z
\end{array}$$

where N_l indicates the number of loci being considered, h_i indicates the i^{th} haplotype, $A_{j,\cdot}$ indicates a possible allele for the j^{th} locus, p_i indicates the proposed proportion of h_i, and z indicates the number of haplotypes proposed in the solution (i.e., the proposed MOI).

We make a distinction between the *true solution*, denoted Sol_n as above, that gave rise to the observed allele proportions for observation n and an estimate, denoted $\widehat{Sol}_{n,j}$, of this true solution where j denotes the sequence or identification number over all estimates for observation n. As we will see later on, we will typically have many estimates for the same observation n.

Among estimates, we make a distinction between those that are *feasible* and those that are infeasible. For a proposed solution $\widehat{Sol}_{n,j}$ to be feasible, it must meet the following conditions:

1. Each haplotype, defined by its alleles, must be unique in the set.
2. For each locus, each allele observed (necessarily with a non-zero proportion) must have a representation in at least one haplotype. For example, if allele $A_{2,9}$ has been observed for an individual, at least one of the proposed haplotypes in $\widehat{Sol}_{n,j}$ must contain this allele in the appropriate column.
3. Similarly, any possible locus allele that has *not* been observed must not be present in the solution.
4. The proportion of each haplotype must be strictly greater than zero. A negative proportion makes no sense and a zero proportion is equivalent to saying that the haplotype in fact does not exist in the solution.
5. The sum of the proportions must equal 1.

2.4.2 Data Collection and Error

The mechanism for collecting the observed allele proportions is described below.

A necessary step before any data collection is to identify the common set of loci that will be examined across all individuals. Given the goal of using loci information to "fingerprint" strains, the loci should be selected to ensure mutual independence across loci.

The initial object of interest is the infected individual. For each individual, the following steps are performed:

- Draw a blood sample, in which live *Plasmodium* organisms are assumed to exist. Note that not all *Plasmodium* organisms will be found in the blood; as described in §1.2, many organisms will infect tissues and thus be absent from the bloodstream.
- The *Plasmodium* organisms in the sample blood are isolated, and their DNA is amplified through polymerase chain reaction and read through a fluorescence assay.
- The assay produces a pooled set of allele intensities per locus, across all organisms detected in the individual's blood sample. Given the microsatellite nature of the

data, the alleles are identified in terms of repeat length, as discussed in §2.3.2.2.

- The relative intensity of the allele readings is used to compute their proportions.

Inherent in this process are read errors, whereby an allele is incorrectly classified and a mismeasurement in allele proportions results. Such errors produce "artifacts", i.e., a misclassification of a read that results in an erroneous intensity level measurement. In some cases, such artifacts create "phantom" alleles; i.e., they give a non-zero intensity to alleles that in fact do not exist. A second complication is the known bias in intensity readings by allele; specifically, it is known that there is an downward bias in the recording of allele intensities as the allele length increases.

This process is depicted in Figure 2.2.

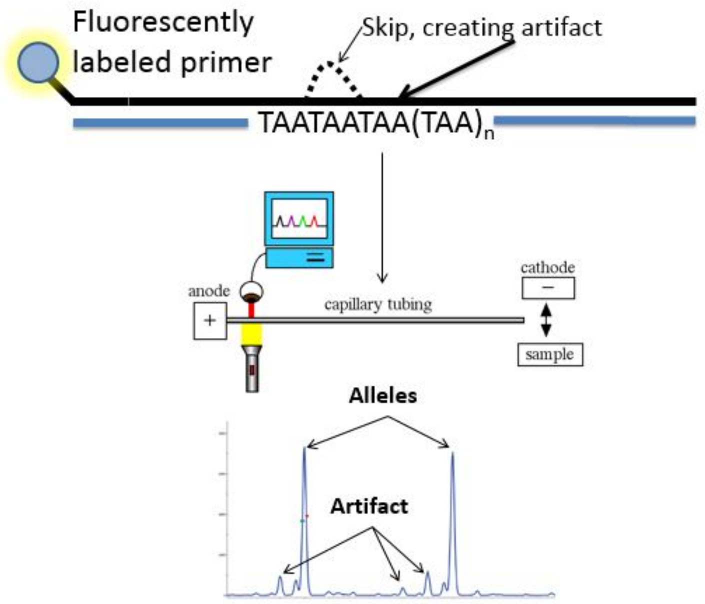


Figure 2.2: Genotyping. The genotyping process computes the relative intensity for each possible allele at a locus. Non-zero intensities assigned to alleles that are not present are termed "artifacts". Source: Greenhouse [15].

The control of these sources of error is the subject of separate research [15]. The key implication of this data collection process is that measurement error will taint the data that are the basis for the `LITSE` algorithm's haplotype inference. Consistent with intuition, it will be shown in §3 that the accuracy of the algorithm suffers as the measurement error increases.

2.5 Research Objective

In this section, we revisit the objective stated in §2.2 and state it more formally using the concepts introduced in the preceding sections. We then illustrate the problem and frame it as a combinatorial search problem. Finally, we review some of the work performed on related problems and explain how the present problem differs.

2.5.1 Restatement of Objective

In light of sections §2.3 and §2.4, we restate the objective in §2.2 as follows,

Objective: For each individual/observation n in a sample of size N_{samp}, using that individual's observed allele proportions for each locus $l \in \{1, \ldots, N_l\}$ but in the absence of the true allele proportions,

1. *identify* a collection of feasible haplotype set solutions $\{\widehat{Sol}_{n,j}\}$, and
2. *compute* the probability of each of these solutions, and
3. *infer* the haplotype population level parameters governing those probabilities.

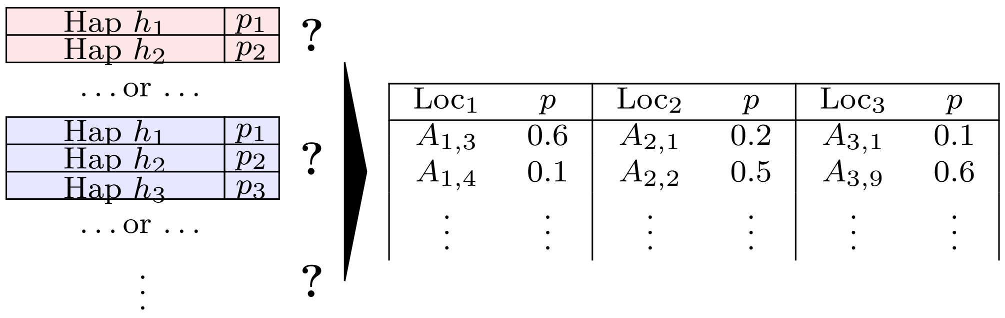
There are two primary benefits of realizing this objective, with the second being of particular interest:

- Identifying infectious strains at the individual patient level.
- Identifying the predominance of infectious strains across individuals in a particular region. Here, the focus is on the population of malarial haplotypes. Implementing such a method in multiple locations or at multiple points in time provides valuable insight into the population genetics of the *Plasmodium* genus.

A formal statistical framework for this objective is presented in section §2.6 and the details of the approach are discussed in §2.7.

2.5.2 Illustration of the Problem

The objective is illustrated in Figure 2.3. As depicted, the goal is to identify candidate, feasible solutions that might have generated the observed allele proportions. As seen in



Hap h_1 p_1
Hap h_2 p_2
... or ...
Hap h_1 p_1
Hap h_2 p_2
Hap h_3 p_3
... or ...
?
?
?
Loc$_1$ p Loc$_2$ p Loc$_3$ p
$A_{1,3}$ 0.6 $A_{2,1}$ 0.2 $A_{3,1}$ 0.1
$A_{1,4}$ 0.1 $A_{2,2}$ 0.5 $A_{3,9}$ 0.6

Figure 2.3: Illustration of Research Question. The left-hand side depicts the space of possible solutions, each of which consists of a set of haplotypes along with proportions for those haplotypes. The correct set is unknown. This set will generate the actual observed data, depicted on the right-hand side. These data come in the form of observed allele proportions per locus, where $A_{i,j}$ indicates the j^{th} possible allele at the i^{th} locus.

the series of examples below, this problem presents a wide range of difficulty depending on the characteristics of the problem.

A fundamental objective of the `LITSE` algorithm – inferring the underlying haplotypes that gave rise to the observed allele proportion data – is a familiar problem in signal processing, where it is known as the "demixing" problem. See Webster [47] for details from an engineering perspective.

Example 2.7 (A Trivial Case). Consider the case where $N_l = 4$ and we are presented with the following observed non-zero allele proportions[5],

$$\left\{ \begin{pmatrix} p(A_{1,a_{1,1}}) = 0.2 \\ p(A_{1,a_{1,2}}) = 0.3 \\ p(A_{1,a_{1,6}}) = 0.5 \end{pmatrix}, \begin{pmatrix} p(A_{2,a_{2,2}}) = 0.3 \\ p(A_{2,a_{2,4}}) = 0.5 \\ p(A_{2,a_{2,7}}) = 0.2 \end{pmatrix}, \begin{pmatrix} p(A_{3,a_{3,1}}) = 0.5 \\ p(A_{3,a_{3,4}}) = 0.3 \\ p(A_{3,a_{3,5}}) = 0.2 \end{pmatrix}, \begin{pmatrix} p(A_{4,a_{4,4}}) = 0.3 \\ p(A_{4,a_{4,5}}) = 0.5 \\ p(A_{4,a_{4,9}}) = 0.2 \end{pmatrix} \right\}$$

In this case, an obvious feasible solution, based on the proportions in each vector, is as

[5]In this example, there is no need to list all possible loci for each locus. We can focus on only those that have a non-zero proportion to determine what haplotypes may have given rise to the observed allele proportions.

follows,

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Prop
h_1	$A_{1,a_{1,1}}$	$A_{2,a_{2,7}}$	$A_{3,a_{3,5}}$	$A_{4,a_{4,9}}$	0.2
h_2	$A_{1,a_{1,2}}$	$A_{2,a_{2,2}}$	$A_{3,a_{3,4}}$	$A_{4,a_{4,4}}$	0.3
h_3	$A_{1,a_{1,6}}$	$A_{2,a_{2,4}}$	$A_{3,a_{3,1}}$	$A_{4,a_{4,5}}$	0.5

Note that this proposed haplotype solution will explain perfectly the observed allele proportions. There is no other solution with the same MOI (3) with this property.

This last statement does not imply that there are no other solutions using higher MOIs that fit the data as well as the present solution. For example, consider the following solution for MOI=4,

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Prop
h_1	$A_{1,a_{1,1}}$	$A_{2,a_{2,7}}$	$A_{3,a_{3,5}}$	$A_{4,a_{4,5}}$	0.2
h_2	$A_{1,a_{1,2}}$	$A_{2,a_{2,4}}$	$A_{3,a_{3,1}}$	$A_{4,a_{4,5}}$	0.3
h_3	$A_{1,a_{1,6}}$	$A_{2,a_{2,2}}$	$A_{3,a_{3,4}}$	$A_{4,a_{4,4}}$	0.3
h_4	$A_{1,a_{1,6}}$	$A_{2,a_{2,4}}$	$A_{3,a_{3,1}}$	$A_{4,a_{4,9}}$	0.2

This solution also has a perfect fit to the observed allele proportions.

□

The identification of the first solution – and to a slightly lesser extent, the second – proposed in Example 2.7 is easy because of the following properties of the set of observed allele proportions:

1. There are few – in this case, four – loci.
2. There are few – in this case, always three – non-zero alleles per locus.
3. Each locus has the same three distinct proportions represented – 0.2, 0.3, and 0.5.

Note that these conditions will rarely be realized in practice since they imply zero measurement error.

The statement concerning using higher MOIs is a general one. With a high enough MOI, we can always construct a perfect solution. At the extreme, we can always list a set of haplotypes that are the Cartesian product of the alleles observed. In Example 2.7, this would involve an MOI of $3^4 = 81$. Yet, as was shown in that example, there is often no need to attempt such a high MOI. For a given fit, we prefer sparse solutions (those with a lower MOI); the `LITSE` algorithm will address this. Section §2.5.3 covers this in more detail.

Consider now a slightly more difficult case building from the first.

Example 2.8 (Two Simple Cases). First, consider a slight twist to Example 2.7; specifically, consider the following observed allele proportions,

$$\left\{ \begin{pmatrix} p(A_{1,a_{1,1}}) = 0.2 \\ p(A_{1,a_{1,2}}) = 0.3 \\ p(A_{1,a_{1,6}}) = 0.5 \end{pmatrix}, \begin{pmatrix} p(A_{2,a_{2,2}}) = 0.8 \\ p(A_{2,a_{2,7}}) = 0.2 \end{pmatrix}, \begin{pmatrix} p(A_{3,a_{3,1}}) = 0.5 \\ p(A_{3,a_{3,4}}) = 0.3 \\ p(A_{3,a_{3,5}}) = 0.2 \end{pmatrix}, \begin{pmatrix} p(A_{4,a_{4,4}}) = 0.3 \\ p(A_{4,a_{4,5}}) = 0.5 \\ p(A_{4,a_{4,9}}) = 0.2 \end{pmatrix} \right\}$$

The only difference is that the second locus now has only two non-zero alleles while the rest still have three. Thus it will be impossible to perform the one-to-one matching as done in the previous example. Nevertheless, it is possible to neatly split the proportion (0.8) for $A_{2,a_{2,2}}$ into two parts (0.3,0.5) and propose the following solution,

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Prop
h_1	$A_{1,a_{1,1}}$	$A_{2,a_{2,7}}$	$A_{3,a_{3,5}}$	$A_{4,a_{4,9}}$	0.2
h_2	$A_{1,a_{1,2}}$	$A_{2,a_{2,2}}$	$A_{3,a_{3,4}}$	$A_{4,a_{4,4}}$	0.3
h_3	$A_{1,a_{1,6}}$	$A_{2,a_{2,2}}$	$A_{3,a_{3,1}}$	$A_{4,a_{4,5}}$	0.5

Thus the allele $A_{2,a_{2,2}}$ appears in two different haplotypes, h_2 and h_3. Once again, the solution explains the allele proportions perfectly. There is no other solution for the same MOI that has this quality.

Second, consider a different but still minor modification to Example 2.7 with the proportions,

$$\left\{ \begin{pmatrix} p(A_{1,a_{1,1}}) = 0.2 \\ p(A_{1,a_{1,2}}) = 0.3 \\ p(A_{1,a_{1,6}}) = 0.5 \end{pmatrix}, \begin{pmatrix} p(A_{2,a_{2,2}}) = 0.5 \\ p(A_{2,a_{2,4}}) = 0.5 \end{pmatrix}, \begin{pmatrix} p(A_{3,a_{3,1}}) = 0.5 \\ p(A_{3,a_{3,4}}) = 0.3 \\ p(A_{3,a_{3,5}}) = 0.2 \end{pmatrix}, \begin{pmatrix} p(A_{4,a_{4,4}}) = 0.3 \\ p(A_{4,a_{4,5}}) = 0.5 \\ p(A_{4,a_{4,9}}) = 0.2 \end{pmatrix} \right\}$$

Once again, the second locus now has only two non-zero alleles while the rest still have three. This time we have a choice regarding which of the two allele proportions we split since they are both equal. In either case, we split the chosen allele's proportion into two parts (0.2,0.3) and leave the other at 0.5. This gives rise to two solutions,

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Prop
h_1	$A_{1,a_{1,1}}$	$A_{2,a_{2,4}}$	$A_{3,a_{3,5}}$	$A_{4,a_{4,9}}$	0.2
h_2	$A_{1,a_{1,2}}$	$A_{2,a_{2,4}}$	$A_{3,a_{3,4}}$	$A_{4,a_{4,4}}$	0.3
h_3	$A_{1,a_{1,6}}$	$A_{2,a_{2,2}}$	$A_{3,a_{3,1}}$	$A_{4,a_{4,5}}$	0.5

and

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Prop
h_1	$A_{1,a_{1,1}}$	$A_{2,a_{2,2}}$	$A_{3,a_{3,5}}$	$A_{4,a_{4,9}}$	0.2
h_2	$A_{1,a_{1,2}}$	$A_{2,a_{2,2}}$	$A_{3,a_{3,4}}$	$A_{4,a_{4,4}}$	0.3
h_3	$A_{1,a_{1,6}}$	$A_{2,a_{2,4}}$	$A_{3,a_{3,1}}$	$A_{4,a_{4,5}}$	0.5

the first of which correspond to splitting the allele $A_{2,a_{2,4}}$ and the second of which corresponds to splitting the allele $A_{2,a_{2,2}}$. Both solutions fit the data perfectly.

$\square$

Two additional lessons can be drawn from the results discussed in Example 2.8 that will be relevant to many situations,

1. Despite the presence of a locus, in this case the second, whose number of non-zero alleles is only two, given that there are other loci with three non-zero alleles we would never consider solutions with an MOI less than 3. Indeed, the rule is that we will consider MOIs only equal to or greater than the maximum number of non-zero alleles across all loci.
2. There will be cases where several solutions tie for a particular fit with the data.

Example 2.9 (A Hard Case). Finally, consider the following problem,

$$\left\{ \begin{pmatrix} p(A_{1,a_{1,1}}) = 0.21 \\ p(A_{1,a_{1,2}}) = 0.39 \\ p(A_{1,a_{1,5}}) = 0.40 \end{pmatrix}, \begin{pmatrix} p(A_{2,a_{2,7}}) = 0.56 \\ p(A_{2,a_{2,4}}) = 0.19 \\ p(A_{2,a_{2,1}}) = 0.25 \end{pmatrix}, \begin{pmatrix} p(A_{3,a_{3,5}}) = 0.27 \\ p(A_{3,a_{3,4}}) = 0.14 \\ p(A_{3,a_{3,1}}) = 0.59 \end{pmatrix}, \begin{pmatrix} p(A_{4,a_{4,9}}) = 0.21 \\ p(A_{4,a_{4,4}}) = 0.54 \\ p(A_{4,a_{4,5}}) = 0.25 \end{pmatrix} \right\}$$

In this case, each locus has three non-zero proportions. Yet it is clear from inspection that there is no solution of MOI=3 that will fit the data. In fact, the above proportions were generated by the following haplotype solution of MOI=5,

$$
Sol_n = \begin{array}{c|cccc|c}
\text{ID} & \text{Loc}_1 & \text{Loc}_2 & \text{Loc}_3 & \text{Loc}_4 & \text{Prop} \\
\hline
h_1 & A_{1,a_{1,1}} & A_{2,a_{2,7}} & A_{3,a_{3,5}} & A_{4,a_{4,9}} & 0.02 \\
h_2 & A_{1,a_{1,2}} & A_{2,a_{2,7}} & A_{3,a_{3,4}} & A_{4,a_{4,4}} & 0.14 \\
h_3 & A_{1,a_{1,1}} & A_{2,a_{2,4}} & A_{3,a_{3,1}} & A_{4,a_{4,9}} & 0.19 \\
h_4 & A_{1,a_{1,2}} & A_{2,a_{2,1}} & A_{3,a_{3,5}} & A_{4,a_{4,5}} & 0.25 \\
h_4 & A_{1,a_{1,5}} & A_{2,a_{2,7}} & A_{3,a_{3,1}} & A_{4,a_{4,4}} & 0.40
\end{array}
$$

$\square$

To put the preceding examples in perspective, note that in all cases the number of loci equaled 4. In the real-world application of the `LITSE` algorithm, it is envisioned to run on cases with circa 30 loci.

2.5.3 A Combinatorial Search

Realizing the first part of the objective – identifying candidate solutions for an individual – necessarily involves a search over many potential solutions. This involves both estimating the MOI and, given that MOI, the precise haplotypes and their proportions. In practice, we will identify a number of possible solutions per individual and assign a probability to each.

We envision scenarios where the number of loci under investigation reaches the region of 30. As we will see later, the number of solutions to the phasing problem increases exponentially as the number of loci increases.

This problem can be viewed as a combinatorial search. In all cases, we are looking for combinations of haplotypes whose implied allele proportions are close to what is actually observed. If there are N_{haps} possible haplotypes and we assume $MOI = z$, then there are $\binom{N_{\text{haps}}}{z}$ different possible solutions. Restricting solutions to those that are feasible helps but only to an extent. This is demonstrated in Table 2.1.

N_l	$\ell(l)$	N_{haps}	z	Number of Poss Solutions
5	5	3125	3	5,081,381,250
10	5	9,765,625	3	1.55e+20
20	5	9.54e+13	3	1.45e+41
30	5	9.31e+20	3	1.35e+62
30	5	9.31e+20	4	3.13e+82

Table 2.1: Sample Potential Number of Solutions. The quantity N_l denotes the number of loci, $\ell(l)$ denotes the number of alleles per each locus, N_{haps} denotes the implied number of possible haplotypes given $N_l, \ell(l)$ (as demonstrated in equation 2.1), and z denotes the assumed MOI. While $\ell(l)$ normally denotes the number of existing alleles, here it can be interpreted as the number of non-zero alleles for a particular observation. The number of possible solutions grows exponentially with N_{haps} and z.

The key point from Table 2.1 is that it will be impractical to perform exhaustive searches of the solution space. The `LITSE` algorithm will need to search through the space in an intelligent, targeted manner. To this end, the method should recognize that a proposed solution is more likely to be correct if the following conditions are true,

1. the haplotypes and their proportions imply observed allele proportions close to those observed, and
2. the suggested haplotypes are common in the haplotype population, and
3. the proposed MOI is common among infected individuals.

This intuition is the foundation for the statistical framework described in §2.6.

2.5.4 Previous Work and `LITSE` Contribution

This section presents the previous work attempting to solve the current problem and explains where the `LITSE` method differs. The research is divided into two categories: those studies that are most closely related, typically because the corresponding method relies on previously computed allele proportions, and those studies that are more loosely related, typically because they are solving a similar problem but are using as input individual reads.

2.5.4.1 Studies Concerning Malarial Haplotype Estimation

The work of Li et al. [24] is the first chronologically in the series of articles closely related to the current study. The authors apply the expectation maximization algorithm to the problem of estimating haplotype phases in *Plasmodium falciparum* but limit themselves to 12 loci, 10 of which are variable. Unlike the `LITSE` algorithm, the authors explicitly model the total number of organisms in each individual as either a constant or truncated Poisson.[6] In their simulation, the authors consider only two three loci, each of which has only two to three possible alleles. In addition, the authors consider only three different haplotype parameter sets rather than allowing this to vary. In this setting, the algorithm performs relatively well, with coverage rates in the range of 95% and confidence intervals with a width typically below 0.10.

Of all research published, the study of Taylor et al. [43] is perhaps closest to that of the current dissertation. Their focus is on estimating frequencies specifically of the *Plasmodium falciparum* species, and their input data are alleles at predetermined loci. Their method shares with the `LITSE` algorithm the goal of imputing the values for missing locus readings though – unlike the current study – they do not consider measurement errors. The method uses a Bayesian framework and includes a Markov Chain Monte Carlo (MCMC) implementation to estimate the posterior distribution of the haplotype population frequencies. In this regard, as will be explained in detail later, it differs from the `LITSE` algorithm, which uses an implementation of a tree search and the Expectation Maximization algorithm. The authors' focus is on single nucleotide polymorphisms (SNPs), while `LITSE` focuses on microsatellites. In particular, the authors focus on two genes known to play a role in drug resistance and consider a total of five loci, each of which takes on one of two possible alleles; the current study considers cases with twenty loci (and is capable of handling higher numbers) with the number of possible alleles per locus in the high single digits (again, higher values can be handled with no change to the method). Less significantly, while the study explicitly considered the concept of MOI, they used only the average MOI in the patient population; as will be seen in §2.6, the

[6] A "truncated Poisson" assigns a non-zero mass to only a subset of the integers $x \in \mathbb{N}$ and deliberately sets the mass for $x = 0$ to the value zero.

`LITSE` algorithm differs in that it estimates the MOI per isolate. The authors evaluate their algorithm using both simulated and field data and report strong results.

Hastings and Smith [18] make the distinction (as do Taylor et al. [43]) between the prevalence of a mutation among blood samples and its frequency in a parasite population, with the latter being more meaningful from a public health perspective. In particular, the former is influenced by both the latter and the average MOI. To this end, the authors present their `MapHaploFreq` algorithm, which takes as input the MOI (unlike the `LITSE` algorithm, for which this is estimated) but not the haplotype definitions. Because of the increased time complexity in considering longer haplotypes, the authors limit the program to estimating haplotypes of up to three codons (analogous to loci), unlike the `LITSE` algorithm which has no limit. The authors assume a multinomial distribution for the selection of haplotypes in a blood sample, though it is not clear that they have considered the fact that this multinomial assumes draws with replacement.[7] The authors test `MapHaploFreq` using both simulated and field data. For the simulation study, they claim success based on frequency point estimates lying in a 95% confidence interval over 95% of the time (the authors do not give much more detail than this). For the field study, where a comparison of the estimates to the truth is impossible since the latter is unknown, the authors content themselves with the fact that, using many different datasets from the same region, the frequency estimates are similar. The operation of the algorithm is discussed in detail in [17]. In Hastings et al. [19], a similar team of authors discuss in more detail the distinction between haplotype frequencies and prevalence in blood samples.

Takala et al. [42] propose an EM-based algorithm for malarial haplotype frequency estimation based on SNP data, albeit focusing on a single protein within *Plasmodium falciparum*. The authors limit themselves to a maximum MOI of three and a maximum number of loci equal to six loci. Of particular interest is the authors' comments regard-

[7]Indeed, the authors state the multinomial probability mass function of selecting 3, 1, and 1 drawn from three categories with frequencies x, y, z as

$$\binom{5}{3,1,1} 3^x 1^y 1^z,$$

thus mistakenly switching the bases and the exponents in the expression. The authors presumably corrected this in their actual implementation.

ing the need to debias raw allele frequency data before submitting it to any haplotype estimation algorithm; the `LITSE` algorithm indeed benefits from such pre-processing and this has been performed on the data obtained from the Greenhouse lab for use in §4.

Focusing on a problem less similar to the current problem, Cheesman et al. [7] propose an adaptation to Real-Time Quantitative - PCR (RTQ-PCR) that aims to measure the relative proportion of rodent malarial parasites carrying different alleles (based on codons) of the same gene with the prerequisite that allele-specific primers are prepared. The authors limited themselves to the case of two strains mixed in a laboratory setting. As such, the method deals primarily with the adaptation of a PCR technique than haplotype phasing; nevertheless, it represents an alternative approach to proportion estimation. Hunt et al. [20] later developed a method extending the work of Cheesman et al. [7] to single nucleotides. The authors test their method successfully by preparing known proportion mixtures of two different malarial strains.

Of all studies, including those unpublished, perhaps the most similar is Chaffee et al. [6], an earlier study by the author and classmates to find a solution to the same fundamental research question. In this study, the input is the same as for the `LITSE` algorithm. However, the underlying algorithm – termed here the `IMH` algorithm (from the paper's title, "Inference of Malarial Haplotypes") – differs substantially. Perhaps the greatest difference is that, as a simplification, the `IMH` algorithm assumes the MOI is equal to the longest locus length; there are clearly cases where this assumption will be false. The `IMH` algorithm proceeds by identifying the set of loci that each have the greatest number of allele proportions among all loci.[8] Each locus in this group has its allele proportions placed in ascending order, and these form the foundation for the solution. The algorithm then proceeds to fill in all remaining loci, one locus at a time, with combinations of observed alleles for that locus. For each locus individually, the algorithm aims to identify the permutation of alleles that minimizes the sum of squared differences between the observed and implied allele proportions (implied by the assumed haplotype proportions). There are $2 \times 2 = 4$ variants, corresponding to the four combinations arising from which of two options has been selected for each of two procedural choices. The first decision

[8]Put differently, it identifies the largest locus block by determining the number of distinct alleles per locus and placing in this block all loci presenting with this locus "length".

is whether or not the estimated haplotype percentages should be fixed using only the longest locus block or all loci. The second is whether to use assumed population allele frequencies for each locus; this differs from the `LITSE` approach, which makes no such assumption. In simulations, there was little variance in the accuracy performance of the four methods; to quote [6, p. 28], "[t]he four algorithms all gave very similar results", with a decrease in accuracy as the variance or true MOI increased. The conclusion includes the following,

> "Our algorithms yield fairly accurate results for simulations with two unique haplotypes, even with a measurement error $\sim N(0, 0.05^2)$, but they rapidly break down when more strains are present.... Indeed, we have only tested the algorithm using simulated samples where our estimate of the number of true haplotypes would be correct...."

The `IMH` algorithm will be compared to the `LITSE` algorithm in §3 and it is this algorithm and its possible improvements that served as the inspiration for the current study.

2.5.4.2 Related Studies Concerning Haplotype Estimation

This category comprises three subcategories: first, studies investigating diploid organisms such as humans rather than an infectious organism such as *Plasmodium falciparum*; second, studies investigating the deliberate or inevitable pooling of genetic data across individuals; and third, studies directly incorporating sequencing techniques.

In the first subcategory, Clark [8] first proposed an iterative method for haplotype reconstruction that first identified all unambiguous haplotypes (from isolates with zero or one heterozygous loci). The algorithm then adds to this list by searching through ambiguous isolates for cases where a previously identified haplotype is feasible; in such cases, the complement of the first haplptype is added to the list of possible haplotypes. The algorithm continues until either all isolates have been phased or it is impossible to complete the exercise. The author acknowledges that one of the weaknesses of the method is that it will be impossible to start the algorithm in the event that all isolates are ambiguous.

As the name of their article implies, Excoffier and Slatkin [13] use an implementation of the Expectation Maximization algorithm to estimate haplotype frequencies in a diploid population. Within the context of the current dissertation, this is similar to considering cases where the MOI is either one or two. The authors limit themselves to considering haplotypes up to 16 loci in length, with a total number of possible haplotypes less than or equal to 16,384; the proposed `LITSE` algorithm in the current dissertation surpasses these constraints. In fairness to the authors, this constraint presumably reflects the computing power available to them in the mid-1990s. To evaluate the performance of the haplotype estimation, the authors use a similarity measure proposed in Renkonen [33], which is employed in this dissertation and is explained in more detail in §3.3.5.2. A noteworthy, and expected, result is that the accuracy of the authors' method improves as the sample size increases. The article concludes by suggesting its application to micro-satellite data, the focus of the `LITSE` algorithm.

A number of studies have been performed on a slightly different problem from that which `LITSE` attempts to address: the imputation of haplotypes per individual in pooled sample; that is, samples containing genetic information from multiple individuals in one set of results. Quade et al. [32] presents an overview of such studies. This situation typically arises when either individual samples are pooled to reduce genotyping expenses or when it is difficult or impossible to separate samples (e.g., from cancerous tissue).

Initially proposed by Stephens et al. [40] and refined by Stephens and Donnelly [38], Stephens and Scheet [39], and Pirinen et al. [28], the PHASE algorithm is a frequently cited algorithm for estimating haplotypes among diploid individuals by explicitly modeling the mutation and recombination processes.

Yang et al. [49] propose an EM algorithm for haplotype estimation under DNA pooling using the EM algorithm and demonstrate through simulations its accuracy when the pooled groups are limited to a few individuals.

Kirkpatrick [22] propose the `HaploPool` method, which estimates haplotypes and their frequency in pooled samples. The method is limited to bi-allelic loci (i.e., each locus has at most two possible alleles). The method uses two algorithms, one using the previously introduced "perfect phylogeny model" and a second using regression. While the objective is similar, this setup differs from that of `LITSE`, whose key input is allele frequencies at the individual level, which handles any number of alleles per locus, and which has no phylogenetic component. The author performs a simulation study and finds the method to be superior to existing methods in terms of both accuracy and runtime.

Finally, several studies have proposed methods for determining the genetic composition of pathogens directly using sequencing reads. As such, they do not use microsatellite reads as input and thus differ somewhat from the present study. Nevertheless, their end goal is similar and it is worthwhile to consider several such studies for perspective.

Zagordi et al. [50] and Zagordi et al. [51] present the ShoRAH (Short Read Assembly into Haplotypes) algorithm to quantify the genetic diversity in mixed samples of bacteria, viruses, and cancerous tumors. As a method to address sequencing errors, the algorithm clusters similar results to form consensus haplotypes from which it is assumed the observed reads were obtained. The algorithm uses a Bayesian framework with a Dirichlet process mixture for haplotype reconstruction and maximum likelihood estimation to estimate all parameters including frequency. The authors test ShoRAH on both simulated and experimental data and report excellent recall (in the range of 0.95 to 1) but with less impressive precision (0.45 to 1).

Similarly, Prosperi et al. [31] present a method for reconstructing viral quasispecies using next-generation sequencing from infected individuals. Their goal is to identify all such variants within a population along with their prevalence. Their method requires that sequence fragments be aligned against an available reference genome with overlapping regions. Similar to the present dissertation, the MOI is explicitly modeled, unknown, and to be estimated. The authors compare their method to the ShoRAH method of [50] using simulated data, where they find similar results, and on clinical data involving the Hepatitis B virus, where their method shows similar recall and superior precision. The algorithm, later named QuRe (QuasiSpecies Reconstruction), is documented in a separate implementation note [30].

Long et al. [26] present `PoolHap`, an algorithm using Next Generation Sequencing (NGS) reads to estimate haplotype frequencies from pooled data where the haplotypes are already known.

Kessner et al. [21] develop and test the `harp` method to estimate haplotype frequencies from pooled sequence reads using the Expectation Maximization algorithm. However, their method assumes that the constituent haplotypes are known, whereas the `LITSE` algorithm attempts to reconstruct these haplotypes. The method `harp` uses the base quality scores from the reads in determining the probabilities for each solution. The authors perform simulation studies and evaluate the accuracy using the metric $\sum_{h=1}^{H}(f_h^{true} - f_h^{est})^2$, where H denotes the total number of haplotypes under consideration; this metric is identical to squared L_2 norm used in the present study for the same purpose. The authors find that their method outperforms all others under consideration, though it remains unclear the degree to which this is due to the advanced knowledge of the haplotype constructions.

2.5.4.3 `LITSE` Contribution

The details of the `LITSE` algorithm appear in the following sections. Nevertheless, it is useful to summarize now how `LITSE` extends and improves upon existing methods. In comparing this algorithm to those from previous studies, `LITSE` stands out in the following ways:

1. Number of loci. Due to the computational considerations, previous methods limited themselves to considering haplotypes up to approximately 10 loci in length; `LITSE` has been tested on 20 loci and, crucially, demonstrates an approximately linear relationship between run time and the number of loci considered.
2. Number of alleles. Previous methods considered up to three possible alleles per locus; `LITSE` faces no such limitation.
3. Modeling of the MOI. Most other work considers the role of the MOI, but only `LITSE` models this explicitly. For example, Li et al. [24] model not the MOI but the total number of infections the individual has suffered, which may include duplicates (e.g., the individual may been bitten by two different mosquitoes carrying the same haplotype; this counts as two different infections under their approach whereas under `LITSE` only completely new [to the individual] haplotypes increase the MOI).
4. Imputation of missing data. `LITSE` imputes missing locus reads for particular individuals; of the preceding methods, only Taylor et al. [43] features such an approach.

2.6 Statistical Framework

In this section, we present the statistical framework for the problem.

2.6.1 Variables and Constants

In this section, we discuss the key random variables representing the data and their assumed distributions.

In structuring the underlying problem in terms of random variables, our basic approach is to "chain together" a series of variables according to an assumed data generating process, with the distribution of each variable conditional on its predecessors. The exception will be the first variable in the chain, whose distribution is not conditional on any other variable.[9]

2.6.1.1 *Z* - Multiplicity of Infection

The first variable, Z, represents the multiplicity of infection in an individual. Recall that the multiplicity of infection indicates the number of distinct strains in an individual. As such, the variable takes on strictly positive integer values; that is,

$$Z \in \mathbb{N}_+ = \{1, 2, \dots\}.$$

This variable is unobserved but will serve as the starting point in constructing the set of variables. It is specific to an infected individual; therefore, we index the variable with the subscript n as Z_n to denote the variable for the n^{th} individual.

Our assumption regarding the distribution of Z is as follows, [10]

$$f_Z(z) = \begin{cases} \text{Multinomial}(1; \mathbf{p} = \{p_1, \dots, p_{\text{maxMOI}}\}) & z \in \{1, 2, \dots, \text{maxMOI}\} \\ 0 & o.w. \end{cases} \tag{2.2}$$

where

[9] We have to "start somewhere".

[10] We might consider a Bayesian approach with a posterior multinomial distribution, in which case the conjugate prior would be the Dirichlet distribution. The prior distribution we put on the p's would penalize higher values.

1. the constant maxMOI is the assumed maximum MOI an individual may achieve and is set based on both domain knowledge and computational considerations (higher values of maxMOI will involve larger search spaces for a solution),
2. the vector $\mathbf{p}$, with $\sum_{i=1}^{\text{maxMOI}} p_i = 1$, is the maxMOI-length vector of probabilities governing the distribution of MOIs across all infected individuals in the geographical region of interest.

Equation 2.2 can be interpreted as saying that an individual randomly is "assigned" a single (hence 1 in the probability function) MOI from maxMOI different possibilities according to the same probability distribution as all other infected individuals.

We also note that given that in equation 2.2 since the number of trials is one, the probability that $Z = j$ (i.e., the j^{th} possible MOI) is

$$\begin{aligned} f_Z(j) &= \frac{1!}{\underbrace{0!0!\ldots 0!}_{1 \text{ to } j-1}\underbrace{1!}_{j}\underbrace{0!\ldots 0!}_{j+1 \text{ to maxMOI}}} p_1^0 p_2^0 \ldots p_{j-1}^0 p_j^1 p_{j+1}^0 \ldots p_k^0 \\ &= p_j \end{aligned}$$

That is, the probability simply reduces to p_j.

Example 2.10 (Z - Multiplicity of Infection). As a simple example, we may have $Z_n = 3$, implying that the n^{th} individual is infected with exactly three distinct strains (with some set of proportions).

□

2.6.1.2 H - Haplotypes Chosen

The second random variable, $\mathbf{H}$, represents which haplotypes are present in an infected individual. Like Z, this variable is unobserved. This variable takes the form of an indicator vector; that is,

$$\mathbf{H} \in \{0, 1\}^{N_{\text{haps}}} \tag{2.3}$$

where

$$H_i = \begin{cases} 1 & \text{haplotype } i \text{ is present in the individual} \\ 0 & o.w. \end{cases}$$

with the condition

$$\sum_{i}^{N_{\text{haps}}} H_i = Z. \tag{2.4}$$

Recall that N_{haps} indicates the total number of possible haplotypes given the loci under consideration, with the haplotypes in a fixed order.

We must have the summation condition in (2.4) to remain consistent with the assumed MOI. This condition, plus the property that $H_i \in \{0, 1\}\ \forall i$, implies that precisely Z entries have value 1 with the rest equal to 0. Thus $\mathbf{H}$ is an indicator vector.

Our assumption regarding the distribution of $\mathbf{H}$ is as follows,

$$f_{H|Z}(\mathbf{h}, z) = \text{Multinomial}(z, \boldsymbol{\pi} = \{\pi_1, \ldots, \pi_{N_{\text{haps}}}\}) \tag{2.5}$$

In equation (2.5), the unknown parameter $\boldsymbol{\pi}$ is a vector of probabilities, with each π_i representing the population frequency of the i^{th} haplotype among all haplotypes in the geographic region of interest. We are thus assuming the likelihood of a haplotype being present in an individual is directly dependent on its population frequency, with a haplotype's chance of being present in an individual increasing as its population frequency increases.

Since we are assuming that $h_i \in \{0, 1\}\ \forall i$, let $\boldsymbol{\iota} = \{\iota_1, \iota_2, \ldots, \iota_z\} = \{i : h_i = 1\}$ denote the indices of the elements in $\mathbf{h}$ that equal 1. Then equation (2.5) reduces to,

$$f_{H|Z}(\mathbf{h}, z) = z!\pi_{\iota_1} \ldots \pi_{\iota_z} \tag{2.6}$$

We use this simplified expression going forward.

Note that this probability is also consistent with equation (2.4). There are, however, two caveats to the use of the multinomial distribution in this context:[11]

1. The multinomial distribution reflects selecting *with* replacement. Here, we are assuming selection (of haplotypes) *without* replacement. However, first we not that $N_{\text{haps}} >> Z$; that is, of all possible haplotypes, we are choosing only a very few. Second, we make the informal assumption that the population frequencies

[11]An alternative would be to use the multivariate hypergeometric distribution; however, the density function for this distribution is more complex and the two distributions converge as the sample grows large.

of the existing haplotypes are spread over many haplotypes; in other words, we assume that there is no haplotype or small group of haplotypes that dominates the population in terms of frequency. With these two assumptions, the distinction between sampling with or without replacement decreases in importance.

2. The multinomial distribution does not enforce equation (2.3). Specifically, it would support vectors where the count for a particular haplotype is greater than one. Nevertheless, once again given that $N_{\text{haps}} >> Z$, the event under the multinomial that $H_i > 1$ for some i is improbable.

Note that $\mathbf{H}$ says nothing about the *proportions* of the various haplotypes in an individual; it only indicates the (binary) presence. This use of $\mathbf{H}$ is consistent with our assumption that the overall haplotype population frequencies do not influence haplotype *concentrations* in the individual; they do, however, influence the *probability* of a haplotype being present.

Example 2.11 (H - Haplotype Indicator). Consider the case where we have 3 loci, the first and third of which have 3 possible alleles and the second of which has 2 alleles. Then $N_{\text{haps}} = 3 \cdot 2 \cdot 3 = 18$ and there are 18 possible haplotypes. Thus $\mathbf{H}$ will have length 18, with each entry corresponding to a particular haplotype. Assume that $Z = 3$. Then the following vector,

$$\mathbf{h} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}^T$$

is valid while

$$\mathbf{h} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T$$

is not (since it has 4 non-zero entries, suggesting that there are $4 \neq Z$ haplotypes present). If, for example,

$$\mathbf{h} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T,$$

and $\pi_2 = 0.01, \pi_6 = 0.08, \pi_{15} = 0.1$, then

$$f_{\mathbf{H}|Z}(\mathbf{h}, z = 3) = \frac{3!}{0! \cdot 1! \cdot (0!)^3 \cdot 1! \cdot (0!)^8 \cdot 1! \cdot (0!)^3} 0.01^1 0.08^1 0.1^1$$

$$= 6 \cdot 8 \cdot 10^{-5}$$
$$= 4.8 \cdot 10^{-4}$$

□

2.6.1.3 Y - Haplotype Proportions

The third random variable, $\mathbf{Y}$, is a vector representing the haplotype *proportions* in an individual. Like the previous two variables, $\mathbf{Y}$ is unobserved. Also like $\mathbf{H}$, $\mathbf{Y}$ is of length N_{haps}. Specifically,

$$\mathbf{Y} \in [0,1]^{N_{\text{haps}}} \tag{2.7}$$

with Y_i indicating the proportion of haplotype i among all haplotypes present in the individual. Since Y_i represents a proportion, we also have the condition,

$$\sum_{i=1}^{N_{\text{haps}}} Y_i = 1 \tag{2.8}$$

Furthermore, $\mathbf{Y}$ needs to be consistent with $\mathbf{H}$. Specifically, we must have,

$$\begin{aligned} &Y_i > 0 \text{ iff } H_i = 1 \\ &Y_i = 0 \text{ iff } H_i = 0 \end{aligned} \tag{2.9}$$

In other words, a haplotype has a non-zero proportion if and only if it is indicated as present in the individual (as indicated by H).

For the distribution of $\mathbf{Y}$, we condition on $\mathbf{H}$ given this dependency in equation (2.9). At the same time, we must recognize the assumption that Y_i is *not* a function of H_i when $H_i = 1$.[12] To capture this, we express the distribution as follows,

$$f_{\mathbf{Y}|\mathbf{H}}(\mathbf{y}, \mathbf{h}) = \begin{cases} \xi & y_i > 0 \iff h_i = 1, e^T\mathbf{y} = 1 \\ 0 & o.w. \end{cases} \tag{2.10}$$

In words, equation (2.10) states that when the conditions are satisfied by a candidate vector $\mathbf{y}$, the likelihood of $\mathbf{y}$ is constant at $\xi > 0$ and equal to the likelihood of any

[12]When $H_i = 0$, clearly the opposite is the case: $\mathbf{Y}_i$ is a constant and is equal to 0.

other feasible candidate vector $\mathbf{y}'$. When the conditions are not met, the likelihood is zero. In short, the conditional distribution is uniform. The value of ξ is unimportant; the parameter ξ is simply a normalizing constant to ensure that $f_{\mathbf{Y}|\mathbf{H}}(\mathbf{y}, \mathbf{h})$ integrates to 1.

Example 2.12 (Y - Haplotype Proportions). Continuing with Example 2.11 from §2.6.1.2, assume that

$$\mathbf{h} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^T$$

is valid. Then consider four candidate $\mathbf{y}$ vectors. First, the vector

$$\mathbf{y} = \begin{pmatrix} 0.1 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 \end{pmatrix}^T$$

is invalid because it violates condition (2.9) since $y_1 > 0$ yet $h_1 = 0$ while $y_2 = 0$ yet $h_2 = 1$. Second, the vector

$$\mathbf{y} = \begin{pmatrix} 0 & 0.1 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 \end{pmatrix}^T$$

is invalid because it violates condition (2.8) although it satisfies equation (2.9). Under equation (2.10), both have a likelihood of zero. Next consider the two vectors,

$$\mathbf{y}' = \begin{pmatrix} 0 & 0.2 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 \end{pmatrix}^T$$

and

$$\mathbf{y}'' = \begin{pmatrix} 0 & 0.7 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \end{pmatrix}^T.$$

Both of these vectors are valid, yet they represent very different distributions for the three implied haplotypes. Despite this difference, each has a likelihood of ξ conditional on $\mathbf{h}$.

□

2.6.1.4 $S^{(l)}$ - Signature Matrices

Next, we construct a series of *constant* matrices, one per locus, called "signature matrices".[13] Each of these is labeled $S^{(l)}$, where $l = 1, \dots, L$ denotes the locus in question.

[13]We call these "signature matrices" because they capture the important identifying feature - in this case, allele composition - of each haplotype.

Each $S^{(l)}$ documents which allele each haplotype features for that locus. The matrix $S^{(l)}$ takes the form,

$$S^{(l)} = \begin{pmatrix} S^{(l)}_{1,1} & S^{(l)}_{1,2} & \dots & S^{(l)}_{1,N_{\text{haps}}} \\ S^{(l)}_{2,1} & S^{(l)}_{2,2} & \dots & S^{(l)}_{2,N_{\text{haps}}} \\ \vdots & \vdots & \ddots & \vdots \\ S^{(l)}_{\ell(l),1} & S^{(l)}_{\ell(l),2} & \dots & S^{(l)}_{\ell(l),N_{\text{haps}}} \end{pmatrix}$$

with

$$S^{(l)}_{i,j} = \begin{cases} 1 & \text{Haplotype } j \text{ has allele } i \text{ at locus } l \\ 0 & o.w. \end{cases}$$

where $\ell(l)$ indicates the number of different possible alleles for locus l and N_{haps}, as before, indicates the total number of possible haplotypes. Thus $S^{(l)}$ is a very wide, short matrix.[14] It follows from the fact that each haplotype will have exactly one allele represented at a locus that the sum of each column will equal 1; in other words, there will be exactly one entry per column equal to 1 with the rest equal to 0.

Example 2.13 (S - Signature Matrix). Consider the simple case where we have a total of only two loci, the first of which has 3 possible alleles (A, B, and C) and the second of which has 4 possible alleles (A, B, C, and D). Recall that by "possible", we mean known to exist but not necessarily present in any sample observation under investigation. Table 2.2 lists the $3 \cdot 4 = 12$ possible haplotypes in this example. The ordering of the

	Haplotype											
Locus	1	2	3	4	5	6	7	8	9	10	11	12
1	A	A	A	A	B	B	B	B	C	C	C	C
2	A	B	C	D	A	B	C	D	A	B	C	D

Table 2.2: Sample Haplotypes

haplotypes is important. Here, as always, we order the haplotypes by ascending allele within locus 1, then within locus 2, and so on.

[14]In practice, there will be no need to construct the entire $S^{(l)}$ since for any particular observation the vast majority of columns will exist for infeasible haplotypes. Instead, we will restrict its construction to only those haplotypes that are relevant.

Since the first locus has 3 possible alleles, $S^{(1)}$ will be a 3×12 matrix as depicted in equation (2.11).

$$S^{(1)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{2.11}$$

As always, the i^{th} row depicts the i^{th} allele within the locus (here, locus 1) and the j^{th} column depicts the j^{th} haplotype. For example, since the first 4 haplotypes all have allele A, the first row has 1's in the first four positions.

Since the second locus has 4 possible alleles, $S^{(2)}$ will be a 4×12 matrix as depicted in equation (2.12).

$$S^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.12}$$

Here, since the 4 possible alleles cycle as we proceed from one haplotype to the next as depicted in Table 2.2, this is represented in $S^{(2)}$.

□

2.6.1.5 $\mathbf{U}^{(l)}$ - Implied Allele Proportions for Locus l

The use of the signature matrices is as follows. Consider the product $S^{(l)}\mathbf{Y}$,

$$S^{(l)}\mathbf{Y} = \begin{pmatrix} S^{(l)}_{1,1} & S^{(l)}_{1,2} & \cdots & S^{(l)}_{1,N_{\text{haps}}} \\ S^{(l)}_{2,1} & S^{(l)}_{2,2} & \cdots & S^{(l)}_{2,N_{\text{haps}}} \\ \vdots & \vdots & \ddots & \vdots \\ S^{(l)}_{\ell(l),1} & S^{(l)}_{\ell(l),2} & \cdots & S^{(l)}_{\ell(l),N_{\text{haps}}} \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ \vdots \\ Y_{N_{\text{haps}}} \end{pmatrix}$$

$$\begin{aligned}
&= \begin{pmatrix} \sum_{j=1}^{N_{\text{haps}}} S_{1,j}^{(l)} Y_j \\ \sum_{j=1}^{N_{\text{haps}}} S_{2,j}^{(l)} Y_j \\ \sum_{j=1}^{N_{\text{haps}}} S_{3,j}^{(l)} Y_j \\ \vdots \\ \sum_{j=1}^{N_{\text{haps}}} S_{\ell(l),j}^{(l)} Y_j \end{pmatrix} \\
&:= \begin{pmatrix} U_1^{(l)} \\ U_2^{(l)} \\ U_3^{(l)} \\ \vdots \\ U_{\ell(l)}^{(l)} \end{pmatrix} \\
&= \mathbf{U}^{(l)} \qquad (2.13)
\end{aligned}$$

Thus the scalar $U_i^{(l)}$ is the implied proportion of allele i for locus l by summing each haplotype's contribution to that proportion. Each $U_i^{(l)}$ – for any locus l and for any allele i at that locus – is entirely determined by $\mathbf{Y}$. Given this dependence, we can state the conditional pdf as follows,

$$f_{\mathbf{U}^{(l)}|\mathbf{Y}}(\mathbf{u}^{(l)}, \mathbf{y}) = \begin{cases} 1 & \mathbf{u}^{(l)} = S^{(l)}\mathbf{y} \\ 0 & o.w. \end{cases} \qquad (2.14)$$

That is, the density for a candidate $\mathbf{u}^{(l)}$ equals one if and only if it equals (is consistent with) what is implied by $\mathbf{y}$; otherwise the density equals zero.

Example 2.14 (U - Implied Allele Proportions). Consider again $S^{(2)}$ from equation (2.12) in Example 2.13. Assume that

$$\mathbf{y} = \begin{pmatrix} 0 & 0.05 & 0 & 0.34 & 0 & 0 & 0 & 0.26 & 0 & 0 & 0 & 0.35 \end{pmatrix}^T$$

is valid. Then the implied allele proportions for locus 2 are as follows,

$$\begin{pmatrix} U_1^{(l)} \\ U_2^{(l)} \\ U_3^{(l)} \\ U_4^{(l)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0.05 \\ 0 \\ 0.34 \\ 0 \\ 0 \\ 0.26 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.35 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0.05 \\ 0.26 \\ 0.69 \end{pmatrix}$$

$\square$

As demonstrated in Example 2.14, we can have certain alleles with an implied proportion of zero (e.g., allele $i = 1$ in the example).

In section §2.6.1.6 it will be clearly desirable to remove any allele proportions in $\mathbf{U}^{(l)}$ equal to zero. To this end, define

$$\mathcal{NZ}(\mathbf{U}^{(l)}) := \{j : U_j^{(l)} > 0\}. \tag{2.15}$$

That is, $\mathcal{NZ}(\mathbf{U}^{(l)})$ is the set of indices of the non-zero implied allele proportions at locus l.

Example 2.15 ($\mathcal{NZ}$ - Extraction of Non-Zero Proportions). Continuing with Example 2.14, given $\mathbf{U}^{(l)}$, we have $\mathcal{NZ}(\mathbf{U}^{(l)}) = \{2, 3, 4\}$ since $U_1^{(l)} = 0$ and $U_i^{(l)} > 0, i \neq 1$. Thus $\mathcal{NZ}(\mathbf{U}^{(l)})_1 = 2, \mathcal{NZ}(\mathbf{U}^{(l)})_2 = 3, \mathcal{NZ}(\mathbf{U}^{(l)})_3 = 4$ and $|\mathcal{NZ}(\mathbf{U}^{(l)})| = 3$.

$\square$

Example 2.16 ($\mathcal{NZ}(\mathbf{U}^{(l)}$ - Implied Non-Zero Proportions). Continuing with Example 2.15, here

$$\mathbf{U}^{(l)}_{\mathcal{NZ}(\mathbf{U}^{(l)})} = \begin{bmatrix} U^{(l)}_{\mathcal{NZ}(\mathbf{U}^{(l)})_1} \\ U^{(l)}_{\mathcal{NZ}(\mathbf{U}^{(l)})_2} \\ U^{(l)}_{\mathcal{NZ}(\mathbf{U}^{(l)})_3} \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.26 \\ 0.69 \end{bmatrix}$$

The only difference between this quantity and that presented in the last line of 2.13 is that the former has deliberately removed all zero entries while still retaining the order of all non-zero entries. This filtering is indicated by the subscript in the leftmost term above.

$\square$

Let $\boldsymbol{U}_n = \{\mathbf{U}^{(1)}_n, \mathbf{U}^{(2)}_n, \ldots, \mathbf{U}^{(N_l)}_n\}$ denote the collection of implied allele proportions across all loci for observation n.

2.6.1.6 $\mathbf{X}^{(l)}$ - Observed Allele Proportions at Locus l

The final random variable, $\mathbf{X}^{(l)}$, denotes the observed allele proportions at locus l. This is the first, and only, observed random variable in the data generating distribution. Similar to $\mathbf{Y}$, which represents the proportion of haplotypes in a population, $\mathbf{X}^{(l)}$ has the following properties,

$$\mathbf{X}^{(l)} \in [0,1]^{\ell(l)} \tag{2.16}$$

with $X^{(l)}_i$ indicating the proportion of allele i at locus l. Since $X^{(l)}_i$ represents a proportion, the following conditions also hold,

$$X^{(l)}_i \geq 0; \qquad \sum_{i=1}^{\ell(l)} X^{(l)}_i = 1 \tag{2.17}$$

That is, for any locus l, each observed allele proportion for that locus must be non-negative and the sum of all such proportions must equal one.

Similar to the set $\mathcal{NZ}(\mathbf{U}^{(l)})$, which denotes the indices of non-zero *implied* proportions of locus l, let $\mathcal{NZ}(\mathbf{X}^{(l)})$ denote the indices of non-zero *observed* proportions of locus l.

Example 2.17 (X - Observed Allele Proportions). Returning to Example 2.14, recall that it is assumed that $\ell(l) = 4$ for locus $l = 1$. One possible $\mathbf{X}^{(l)}$ would be

$$\mathbf{X}^{(l)} = \begin{bmatrix} 0 \\ 0.04 \\ 0.30 \\ 0.66 \end{bmatrix}$$

which is reasonably "close" to $\mathbf{U}^{(l)}$ itself. Another possible $\mathbf{X}^{(l)}$ is

$$\mathbf{X}^{(l)} = \begin{bmatrix} 0 \\ 0.25 \\ 0.37 \\ 0.38 \end{bmatrix}$$

which is far more dissimilar. In each case, $\mathcal{NZ}(\mathbf{X}^{(l)}) = \{2, 3, 4\}$.

$\square$

A final assumption is that $\mathbf{X}^{(l)}$ will have exactly the same zero entries, if any, as $\mathbf{U}^{(l)}$; in other words, the two should be consistent in which alleles are expressed (though the actual values can differ significantly, albeit with decreasing likelihood). This requirement is captured formally in the conditional distribution of $\mathbf{X}^{(l)}$ as stated below,

$$f_{\mathbf{X}^{(l)}|\mathbf{U}^{(l)}}(\mathbf{x}, \mathbf{u}) = \begin{cases} \text{Dirichlet}\left(\mathbf{x}; c^{(l)}\mathbf{u} + 1\right) & \mathcal{NZ}(\mathbf{u}) = \mathcal{NZ}(\mathbf{x}) \\ 0 & o.w. \end{cases} \tag{2.18}$$

where $c^{(l)} > 0$ is a dispersion parameter for locus l. We will discuss this distribution in detail below, but as a very brief introduction we can say that the above distribution will serve to center the observed allele proportions $\mathbf{X}^{(l)}$ around $\mathbf{U}^{(l)}$ and control the dispersion from this center via the parameter $c^{(l)}$.

The use of the distribution in equation (2.18) can be justified by reviewing some of the general properties of the Dirichlet distribution and demonstrating how it can be incorporated in the current framework.

2.6.1.6.1 Properties of the Dirichlet Distribution

The standard representation of the probability density function for a random vector $\mathbf{o}$ under this distribution is as follows,

$$f(\mathbf{o}; \boldsymbol{\alpha}) = \begin{cases} \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{d} o_i^{\alpha_i - 1} & o_i \geq 0 \ \forall i; \sum_i o_i = 1 \\ 0 & o.w. \end{cases}, \tag{2.19}$$

where d is the length of $\mathbf{o}$,

$$\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_d\}, \alpha_i > 0 \tag{2.20}$$

is the set of parameters[15], and

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{d} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{d} \alpha_i\right)}, \tag{2.21}$$

incorporating $\Gamma(\cdot)$, the gamma function.

Note that the requirement that $\sum_i o_i = 1; o_i \geq 0$ in equation (2.19) lends itself to using the Dirichlet distribution to model *proportions*, since proportions naturally follow such constraints.

The mean of each o_i is

$$E[o_i] = \frac{\alpha_i}{\alpha_0}, \tag{2.22}$$

where

$$\alpha_0 = \sum_i \alpha_i.$$

That is, each o_i is centered at the proportion of α_i vis-à-vis the sum of all α_i.

The variance is as follows,

$$\text{Var}[o_i] = \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)}. \tag{2.23}$$

Note that the variance decreases as a_0 increases, a useful property that we will exploit.

The covariance between $o_i, o_j; i \neq j$ is

$$\text{Cov}[o_i, o_j] = \frac{-\alpha_i \alpha_j}{\alpha_0^2(\alpha_0 + 1)}; i \neq j.$$

[15]Note that there is no requirement that $\sum_i \alpha_i = 1$. The only requirement is that each α_i be strictly positive. However, from equation (2.24), we see that we need the more restrictive condition that $\alpha_i > 1$ for the mode to exist. Since we will make use of the mode, we will need this restriction.

In particular, this covariance is negative as all constants in the expression are strictly positive by definition. This is intuitive; whenever any one proportion increases (decreases), one would expect the others to decrease (increase).

Finally, the mode of the distribution occurs at the following point,

$$\text{mode}(f(\mathbf{o};\boldsymbol{\alpha})) = \left(\frac{\alpha_1 - 1}{a_0 - d}, \frac{\alpha_2 - 1}{a_0 - d}, \; \dots \; , \frac{\alpha_d - 1}{a_0 - d}\right); \alpha_i > 1 \tag{2.24}$$

which converges to the mean as the elements of $\boldsymbol{\alpha}$ increase. Note that for the mode to exist, we have the condition $\alpha_i > 1$. Since this dissertation will implicitly make use of the mode in the parameter estimation procedure described in §2.7.4.1, it is further assumed that $\alpha_i > 1$.[16]

2.6.1.6.2 Application of the Dirichlet Distribution

Because the only requirement on α_i in equation (2.19) is that $\alpha_i > 1$, the Dirichlet distribution can be reparameterized as follows. When modeling $\mathbf{X}^{(l)}$, it is desirable to ensure that the expected location where the likelihood is maximized is consistent with the parameterization. As such, one makes use of the mode in equation (2.24) and proceeds as follows. We first assume that the entries of the vector $\boldsymbol{\alpha}$ are positive and sum to 1; that is, $\boldsymbol{\alpha}$ is a vector of proportions. One constructs a new $\boldsymbol{\alpha}'$ vector as $\alpha'_i = c\alpha_i + 1 \; \forall i$. Here $c > 0$ represents a scaling factor whose value will control the concentration of the distribution. Now, by construction, $\alpha'_i > 1 \; \forall i$.

Under this reparameterization, the pdf in equation (2.19) becomes

$$\begin{aligned} f(\mathbf{o};\boldsymbol{\alpha}') &= \begin{cases} \frac{1}{B(\boldsymbol{\alpha}')} \prod_{i=1}^{d} o_i^{\alpha'_i - 1} & o_i \geq 0 \; \forall i; \sum_i o_i = 1 \\ 0 & o.w. \end{cases} \\ &= \begin{cases} \frac{1}{B(c\boldsymbol{\alpha}+1)} \prod_{i=1}^{d} o_i^{c\alpha_i} & o_i \geq 0 \; \forall i; \sum_i o_i = 1 \\ 0 & o.w. \end{cases} \end{aligned} \tag{2.25}$$

and equation (2.20) becomes

$$\boldsymbol{\alpha}' = \{c\alpha_1 + 1, \dots, c\alpha_d + 1\}; \alpha_i > 0, \sum_i \alpha_i = 1, c > 0. \tag{2.26}$$

[16] Rather than simply carrying forward with the less restrictive condition that $\alpha_i > 0$ as stated in equation (2.20).

Consider the mean and variance under this new form, which we can compute through substitution in equations (2.22) and (2.23). First, note that $\alpha'_0 = \sum_i \alpha'_i = \sum_i (c\alpha_i + 1) = c + d$. The mean of each o_i is thus

$$E[o_i] = \frac{\alpha'_i}{a'_0} = \frac{c\alpha_i + 1}{c + d},$$

with

$$\begin{aligned} \lim_{c\to\infty} \frac{c\alpha_i + 1}{c + d} &= \lim_{c\to\infty} \frac{\alpha_i + 1/c}{1 + d/c} \\ &= \frac{\alpha_i}{1} = \alpha_i. \end{aligned}$$

That is, the expectation of the i^{th} element in $\mathbf{o}$ converges to the proportion α_i as the concentration increases.

For the variance,

$$\begin{aligned} \text{Var}[o_i] &= \frac{\alpha'_i(\alpha'_0 - \alpha'_i)}{(\alpha'_0)^2(\alpha'_0 + 1)} \\ &= \frac{(c\alpha_i + 1)(c + d - c\alpha_i - 1)}{(c + d)^2(c + d + 1)} \end{aligned}$$

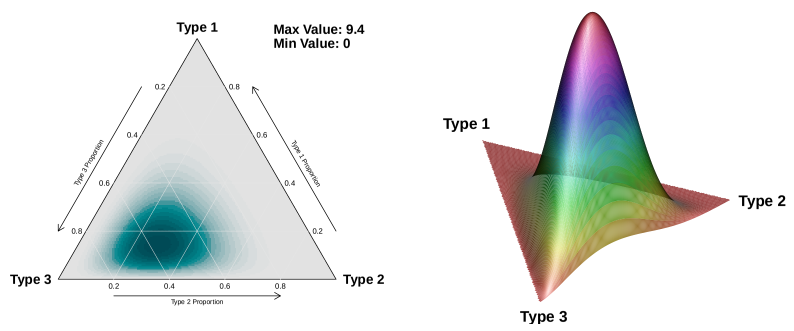
with

$$\lim_{c\to\infty} \frac{(c\alpha_i + 1)(c + d - c\alpha_i - 1)}{(c + d)^2(c + d + 1)} = 0.$$

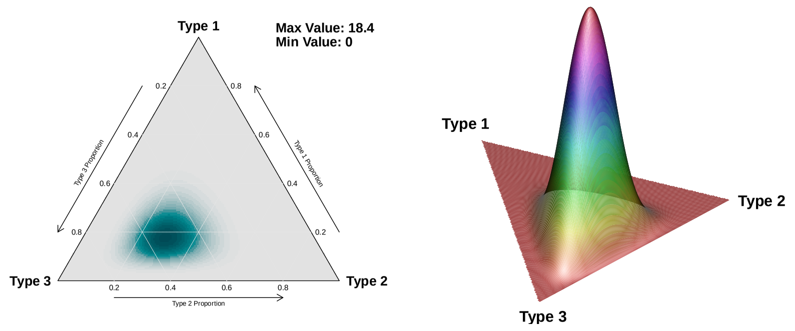
Since $\alpha'_0 > \alpha'_i > 0$, the variance will always be non-negative. We see that the variance decreases (increases) as c increases (decreases) for any fixed value of α_i, with the variance converging to zero asymptotically. Thus c determines the dispersion of the distribution.

This property of c is depicted in Figure 2.4. We fix $\boldsymbol{\alpha} = 0.2/0.3/0.5$ and investigate the behavior of the Dirichlet pdf in equation (2.25) as c varies across values 10, 20, 50, and 100. Each panel has two components: a ternary diagram (see Stover [41] for an explanation) on the left and a 3D wireframe plot on the right depicting the value of the pdf for all valid combinations of $\mathbf{o} = \{o_1, o_2, o_3\}$[17]. In Figure 2.4(a) for $c = 10$, the region of peak values is relatively disperse. As c increases, the ternary diagram shows a tighter region of peak values and the wireframe plot has a sharper peak.

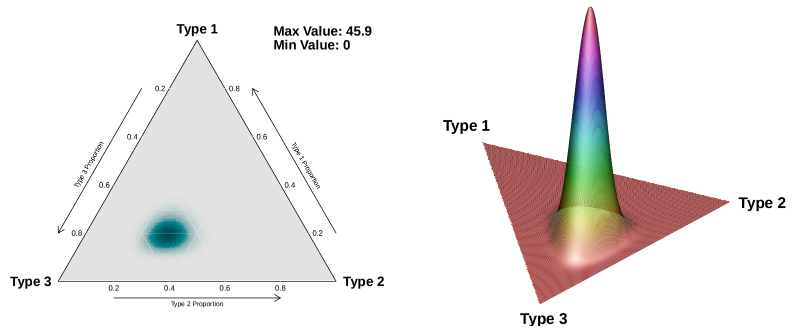
To consider this property of c from a slightly different perspective, consider the distribution of $n = 1000$ iid samples from the Dirichlet distribution as c varies but



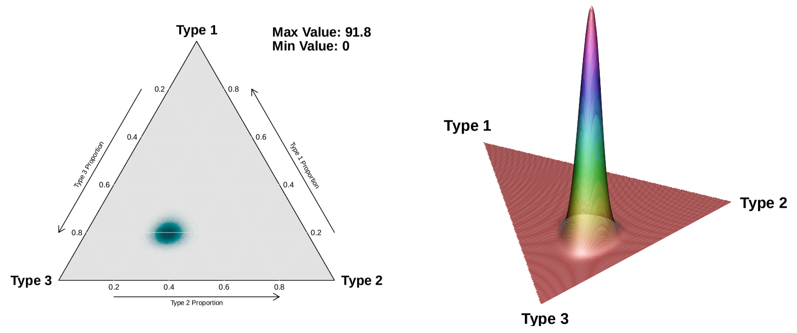
(a) $c = 10$



(b) $c = 20$

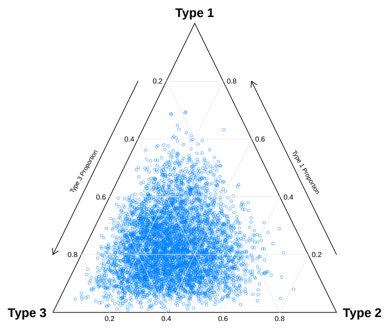


(c) $c = 50$

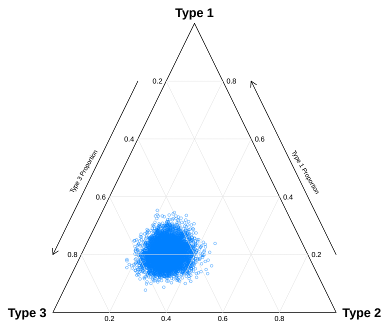


(d) $c = 100$

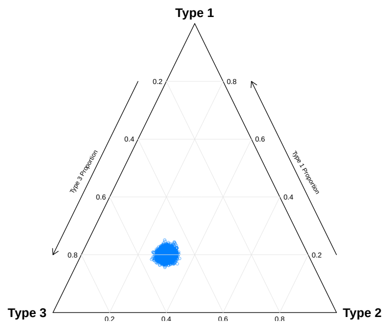
Figure 2.4: Density plots for $\boldsymbol{\alpha} = 0.2/0.3/0.5$. Each of the four panels above presents the behavior of the pdf value for the Dirichlet distribution when $\boldsymbol{\alpha} = 0.2/0.3/0.5$ but c varies for all valid values of $\mathbf{o}$. Each panel has two plots: a ternary diagram and a wireframe plot. (For an introduction to ternary diagrams, see [27].) As c increases, the region of the greatest values becomes smaller, the peak in the wireframe plot becomes sharper, and the peak values focus increasingly on α. Note that the scale of the level plots on the left hand side of each panel varies from panel to panel.



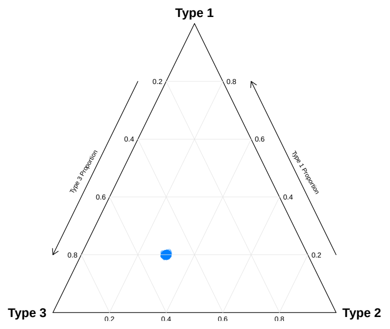
(a) $c = 10$



(b) $c = 100$



(c) $c = 1000$



(d) $c = 10000$

Figure 2.5: Scatter plots for $\boldsymbol{\alpha} = 0.2/0.3/0.5$. Each of the four panels above presents the a scatter plot of $n = 1000$ points in blue randomly generated under the Dirichlet distribution when $\boldsymbol{\alpha} = 0.2/0.3/0.5$ but c varies. As c increases, the region where the points are present shrinks and converges to 0.2/0.3/0.5.

$\boldsymbol{\alpha} = 0.2/0.3/0.5$ remains fixed. This is depicted in Figure 2.5. Analogous to the previous result, we see that the range of values shrinks as $c \to \infty$.

Having reviewed some of the properties of the Dirichlet distribution, we are in a position to explain equation (2.18) as the conditional distribution for $\mathbf{X}^{(l)}$, the observed allele proportions at locus l. The vector $\mathbf{X}^{(l)}$ will take the place of the $\mathbf{o}$ vector in equation (2.25). Since we want the non-zero observed allele proportions to be centered at the implied non-zero proportions represented by $\mathbf{U}^{(l)}_{\mathcal{NZ}(\mathbf{U}^{(l)})}$, this vector replaces $\boldsymbol{\alpha}$ in equation (2.26). The justification is as follows. Note that any $\mathbf{X}^{(l)}$ will be composed of at least one non-zero allele (perhaps all of the alleles will be non-zero) with possibly one or more zero alleles. As such, the entire $\mathbf{X}^{(l)}$ vector is determined when only the

[17]Recall that we must have $o_i \geq 0$ and $\sum_i o_i = 1$.

non-zero alleles have been determined. Given the previously mentioned assumption that $\mathcal{NZ}(\mathbf{U}^{(l)}) = \mathcal{NZ}(\mathbf{X}^{(l)})$, we would view as incompatible proposed pairing between implied and observed proportions with, for example, a $\mathbf{U}^{(l)}$ vector with $U_3^{(l)} = 0$ and an $\mathbf{X}^{(l)}$ vector with $X_3^l = 0.23$ (or vice versa). With this assumption, we base the conditional probability of $\mathbf{X}^{(l)}$ on $\mathbf{U}^{(l)}_{\mathcal{NZ}(\mathbf{U}^{(l)})}$ and $c^{(l)}$. The parameter $c^{(l)}$, to be estimated from the data, allows us to control the dispersion around the mean for the locus l in question and let this dispersion vary by locus. "Noisy" loci – those subject to relatively high measurement error – would be expected to have a low value for $c^{(l)}$ while "clean" loci would be expected to have a high value for $c^{(l)}$; modeling this concentration parameter per locus allows for a mixture of noisy and clean loci.

Example 2.18 (Observed vs. Implied Allele Proportions). Consider the case where

$$\mathbf{u}^{(l)} = \begin{pmatrix}0 & 0.1 & 0 & 0.2 & 0 & 0.4 & 0.3\end{pmatrix}^T.$$

From $\mathbf{u}^{(l)}$, we see that in this example, the total number of alleles at the locus is 7 and the number of non-zero alleles is 4. Thus $\ell(l) = 7$, $\mathcal{NZ}(\mathbf{u}^{(l)}) = \{2, 4, 6, 7\}$, and $|\mathcal{NZ}(\mathbf{u}^{(l)})| = 4$.

So we have,

$$\begin{aligned}\mathbf{u}^{(l)}_{\mathcal{NZ}(\mathbf{u}^{(l)})} &= \begin{pmatrix}u^{(l)}_{\mathcal{NZ}(\mathbf{u}^{(l)})_1} & u^{(l)}_{\mathcal{NZ}(\mathbf{u}^{(l)})_2} & u^{(l)}_{\mathcal{NZ}(\mathbf{u}^{(l)})_3} & u^{(l)}_{\mathcal{NZ}(\mathbf{u}^{(l)})_4}\end{pmatrix}^T \\ &= \begin{pmatrix}u_2^{(l)} & u_4^{(l)} & u_6^{(l)} & u_7^{(l)}\end{pmatrix}^T \\ &= \begin{pmatrix}0.1 & 0.2 & 0.4 & 0.3\end{pmatrix}^T\end{aligned}$$

Consistent with equation (2.26),

$$\boldsymbol{\alpha}' = \begin{pmatrix}c^l \cdot 0.1 + 1 & c^l \cdot 0.2 + 1 & c^l \cdot 0.4 + 1 & c^l \cdot 0.3 + 1\end{pmatrix}$$

for some $c^l > 0$.

Now consider four different candidate $\mathbf{x}^{(l)}$: $\mathbf{x}^{(l)*}$, $\mathbf{x}^{(l)**}$, $\mathbf{x}^{(l)***}$, and $\mathbf{x}^{(l)****}$. Let

$$\mathbf{x}^{(l)*} = \begin{pmatrix}0 & 0 & 0 & 0.5 & 0 & 0.35 & 0.15\end{pmatrix}$$

The vector $\mathbf{x}^{(l)*}$ is not compatible with $\mathbf{u}^{(l)}$ since there is a mismatch in the non-zero elements. While $x_i^{(l)*}$ is indeed zero whenever $u_i^{(l)}$ is zero, it is also true that $x_2^{l*} = 0$ while $u_2^{(l)} = 0.1$. Thus $f_{\mathbf{X}^{(l)}|\mathbf{U}^{(l)}}(\mathbf{x}^{(l)*}, \mathbf{u}^{(l)}) = 0$.

Similarly, let

$$\mathbf{x}^{(l)**} = \begin{pmatrix} 0 & 0 & 0 & 0.5 & 0.05 & 0.3 & 0.15 \end{pmatrix}$$

The vector $\mathbf{x}^{(l)**}$ is also incompatible with $\mathbf{u}^{(l)}$ since there is a mismatch in the non-zero elements. Specifically, $u_2^{(l)} = 0.1$ while $x_2^{(l)**} = 0$ and $u_5^{(l)} = 0$ while $x_5^{(l)**} = 0.05$. Thus $f_{\mathbf{X}^{(l)}|\mathbf{U}^{(l)}}(\mathbf{x}^{(l)**}, \mathbf{u}^{(l)}) = 0$ as well.

Now let

$$\mathbf{x}^{(l)***} = \begin{pmatrix} 0 & 0.05 & 0 & 0.5 & 0 & 0.3 & 0.15 \end{pmatrix}$$

and

$$\mathbf{x}^{(l)****} = \begin{pmatrix} 0 & 0.09 & 0 & 0.19 & 0 & 0.4 & 0.32 \end{pmatrix}$$

Both vectors are compatible with $\mathbf{u}^{(l)}$ since both have zero entries in precisely the same location as $\mathbf{u}^{(l)}$. Their pdf values are

$$f(\mathbf{x}^{(l)***}; \boldsymbol{\alpha}') = \frac{1}{B(\boldsymbol{\alpha}')} \left(0.05^{c\cdot 0.1+1} 0.5^{c\cdot 0.2+1} 0.3^{c\cdot 0.4+1} 0.15^{c\cdot 0.3+1}\right)$$

and

$$f(\mathbf{x}^{(l)****}; \boldsymbol{\alpha}') = \frac{1}{B(\boldsymbol{\alpha}')} \left(0.09^{c\cdot 0.1+1} 0.19^{c\cdot 0.2+1} 0.4^{c\cdot 0.4+1} 0.32^{c\cdot 0.3+1}\right),$$

respectively.

We note that $\mathbf{x}^{(l)****}$ is "closer" to $\mathbf{u}^{(l)}$ in terms of the distribution of the proportions represented. Figure 2.6 presents the value of the pdf in equation (2.25) for each of the two vectors for different values of c. We see that, as expected, the second vector has a higher density value for all values of $c^{(l)}$ investigated. As the value of $c^{(l)}$ increases, the relative advantage of the $x^{(l)****}$ increases due to its closeness to the mode while the density for $x^{(l)***}$ remains relatively flat.

□

Applying the discussed reparameterization of the Dirichlet distribution to the original likelihood function stated for $\mathbf{X}^{(l)}$ in equation (2.18) yields,

$$f_{\mathbf{X}^{(l)}|\mathbf{U}^{(l)}}(\mathbf{x}, \mathbf{u}) = \begin{cases} \frac{1}{B(\{c^{(l)}u_i+1; i\in\mathcal{NZ}(\mathbf{u})\})} \prod_{i\in\mathcal{NZ}(\mathbf{u})} x_i^{c^{(l)}u_i} & \mathcal{NZ}(\mathbf{u}) = \mathcal{NZ}(\mathbf{x}) \\ 0 & o.w. \end{cases} \tag{2.27}$$

It is this representation that will be used going forward.

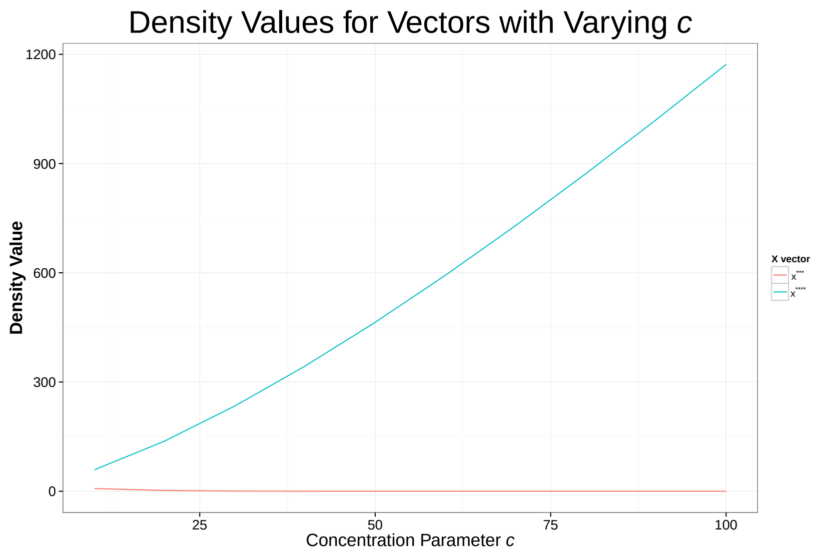


Figure 2.6: Density values for Example 2.18. The vector $x^{(l)****}$ clearly dominates $x^{(l)***}$, increasingly so as $c \to \infty$.

Finally, analogous to $\boldsymbol{U}$, let $\boldsymbol{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N_l)}\}$ be the collection of observed allele proportions across all loci and let $\mathbf{c} = \{c^{(l)}; l = 1, \ldots, N_l\}$ denote the collection of concentration parameters.

2.6.1.7 Recap of Variables, Constants, and Parameters

Table 2.3 summarizes the variables and constants discussed in this section.

Symbol	Description	Obs?	Type	Form	|?	Params
Z	MOI	N	variable	scalar	none	$\mathbf{p}$
$\mathbf{H}$	Indicator of haplotypes	N	variable	vector	Z	$\boldsymbol{\pi}$
$\mathbf{Y}$	Proportions for haplotypes	N	variable	vector	$\mathbf{H}$	ξ
$S^{(l)}$	Signature matrix	Y	constant	matrix	none	none
$\mathbf{U}^{(l)}$	Implied allele proportions	N	variable	vector	$S^{(l)}, \mathbf{Y}$	none
$\mathbf{X}^{(l)}$	Observed allele proportions	Y	variable	vector	$\mathbf{U}^{(l)}$	$c^{(l)}$

Table 2.3: Summary of Variables and Constants. "Obs?" states if observed. "|?" indicates any conditioning variable(s). $S^{(l)}, \mathbf{U}^{(l)}, \mathbf{X}^{(l)}$ exist for locus $l, l = 1, \ldots, N_l$.

Table 2.4 summarizes the parameters in the distributions of the random variables. All parameter values are unknown and will need to be estimated with the exception of ξ.

Symbol	Description	Form
$\mathbf{p}$	MOI population frequencies	Vector of length maxMOI
$\boldsymbol{\pi}$	Haplotype population frequencies	Vector of length N_{haps}, one entry per haplotype denoted π_i
$\mathbf{c}$	Concentration parameters	Vector of length N_l, one entry per locus denoted $c^{(l)}$
ξ	Constant pdf value for valid $\mathbf{Y}$ vectors	Scalar > 0

Table 2.4: Summary of Parameters. There are three primary parameters – $\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}$ – while ξ is a nuisance parameter.

2.6.2 Data Generation

In this section, we discuss the process by which the data are generated. This process is the same for all observations.

Figure 2.7 presents a high level overview of the process. The steps in the process correspond closely to the random variables discussed in §2.6.1. The diagram demonstrates the chain structure of the random variables in the ultimate production of a set of observed allele proportions for each locus.

Each step is discussed in more detail below.

2.6.2.1 Determine MOI

In this step, the individual is assigned an MOI according to the distribution outlined in §2.6.1.1. We assume that there is a standard parameter $\mathbf{p}$ valid for all observations. All observations are assigned an MOI in an independent and identical fashion. While it might be argued that the MOI is a characteristic of an individual's infection, we model it explicitly. It is convenient to start with this variable in the chain since other variables are naturally dependent on it.

2.6.2.2 Determine Haplotypes

In this step, the specific haplotypes infecting the individual are selected. The number of distinct haplotypes must be consistent with the MOI, hence the dependence in Figure 2.7

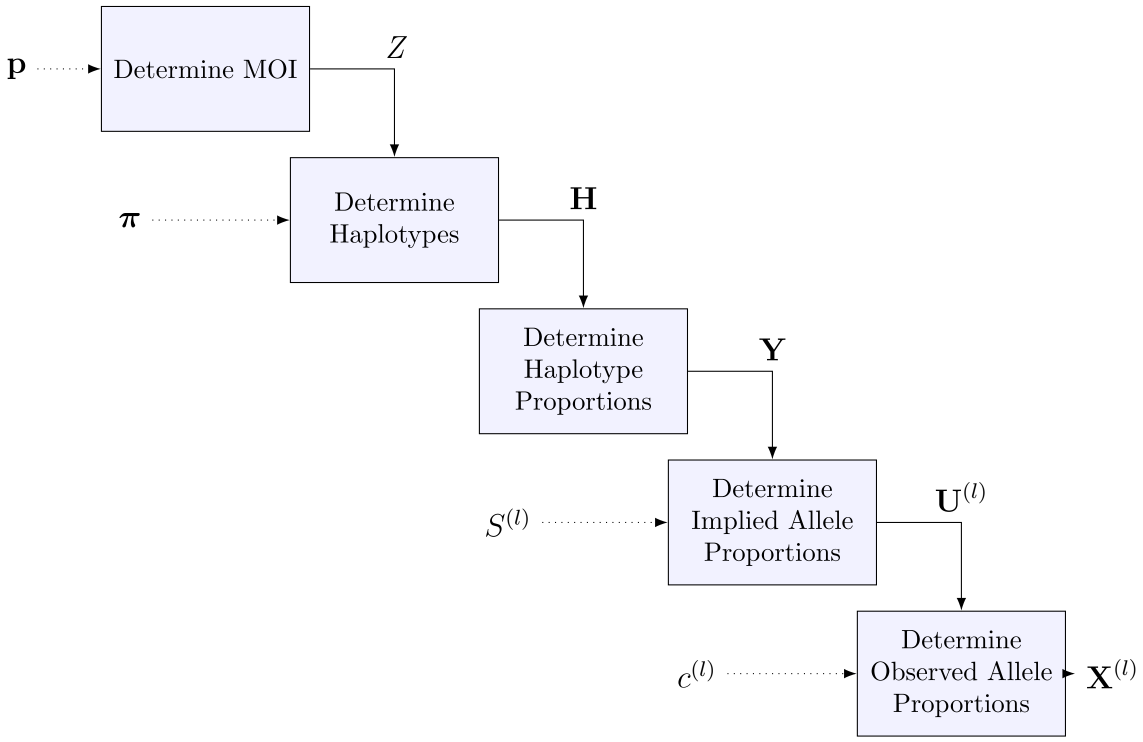


Figure 2.7: Data Generating Process for an Observation.Each step is represented by a box. Incoming parameters for a process are to the left of the relevant box with an incoming dotted arrow. Outgoing random variables are to the right of the relevant box next to the arrow leading in to the following process. Each process typically uses a combination of parameters and previously produced random variables. The process ends when $\mathbf{X}^{(l)}$, the observed allele proportions, are produced.

on the variable Z. The haplotypes are chosen according to how prevalent they are in the relevant geographic region, as captured by the haplotype population frequency vector $\boldsymbol{\pi}$ and described in §2.6.1.2.

2.6.2.3 Determine Haplotype Proportions

In this step, the proportions of the MOI distinct haplotypes in the individual are determined, as described in §2.6.1.3. A person infected infected with multiple strains (i.e., the case where MOI>1) may well see strains propagate at different rates and we have no basis on which to favor one strain over another, we assume that all proportion combinations are equally likely, with probability equal to ξ. The only requirement is that it is precisely the strains indicated in the random variable $\mathbf{H}$ that have non-zero proportions.

Since ξ does not influence the selection of haplotype proportions in this step, it does not appear in Figure 2.7.

2.6.2.4 Determine Implied Allele Proportions

In this step, we simply derive the implied allele proportions for each locus l in the form of $\mathbf{U}^{(l)}$, as described in §2.6.1.4 and §2.6.1.5. By "implied", we mean what would be observed based solely on the haplotypes chosen and their relative proportions and in the absence of any measurement error. Recall that $S^{(l)}$ is fixed and known; its only role is to generate $\mathbf{U}^{(l)}$ through matrix multiplication. Figure 2.7 presents $S^{(l)}$ as an input parameter to this process; while this is technically true, it is also true that it never varies and there is no need to estimate it. It is constructed from our knowledge of the entire set of haplotypes and loci in question.

2.6.2.5 Determine Observed Allele Proportions

In this final step, as discussed in §2.6.1.6, we derive the only variable that is actually observed: $\mathbf{X}^{(l)}$, the observed allele proportions at locus l. This is a function of two quantities: $\mathbf{U}_{\tau}^{(l)}$, the implied non-zero allele proportions and $c^{(l)}$, a concentration parameter that controls the error in the observed values around the implied values.

2.6.3 Data Likelihood

In this section, we discuss the likelihood function for the data. This is with the view to estimate certain parameters that will influence the likelihood of individual solutions.

2.6.3.1 Background

Recall that we are assuming a collection of blood samples, one from each person tested. As such, all the variables discussed in §2.6.1 and summarized in Table 2.3 – with the exception of $S^{(l)}$ which is a constant – need to be indexed for the individual in question. To do this, we add the subscript $n = 1, \ldots, N_{\text{samp}}$ to the variable for the n^{th} individual;

e.g., X_5^3 will denote the observed allele proportions for the 5th sample at the 3rd locus and Z_{10} will denote the MOI for the 10th observation.

2.6.3.2 Key Assumptions

We make the following key assumptions when deriving the likelihood equations:

1. The parameter $\mathbf{p}$ governing the choice of the MOI is constant and identical across all individuals.
2. The parameter $\boldsymbol{\pi}$ representing the haplotype population frequencies is constant and identical across all individuals. Put differently, there is a single set of haplotype frequencies that is independent from the individuals under consideration.
3. The parameter $c^{(l)}$ representing the concentration of oberved allele proportions around the mean at locus l is constant and identical across all individuals. It can, however, vary from locus to locus. Some loci will likely be "noisier" than others.
4. The loci have been chosen such that they are independent within each sample. More precisely, we assume that $\mathbf{U}_n^{(l_1)}$ and $\mathbf{U}_n^{(l_2)}$ are independent for sample n for all loci $l_1 \neq l_2$. The justification for this assumption is that the loci are scattered across all 14 chromosomes.
5. All variables are independently and identically distributed across all samples.

2.6.3.3 Joint Likelihood for One Sample

Each sample will have realized a value for each of the following random variables, in the following order:

1. Z_n
2. $\mathbf{H}_n$
3. $\mathbf{Y}_n$
4. $\boldsymbol{U}_n := \{\mathbf{U}_n^{(l)}; l = 1, \ldots, N_l\}$, determined entirely by $\mathbf{Y}_n$ (see §2.6.1.4)
5. $\boldsymbol{X}_n := \{\mathbf{X}_n^{(l)}; l = 1, \ldots, N_l\}$

where each of the last two variables is a set of variables with one instance occurring at each locus. This is depicted in Figure 2.8, which shows the dependence relationships among the variables for a single sample. We see two features of the data structure:

1. the pair of variables labeled $U_n^{(l)}, X_n^{(l)}$ is represented once for each locus; hence the stacking from $1, \ldots, N_l$
2. the variables $\mathbf{Y}_n$, $\boldsymbol{U}_n$ are all derived from the same single source – the haplotype proportions.

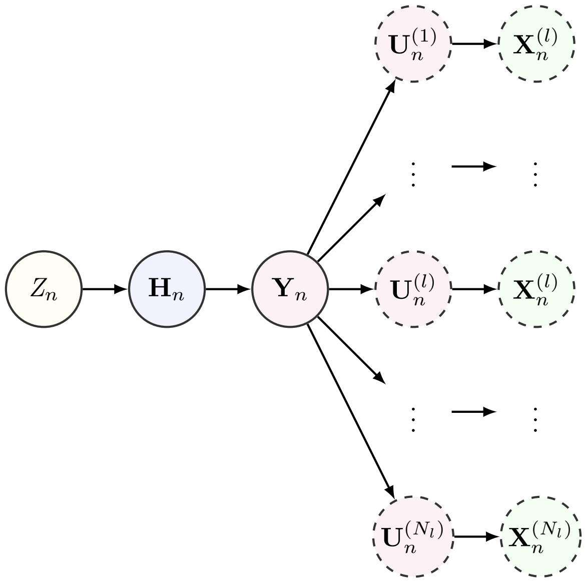


Figure 2.8: Graph of Variables for a Single Observation. Each variable is represented as a node with its name inside. The variables can be split into four groups, as indicated by the four node colors; in particular, the variables in light purple are all determined entirely by the haplotype proportions. Each arrow indicates the dependence of the second node on the first. The "loci variables" are represented by nodes with dotted borders, with each row representing the same pair of variables $(\mathbf{U}_n^l, \mathbf{X}_n^l)$ that occur together at locus l. The superscript in the variable name indicates to which locus it belongs.

Let

$$\Theta := \{\mathbf{p}; \boldsymbol{\pi}; \mathbf{c}\} \tag{2.28}$$

and consider the single sample n. The joint likelihood of all data, both observed and unobserved, for a single sample n is as follows:

$$\begin{aligned} &\mathbb{L}(\Theta | Z_n = z, \mathbf{H}_n = \mathbf{h}, \mathbf{Y}_n = \mathbf{y}, \boldsymbol{U}_n = \boldsymbol{u}, \boldsymbol{X}_n = \boldsymbol{x}) \\ &= \left[\prod_{l=1}^{N_l} f_{\mathbf{X}_n^{(l)} | \mathbf{U}_n^{(l)}}(\mathbf{x}, \mathbf{u}; \mathbf{c}) f_{\mathbf{U}_n^{(l)} | \mathbf{Y}_n}(\mathbf{u}, \mathbf{y}) \right] f_{\mathbf{Y}_n | \mathbf{H}_n}(\mathbf{y}, \mathbf{h}; \xi) f_{\mathbf{H} | Z_n}(\mathbf{h}, z; \boldsymbol{\pi}) f_{Z_n}(z; \mathbf{p}) \end{aligned} \tag{2.29}$$

2.6.3.4 Joint Likelihood for All Samples

Under the assumption that all variables above are distributed iid[18], the joint likelihood for all data in all samples,

$$\mathcal{C} = \{Z_n, \mathbf{H}_n, \mathbf{Y}_n, \boldsymbol{U}_n, \boldsymbol{X}_n\}_{n=1,\ldots,N_{\text{samp}}}, \tag{2.30}$$

is simply

$$\mathbb{L}(\mathcal{C}) = \prod_{n=1}^{N_{\text{samp}}} \left[\prod_{l=1}^{N_l} f_{\mathbf{X}_n^{(l)}|\mathbf{U}_n^{(l)}}(\mathbf{x}, \mathbf{u}; \mathbf{c}) f_{\mathbf{U}_n^{(l)}|\mathbf{Y}_n}(\mathbf{u}, \mathbf{y}) \right] f_{\mathbf{Y}_n|\mathbf{H}_n}(\mathbf{y}, \mathbf{h}; \xi) f_{\mathbf{H}_n|Z_n}(\mathbf{h}, z; \boldsymbol{\pi}) f_{Z_n}(z; \mathbf{p}). \tag{2.31}$$

We use the letter “C” in equation (2.30) to indicate that this represents the complete data, most of which is actually unobserved. This is discussed in further detail in §2.7.4.1.

2.7 Details of the `LITSE` Algorithm

In this section, we discuss the details of the Learning with Iteration and Tree-based Search Estimation (`LITSE`) algorithm. As the name implies, the algorithm is iterative in that it repeatedly re-estimates the underlying population parameters to refine the predictions for individual observations.

2.7.1 Overview

The algorithm is summarized in Algorithm 1 below.

[18]Note that all variables other than Z_n have *conditional* distributions.

```
LITSE() Summary
    Main Input(s)  : li, loci information; oap, observed allele proportions per
                     locus/observation; $\hat{\mathbf{p}}^{(0)}, \hat{\boldsymbol{\pi}}^{(0)}, \hat{\mathbf{c}}^{(0)}$, initial parameter estimates;
                     run parameters.
    Main Output(s): finalSols, proposed haplotype solutions per observation;
                     $\hat{\mathbf{p}}, \hat{\boldsymbol{\pi}}, \hat{\mathbf{c}}$, final parameter estimates.
1   Initialize all parameters and counts/totals.
2   Perform validity checks.
3   propSols $\leftarrow$ $SolutionSetIdentification$(li, oap).
4   propSols $\leftarrow$ $SolutionSetCompletion$(li, propSols).
5   [finalSols; $\hat{\mathbf{p}}, \hat{\boldsymbol{\pi}}, \hat{\mathbf{c}}$] $\leftarrow$ $SolutionSetRefinement$(li, propSols, $\hat{\mathbf{p}}^{(0)}, \hat{\boldsymbol{\pi}}^{(0)}, \hat{\mathbf{c}}^{(0)}$).
6   Return (finalSols; $\hat{\mathbf{p}}, \hat{\boldsymbol{\pi}}, \hat{\mathbf{c}}$).
```

Algorithm 1: Summary of LITSE Algorithm. Summary; details excluded.

The `LITSE` algorithm operates in three main steps:

1. **Solution Set Identification** (SSI). In this first step, depicted on line 3, the algorithm identifies candidate solutions for each observation individually based solely on the degree to which a proposed solution aligns with the observed allele proportions. In the case where an observation is missing observed allele data for a locus, all candidate solutions for that observation will be blank for the locus in question.
2. **Solution Set Completion** SSC. In this second step, depicted on line 4, the algorithm identifies and completes all partial solutions from the previous step with missing locus information.
3. **Solution Set Refinement** (SSR). In this third step, depicted on line 5, the algorithm iteratively computes the probability of each solution by computing the probability of all the components of each solution. The results returned include a set of proposed solutions for each observation and the probability of each, along with the computed parameter estimates.

As the names imply, the SSI procedure is a precursor to the SSR procedure. The SSI procedure determines a set of possible solutions for each individual observation in isolation, the SSC step imputes the allele composition for each solution missing data for one or more loci, and the SSR procedure refines features of these proposals.

The `LITSE` algorithm can be viewed as a tree-based search with a refinement through the Expectation-Maximization (EM) algorithm. While the SSR procedure, given its iterations of expectation and maximization sub-steps, can be more closely identified with the EM algorithm, the SSI and SSC steps can be viewed as precursors.

2.7.2 Solution Set Identification

The primary purpose of the SSI procedure is to identify per individual observation but without regard to the results obtained for other observations, a set of candidate solutions $\mathcal{S}_n^f$ for an observation as described in §2.7.4.1.5 and Table 2.10. Members of this set will be those whose solution error, as discussed below, is low.

The fundamental task of the SSI procedure, discussed in detail below, is to search through potential solutions for the observation and to structure this solution space as a tree. Each level in the tree processes a single locus, with the order in which the loci are processed determined beforehand. Thus the total number of levels in the tree equals $N_l + 1$, counting a root node from which the search begins.

Consistent with the form a solution takes as outlined in §2.4.1.3 – a set of ordered columns, one per locus – the SSI procedure builds a solution by starting out with a "blank" solution (i.e., one whose number of rows equals the assumed MOI and with one column per locus). It progressively fills in columns, one by one, in a predetermined order. Since there will typically be many feasible vectors for a given column, the process of filling in the columns can be viewed as a search, organized in a tree data structure, whereby each node represents a partial solution that "inherits" a presumed set of alleles for previously completed loci and is now considering a configuration for the locus at its level.

This section introduces the definition of the error, gives an overview of the composite steps, and gives details on some of the most involved steps, in particular the tree search for solutions.

2.7.2.1 Summary

The component steps of the SSI procedure are specified in Algorithm 2, with each line explained below.

```
LITSE Solution Set Identification() Procedure
    Main Input(s)  : li, loci information; oap, observed allele proportions per
                     locus/observation; run parameters incl smnc, hmnc, mfl, and mer.
    Main Output(s): propSols, proposed haplotype solutions per observation/MOI.
1   Initialize all variables.
2   Preprocess input data.
3   for $n = 1$ to $N_{samp}$ do
4       lblocks ← GetLocusBlockOrder($\texttt{oap}_n$, li).
5       Identify set $\texttt{MOIs}_n$ to explore using am.
6       for $z \in \texttt{MOIs}_n$ do
7           [combs, perms] ← GetCombsPerms($\texttt{oap}_n, z$).
8           propSols ← SearchSolSpace($\texttt{oap}_n$, li, lblocks, combs, perms, $z$, run pars).
9           Remove infeasible solutions from propSols.
10          Assess MOI and break if $\min_z(error)/\min_{z-1}(error) >$ mer.
11  Return (propSols) [for all observations].
```

Algorithm 2: LITSE Solution Set Identification Procedure. Summary; details excluded.

Lines 1 and 2 perform standard validation and initialization.

Line 4 analyzes all loci for the observation and determines the order in which the loci will be processed in the tree; it is discussed in more detail in §2.7.2.2.

Line 5 computes the range of MOIs that will be attempted for the observation. We first determine the minimum value minMOI_n for z, as described in §2.7.4.2.3 and illustrated in Example 2.29. The range is then computed as $\{\text{minMOI}_n, \ldots, \text{minMOI}_n + \texttt{am}\}$, where `am` is discussed in Table 2.5.

Line 7 determines the combinations and permutations used at each level to construct the tree; it is discussed in more detail in §2.7.2.3.

Line 8 represents the most important step in the procedure, the actual tree search, and is discussed in detail in §2.7.2.4. Three key parameters are `smnc`, `hmnc`, and `mfl`.

Finally, in line 9 the procedure removes any infeasible solutions that may have been produced during the tree search. In line 10, the entire MOI is assessed for the observation with one of two outcomes: (1) the MOI is deemed invalid or incorrect, all solutions pertaining to the MOI are dropped, and no further MOIs are investigated for the observation (i.e., the algorithm breaks out of the loop) or (2) the MOI is deemed valid, all solutions are retained, and the next MOI is attempted (assuming this is permitted by the `am` parameter).

Among the inputs to the SSI step, there are five run time parameters or "settings" – not to be confused with the population-level parameters $\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}$ to be estimated – that control the precise functioning of the SSI procedure. These are presented in Table 2.5 below. The first parameter will be used to control the width of the search tree search, as

Parameter	Description
`smnc`	the "soft" maximum number of nodes, after all nodes for a level have been placed in ascending order by error, chosen at one level to spawn children at the next level. This parameter is used to control the size of the search tree and is discussed in detail in §2.7.2.4.2.
`hmnc`	the corresponding "hard" maximum number of nodes chosen at one level to spawn children at the next level. This parameter is related to the one above and is also discussed in detail in §2.7.2.4.2.
`mfl`	the maximum floating level (the lowest level in the tree at which proportions for partial solutions will be recomputed, after which the proportions for all descendants remain fixed).
`am`	the additional MOI increment. This is the maximum over the minimum feasible MOI for an observation for which the LITSE algorithm is applied. For example, a value of 2 indicates that an observation with $\text{minMOI}_n = k$ will be checked at assumed MOIs of $k, k+1, k+2$.
`mer`	the maximum error ratio. This is the maximum permissible value for the ratio of the lowest error among all solutions for one MOI versus the equivalent value for the preceding MOI. Any ratio above this maximum will result in `LITSE`dropping the MOI for the observation and ceasing to explore higher MOIs. Discussed in detail in §2.7.2.5.

Table 2.5: Key Solution Set IdentificationParameters.

discussed in §2.7.2.4 below, while the second is used to determine the number of different MOIs to try. As either increases, the breadth of the search widens and the runtime increases.

The remainder of section §2.7.2 describes three key steps – ordering the loci, determining the combinations/permutations and performing the tree search – in greater detail.

2.7.2.2 Get Locus Blocks

In this step, the procedure groups the alleles at each locus by their number of non-zero alleles and creates a sorted list of loci numbers. The underlying idea is that when we create the search tree, with one locus per level, we want to start with the longest loci and proceed in descending fashion to those with the fewest non-zero alleles (perhaps just one allele if that is all that is observed at a particular locus). We group the loci first into *blocks*, sort within a block when the number of members is greater than two based on a hierarchical clustering procedure in which we first take those loci that are "closest" to one another in the group, and produce an ordered list of loci.

Example 2.19 (Computing Locus Blocks and Order). Consider the case where there are eight loci with the following observed allele proportions, at first sorted within each locus by allele label.

Locus

1	2	3	4	5	6	7	8
$A_{1,2}$:0.30	$A_{2,2}$:0.20	$A_{3,2}$:1.00	$A_{4,1}$:0.10	$A_{5,3}$:0.19	$A_{6,2}$:0.35	$A_{7,3}$:0.70	$A_{8,2}$:0.15
$A_{1,3}$:0.39	$A_{2,4}$:0.16		$A_{4,4}$:0.81	$A_{5,4}$:0.40	$A_{6,3}$:0.35	$A_{7,4}$:0.20	$A_{8,4}$:0.85
$A_{1,6}$:0.20	$A_{2,5}$:0.29		$A_{4,5}$:0.09	$A_{5,5}$:0.30	$A_{6,5}$:0.30	$A_{7,7}$:0.10	
$A_{1,9}$:0.11	$A_{2,6}$:0.11			$A_{5,7}$:0.11			
	$A_{2,9}$:0.24						

As a first step, group the loci according to their number of non-zero alleles into blocks and order the locis in each block by descending proportion. Each shade represents a different block.

Locus

1	2	3	4	5	6	7	8
$A_{1,3}$:0.39	$A_{2,5}$:0.29	$A_{3,2}$:1.00	$A_{4,4}$:0.81	$A_{5,4}$:0.40	$A_{6,2}$:0.35	$A_{7,3}$:0.70	$A_{8,4}$:0.85
$A_{1,2}$:0.30	$A_{2,9}$:0.24		$A_{4,1}$:0.10	$A_{5,5}$:0.30	$A_{6,3}$:0.35	$A_{7,4}$:0.20	$A_{8,2}$:0.15
$A_{1,6}$:0.20	$A_{2,2}$:0.20		$A_{4,5}$:0.09	$A_{5,3}$:0.19	$A_{6,5}$:0.30	$A_{7,7}$:0.10	
$A_{1,9}$:0.11	$A_{2,4}$:0.16			$A_{5,7}$:0.11			
	$A_{2,6}$:0.11						

Here, locus 2 alone belongs to the first block (composed of all loci with length 5), loci 1 and 5 belong to the next block (length 4), and so on.

Thus we know we will first process the block of length 5, then of length 4, then of 3, 2, and 1, in that order. It remains to determine the order within each block with more than one member. For blocks with two members, such as the block including loci 1 and 5, we choose arbitrarily as first and let the other be second. For blocks with more than two members, such as the block including loci 4, 6, and 7, we perform hierarchical clustering and first choose the closest pair, then the third member that is closest to this first pair, and so on. In this example loci 4 and 7 are the closest of the three possible pairs, so they are included first (in either order) for the block. Locus 6 is the remainder, so it comes last in the block.

Putting this all together, the ordered loci list is $\{2, 1, 5, 4, 7, 6, 8, 3\}$. Thus locus 2 will be processed at the first level and locus 3 will be processed at the eighth level, will all other loci processed in between.

□

The ordered list is used later in the tree building step.

2.7.2.3 Identify Combinations and Permutations

As indicated in Algorithm 2, this step takes as input the observed alleles and the presumed MOI z and generates all *feasible* columns for each locus of length z. A column is feasible for a locus if it meets the following two conditions:

1. it has length z, and
2. every allele with non-zero proportion has at least one entry

The actual observed proportions for the alleles are irrelevant for this purpose; what matters is simply the set of alleles.

In addition, we make a distinction between combinations and permutations. In the former, the position of an entry does not matter, while in the latter it does. We will start out choosing a combination for a particular locus and then insert permutations in the solution columns for the remaining loci.

Example 2.20 (Feasible Locus Column). Assume that for a given locus l the following three alleles are observed with non-zero proportions: $\{A_{l,1}, A_{l,3}, A_{l,7}\}$.

Now assume that $z = 3$. Then there is only one feasible combination column,

$$\begin{pmatrix} A_{l,1} \\ A_{l,3} \\ A_{l,7} \end{pmatrix}$$

and there are six feasible permutation columns,

$$\begin{pmatrix} A_{l,1} \\ A_{l,3} \\ A_{l,7} \end{pmatrix}, \begin{pmatrix} A_{l,1} \\ A_{l,7} \\ A_{l,3} \end{pmatrix}, \begin{pmatrix} A_{l,3} \\ A_{l,1} \\ A_{l,7} \end{pmatrix}, \begin{pmatrix} A_{l,3} \\ A_{l,7} \\ A_{l,1} \end{pmatrix}, \begin{pmatrix} A_{l,7} \\ A_{l,1} \\ A_{l,3} \end{pmatrix}, \begin{pmatrix} A_{l,7} \\ A_{l,3} \\ A_{l,1} \end{pmatrix}$$

The following columns are infeasible,

$$\begin{pmatrix} A_{l,1} & A_{l,3} & A_{l,1} \end{pmatrix}^T, \begin{pmatrix} A_{l,1} & A_{l,3} & A_{l,7} & A_{l,1} \end{pmatrix}^T,$$

the first because it is missing an allele and the second because it is incompatible with $z = 3$.

If, instead, $z = 4$, then there are three feasible combination vectors,

$$\begin{pmatrix} A_{l,1} \\ A_{l,3} \\ A_{l,7} \\ A_{1,1} \end{pmatrix}, \begin{pmatrix} A_{l,1} \\ A_{l,3} \\ A_{l,7} \\ A_{1,3} \end{pmatrix}, \begin{pmatrix} A_{l,1} \\ A_{l,3} \\ A_{l,7} \\ A_{1,7} \end{pmatrix},$$

and 36 different feasible permutation columns.

□

Note in particular that the number of feasible columns – combinations or permutations – grows exponentially with z. In the example above, with the same set of observed alleles, the number of combinations tripled and the number of permutations grew sixfold when moving from $z = 3$ to $z = 4$. This property is discussed in more detail in the next section.

The combinations and permutations are computed once per locus for a given MOI. Each set forms of a collection of possible entries in the locus column of interest.

2.7.2.4 Search Solution Space

This step, the most computationally intensive of the algorithm, explores the solution space for an observation and assumed MOI for that observation. By *solution space*, we mean the set of all possible feasible solutions of a given assumed MOI.

The step adopts a tree search approach whereby each solution, partial or complete, is represented by a node and each level in the tree represents the processing of a given locus. This process is illustrated in Figure 2.9. Here, all nodes other than the leaves

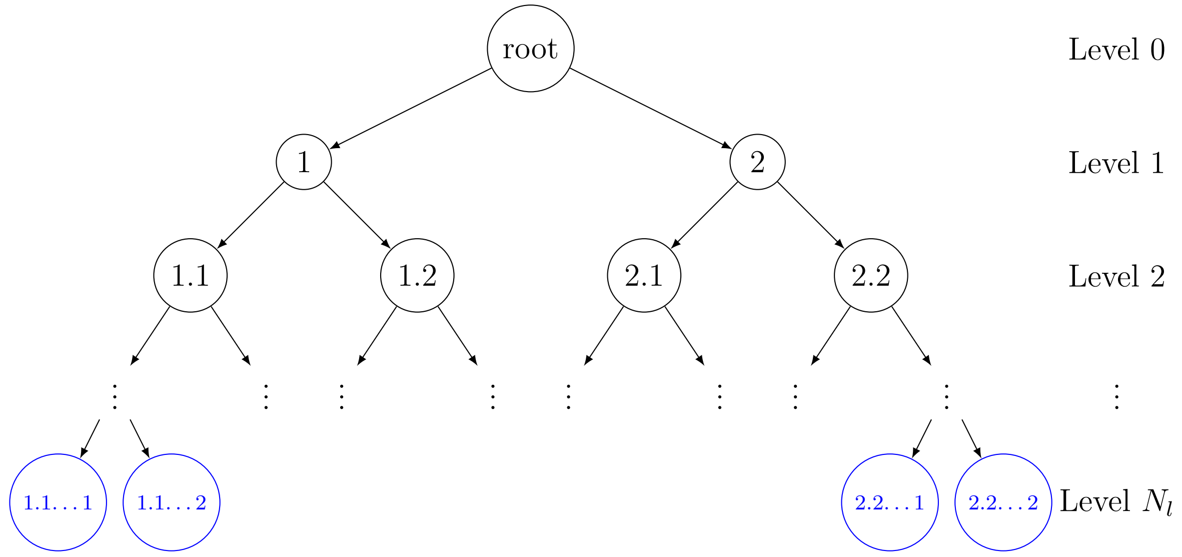


Figure 2.9: Diagram of Search Tree. The tree begins at the top with a root node, indicating an empty solution. The tree is built layer by layer starting at level 0, then level 1, and so on down to level N_l. Each node not at level Nl represents a partial solution, which inherits (except for the root) a partial solution from its parent and furthers the construction of the solution by proposing a feasible combination or permutation vector for the relevant locus. The nodes at level N_l, outlined in blue, are leaves in the tree and represent complete solutions; i.e., all loci columns have been filled. The edges represent the inheritance of a partial solution from a parent. In this illustration, the tree is binary. This is a specific case; in general, the number of children will vary by node.

represent partial solutions and all leaves represent final solutions. The step begins with a a root node at the top with an empty solution. It then descends level-by-level, each level representing the processing of a particular locus, and with each node having a parent from which it inherits a partial solution. This inherited partial solution is augmented with a proposed feasible combination or permutation vector (see §2.7.2.3) to construct a solution one locus closer to completion. Thus each edge from one node to another represents the addition of a feasible column for the corresponding locus in the solution matrix. This implies that the number of outbound edges from a node is equal to the number of feasible columns for that locus (e.g., 36 in Example 2.20).

Example 2.21 (Tree Search Path). Consider this sample path, one of many, through the first few levels of the tree presented in Figure 2.9 in the case where the assumed MOI equals 4,

ID	Loc_1	Loc_2	Loc_3	Loc_4	Loc_5	...	Loc_{N_l}	Prop
h_1	$A_{1,?}$	$A_{2,?}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	?
h_2	$A_{1,?}$	$A_{2,?}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	?
h_3	$A_{1,?}$	$A_{2,?}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	?
h_4	$A_{1,?}$	$A_{2,?}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	?

▼

ID	Loc_1	Loc_2	Loc_3	Loc_4	Loc_5	...	Loc_{N_l}	Prop
h_1	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	0.30
h_2	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	0.23
h_3	$A_{1,?}$	$A_{2,3}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	0.43
h_4	$A_{1,?}$	$A_{2,5}$	$A_{3,?}$	$A_{4,?}$	$A_{5,?}$	...	$A_{N_l,?}$	0.04

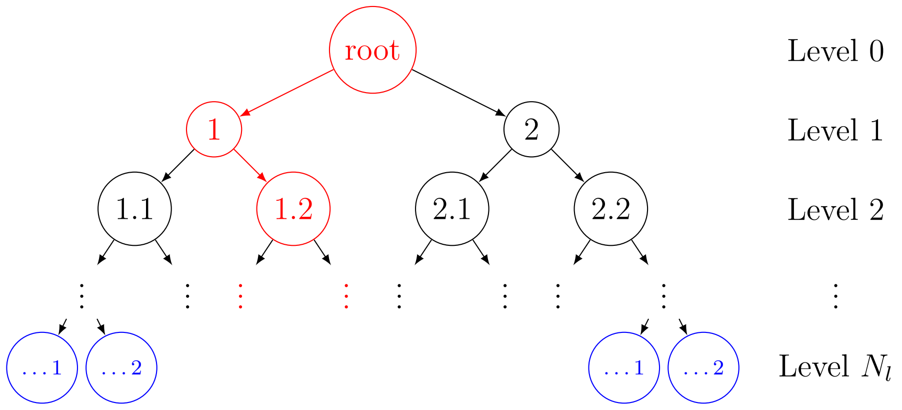
▼

ID	Loc_1	Loc_2	Loc_3	Loc_4	Loc_5	...	Loc_{N_l}	Prop
h_1	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,?}$	$A_{5,1}$	...	$A_{N_l,?}$	0.31
h_2	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,?}$	$A_{5,3}$	...	$A_{N_l,?}$	0.25
h_3	$A_{1,?}$	$A_{2,3}$	$A_{3,?}$	$A_{4,?}$	$A_{5,4}$	...	$A_{N_l,?}$	0.39
h_4	$A_{1,?}$	$A_{2,5}$	$A_{3,?}$	$A_{4,?}$	$A_{5,5}$	...	$A_{N_l,?}$	0.05

▼

...

corresponding to the following path in the tree,



as indicated in red. Note that in this example, the first locus processed was 2 and the second was 5. This is evident by looking at which column has been newly completed at each level. Also, the proposed proportions were tentatively set at level 1 but were then refined at the next level. The fixing of proportions is discussed in more detail below.

□

As stated previously, the number of outbound branches of a node – equivalent to the number of feasible solutions for the corresponding locus – grows exponentially with several quantities. This is stated formally below as a proposition, illustrated in more detail, and then proven in §2.A.

Proposition 1 (Outbound Node Degree). Let m denote the assumed MOI for an observation and let n denote the number of unique observed alleles for locus l (i.e., the "locus length") at level v in the search tree. Then the number of outbound edges from each node at level $v-1$ to level v equals

$$\mathcal{C}(m,n) = \binom{m-1}{m-n}, \; m \geq n,$$

if the algorithm is considering *combinations* for locus l and

$$\mathcal{P}(m,n) = \sum_{j=0}^{n-1} (-1)^j \binom{n}{j} (n-j)^m, \; m \geq n,$$

if the algorithm is considering *permutations* for locus l.

□

Recall that by construction, $m \geq n$. Furthermore, the algorithm considers combinations only when branching from the root node to level 1. For all other levels, it considers permutations.

Given this result, it is clear that the number of combinations, and particularly permutations, grows exponentially with m and n. Table 2.6 presents the number of outbound edges for various combinations of m and n, while 2.7 does the same for permutations.

	m									
n	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2		1	2	3	4	5	6	7	8	9
3			1	3	6	10	15	21	28	36
4				1	4	10	20	35	56	84
5					1	5	15	35	70	126
6						1	6	21	56	126
7							1	7	28	84
8								1	8	36
9									1	9
10										1

Table 2.6: Outbound Node Degree for Combinations. The number of valid combinations for each value of m (MOI) and n (number of distinct alleles) is presented. The results form a Pascal triangle.

These values were derived using the equations in Proposition 1 and confirmed by generating all combinations and permutations for each combination of m, n and counting only valid combinations and permutations.

As a concrete illustration of how the number of nodes in a tree expands as the number of levels (corresponding to loci) and MOI (corresponding to the length of the combination and permutation vectors) vary, consider Figure 2.10 below, generated by constructing unrestricted trees in `LITSE` for a variety of values for number of loci and MOI. By "unrestricted", we mean in the absence of any control on the branching from one level in the search tree to the next. In Figure 2.10, it is evident that as any one of the three factors increases, the number of nodes grows exponentially. Indeed, the values grow so quickly that it is problematic to observe the tree sizes for lower factor values when plotted

	m									
n	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2		2	6	14	30	62	126	254	510	1022
3			6	36	150	540	1806	5796	18150	55980
4				24	240	1560	8400	40824	186480	818520
5					120	1800	16800	126000	834120	5103000
6						720	15120	191520	1905120	16435440
7							5040	141120	2328480	29635200
8								40320	1451520	30240000
9									362880	16329600
10										3628800

Table 2.7: Outbound Node Degree for Permutations. The number of valid permutations for each value of m (MOI) and n (number of distinct alleles) is presented.

together with higher values. This behavior is the primary justification for introducing controls on the branching of the tree, as discussed in detail in §2.7.2.4.2.

For each node, whether an interim node with a partial solution or a leaf node with a final solution, the solution error is computed as described in §2.7.2.4.1. Recall that the computation of the error involves the corresponding S matrix for the solution. Since this S matrix is a representation of the chosen allele at each locus for a haplotype, and by definition a partial solution has one or more locus columns blank, the sections in the matrix relevant to the unprocessed loci will be all zeros.

2.7.2.4.1 Solution Error

The primary method for assessing a solution $\widehat{Sol}_{n,j}$, in the absence of any likelihood data regarding the MOI chosen or haplotypes involved, is the *solution error* denoted as $\mathrm{err}(\widehat{Sol}_{n,j})$. This error is the mean squared error, defined as follows,

$$\mathrm{err}(\widehat{Sol}_{n,j}) = \frac{1}{\sum_l \ell(l)} \left\| S\mathbf{y}(\widehat{Sol}_{n,j}) - \mathbf{x} \right\|_2^2 \tag{2.32}$$

where S is the set of stacked matrices $S^{(l)}$, $\mathbf{x}$ is the set of stacked observed allele proportions $\mathbf{x}^{(l)}$ by the same set of loci, and $\ell(l)$ is the number of alleles in locus l and $\sum_l \ell(l)$ is the count of all possible alleles. This definition simply takes the L_2 norm of the difference

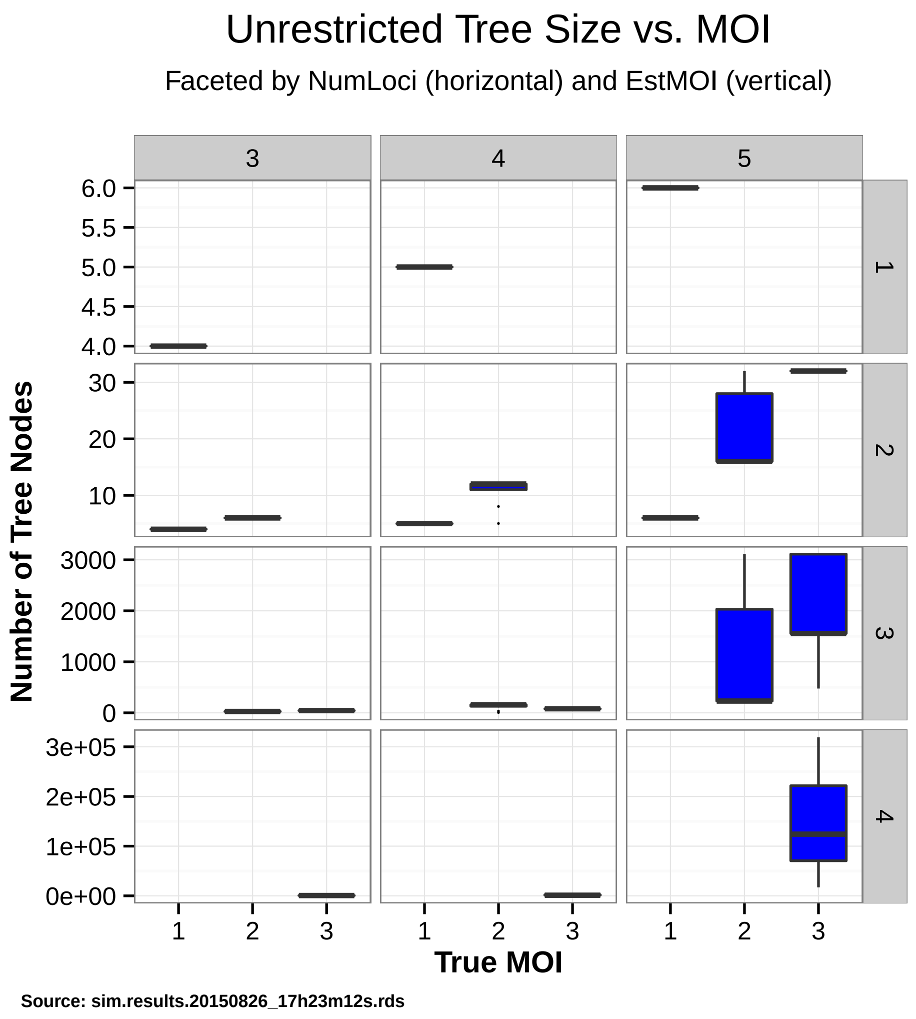


Figure 2.10: Unrestricted Treesize versus MOI. Each panel presents the distribution of the unrestricted number of nodes in a tree for different values of true MOI. The panels are faceted by number of loci (horizontally) and estimated MOI (vertically).

between the implied allele proportions and the observed allele proportions, each in the form of a column vector of length $\sum_l \ell(l)$, squares the entries, sums the squares, and then takes the average.

The justification for using the solution error is that it compares a proposed solution to the only observed data we have, namely, the observed allele proportions and measures the compatibility between the two.

Example 2.22 (Solution Errors). Assume there are three loci, each with three possible alleles labeled $A_{l,\{1,2,3\}}$, and we have the following observed allele proportions,

$$\mathbf{x} = \begin{pmatrix} 0 & 0.2 & 0.8 & | & 0.3 & 0.7 & 0 & | & 0 & 0.4 & 0.6 \end{pmatrix}^T$$

where the vertical bars simply separate the proportions belonging to a particular locus from those of other loci.

Now consider the several proposed solutions. First, consider

$$\widehat{Sol}_{n,j} = \begin{array}{c|ccc|r} \text{ID} & \text{Loc}_1 & \text{Loc}_2 & \text{Loc}_3 & \text{Prop} \\ \hline h_1 & A_{1,2} & A_{2,1} & A_{3,3} & 0.2 \\ h_2 & A_{1,2} & A_{2,1} & A_{3,2} & 0.3 \\ h_3 & A_{1,3} & A_{2,2} & A_{3,3} & 0.5 \end{array}$$

The S matrix corresponding to this solution is the following,

$$S^T = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Thus

$$\begin{aligned} S\mathbf{y}(\widehat{Sol}_{n,j}) - \mathbf{x} &= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.2 \\ 0.3 \\ 0.5 \end{pmatrix} - \begin{pmatrix} 0 \\ 0.2 \\ 0.8 \\ 0.3 \\ 0.7 \\ 0 \\ 0 \\ 0.4 \\ 0.6 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0.5 & 0.5 & 0.5 & 0.5 & 0 & 0 & 0.3 & 0.7 \end{pmatrix}^T - \\ &\quad \begin{pmatrix} 0 & 0.2 & 0.8 & 0.3 & 0.7 & 0 & 0 & 0.4 & 0.6 \end{pmatrix}^T \\ &= \begin{pmatrix} 0 & 0.3 & -0.3 & 0.2 & -0.2 & 0 & 0 & -0.1 & 0.1 \end{pmatrix}^T \end{aligned}$$

and

$$\begin{aligned}\mathrm{err}(\widehat{Sol}_{n,j}) &= (0^2 + 0.3^2 + (-0.3)^2 + 0.2^2 + (-0.2)^2 + 0^2 + 0^2 + (-0.1)^2 + 0.1^2)/9 \\ &= 0.28/9 = 0.0311\end{aligned}$$

Second, consider another solution,

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Prop
h_1	$A_{1,2}$	$A_{2,2}$	$A_{3,3}$	0.25
h_2	$A_{1,3}$	$A_{2,1}$	$A_{3,2}$	0.25
h_3	$A_{1,3}$	$A_{2,2}$	$A_{3,2}$	0.2
h_4	$A_{1,3}$	$A_{2,2}$	$A_{3,3}$	0.3

Now,

$$\begin{aligned}S\mathbf{y}(\widehat{Sol}_{n,j}) - \mathbf{x} &= \begin{pmatrix}0 & 0.25 & 0.75 & 0.25 & 0.75 & 0 & 0 & 0.45 & 0.55\end{pmatrix}^T - \\ &\quad \begin{pmatrix}0 & 0.2 & 0.8 & 0.3 & 0.7 & 0 & 0 & 0.4 & 0.6\end{pmatrix}^T \\ &= \begin{pmatrix}0 & 0.05 & -0.05 & -0.05 & 0.05 & 0 & 0 & 0.05 & -0.05\end{pmatrix}^T\end{aligned}$$

and it follows that $\mathrm{err}(\widehat{Sol}_{n,j}) = 0.015/9 = 0.00167$. This second solution was clearly a better one in terms of the implied fit to the observed allele proportions, as witnessed in the error.

Third, consider the proposed solution,

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Prop
h_1	$A_{1,2}$	$A_{2,1}$	$A_{3,2}$	0.2
h_2	$A_{1,3}$	$A_{2,1}$	$A_{3,2}$	0.1
h_3	$A_{1,3}$	$A_{2,2}$	$A_{3,2}$	0.1
h_4	$A_{1,3}$	$A_{2,2}$	$A_{3,3}$	0.6

In this case,

$$S\mathbf{y}(\widehat{Sol}_{n,j}) = \begin{pmatrix}0 & 0.2 & 0.8 & 0.3 & 0.7 & 0 & 0 & 0.4 & 0.6\end{pmatrix}^T,$$

which is the same as the observed allele proportions $\mathbf{x}$, so $\mathrm{err}(\widehat{Sol}_{n,j}) = 0$.

Finally, consider the following solution, different from the third,

$\widehat{Sol}_{n,j} =$

ID	Loc$_1$	Loc$_2$	Loc$_3$	Prop
h_1	$A_{1,2}$	$A_{2,2}$	$A_{3,3}$	0.2
h_2	$A_{1,3}$	$A_{2,1}$	$A_{3,2}$	0.3
h_3	$A_{1,3}$	$A_{2,2}$	$A_{3,2}$	0.1
h_4	$A_{1,3}$	$A_{2,2}$	$A_{3,3}$	0.4

Here again, $S\mathbf{y}(\widehat{Sol}_{n,j}) = \mathbf{x}$, and thus $\text{err}(\widehat{Sol}_{n,j}) = 0$. There are two differences between this and the previous solution: the first haplotype has changed slightly and the proportions have changed. Since there can be only one true solution, a maximum of one (and perhaps none) of these solutions can be correct.

$\square$

The last case in Example 2.22 demonstrates an important point: a solution with an error of zero is not necessarily the correct one. In the absence of measurement error, the converse is true; yet in the presence of measurement error, the converse is also false: a correct solution's implied allele proportions may not equal those observed.

On the basis of the solution error alone, there is nothing to recommend either the third or the fourth solution over the other. However, when the probabilistic framework from §2.7.4.2.4 is applied in the SSR procedure, the presence of different haplotypes in the solutions will lead (assuming different population frequencies for the differing haplotypes) to different probabilities for the two solutions.

Example 2.23 (Computation of Error for Partial Solution). In Example 2.21, node 1.2 corresponds to the following partial solution,

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Loc$_5$	…	Loc$_{N_l}$	Prop
h_1	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,?}$	$A_{5,1}$	…	$A_{N_l,?}$	0.31
h_2	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,?}$	$A_{5,3}$	…	$A_{N_l,?}$	0.25
h_3	$A_{1,?}$	$A_{2,3}$	$A_{3,?}$	$A_{4,?}$	$A_{5,4}$	…	$A_{N_l,?}$	0.39
h_4	$A_{1,?}$	$A_{2,5}$	$A_{3,?}$	$A_{4,?}$	$A_{5,5}$	…	$A_{N_l,?}$	0.05

Here only two of the N_l loci have been filled in; all others are blank. Assume further that $N_l = 6, \ell(l) = 5$ for all loci, and that the observed allele proportions are as follows,

$$\mathbf{x} = \big[0\ 0.2\ 0.8\ 0\ 0 \mid 0.3\ 0.7\ 0\ 0\ 0 \mid 0\ 0.4\ 0.6\ 0\ 0 \mid \\ 0\ 0.1\ 0.9\ 0\ 0 \mid 0\ 0.7\ 0.2\ 0.1\ 0 \mid 0\ 0\ 0.3\ 0.4\ 0.3\big]^T$$

Then if S is the relevant signature matrix and $\widehat{Sol}_{n,j}$ denotes the partial solution, the error vector is,

$$S\mathbf{y}(\widehat{Sol}_{n,j}) \quad - \quad \mathbf{x} \quad = \quad \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\ \hline 0&0&0&0\\1&1&0&0\\0&0&1&0\\0&0&0&0\\0&0&0&1\\ \hline 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\ \hline 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\ \hline 1&0&0&0\\0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\\ \hline 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \begin{pmatrix} 0.31\\0.25\\0.39\\0.05 \end{pmatrix} - \begin{pmatrix} 0\\0.2\\0.8\\0\\0\\ \hline 0.3\\0.7\\0\\0\\0\\ \hline 0\\0.4\\0.6\\0\\0\\ \hline 0\\0.1\\0.9\\0\\0\\ \hline 0\\0.7\\0.2\\0.1\\0\\ \hline 0\\0\\0.3\\0.4\\0.3 \end{pmatrix} \quad = \quad \begin{pmatrix} 0\\-0.2\\-0.8\\0\\0\\ \hline -0.3\\-0.14\\0.39\\0\\0.05\\ \hline 0\\-0.4\\-0.6\\0\\0\\ \hline 0\\-0.1\\-0.9\\0\\0\\ \hline 0.31\\-0.7\\0.05\\0.29\\0.05\\ \hline 0\\0\\-0.3\\-0.4\\-0.3 \end{pmatrix}$$

and thus $\text{err}(\widehat{Sol}_{n,j}) = 3.2994$. Note in particular that the S matrix for the solution contains all zeros for the loci (1,3,4,6) that have not yet been processed in the tree. All nodes at the same level will have the same zeroed locus blocks in the S matrix.

□

The details of the SSS routine are presented in Algorithm 3. There are several details worth noting in this procedure.

2.7.2.4.2 Branching Control for a Heuristic Search

On lines 3 and 25 in Algorithm 3, the routine identifies the top node(s) at the current level in the tree (either root or elsewhere) deemed most likely to lead to a low error final solution.[19] This is to address the issue discussed in §2.7.4.1.5 and Example 2.30: if left unbound, the number of potential solutions grows exponentially as z or N_l increases. The solution is to adopt a heuristic search approach that limits the number of solutions considered. Here, the error of the *partial* solution at a particular level is used as the heuristic, an indicator that the error of the final solutions below this node in the tree will also have low errors. We identify two parameters, $\texttt{smnc}, \texttt{hmnc} \in \mathbb{N}, \texttt{smnc} \leq \texttt{hmnc}$ (soft and hard "maximum nodes chosen"), to control how many nodes will have children in the next level. All nodes for a level are first ordered by error in ascending order. Note that there may be ties in this ordering. The procedure then steps through the ordered list and chooses each node until one of the following happens:

1. all nodes at the level have been chosen, or
2. at least `smnc`, but no more than `hmnc`, nodes have been chosen and the next node has an error strictly greater than that of each of the chosen nodes.[20]

The parameters `smnc` and `hmnc` work together in such a way that `smnc` sets a general goal regarding the number of nodes desired, allowing all members of ties of sufficiently low error to be included while `hmnc` ensures that in cases where there are many members in such ties the final number of accepted nodes remains reasonable. Thus the parameter `smnc` will typically be lower than, but never greater than, the parameter `hmnc`. Once `hmnc` nodes have been chosen, no further nodes will be selected, even if their error is equal to that of one of the selected nodes. The parameter `hmnc` should thus be set to a

[19]In the case of the root node, this is chosen automatically as by definition it is the only node at its level.

[20]As shown in Example 2.24, while nodes are processed in order of increasing error, *within* a series of nodes of the same error the order is arbitrary. Thus the parameter `hmnc`may eliminate nodes that were simply arbitrarily placed late in a queue of nodes of identical error. As long as `hmnc`is set high enough, such eliminated nodes will have a small chance of leading to a correct solution.

```
Search Solution Space() Routine
    Main Input(s) : oap_n, li, lblocks, combs, perms, $z$, smnc, hmnc.
    Main Output(s): propSols for $z$.
1   Create root node with empty solution.
2   Compute root node solution error vs. oap_n for reference.
3   Mark root node as chosen.
4   Create lociOrdered from lblocks.
5   Set fixedProps to FALSE.
6   Initialize set cumAlleles of all allele proportions so far.
7   for $l = 1, \ldots, N_l$ do
8       Identify current locus from lociOrdered($l$).
9       Identify chosen nodes chosenParNodes from level $l - 1$.
10      if !fixedProps then
11          Add locus alleles to cumAlleles.
12      if $l == 1$ then
13          configs ← combs.
14      else
15          configs ← perms.
16      for chosenNode in chosenParNodes do
17          for config in configs do
18              Create node at level $l$, establish parent-child relationship, inherit solution
                from chosenNode and set current locus column to config.
19              if fixedProps then
20                  Inherit existing proportions from parent node.
21              else
22                  Compute revised proportions through NNLS.
23              Compute and record solution error of node.
24      Rank all nodes by solution error.
25      Identify top smnc nodes by solution error, allowing for ties, but cut off at
        hmnc nodes. Mark these as chosen.
26      if $l == N_l$ then
27          propSols ← all nodes at level.
28      Determine whether to fix proportions (fixedProps ← TRUE).
29  Return (propSols).
```

Algorithm 3: LITSE Search Solution Space() Routine. Summary; details excluded.

high enough value to ensure that nodes with a non-negligible chance of being on the path to the correct solution will be included while excluding those with no reasonable chance. The setting of these parameters will be discussed in §3. We use the term "branching control" to denote this process of selecting a subset of nodes for continuation.

Their use is illustrated in Example 2.24 below.

Example 2.24 (Application of `smnc` and `hmnc`). Consider the case where the routine is processing level l in the search tree and all nodes have been ordered by increasing error (possibly with ties at certain points in the series). As such, it needs to select which nodes from level $l-1$ will have children at level l. Assume that level $l-1$ has 100 nodes. The set of selected nodes depends on the specific characteristics of the nodes at level $l-1$ and the choice of parameter values.

1. If $\texttt{smnc} \geq 100$, all nodes will be chosen.
2. If $\texttt{smnc} < 100$ and
   a) (case 1) $\text{err}(\widehat{Sol}_{n,\texttt{smnc}+1}) > \text{err}(\widehat{Sol}_{n,\texttt{smnc}})$, then only the first `smnc`nodes will be chosen,
   b) (case 2) there exists some $k > 0$ such that $\text{err}(\widehat{Sol}_{n,\texttt{smnc}+k}) = \text{err}(\widehat{Sol}_{n,\texttt{smnc}})$ but $\text{err}(\widehat{Sol}_{n,\texttt{smnc}+k+1}) > \text{err}(\widehat{Sol}_{n,\texttt{smnc}})$, then the first $\min\{ssmnc + k, \texttt{hmnc}\}$ nodes will be chosen

Consider some concrete cases with 100 nodes. Say $\texttt{smnc} = \texttt{hmnc} = 30$; then the first 30 nodes will be chosen, even if the 31st shares the same error as the 30th. Say $\texttt{smnc} = 30, \texttt{hmnc} = 50$ and $\text{err}(\widehat{Sol}_{n,38}) = \text{err}(\widehat{Sol}_{n,30})$ but $\text{err}(\widehat{Sol}_{n,39}) > \text{err}(\widehat{Sol}_{n,30})$; then 38 nodes will be chosen. Finally, say Say $\texttt{smnc} = 30, \texttt{hmnc} = 50$ and $\text{err}(\widehat{Sol}_{n,51}) = \text{err}(\widehat{Sol}_{n,30})$; then only the first 50 nodes will be chosen even though the 51st had a "sufficient" error.

□

Using these parameters, we can roughly control the "width" of the search tree, since levels are capped in terms of the number of nodes they will contain. Under this approach the maximum number of nodes at a level will equal $\texttt{hmnc} \cdot |\texttt{configs}|$ (permutations or combinations), where `configs` is the object of the same name appearing in line 17.

In summary,

- low values for `smnc` and `hmnc` correspond to a restrictive search,
- high values correspond to a more exhaustive search,
- in the limit, as `smnc` $\to \infty$, we evaluate all possibilities.

2.7.2.4.3 Compute Proportions

In any tree search, the SSS routine will need to compute the proportions of the proposed haplotypes. The proportions should be such that, when applied to the composition of the haplotypes, they generate allele proportions close to what was observed. In all cases, any set of proportions computed must meet the following two obvious conditions,

1. they must be non negative, and
2. they must sum to 1.

These two conditions are obvious for any set of proportions. A third condition, related to the MOI, is discussed in §2.7.2.4.4.

The general idea is to choose one of the tree levels as a "proportion fixation" level in the tree. Before and at this level, the proportions are updated as the routine descends the tree. That is, while each new node inherits the (partial) haplotype definitions from its parent, it does not inherit the proportions; instead, these are recomputed to reflect the additional data, in particular, the new locus added. Beyond the fixation level, each node inherits the proportions from its parent and no recomputation is performed.

The check on whether to fix the proportions at the current level is stated on line 28 in Algorithm 3. There are two broad conditions under which the proportions are fixed:

1. the level in the tree reaches the value of the parameter `mfl`, discussed below, and/or
2. the level in the tree represents the end of the processing of one or more locus blocks, split into the following two sub-cases,
    a) when the assumed MOI z is equal to the length of the longest locus block, and the level represents the last locus in that block, or
    b) when the assumed MOI z is greater than this length, [21] and the level represents the last locus of the second block.

[21]Recall that we will never have the case where the assumed MOI is less than this number.

Regarding the first condition, the purpose of the parameter `mfl` is to avoid the computational cost of redetermining the proportions at each node when the solutions have reached a point where their proportions are likely to change little from one level to the next. Thus a high (low) value for `mfl` corresponds to a lax (restrictive) approach to recomputing proportions. Use of this parameter will prevent cases where the algorithm is recomputing proportions deep into the tree when the second condition has not yet been met.

Regarding the second condition, the idea is to stop recomputing proportions when sufficient information, in the form of loci allele proportions, has been processed. The first sub-case is straightforward; here, there are one or more loci that can determine the proportions since their length equals the assumed MOI. The second sub-case is stated on line 22 in Algorithm 3. Here, since the assumed MOI z is greater than the length of any of the loci, the goal is to determine how to best split each locus's allele proportions over z haplotypes whose number is greater than the number of allele proportions for any one locus.

Given the constraints on the proportions, the routine uses non-negative least squares (NNLS), an iterative procedure, to find proportions that minimize the squared error of the difference between the optimal haplotype proportions across the loci being considered. When performed, this procedure is *cumulative* operates on two or more loci at once.

In the standard NNLS setup, we seek to solve the following constrained optimization,

$$\min \|A\mathbf{v} - \mathbf{b}\|_2 \;\; s.t. \; \mathbf{v} \geq 0$$

where A is a matrix and $\mathbf{b}$ is a vector that are both given. The vector $\mathbf{v}$ is to be determined [23].

Applying this framework to the problem of determining haplotype proportions with information on two or more loci,

1. the matrix A represents the structure of the locus data along with the constraints of the problem, with number of columns equal to $|loci| \cdot z$,
2. the vector $\mathbf{b}$ represents the stacked observed allele proportions, with one block per locus, and

3. the vector $\mathbf{v}$ represents the stacked proportions to be determined, with one block per locus.

Note that by matrix compatibility, the vector $\mathbf{v}$ will have dimensions $(|loci| \cdot z) \times 1$

Once the NNLS regression has been performed and a $\mathbf{v}$ vector has been identified, this $\mathbf{v}$ vector will be a set of stacked proportions, one per locus. If the loci have aligned well, the proportions will be similar. In any event, we need a single set of final proportions, which we compute by averaging for each haplotype. This is similar to what was performed in the first case.

The details of the computation are best explained through an illustration as provided in Example 2.25.

Example 2.25 (Compute Proportions). Assume that the Search Solution Space routine is currently at level 3 and is processing a particular node. Assume that the proposed solution corresponding to this node is as follows,

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Loc$_5$
h_1	$A_{1,?}$	$A_{2,3}$	$A_{3,?}$	$A_{4,1}$	$A_{5,1}$
h_2	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,2}$	$A_{5,3}$
h_3	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,2}$	$A_{5,4}$
h_4	$A_{1,?}$	$A_{2,5}$	$A_{3,?}$	$A_{4,5}$	$A_{5,4}$

In this example, $z = 4$. Since we are at level 3, exactly three of the loci have been filled (2, 4, and 5). Also, each of the loci has exactly three ($< z$) unique alleles represented, thus categorizing this as a case 2 problem.

Assume further that the observed allele proportions for the relevant loci are as follows:

Loc$_2$	p	Loc$_4$	p	Loc$_5$	p
$A_{2,2}$	0.40	$A_{4,1}$	0.20	$A_{5,1}$	0.15
$A_{2,3}$	0.20	$A_{4,2}$	0.50	$A_{5,3}$	0.20
$A_{2,5}$	0.40	$A_{4,5}$	0.30	$A_{5,4}$	0.65

The A matrix is as follows,

$$A = \left[\begin{array}{cccccccccccc}
\delta & 0 & 0 & 0 & -\delta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \delta & 0 & 0 & 0 & -\delta & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \delta & 0 & 0 & 0 & -\delta & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \delta & 0 & 0 & 0 & -\delta & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & -\delta & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & -\delta & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & -\delta & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta & 0 & 0 & 0 & -\delta \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\hline
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}\right]$$

The entire matrix is constructed as a series of vertically stacked blocks, as indicated by the horizontal lines. In addition, the matrix is viewed as three (due to three loci) vertical strips, each with a number of columns equal to z. Thus the number of columns is $3 \cdot 4 = 12$. The first block contains diagonal entries of value δ, where δ defaults to $1e-3$ is a small constant. The purpose of this block is to steer the proportions proposed for the first locus to equal the proportions for the second. This will become clearer when the **b** vector is constructed. Similarly, the second block guides the proportions of the second locus to equal those of the third. The value δ is chosen to be small because we want this "guidance" towards equality to avoid conflicting with the constraints, to be discussed next. The first set of constraints, represented by blocks 3-5, is that the proportions for each locus generate the observed allele proportions for that locus. Thus the third block governs locus 2 (the first of the loci to be filled), block 4 governs locus 4, and block 5 governs locus 5. Finally, one of the constraints is that the proportions sum to 1. To achieve this constraint, we introduce blocks 6-8, one per locus as before.

We next construct the **b** vector as follows,

$$\mathbf{b} \;=\; \big[0\;\;0\;\;0\;\;0\;\mid\;0\;\;0\;\;0\;\;0\;\mid 0.2\;\;0.4\;\;0.4\;\mid\;0.2\;\;0.5\;\;0.3\;\mid\;0.15\;\;0.2\;\;0.65\;\mid 1\;\mid\;1\;\mid\;1\big]^T$$

Here, the first two blocks of entries are always zero. The idea is that if the proportions for the first pair of loci are the same or similar, their difference (as constructed using the A matrix) should be zero. The next three blocks are the observed allele proportions per locus. The corresponding entries in the A matrix will sum the relevant rows in the haplotypes, and these sums are expected to equal the observed allele proportions. Finally, the last three blocks, coupled with their corresponding entries in the A matrix, ensure that each of the proposed haplotype proportions sum to 1.

When we submit A and **b** to the Lawson-Hanson NNLS routine, we get the following result for **v** (rounded to the nearest thousandth),

$$\mathbf{v} = \big[0.200\;0.133\;0.267\;0.400\;\mid\;0.200\;0.183\;0.317\;0.300\;\mid\;0.150\;0.200\;0.333\;0.317\big]^T$$

and the final set of proportions that will be proposed for this node are,

$$(0.200+0.200+0.150)/3 = 0.183,\;\; (0.133+0.183+0.200)/3 = 0.172,$$
$$(0.267+0.317+0.333)/3 = 0.306,\;\; (0.400+0.300+0.317)/3 = 0.339$$

We see that the sum of these proportions is 1 as required. Thus the solution for the node will have its proportions updated, as follows,

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Loc$_5$	Prop
h_1	$A_{1,?}$	$A_{2,3}$	$A_{3,?}$	$A_{4,1}$	$A_{5,1}$	0.183
h_2	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,2}$	$A_{5,3}$	0.172
h_3	$A_{1,?}$	$A_{2,2}$	$A_{3,?}$	$A_{4,2}$	$A_{5,4}$	0.306
h_4	$A_{1,?}$	$A_{2,5}$	$A_{3,?}$	$A_{4,5}$	$A_{5,4}$	0.339

This single set of proportions will then be used to compute the error. The relative "goodness of fit" of this node will be determined by comparing its error to those of all other nodes at the same level.

□

2.7.2.4.4 Assess Validity of Solution

The `LITSE` algorithm is designed to ensure that all proposed solutions, no matter quality of the fit with the observed allele proportions, are feasible as defined in §2.4.1.3.

The validity check here is thus an extension of the two conditions regarding proportions mentioned at the beginning of §2.7.2.4.3. There is an additional constraint that every proportion in a solution meet or exceed some user-defined minimum $\epsilon > 0$. The purpose of this control is identify and eliminate solutions where one or more haplotypes is essentially non-existent.

This control is particularly useful when attempting higher MOIs on observed data for which a lower MOI already gives an excellent fit. Consider the following example, generated using code for simulating observational data and analyzed using `LITSE`.

Example 2.26 (Valid and Invalid Proportions). Assume five loci and an observation with the following true haplotypes.

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Loc$_5$	p
h_1	$A_{1,3}$	$A_{2,8}$	$A_{3,3}$	$A_{4,5}$	$A_{5,1}$	0.70995
h_2	$A_{1,5}$	$A_{2,8}$	$A_{3,3}$	$A_{4,3}$	$A_{5,1}$	0.29005

This implies that the true MOI is 2. Assume that with some level of measurement error the observed proportions are as follows,

Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Loc$_5$
$A_{1,3} : 0.7173$	$A_{2,8} : 1$	$A_{3,3} : 1$	$A_{4,5} : 0.7362$	$A_{5,1} : 1$
$A_{1,5} : 0.2827$			$A_{4,3} : 0.2638$	

Since loci 1 and 4 have length 2 and the others length 1, the `LITSE` algorithm will start with the assumption that 2 is the correct MOI and will compute the following proportion estimates for one solution.[22]

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Loc$_5$	p
h_1	$A_{1,3}$	$A_{2,8}$	$A_{3,3}$	$A_{4,5}$	$A_{5,1}$	0.7268
h_2	$A_{1,5}$	$A_{2,8}$	$A_{3,3}$	$A_{4,3}$	$A_{5,1}$	0.2732

Since it provides a good fit, this solution will produce a low error vis-à-vis the observed allele proportions.

When the algorithm begins attempting solutions with an MOI of 3, it will consider all combinations/permutations of the alleles expressed per locus. One such potential solution will be the following,

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Loc$_5$	p
h_1	$A_{1,3}$	$A_{2,8}$	$A_{3,3}$	$A_{4,5}$	$A_{5,1}$	0.7268
h_2	$A_{1,5}$	$A_{2,8}$	$A_{3,3}$	$A_{4,3}$	$A_{5,1}$	0.2732
h_3	$A_{1,3}$	$A_{2,8}$	$A_{3,3}$	$A_{4,3}$	$A_{5,1}$	1.34e-13

This solution is clearly feasible, in that it meets all the criteria of §2.4.1.3. Yet it is also clear that by forcing a solution with three haplotypes, the solution simply added a feasible but erroneous haplotype and assigned it a miniscule proportion.

This solution of three haplotypes would be deemed invalid if ϵ was set to anything greater than $1.34e-13$.

$\square$

The purpose of this control is to identify and remove such forced solutions, as these are indicative of attempting an excessively high MOI.

While the value of $\epsilon > 0$ can be defined by the user, for all simulations in §3 it was set to 0.001; that is, for a solution to be valid, each haplotype must have a proportion equal to at least 1/1000.

2.7.2.5 Assess Attempted MOI

One of the key tasks facing the `LITSE` algorithm is to correctly identify the MOI for each observation. In an effort to achieve this, and similar to assessing individual solutions,

[22]This is one among many solutions. In this case, we know that it is correct even if the proportions are slightly off. The estimated proportions were computed by averaging the proportions of loci 1 and 4.

the algorithm will assess the entire set of solutions for an observation/MOI combination.

Once all solutions for a particular MOI and particular observation have been gathered, the algorithm will reject the entire MOI for that observation if each individual solution at that MOI for that observation was deemed invalid. It will also reject the entire MOI if the best solution from that MOI does not significantly improve on the best error from the preceding MOI. More precisely, the MOI $k + 1$ will be rejected if the ratio $\min(error)_{k+1}/\min(error)_k >$ `mer` and no greater MOIs will be attempted.

The reasoning behind this filtering is as follows. As the attempted MOI increases, the solutions have more flexibility to more closely match the observed allele proportions and achieve lower errors. Yet when the correct MOI is attempted, we expect at least some solutions to produce implied allele proportions close to those observed.[23] Thus the next higher (and incorrect) MOI will not produce significantly better results in terms of error.

To illustrate, consider the case where the maximum locus length for an observation is 2 and `am` $= 2$. This means that `LITSE` will normally consider MOIs of $z \in \{2, 3, 4\}$. However, if the MOI 3 is rejected for the reasons mentioned above, then the algorithm will retain solutions for $z = 2$, reject all solutions from $z = 3$, and no longer attempt solutions for $z = 4$. This early termination also has the advantage of eliminating costly computations for MOIs unlikely to be correct.

2.7.2.6 Illustration of a `LITSE` Tree

To provide some additional context for the simulation exercise and the trees produced, please refer to §2.B in the Appendix for an illustration of a simple tree generated by `LITSE` for a single observation.[24] The `LITSE` algorithm includes an interface to the `igraph` package [9] in `R` , enabling the production of graphs of the smaller trees produced.

[23]This is complicated by the presence of measurement error, whose impact will be assessed in §3.

[24]This tree was not part of any simulation run and was produced separately.

2.7.3 Solution Set Completion

As presented in Algorithm 4 below, the SSC step imputes any missing loci data for each solution produced in the SSI step.

```
LITSE Solution Set Completion() Procedure
    Input  : propSols, proposed solutions for each observation/MOI from Solution Set
             Identificationstep.
    Output: propSols, complete proposed solutions for each observation/MOI.
1   incompleteSols←Create list of incomplete solutions in propSols.
2   incompleteHaps←Create list of unique incomplete haplotypes in incompleteSols.
3   distMat←Build matrix (using Hamming distance) with distance between (a) each
    member incompleteHap of incompleteHaps and (2) all haplotypes in propSols not
    sharing missing loci with incompleteHap.
4   for incompleteSol in incompleteSols do
5       for incompleteHap in incompleteSol do
6           closestHaps(incompleteHap)←Identify closest haplotype(s) from distMat.
7       Split incompleteSol into Cartesian product of incompleteSol and
        closestHaps(incompleteHap) over all members in
        closestHaps(incompleteHap).
8       Remove incompleteSol from propSols.
9       Add (newly formed) complete solutions to propSols.
10  Return (propSols).
```

Algorithm 4: LITSE Solution Set Completion Procedure. Summary; details excluded.

Note that, as explained in §2.4.1.2.2, it is a particular observation/locus combination that will be complete or incomplete. When an observation has an incomplete locus, this will be present for all solutions belonging to that observation since none of these solutions will have had the necessary observed allele information for that locus. Because distances are computed only for haplotype pairs with no shared missing loci, it follows that an incomplete solution will never have its haplotypes measured against other haplotypes from other solutions for the same observation. It may well, however, have its haplotypes measured against other haplotypes from other incomplete solutions for different observations (as long as they share no missing loci).

In cases where all solutions from the previous step are complete, the Solution Set Completion step is unnecessary and is skipped.

To illustrate this process, consider the following example.

Example 2.27 (Impute Missing Locus Information). Consider the case where there are four loci in question and one solution is as follows,

ID	Loc_1	Loc_2	Loc_3	Loc_4	Prop
h_1	?	$A_{2,33}$	$A_{3,12}$	$A_{4,42}$	0.5
h_2	?	$A_{2,102}$	$A_{3,21}$	$A_{4,51}$	0.4
h_3	?	$A_{2,36}$	$A_{3,30}$	$A_{4,54}$	0.1

That is, the solution is missing data for locus 1 only.

Now consider two scenarios. In the first, each of the haplotypes has a single closest haplotype. Assume that these are $(A_{1,6}, ?, A_{3,12}, A_{4,42})$, $(A_{1,9}, A_{2,102}, A_{3,21}, A_{4,54})$, and $(A_{1,9}, A_{2,36}, A_{3,30}, A_{4,54})$ for haplotypes 1,2, and 3, respectively. Then we replace the existing incomplete solution with $1 \times 1 \times 1 = 1$ solution, namely,

ID	Loc_1	Loc_2	Loc_3	Loc_4	Prop
h_1	$A_{1,6}$	$A_{2,33}$	$A_{3,12}$	$A_{4,42}$	0.5
h_2	$A_{1,9}$	$A_{2,102}$	$A_{3,21}$	$A_{4,51}$	0.4
h_3	$A_{1,9}$	$A_{2,36}$	$A_{3,30}$	$A_{4,54}$	0.1

This example demonstrates two points. First, there may be cases where the closest haplotype also has missing loci, albeit in different locus positions. Second, we never overwrite a non-missing locus value in the original solution; for example, for the second haplotype, we retained the fourth locus as $A_{4,51}$ even though the nearest haplotype had $A_{4,54}$ in this position.

In the second scenario, for the same incomplete solution, assume that the first haplotype had a total of three distinct haplotypes that tied for proximity, the second haplotype still had only one, and the third had two. The new haplotypes in each of the three sets differ at the first locus, so when "merging" each with the corresponding original haplotypes they produce different results.[25] Then the original incomplete solution would be replaced by $3 \times 1 \times 2 = 6$ separate complete solutions, the Cartesian product of the adjusted haplotype sets. One of these six solutions would be that appearing above.

$\square$

[25]Here, by "set" we mean the collection of proximal, replacement haplotypes for a given haplotype.

2.7.4 Solution Set Refinement

This procedure essentially implements the EM algorithm using the proposed solution identified in Solution Set Identification. At this point in the overall `LITSE` algorithm, solutions have been identified for each MOI considered for each observation. The details of the SSR procedure are presented in Algorithm 5 below.

```
LITSE Solution Set Refinement()  Procedure
    Input  : propSols, proposed solutions for each observation/MOI from Solution Set
             Identification procedure; $\hat{\mathbf{p}}^{(0)}, \hat{\boldsymbol{\pi}}^{(0)}, \hat{\mathbf{c}}^{(0)}$, parameter estimates; run settings.
    Output: finalSols, per observation, a list of possible solutions with probability of
             each; $\hat{\mathbf{p}}, \hat{\boldsymbol{\pi}}, \hat{\mathbf{c}}$, final parameter estimates
1   Determine unique haplotypes in propSols.
2   finalSols $\leftarrow$ propSols.
3   while itercnt $<$ itermax and convergence criteria not met do
4       Increment itercnt.
5       Do Expectation step (will update finalSols).
6       Do Maximization step (will update $\hat{\mathbf{p}}, \hat{\boldsymbol{\pi}}, \hat{\mathbf{c}}$).
7       Compute value of $Q$ function.
8   Return (finalSols).
```

Algorithm 5: LITSE Solution Set Refinement Procedure. Summary; details excluded.

In line 1, the procedure identifies all unique haplotypes used across all solutions for all observations. This list forms the set of haplotypes deemed to have some evidence for existing in the population.

Line 5 executes the Expectation step of the EM algorithm, as documented in 2.7.4.2.4. Similarly, line 6 executes the Expectation step of the EM algorithm, as documented in 2.7.4.2.5. The result of these two steps will be a constantly evolving set of probabilities for the solutions and estimates for the parameters. Finally, line 7 computes the Q function value, whose change from the previous iteration is one of the stopping criteria.

2.7.4.1 Role of the Expectation-Maximization Algorithm

In this section, we briefly discuss the general setup of the Expectation Maximization algorithm and then cover how it is applied in the current situation.

2.7.4.1.1 General Framework

The Expectation-Maximization (EM) algorithm is used to find the maximum likelihood parameters of a model in cases where there is no closed form solution for the parameter estimates. Such situations frequently involve latent variables in addition to data observations. Presented in Dempster et al. [10], it has been used widely across many areas in applied statistics.

To paraphrase [4], the EM algorithm can be thought of as reversing the generative process to find the hidden structure that likely generated the observed data.

As suggested by its name, the EM algorithm comprises two steps – an expectation step and a maximization step – which are performed iteratively until some stopping criterion has been reached. Typical choices for a stopping criterion include convergence in the parameter estimates, convergence in the likelihood of the data, or maximum number of iterations reached.

There are several things to consider in the application of the EM algorithm to the current problem.

1. the distinction between the observed, unobserved, and complete data
2. the parameters to be estimated
3. the $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function
4. the implementation of the expectation step
5. the implementation of the maximization step

2.7.4.1.2 Data classification

The EM algorithm considers three sets of data,

1. the observed data, denoted $\Omega = \{\Omega_n\}_{n=1,\ldots,N_{\text{samp}}}$
2. the unobserved/latent data, denoted $\Upsilon = \{\Upsilon_n\}_{n=1,\ldots,N_{\text{samp}}}$
3. the complete data, which is simply the union of the two previous sets, denoted $\mathcal{C} = \{\Omega_n, \Upsilon_n\}_{n=1,\ldots,N_{\text{samp}}}$

2.7.4.1.3 Parameters

The set of parameters across all variables in $\mathcal{C}$ is denoted as Θ. There will be two versions of Θ: $\Theta^{(t)}$ represents the current values for the parameter estimates and $\Theta^{(t-1)}$ represents the set of values estimated in the previous iteration.

2.7.4.1.4 $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function

Within the general framework of the EM algorithm, the $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function is defined as follows

$$Q(\Theta^{(t)}|\Theta^{(t-1)}) := E[\log f(\mathcal{C};\Theta^{(t)})|\Omega,\Theta^{(t-1)}] \tag{2.33}$$

This function is the expectation of the log of the likelihood of the complete set of data conditional on the observed data.

2.7.4.1.5 Expectation Step

In the expectation step, we compute the expectation in equation (2.33). As we will see in §2.7.4.2.4, given the structure of $Q(\Theta^{(t)}|\Theta^{(t-1)})$, this typically involves computing several probabilities. These probabilities will be a function of $\Theta^{(t-1)}$.

2.7.4.1.6 Maximization Step

In the maximization step, we recompute the parameter values to maximize the $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function. Note that the probabilities computed in equation (2.7.4.1.5) remain unchanged in this step (though they will be recomputed in the next iteration's E step).

2.7.4.2 Application of EM Algorithm to LITSE

Each aspect of the general EM algorithm is considered below.

2.7.4.2.1 Data classification

As shown in Table 2.3, all variables other than the observed allele proportions $\boldsymbol{X}_n$ are latent. Thus $\Omega_n = \{\boldsymbol{X}_n\}$ and $\Upsilon_n = \{Z_n, \mathbf{H}_n, \mathbf{Y}_n, \mathcal{U}_n\}$.

2.7.4.2.2 Parameters

As shown in Table 2.4, the parameters to estimate include $\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}$. There will be no need to estimate ξ. Thus $\Theta = \{\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}\}$ in our implementation.

2.7.4.2.3 $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function

Consider the first component of the $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function, $f(\mathcal{C};\Theta^{(t)})$. We already have a representation for this from equation (2.31); that is,

$$\begin{aligned} f(\mathcal{C};\Theta^{(t)}) &= \mathbb{L}(\mathcal{C}) \\ &= \prod_{n=1}^{N_{\text{samp}}} \left[\prod_{l=1}^{N_l} f_{\mathbf{X}_n^{(l)}|\mathbf{U}_n^{(l)}}(\mathbf{x},\mathbf{u};\mathbf{c}) f_{\mathbf{U}_n^{(l)}|\mathbf{Y}_n}(\mathbf{u},\mathbf{y})\right] f_{\mathbf{Y}_n|\mathbf{H}_n}(\mathbf{y},\mathbf{h};\xi) f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi}) f_{Z_n}(z;\mathbf{p}). \end{aligned}$$

One feature of this expression is that all variables are indexed by n; that is, for each observation n we include only the realized values for the variables, only one ($\mathbf{X}_n$) of which we will have actually observed.

With the definition of $Q(\Theta^{(t)}|\Theta^{(t-1)})$ in mind, one can rewrite this expression initially as the following,

$$\begin{aligned} &f(\mathcal{C};\Theta^{(t)}) = \\ &\prod_{n=1}^{N_{\text{samp}}} \prod_{\boldsymbol{u}\in\boldsymbol{U}} \prod_{\mathbf{y}\in\mathcal{Y}} \prod_{\mathbf{h}\in\mathcal{H}} \prod_{z\in\mathcal{Z}} \Bigg[\left[\prod_{l=1}^{N_l} f_{\mathbf{X}_n^{(l)}|\mathbf{U}_n^{(l)}}(\mathbf{x},\mathbf{u};\mathbf{c}) f_{\mathbf{U}_n^{(l)}|\mathbf{Y}_n}(\mathbf{u},\mathbf{y})\right] \cdot \\ &\qquad f_{\mathbf{Y}_n|\mathbf{H}_n}(\mathbf{y},\mathbf{h};\xi) f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi}) f_{Z_n}(z;\mathbf{p})\Bigg]^{I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)} \end{aligned} \tag{2.34}$$

where

$$I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) := \mathbb{1}(\mathcal{U}_n = \boldsymbol{u}) \cdot \mathbb{1}(\mathbf{Y}_n = \mathbf{y}) \cdot \mathbb{1}(\mathbf{H}_n = \mathbf{h}) \cdot \mathbb{1}(Z_n = z) \tag{2.35}$$

and the sets over which we are computing the product (i.e., the terms under the multiplication operators in the first part of equation (2.35)) are documented in Table 2.8. Note that $I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)$ defines a proposed solution for observation n, as it stipulates the MOI, haplotypes, haplotype proportions, and implied allele proportions.

Set	Description
$\mathcal{Z}$	Set of all numbers $\{1, 2, \ldots, \text{maxMOI}\}$ (the maximum MOI considered). This set is finite and countable.
$\mathcal{H}$	Set of all vectors comprising numbers $\{0, 1\}$ and of length Nh (the total number of possible haplotypes). This set is finite and countable.
$\mathcal{Y}$	Set of all vectors comprising numbers between 0 and 1 inclusive and of length N_{haps} (the total number of possible haplotypes). This set is uncountable.
$\boldsymbol{U}$	Collection of all sets of size N_l, all of whose members are vectors meeting certain criteria. Let $\boldsymbol{u}_\lambda \in \boldsymbol{U}$. Then $\boldsymbol{u}_\lambda$ will have N_l elements with the j^{th} element $\boldsymbol{u}_{\lambda,j}$ being a vector of length $\ell(j)$ with entries between 0 and 1 inclusive. In other words, $\boldsymbol{u}_{\lambda,j} \in [0,1]^{\ell(j)}$. This set is uncountable.

Table 2.8: Search Set - Exhaustive ($\mathcal{S} = \{\mathcal{Z}, \mathcal{H}, \mathcal{Y}, \boldsymbol{U}\}$). These sets include few restrictions on the values for each variable. $\mathcal{S}$ is simply the collection of these four sets. Note that these sets are identical for all observations.

We call these sets "exhaustive" because they include minimal restrictions on the elements in each set. The sets in Table 2.8 are intended to include all possible values that each variable may take on; indeed, as we will see below, they cover many infeasible values as well. The sets are illustrated in the next example.

Example 2.28 (Search Sets). Assume that there are 3 loci in total, the first two of which have two possible alleles and the third of which has three possible alleles. Then $N_l = 3$, $N_{\text{haps}} = 2 \cdot 2 \cdot 3 = 12$, and $\ell(j) = 2, 2, 3$ for $j = 1, 2, 3$, respectively.

If, for example, maxMOI $= 5$, then $\mathcal{Z}$ is simply $\mathcal{Z} = \{1, 2, 3, 4, 5\}$.

The set $\mathcal{H}$ will contain all vectors of length 12 with 0's and 1's throughout. For example,

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}^T$$

and

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}^T$$

would be just two elements of $\mathcal{H}$. Based on their number of non-zero entries, the first vector is consistent with $z = 4$ while the second is consistent with $z = 6$. Since maxMOI $=$

5, the second vector would be infeasible when assessed as a potential haplotype indicator vector as described in §2.6.1.2.

Similarly, the set $\mathcal{Y}$ will contain all vectors of length 12 with values between 0 and 1 throughout. For example,

$$\begin{pmatrix}0 & 0 & 0 & 0 & 0 & 0.20 & 0.11 & 0 & 0.24 & 0 & 0 & 0.45\end{pmatrix}^T$$

and

$$\begin{pmatrix}0.1 & 0.9 & 1 & 0.34 & 0 & 0.89 & 0.11 & 0.43 & 0.24 & 0.87 & 0.12 & 0.98\end{pmatrix}^T$$

would be just two elements of $\mathcal{Y}$. While both are included in $\mathcal{Y}$, when assessed as potential haplotype proportion vectors as described in §2.6.1.3 the first vector is feasible while the second vector falls afoul for two reasons: first, it implies that $z = 11$, which is clearly greater than maxMOI, and second, the sum of the entries exceeds 1.

Finally, every element of the collection $\boldsymbol{U}$ will itself be a set with $N_l = 3$ vectors. The first of these vectors will have two entries, the second will have two, and the third will have three. Each vector will have entries in the range 0 through 1. Thus two sets in the collection $\boldsymbol{U}$ would look like the following,

$$\left\{\begin{pmatrix}0.23 & 0.77\end{pmatrix}^T, \begin{pmatrix}0.33 & 0.67\end{pmatrix}^T, \begin{pmatrix}0.31 & 0.34 & 0.35\end{pmatrix}^T\right\}$$

and

$$\left\{\begin{pmatrix}0.23 & 0.98\end{pmatrix}^T, \begin{pmatrix}0.13 & 0.45\end{pmatrix}^T, \begin{pmatrix}0.34 & 0.34 & 0.87\end{pmatrix}^T\right\}.$$

While both are included in $\boldsymbol{U}$, when assessed as potential implied allele proportions as described in §2.6.1.5 the first set is feasible while the second is invalid because for none of the three locis does the total of the implied proportions sum to 1.

$\square$

The alternative formulation in equation (2.34) achieves the same result as equation (2.31) but does so in a different way. Here, we are theoretically computing products over a much larger space, as indicated by the sets below each product symbol and documented in Table 2.8. For example, the product range for z is $\mathcal{Z}$, the *entire* set of integers in the range 1 to maxMOI, rather than using the single value z_n as in equation (2.31). The same is true for all other variables in the equation. However, we are controlling this larger space for each variable through the use of equation (2.35). In $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z)$, we use indicator functions to "switch on" a particular multiplicand only when all the values used in the multiplicand equal the correct values for the relevant variables for observation n. Note that $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) = 1$ if and only if all component indicator functions equal 1; otherwise, $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) = 0$. When $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) = 1$ the multiplicand is used in the product; when $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) = 0$, the base is raised to power of 0 and the result is to simply multiply by 1.[26] In summary, equation (2.34) can be viewed as the product of many, many terms, the vast majority of which equal 1 and a small minority of which equal the values corresponding to the true complete data.

Note that the sets in Table 2.8 do not distinguish between what is compatible and incompatible with $\boldsymbol{X}_n$, the observed allele proportions for each locus for observation n, or indeed even what is feasible in general (as discussed in Example 2.28). While equation (2.34) is correct, it implies computing many bases whose exponent we know will equal zero, resulting in a multiplicand equaling 1, because we know that a particular variable will be incompatible with $\boldsymbol{X}_n$. As such, the sets in Table 2.8 are needlessly large.

It is to our advantage to shrink each set in Table 2.8 by selecting for each a subset *limited to feasible values*. The method we propose is to start with the simplest variable, Z_n, the assumed MOI. Since $\boldsymbol{X}_n$ includes the observed proportions for each allele at each locus and each haplotype contains exactly one allele per locus, the MOI generating $\boldsymbol{X}_n$ must be at least equal to the maximum number of non-zero alleles across all loci.[27] We call this minimum value minMOI_n.

Example 2.29 (Minimum Value for z). Assume that there are three loci and

$$\boldsymbol{X}_n = \{\mathbf{x}_n^1, \mathbf{x}_n^2, \mathbf{x}_n^3, \mathbf{x}_n^4\}$$

[26]To avoid any confusion, note that equation (2.34) takes the form

$$f(\mathcal{C}; \Theta^t) = \prod \underbrace{\text{base}^{\text{exponent}}}_{\text{multiplicand}}.$$

That is, the string of f functions form the base and $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z)$ forms the exponent. This is the view we have in mind when discussing the properties of the likelihood function.

[27]This is the minimum MOI for observation n; a higher MOI is possible in the case where haplotypes "double up".

$$= \left\{ \begin{bmatrix} 0 \\ 0 \\ 0.2 \\ 0.34 \\ 0 \\ 0 \\ 0.46 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.4 \\ 0.2 \\ 0 \\ 0.4 \end{bmatrix}, \begin{bmatrix} 0.1 \\ 0.3 \\ 0 \\ 0 \\ 0.25 \\ 0 \\ 0 \\ 0.35 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.3 \\ 0.7 \\ 0 \end{bmatrix} \right\}$$

The number of non-zero alleles in $\boldsymbol{X}_n$ per locus is 3,3,4,2. Thus we know that at least $4 = \max\{3, 3, 4, 2\}$ haplotypes must have generated the data; that is, $\text{minMOI}_n = 4$. So in this case, we would want to consider only $\{z \in \mathbb{Z} : 4 \leq z \leq \text{maxMOI}\}$; that is, we would skip $z \in \{1, 2, 3\}$.

$\square$

Having identified valid values for Z_n, we then work our way backwards through the sets, starting with $\mathcal{H}$ and reducing each set to only what is feasible given all previous assumptions. Note that we do this per observation. This exercise leads to the sets in Table 2.9. We call these sets "feasible" search sets because they include only values that are feasible for each variable. To be clear, the purpose of these sets is to reduce the number of computations while still evaluating *all* valid variable values.[28]

With these search sets, we can state the likelihood as follows,

$$\begin{aligned} f(\mathcal{C}; \Theta^{(t)}) &= \prod_{n=1}^{N_{\text{samp}}} \prod_{\boldsymbol{u} \in \boldsymbol{U}^f_{n,\mathbf{y}}} \prod_{\mathbf{y} \in \mathcal{Y}^f_{n,\mathbf{h}}} \prod_{\mathbf{h} \in \mathcal{H}^f_{n,z}} \prod_{z \in \mathcal{Z}^f_n} \Bigg[\Bigg[\prod_{l=1}^{N_l} f_{\mathbf{X}^{(l)}_n | \mathbf{U}^{(l)}_n}(\mathbf{x}, \mathbf{u}; \mathbf{c}) f_{\mathbf{U}^{(l)}_n | \mathbf{Y}_n}(\mathbf{u}, \mathbf{y}) \Bigg] \cdot \\ &\qquad f_{\mathbf{Y}_n | \mathbf{H}_n}(\mathbf{y}, \mathbf{h}; \xi) f_{\mathbf{H}_n | Z_n}(\mathbf{h}, z; \boldsymbol{\pi}) f_{Z_n}(z; \mathbf{p}) \Bigg]^{I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z)} \\ &= \prod_{n=1}^{N_{\text{samp}}} \prod_{\boldsymbol{u} \in \boldsymbol{U}^f_{n,\mathbf{y}}} \prod_{\mathbf{y} \in \mathcal{Y}^f_{n,\mathbf{h}}} \prod_{\mathbf{h} \in \mathcal{H}^f_{n,z}} \prod_{z \in \mathcal{Z}^f_n} \Bigg[\Bigg[\prod_{l=1}^{N_l} f_{\mathbf{X}^{(l)}_n | \mathbf{U}^{(l)}_n}(\mathbf{x}, \mathbf{u}; \mathbf{c}) \cdot 1 \Bigg] \cdot \\ &\qquad \xi \cdot f_{\mathbf{H}_n | Z_n}(\mathbf{h}, z; \boldsymbol{\pi}) f_{Z_n}(z; \mathbf{p}) \Bigg]^{I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z)} \end{aligned} \tag{2.36}$$

[28]By "feasible", we simply mean mathematically possible as opposed to computationally practical.

Set	Description
$\mathcal{Z}^f_n$	Set of all integers $\{z : z \leq \text{minMOI}\}$.
$\mathcal{H}^f_{n,z}$	Set of all vectors comprising numbers $\{0, 1\}$, of length N_{haps}, with precisely z 1's, and consistent with the non-zero alleles observed in $\boldsymbol{X}_n$. By "consistent", we mean having only haplotype sets that span all alleles observed in the data and only those alleles. This set can be constructed only once z has been chosen.
$\mathcal{Y}^f_{n,\mathbf{h}}$	Set of all vectors comprising numbers between 0 and 1 (inclusive), of length N_{haps}, summing to 1, and consistent with $\mathbf{h}$. Ensuring a candidate vector is consistent with $\mathbf{h}$ will ensure it is consistent with $\boldsymbol{X}_n$, the observed data. By "consistent", we mean $y_i > 0 \iff h_i = 1, i \in \{1, 2, \ldots, N_{\text{haps}}\}$. This set can be constructed only once $\mathbf{h}$ has been chosen.
$\boldsymbol{U}^f_{n,\mathbf{y}}$	The set of vectors $\{\mathbf{U}^{(l)}, l = 1, 2, \ldots, N_l\}$ consistent with $\mathbf{y}$. Note that any given $\mathbf{y}$ will completely determine this set; thus there is *only one* such set. This implies that, despite the possible appearance to the contrary, $\lvert \boldsymbol{u} \in \boldsymbol{U}^f_{n,\mathbf{y}} \rvert = 1$. This set can be constructed only once $\mathbf{y}$ has been chosen.

Table 2.9: Search Set - Feasible ($\mathcal{S}^f_n = \{\mathcal{Z}^f_n, \mathcal{H}^f_{n,z}, \mathcal{Y}^f_{n,\mathbf{h}}, \boldsymbol{U}^f_{n,\mathbf{y}}\}$). Each set is a subset of its counterpart in $\mathcal{S}$ while still including *all* valid variable values. For example, $\mathcal{H}^f_{n,z} \subset \mathcal{H}$. The entity $\mathcal{S}^f_n$ is simply the collection of these four sets. Note that these sets are observation-specific, hence the n in the subscript.

$$= \prod_{n=1}^{N_{\text{samp}}} \prod_{\boldsymbol{u} \in \boldsymbol{U}^f_{n,\mathbf{y}}} \prod_{\mathbf{y} \in \mathcal{Y}^f_{n,\mathbf{h}}} \prod_{\mathbf{h} \in \mathcal{H}^f_{n,z}} \prod_{z \in \mathcal{Z}^f_n} \left[\left[\prod_{l=1}^{N_l} f_{\mathbf{X}^{(l)}_n | \mathbf{U}^{(l)}_n}(\mathbf{x}, \mathbf{u}; \mathbf{c}) \right] \xi f_{\mathbf{H}_n | Z_n}(\mathbf{h}, z; \boldsymbol{\pi}) f_{Z_n}(z; \mathbf{p}) \right]^{I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z)} \tag{2.37}$$

after applying the sets from Table 2.9 and where $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z)$ was introduced in equation (2.35). Note that equation (2.36) follows from the construction of the sets $\mathcal{Y}^f_{n,\mathbf{h}}$ and $\boldsymbol{U}^f_{n,\mathbf{y}}$ and equations (2.10) and (2.14).

Applying the definition of the $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function from §2.7.4.2.3 to equations (2.37) and (2.35), in the current implementation,

$$Q(\Theta^{(t)}|\Theta^{(t-1)}) = E[\log f(\mathcal{C}; \Theta^{(t)})|\Omega, \Theta^{(t-1)}]$$

$$= E\left[\log\left[\prod_{n=1}^{N_{\text{samp}}}\prod_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\prod_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\prod_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\prod_{z\in\mathcal{Z}^f_n}\left[\left[\prod_{l=1}^{N_l} f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u};\mathbf{c})\right]\cdot\right.\right.\right.$$

$$\left.\left.\left.\xi f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi})f_{Z_n}(z;\mathbf{p})\right]^{I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)}\right]\middle|\Omega,\Theta^{(t-1)}\right]$$

$$= E\left[\left[\sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n} I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\log\left[\left[\prod_{l=1}^{N_l} f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u};\mathbf{c})\right]\cdot\right.\right.\right.$$

$$\left.\left.\left.\xi f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi})f_{Z_n}(z;\mathbf{p})\right]\right]\middle|\Omega,\Theta^{(t-1)}\right] \tag{2.38}$$

$$= E\left[\left[\sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n} I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\left[\sum_{l=1}^{N_l}\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u};\mathbf{c})\right)+\right.\right.\right.$$

$$\left.\left.\left.\log\xi+\log\left(f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi})\right)+\log\left(f_{Z_n}(z;\mathbf{p})\right)\right]\right]\middle|\Omega,\Theta^{(t-1)}\right] \tag{2.39}$$

$$= \sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n} E\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\left[\sum_{l=1}^{N_l}\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u};\mathbf{c})\right)+\right.$$

$$\left.\log\xi+\log\left(f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi})\right)+\log\left(f_{Z_n}(z;\mathbf{p})\right)\right] \tag{2.40}$$

$$= \sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n}\underbrace{\Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]}_{\substack{\text{Expectation}\\\text{component}}}\cdot$$

$$\left[\sum_{l=1}^{N_l}\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u};\mathbf{c})\right)+\log\xi+\log\left(f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi})\right)+\log\left(f_{Z_n}(z;\mathbf{p})\right)\right] \tag{2.41}$$

where equations (2.38) and (2.39) follows from the properties of the log function, equation (2.40) follows because from the previous line only $I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)$ is a random variable, and equation (2.41) follows from the properties of the indicator function.

For the expectation component in equation (2.41), we note that

$$\Pr[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)|\Omega,\Theta^{(t-1)}] = \Pr[\mathcal{U}_n=\boldsymbol{u},\mathbf{Y}_n=\mathbf{y},\mathbf{H}_n=\mathbf{h},Z_n=z|\boldsymbol{X}_n,\Theta^{(t-1)}] \tag{2.42}$$

$$= \frac{\Pr[\boldsymbol{X}_n = \boldsymbol{x}, \mathcal{U}_n = \boldsymbol{u}, \mathbf{Y}_n = \mathbf{y}, \mathbf{H}_n = \mathbf{h}, Z_n = z | \Theta^{(t-1)}]}{\Pr[\boldsymbol{X}_n = \boldsymbol{x} | \Theta^{(t-1)}]} \tag{2.43}$$

where

$$\begin{aligned}
&\Pr[\boldsymbol{X}_n = \boldsymbol{x}, \mathcal{U}_n = \boldsymbol{u}, \mathbf{Y}_n = \mathbf{y}, \mathbf{H}_n = \mathbf{h}, Z_n = z | \Theta^{(t-1)}] \\
&\quad = \Pr[\boldsymbol{X}_n = \boldsymbol{x} | \mathcal{U}_n = \boldsymbol{u}, \Theta^{(t-1)}] \cdot \Pr[\mathcal{U}_n = \boldsymbol{u} | \mathbf{Y}_n = \mathbf{y}, \Theta^{(t-1)}] \cdot \\
&\qquad \Pr[\mathbf{Y}_n = \mathbf{y} | \mathbf{H}_n = \mathbf{h}, \Theta^{(t-1)}] \cdot \Pr[\mathbf{H}_n = \mathbf{h} | Z_n = z, \Theta^{(t-1)}] \cdot \\
&\qquad \Pr[Z_n = z | \Theta^{(t-1)}] \\
&\quad = \prod_{l=1}^{N_l} \mathrm{Dir}(\mathbf{x}^{(l)}; \boldsymbol{u}, c^{(l),\Theta^{(t-1)}}) \cdot 1 \cdot \xi \cdot (z! \pi_{\iota_1}^{(t-1)} \cdot \dots \cdot \pi_{\iota_z}^{(t-1)}) \cdot p_z^{(t-1)} \\
&\quad = \left[\prod_{l=1}^{N_l} \mathrm{Dir}(\mathbf{x}^{(l)}; \boldsymbol{u}, c^{(l),\Theta^{(t-1)}}) \right] \xi (z! \pi_{\iota_1}^{(t-1)} \cdot \dots \cdot \pi_{\iota_z}^{(t-1)}) p_z^{(t-1)} \qquad (2.44)
\end{aligned}$$

for the observed data $\mathbf{x}$ and a particular proposed solution $I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z)$, and

$$\begin{aligned}
&\Pr[\boldsymbol{X}_n = \boldsymbol{x} | \Theta^{(t-1)}] \\
&\quad = \sum_{\boldsymbol{u} \in \boldsymbol{U}_{n,\mathbf{y}}^f} \sum_{\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^f} \sum_{\mathbf{h} \in \mathcal{H}_{n,z}^f} \sum_{z \in \mathcal{Z}_n^f} \Pr[\boldsymbol{X}_n = \boldsymbol{x}, \mathcal{U}_n = \boldsymbol{u}, \mathbf{Y}_n = \mathbf{y}, \mathbf{H}_n = \mathbf{h}, Z_n = z | \Theta^{(t-1)}]
\end{aligned} \tag{2.45}$$

where in equation (2.44) $\Pr[\mathcal{U}_n = \boldsymbol{u} | \mathbf{Y}_n = \mathbf{y}, \Theta^{(t-1)}] = 1$ by the construction of the set $\boldsymbol{U}_{n,\mathbf{y}}^f$ (see Table 2.9), ξ is a constant, and $\iota_j, j = 1, \dots, z$ signifies the index for the j^{th} haplotype present.

Note that equation (2.42) represents the conditional probability for any potential solution to the fundamental question: *which haplotypes, and in what proportion, is the sample likely to be infected with given the observed data?.*

Equation (2.44) is a measure of the relative probability of any particular solution to the primary question of interest. It can be interpreted as follows,

$$\underbrace{\left[\prod_{l=1}^{N_l} \mathrm{Dir}(\mathbf{x}^{(l)}; \boldsymbol{u}_n, c^{(l),\Theta^{(t-1)}}) \right]}_{\substack{\text{likelihood} \\ \text{of data} \\ \text{given assumed} \\ \text{proportions}}} \underbrace{\xi}_{\substack{\text{likelihood} \\ \text{of proportions} \\ \text{given assumed} \\ \text{haplotypes}}} \underbrace{(z! \pi_{\iota_1}^{(t-1)} \cdot \dots \cdot \pi_{\iota_z}^{(t-1)})}_{\substack{\text{likelihood} \\ \text{of haplotypes} \\ \text{given assumed} \\ \text{MOI}}} \underbrace{p_z^{(t-1)}}_{\substack{\text{likelihood} \\ \text{of MOI}}}$$

In other words, the overall probability of a potential solution is the combination (product) of the probabilities of its components. We also note, as previously discussed, that the probability of a particular set of proportions given an assumed sets of haplotypes is constant and thus does not play a role in the relative probability.[29] Intuitively, to achieve a "high" probability, a solution will need to achieve relatively high component probabilities. Conversely, a solution with a low component probability will likely have a low overall probability. For example, a solution with a high probability assumed MOI and choice of haplotypes but a low probability fit with the data (due to an implausible choice of haplotype proportions) will have a low overall probability.

There are two additional observations to be made from equations (2.43), (2.44), and (2.45). First, we note that from equation (2.45) that the denominator in equation (2.43) is simply the sum of all numerators considered in equation (2.43). Thus for each observation we will need to compute each numerator once, using the expression in (2.44), for a collection of valid sets for each observation. We then just sum these per observation to compute equation (2.45); finally, we use this result in equation (2.43) for each combination. Second, recall that we argued earlier that the value ξ would not play a role in the probability of a particular solution or the estimation of any parameters. The three equations show that this is indeed the case. Specifically, ξ appears directly in equation (2.44) and thus indirectly – but still linearly – in equation (2.45). Thus the ξ's cancel each other and ξ plays no role in the value of equation (2.43).

2.7.4.2.4 Expectation Step

In this step, we need to compute the expectation component of $Q(\Theta^{(t)}|\Theta^{(t-1)})$. In the current implementation, this means computing equation (2.43).

In theory, we need to compute this expectation for *all* combinations of the sets $\mathcal{Z}_n^f$,$\mathcal{H}_{n,z}^f$,$\mathcal{Y}_{n,\mathbf{h}}^f$, and $\boldsymbol{U}_{n,\mathbf{y}}^f$. However, to do so would be both

1. impractical, given the potentially large number of elements in $\mathcal{H}_{n,z}^f$ and the infinite number of elements in $\mathcal{Y}_{n,\mathbf{h}}^f$ (and thus $\boldsymbol{U}_{n,\mathbf{y}}^f$), and

[29]Consistent with this claim, we note that in equation (2.43) the value ξ will appear in both the numerator and denominator and thus drop out of the entire expression.

2. of limited value, given that the vast majority of combinations will result in a low relative probability as measured in equation (2.44).

The impracticality of considering all members in a set from $\mathcal{S}_n^f$ comes primarily from the number of haplotype and proportion combinations consistent with a set of observed allele proportions. Expressed using our chosen variables, for any observation the cardinality of the sets $\mathcal{H}_{n,z}^f$ and $\mathcal{Y}_{n,\mathbf{h}}^f$ will be too high to consider all members when we consider situations with a desired number of loci as discussed earlier in §2.2. Example 2.30 demonstrates this impracticality.

Example 2.30 (Impracticality of Exhaustive Search). Consider the case where $N_l = 8$ and each locus has 4 possible alleles. We observe the following allele proportions for an observation. Since the first locus has four non-zero allele proportions, we know

Locus							
1	2	3	4	5	6	7	8
$A_{1,1} : 0.30$	$A_{2,1} : 0.20$	$A_{3,1} : 0.80$	$A_{4,1} : 0.10$	$A_{5,1} : 0.29$	$A_{6,1} : 0.35$	$A_{7,1} : 0.00$	$A_{8,1} : 0.00$
$A_{1,2} : 0.39$	$A_{2,2} : 0.16$	$A_{3,2} : 0.00$	$A_{4,2} : 0.81$	$A_{5,2} : 0.30$	$A_{6,2} : 0.35$	$A_{7,2} : 0.10$	$A_{8,2} : 0.00$
$A_{1,3} : 0.20$	$A_{2,3} : 0.29$	$A_{3,3} : 0.00$	$A_{4,3} : 0.09$	$A_{5,3} : 0.30$	$A_{6,3} : 0.00$	$A_{7,3} : 0.20$	$A_{8,3} : 0.40$
$A_{1,4} : 0.11$	$A_{2,4} : 0.11$	$A_{3,4} : 0.20$	$A_{4,4} : 0.00$	$A_{5,4} : 0.11$	$A_{6,4} : 0.30$	$A_{7,4} : 0.70$	$A_{8,4} : 0.60$

that the MOI that generated this observation is at least 4. Since each locus has four *possible* alleles, there are $4^8 = 65536$ different possible haplotypes before considering which haplotypes are even consistent with the data. Since the vectors in the set $\mathcal{H}$ capture the combinations of size 4 of these 65536 haplotypes, this set will have approximately $\binom{65536}{4} = 7.69\text{e}{+}17$ members. For each member of $\mathcal{H}$, we would then have an infinite number of elements in $\mathcal{Y}$ for the proportions of the four selected haplotypes. This is simply a property of each $\mathbf{y} \in \mathcal{Y}$ being real-valued.

Even when we move to $\mathcal{S}_n^f$, there are $4 \cdot 4 \cdot 2 \cdot 3 \cdot 4 \cdot 3 \cdot 2 \cdot 2 = 4608$ different *feasible* haplotypes. While this represents a reduction of 93%, we still have $\binom{4608}{4} = 1.87\text{e}{+}13$ different members in $\mathcal{H}_{n,z}^f$. And each still has an infinite number of possible proportions that are consistent, even with the restrictions that have been placed on this set.

□

Regarding the low relative probabilities mentioned above, this will be evident in equation (2.44) and due to one or more low component probabilities. This could be due to the choice of an unlikely MOI, an unlikely set of haplotypes, or an unlikely/poor alignment between the implied and observed allele proportions. Example 2.18 demonstrated how this might happen due to a particular cause: the low probability of observing $\mathcal{X}_n$ given $\mathcal{U}_n$. A low value for this equation will have two consequences: first, the impact on the computation in equation (2.45) will be negligible, and second, the probability for the corresponding solution, as captured in equation (2.43), will be low due to a low value in the denominator. In short, such a combination can be safely discarded knowing that it has a low probability of the being the solution of interest and that it has a minimal impact on the probability (through equation (2.45)) of other potential solutions.

We propose a reduction from the feasible set $\mathcal{S}_n^f$ to the *reduced set* $\mathcal{S}_n^r = \{\mathcal{Z}_n^r, \mathcal{H}_{n,z}^r, \mathcal{Y}_{n,\mathbf{h}}^r, \mathcal{U}_{n,\mathbf{y}}^r\}$ that makes the problem computationally tractable. Each component is presented in detail in Table 2.10.

Set	Description
$\mathcal{Z}_n^r$	Set of all integers $\{z : \text{minMOI} \leq z \leq \text{maxMOI}\}$.
$\mathcal{H}_{n,z}^r$	Set of haplotype vectors found during tree search. By construction, $h_i \in \{0,1\}$, is of length N_{haps}, and $\sum_i h_i = z$, and is consistent with the non-zero alleles observed in $\boldsymbol{X}_n$.
$\mathcal{Y}_{n,\mathbf{h}}^r$	For each vector $\mathbf{h} \in \mathcal{H}_{n,z}^r$, there is a single vector in $\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^r$ with the proportions for each haplotype in $\mathbf{h}$. By construction, $\mathbf{y}$ is consistent with its corresponding $\mathbf{h}$.
$\boldsymbol{U}_{n,\mathbf{y}}^r$	For each vector $\mathbf{h} \in \mathcal{H}_{n,z}^r$, there is a single set of vectors $\{\mathbf{U}^{(l)}, l = 1, 2, \ldots, N_l\}$ consistent with $\mathbf{h}$ (and thus $\mathbf{y}$).

Table 2.10: Search Set - Restricted ($\mathcal{S}_n^r = \{\mathcal{Z}_n^r, \mathcal{H}_{n,z}^r, \mathcal{Y}_{n,\mathbf{h}}^r, \boldsymbol{U}_{n,\mathbf{y}}^r\}$). Each set is a subset of its counterpart in $\mathcal{S}_n^f$ and is determined through the tree search algorithm described in §2.7.2.

The nested relationship between the three collections of variables – as documented in Tables 2.8, 2.9, and 2.10 – is depicted in Figure 2.11.

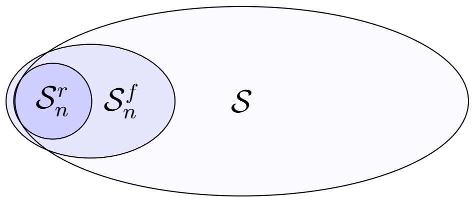


Figure 2.11: Three Collections of Variables. The diagram shows the relationship between the three collections. In particular, $\mathcal{S}_n^r \subset \mathcal{S}_n^f \subset \mathcal{S}$. Note that the diagram is not drawn to scale.

Viewed in this framework, the purpose of the SSI step discussed in §2.7.2 is to construct this set $\mathcal{S}_n^r$ for any observation n. In other words, its purpose is to reduce the total number of solutions considered in the EM phase of the algorithm.

Finally, it is useful to relate the variables in Table 2.10 to the concept of a proposed solution introduced in §2.4.1.3. The information in a proposed solution $\widehat{Sol}_{n,j}$ – namely, which haplotypes are proposed (and thus how many) and in what proportions – is exactly what is captured by some $\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^r$.[30] Thus we can write

$$\mathbf{y}(\widehat{Sol}_{n,j})$$

to get the implied haplotype proportions vector from a particular solution and we can write

$$\widehat{Sol}_{n,j}(\mathbf{y})$$

to indicate the solution equivalent to a particular value $\mathbf{y}$.

2.7.4.2.5 Maximization Step

In the maximization step, we need to find the estimates of the parameter set Θ that maximize the expression $Q(\Theta^{(t)}|\Theta^{(t-1)})$, which itself is a function of the current iteration's expected values in equation (2.41).

[30]Recall that the elements of $\mathcal{Z}_n^r$ simply indicate how many haplotypes and the elements of $\mathcal{H}_{n,z}^r$ indicate which haplotypes have been chosen. Neither indicates the proportions, yet the value for each is implied by a particular $\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^r$. On the other hand, the variable set $\{\mathbf{U}^{(l)}, l = 1, 2, \ldots, N_l\} \in \boldsymbol{U}_{n,\mathbf{y}}^r$ states the implied allele proportions but does not indicate which haplotypes were used or in what proportions.

Parameter p. For the parameter $\mathbf{p}$, we perform a constrained optimization of equation (2.41). The constraint is that $\sum_{i=1}^{\text{maxMOI}} \hat{p}_i^{(t)} = 1$. The Lagrangian is thus (2.41) minus a term representing the constraint:

$$\mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)})) = Q(\Theta^{(t)}|\Theta^{(t-1)}) - \lambda\left(\sum\nolimits_i \hat{p}_i^{(t)} - 1\right) \tag{2.46}$$

We see in equation (2.41) that the only place where $p_i^{(t)}$ appears is in term $\log(f_{Z_n}(z))$. Given the assumed distribution on Z_n,

$$\begin{aligned}
\log(f_{Z_n}(z)) &= \log[(1!/(n_1! \ldots n_{\text{maxMOI}}!)(p_1^{(t)})^{n_1} \ldots (p_{\text{maxMOI}}^{(t)})^{n_{\text{maxMOI}}}] \\
&= \log[(1!/(n_1! \ldots n_{\text{maxMOI}}!)] + \sum_{i=1}^{n_{\text{maxMOI}}} n_i \log(p_i^{(t)}) \\
\frac{\partial \log(f_{Z_n}(z))}{\partial p_i^{(t)}} &= n_i/p_i^{(t)} \\
&= 1/p_i^{(t)}
\end{aligned} \tag{2.47}$$

Using the result above and differentiating (2.46) with respect to $\hat{p}_i^{(t)}$ and setting the result to zero yields,

$$\begin{aligned}
\frac{\partial \mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)}))}{\partial \hat{p}_i^{(t)}} &= \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}_{n,\mathbf{y}}^f} \sum_{\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^f} \sum_{\mathbf{h} \in \mathcal{H}_{n,z}^f} \sum_{z \in \mathcal{Z}_n^f} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)} \right] \cdot \frac{\mathbb{1}(z = i)}{\hat{p}_i^{(t)}} - \lambda \\
&\triangleq 0 \implies \\
\lambda &= \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}_{n,\mathbf{y}}^f} \sum_{\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^f} \sum_{\mathbf{h} \in \mathcal{H}_{n,z}^f} \sum_{z \in \mathcal{Z}_n^f} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)} \right] \cdot \frac{\mathbb{1}(z = i)}{\hat{p}_i^{(t)}} \\
\hat{p}_i^{(t)} &= \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}_{n,\mathbf{y}}^f} \sum_{\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^f} \sum_{\mathbf{h} \in \mathcal{H}_{n,z}^f} \sum_{z \in \mathcal{Z}_n^f} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)} \right] \cdot \frac{\mathbb{1}(z = i)}{\lambda}
\end{aligned} \tag{2.48}$$

Differentiating (2.46) with respect to λ and setting the result to zero yields,

$$\begin{aligned}
\frac{\partial \mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)}))}{\partial \lambda} &= \sum\nolimits_i \hat{p}_i^{(t)} - 1 \\
\frac{\partial \mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)}))}{\partial \lambda} &\triangleq 0 \implies
\end{aligned}$$

$$\sum_i \hat{p}_i^{(t)} = 1 \tag{2.49}$$

Using the result in (2.49) and applying it to (2.48) gives,

$$\sum_i \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\cdot\frac{\mathbb{1}(z=i)}{\lambda} = \sum_i \hat{p}_i^{(t)}$$

$$= 1 \implies$$

$$\sum_i \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\mathbb{1}(z=i) = \lambda \tag{2.50}$$

Finally, applying (2.50) to (2.48) leads to,

$$\hat{p}_i^{(t)} = \frac{\sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\mathbb{1}(z=i)}{\sum_i \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\mathbb{1}(z=i)}$$

$$= \frac{\sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\mathbb{1}(z=i)}{\sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]} \tag{2.51}$$

Note as well from equation (2.47) that,

$$\frac{\partial^2 \log(f_{Z_n}(z))}{\partial p_i^{(t)2}} = -1/p_i^{(t)2} < 0,$$

indicating that the function is concave and thus identifying the zero of the first partial derivative will yield a maximum.

We see in equation (2.51) that the estimate $\hat{p}_i^{(i)}$ is an increasing function of both its frequency of occurrence in the set Z_n[31] (as captured in $\mathbb{1}(z=i)$) and the probability of the solution it is associated with (as captured in the expectation component). This is intuitive: MOIs that are frequently feasible and lead to good fits with the observed data should be presumed to be more frequent.

[31]Recall that different observations may have sets of valid z's. That is, we may have $Z_n \neq Z_m$ for $n \neq m$.

Parameter $\boldsymbol{\pi}$. Like $\mathbf{p}$, $\boldsymbol{\pi}$ is a parameter in a multinomial distribution; as such, its estimation is similar. Once again, we are performing an optimization with a similar constraint, namely $\sum_{i=1}^{N_{\text{haps}}} \hat{\pi}_i^{(t)} = 1$. The Lagrangian is thus

$$\mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)})) = Q(\Theta^{(t)}|\Theta^{(t-1)}) - \lambda \left(\sum\nolimits_i \hat{\pi}_i^{(t)} - 1\right) \tag{2.52}$$

The parameter $\pi_i^{(t)}$ appears only in the term $\log(f_{H_n|Z_n}(\mathbf{h}, z; \boldsymbol{\pi}))$ with,

$$\begin{aligned}
\log(f_{\mathbf{H}_n|Z_n}(\mathbf{h}, z; \boldsymbol{\pi})) &= \log[(z!/(n_1! \dots n_{N_{\text{haps}}}!)(\pi_1^{(t)})^{n_1} \dots (\pi_{N_{\text{haps}}}^{(t)})^{n_{N_{\text{haps}}}}] \\
&= \log[(z!/(n_1! \dots n_{N_{\text{haps}}}!)] + \sum_{i=1}^{N_{\text{haps}}} n_i \log(\pi_i^{(t)}) \\
\frac{\partial \log(f_{\mathbf{H}|Z_n}(\mathbf{h}, z; \boldsymbol{\pi}))}{\pi_i^{(t)}} &= n_i / p_i^{(t)} \\
&= (1/\pi_i^{(t)}) h_i
\end{aligned}$$

In the last line, recall that by construction $h_i \in \{0, 1\}$ indicates whether or not haplotype i is present in $\mathbf{h}$.[32] Differentiating (2.52) with respect to $\hat{\pi}_i^{(t)}$ and setting the result to zero yields,

$$\begin{aligned}
\frac{\partial \mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)}))}{\partial \hat{\pi}_i^{(t)}} &= \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}_{n,\mathbf{y}}^f} \sum_{\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^f} \sum_{\mathbf{h} \in \mathcal{H}_{n,z}^f} \sum_{z \in \mathcal{Z}_n^f} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)} \right] \cdot \frac{h_i}{\hat{\pi}_i^{(t)}} - \lambda \\
&\triangleq 0 \implies \\
\lambda &= \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}_{n,\mathbf{y}}^f} \sum_{\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^f} \sum_{\mathbf{h} \in \mathcal{H}_{n,z}^f} \sum_{z \in \mathcal{Z}_n^f} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)} \right] \cdot \frac{h_i}{\hat{\pi}_i^{(t)}} \\
\hat{\pi}_i^{(t)} &= \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}_{n,\mathbf{y}}^f} \sum_{\mathbf{y} \in \mathcal{Y}_{n,\mathbf{h}}^f} \sum_{\mathbf{h} \in \mathcal{H}_{n,z}^f} \sum_{z \in \mathcal{Z}_n^f} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)} \right] \cdot \frac{h_i}{\lambda}
\end{aligned} \tag{2.53}$$

Differentiating (2.52) with respect to λ and setting the result to zero yields,

$$\frac{\partial \mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)}))}{\partial \lambda} = \sum\nolimits_i \hat{\pi}_i^{(t)} - 1$$

[32]Recall that $\mathbf{h}$ is an indicator vector, whose ith entry indicates whether haplotype i is present.

$$\begin{aligned}\frac{\partial \mathcal{L}(Q(\Theta^{(t)}|\Theta^{(t-1)}))}{\partial \lambda} &\triangleq 0 \implies \\ \textstyle\sum_i \hat{\pi}_i^{(t)} &= 1\end{aligned} \tag{2.54}$$

Using the result in (2.54) and applying it to (2.53) gives,

$$\begin{aligned}\sum_i \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in \mathcal{H}^f_{n,z}} \sum_{z\in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right] \cdot \frac{h_i}{\lambda} &= \textstyle\sum_i \hat{\pi}_i^{(t)} \\ &= 1 \implies \end{aligned}$$

$$\sum_i \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in \mathcal{H}^f_{n,z}} \sum_{z\in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right] h_i = \lambda \tag{2.55}$$

And applying (2.55) to (2.53) gives,

$$\begin{aligned}\hat{\pi}_i^{(t)} &= \frac{\sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in \mathcal{H}^f_{n,z}} \sum_{z\in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right] h_i}{\sum_i \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in \mathcal{H}^f_{n,z}} \sum_{z\in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right] h_i} \\ &= \frac{\sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in \mathcal{H}^f_{n,z}} \sum_{z\in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right] h_i}{\sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in \mathcal{H}^f_{n,z}} \sum_{z\in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right]}\end{aligned} \tag{2.56}$$

As for p_i above, the second derivative of the function is negative, indicating the function is concave and that a global maximum will be found.

Note that the estimate $\hat{\pi}_i^{(i)}$ is an increasing function of both its frequency of occurrence in the set $\mathcal{H}^f_{n,z}$ (as captured by h_i) and the probability of the solution it is associated with (as captured in the expectation component). An important implication of this approach is that if in a collection of observations a given haplotype i is never consistent with the observed data, then its estimated frequency will equal zero, since there will be no element $\mathbf{h} \in \mathcal{H}^f_{n,z}$ for any n such that $h_i = 1$.

Parameter c. For $\mathbf{c} = \{c^{(l)}, l = 1, \ldots, N_l\}$, each $c^{(l)}$ will be estimated separately using the same method. As such, each locus will be handled separately for its own parameter.

Here, there is no constraint that the parameters sum to any particular number. The only constraint is that any estimate be feasible; that is, $\hat{c}^{(l,t)} > 0$ where l and t indicate the locus and iteration number, respectively.

As always in the maximization step, we seek an estimate $\hat{c}^{(l,t)}$ for our parameter of interest such that,

$$\begin{aligned}\hat{c}^{(l,t)} &:= \underset{c^{(l)}>0}{\operatorname{argmax}}\, Q(\Theta^{(t)}|\Theta^{(t-1)}) \\ &= \underset{c^{(l)}>0}{\operatorname{argmax}} \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right] \cdot \\ &\qquad \left[\sum_{k=1}^{N_l} \log\left(f_{\mathbf{X}^{(k)}_n|\mathbf{U}^{(k)}_n}(\mathbf{x},\mathbf{u}|c^{(k)})\right) + \log\xi + \log\left(f_{\mathbf{H}_n|Z_n}(\mathbf{h},z;\boldsymbol{\pi})\right) + \log\left(f_{Z_n}(z;\mathbf{p})\right)\right]\end{aligned} \tag{2.57}$$

However, since each $c^{(l)}$ appears only in the first term of the second line of equation (2.57), and such terms are relevant only when $k = l$, this equation is equivalent to,

$$\hat{c}^{(l,t)} = \underset{c^{(l)}>0}{\operatorname{argmax}} \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u}\in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}} \sum_{z\in\mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z) \middle| \Omega, \Theta^{(t-1)}\right] \left[\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u}|c^{(l)})\right)\right] \tag{2.58}$$

That is, we seek a single value for $c^{(l)}$, used by all observations in assessing their fit with the observed data at locus l, that maximizes the objective function in equation (2.58). Equation (2.58) can be viewed as a weighted sum, with the weight equal to the $\Pr[\cdot]$ component. Thus solutions with a higher degree of probability have a higher weight in determining $\hat{c}^{(l,t)}$.[33]

The log expression in the final pair of square brackets in equation (2.58) can be expressed after substitution from equation (2.27) as,

$$\begin{aligned}&\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u};c^{(l)})\right) \\ &\quad = \log\left[\frac{1}{B(\{c^{(l)}u_i + 1; i\in\mathcal{NZ}(\mathbf{u})\})} \prod_{i\in\mathcal{NZ}(\mathbf{u})} x_i^{c^{(l)}u_i}\right]\end{aligned}$$

[33]This is in fact true of the estimation of all parameters in the model.

$$
\begin{aligned}
&= -\log\left[B(\{c^{(l)}u_i+1; i\in\mathcal{NZ}(\mathbf{u})\})\right] + \sum_{i\in\mathcal{NZ}(\mathbf{u})} c^{(l)}u_i\log(x_i) \\
&= -\log\left[\frac{\prod_{i\in\mathcal{NZ}(\mathbf{u})}\Gamma(c^{(l)}u_i+1)}{\Gamma\left(\sum_{i\in\mathcal{NZ}(\mathbf{u})} c^{(l)}u_i+1\right)}\right] + \sum_{i\in\mathcal{NZ}(\mathbf{u})} c^{(l)}u_i\log(x_i) \\
&= -\sum_{i\in\mathcal{NZ}(\mathbf{u})}\log\left[\Gamma(c^{(l)}u_i+1)\right] + \log\left[\Gamma\left(c^{(l)}+|\mathcal{NZ}(\mathbf{u})|\right)\right] + \sum_{i\in\mathcal{NZ}(\mathbf{u})} c^{(l)}u_i\log(x_i) \\
&= \log\left[\Gamma\left(c^{(l)}+|\mathcal{NZ}(\mathbf{u})|\right)\right] + \sum_{i\in\mathcal{NZ}(\mathbf{u})} c^{(l)}u_i\log(x_i) - \log\left[\Gamma(c^{(l)}u_i+1)\right] \qquad (2.59)
\end{aligned}
$$

Recall that the goal is to find the value of $c^{(l)}$ that maximizes the expression in equation (2.58), of which equation (2.59) is a critical part. There are two possible numerical approaches to finding $\hat{c}^{(l,t)}$,

1. use a gradient descent approach, varying $c^{(l)}$ along the way, to attempt to find a global maximum, or
2. compute the partial derivative of equation (2.58) with respect to $c^{(l)}$ and find the value of this parameter that sets the derivative to zero.

Both methods were implemented and tested; however, the second proved to be approximately 50 times faster with no loss in accuracy. It was therefore chosen and is explained in more detail next.

Differentiating equation (2.58) with respect to $\mathbf{c}^{(l)}$ yields,

$$
\begin{aligned}
\frac{\partial Q(\Theta^{(t)}|\Theta^{(t-1)})}{\partial c^{(l)}} &= \frac{\partial}{\partial c^{(l)}}\left[\sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n}\Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\cdot\right. \\
&\qquad\left.\left[\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u}|c^{(l)})\right)\right]\right] \\
&= \sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n}\Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\cdot \\
&\qquad\frac{\partial}{\partial c^{(l)}}\left[\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u}|c^{(l)})\right)\right] \qquad (2.60)
\end{aligned}
$$

Given equation (2.59), the derivative in equation (2.60) is as follows,

$$
\frac{\partial}{\partial c^{(l)}}\left[\log\left(f_{\mathbf{X}^{(l)}_n|\mathbf{U}^{(l)}_n}(\mathbf{x},\mathbf{u};c^{(l)})\right)\right]
$$

$$= \digamma\left(c^{(l)} + |\mathcal{NZ}(\mathbf{u})|\right) + \sum_{i \in \mathcal{NZ}(\mathbf{u})} \left[u_i \log(x_i) - u_i \digamma(c^{(l)} u_i + 1)\right] \tag{2.61}$$

where $\digamma[\cdot]$ is the digamma function equal to the derivative of the log of the gamma function [1, p. 258].

From (2.61), we see that when $|\mathcal{NZ}(\mathbf{u})| = 1$ and thus $u_i = 1$ for a single i, the derivative is zero. This is intuitive; in our particular implementation, in cases where at a locus for an observation there is only one non-zero allele, $u_i = x_i = 1$ by definition and no information is contained regarding the degree to which proportions vary from their implied values.

For all other cases, putting the previous results together, the objective is to find $c^{(l)}$ such that,

$$0 = \frac{\partial}{\partial c^{(l)}} \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y} \in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h} \in \mathcal{H}^f_{n,z}} \sum_{z \in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)}\right] \left[\log\left(f_{\mathbf{X}^{(l)}_n | \mathbf{U}^{(l)}_n}(\mathbf{x}, \mathbf{u}; c^{(l)})\right)\right]$$

$$= \sum_{n=1}^{N_{\text{samp}}} \sum_{\boldsymbol{u} \in \boldsymbol{U}^f_{n,\mathbf{y}}} \sum_{\mathbf{y} \in \mathcal{Y}^f_{n,\mathbf{h}}} \sum_{\mathbf{h} \in \mathcal{H}^f_{n,z}} \sum_{z \in \mathcal{Z}^f_n} \Pr\left[I_n(\boldsymbol{u}, \mathbf{y}, \mathbf{h}, z) \middle| \Omega, \Theta^{(t-1)}\right] \cdot$$

$$\left[\digamma\left(c^{(l)} + |\mathcal{NZ}(\mathbf{u})|\right) + \sum_{i \in \mathcal{NZ}(\mathbf{u})} \left[u_i \log(x_i) - u_i \digamma(c^{(l)} u_i + 1)\right]\right]$$

Note that no initial estimate for any $c^{(l)}$ is ever used. The path of estimates for $\mathbf{c}$ will be dependent on the estimates for the other parameters, however.

An assumption of this approach is that $Q(\Theta^{(t)}|\Theta^{(t-1)})$ is concave with respect to $c^{(l)}$. This is stated as a proposition and proved in §2.A.

Proposition 2 (Concavity of $Q(\Theta^{(t)}|\Theta^{(t-1)})$ with respect to $c^{(l)}$). The function $Q(\Theta^{(t)}|\Theta^{(t-1)})$ in equation (2.41) is concave with respect to the concentration parameter $c^{(l)}$.

$\square$

To demonstrate the validity of this approach, consider the following checks. As a first test, we set the concentration parameter c to the value 100 and let $\alpha = (0.2, 0.3, 0.5)$;[34]

[34]As will be seen in §4, this value for c is close the value for a real world dataset.

therefore, the parameter vector is $c\alpha + 1 = \{21, 31, 51\}$. We then use this parameter vector to produce four different sets of randomly generated IID observations of size 20, 50, 100, and 1000. For each set, we plot the value of the log likelihood of the data over 1001 equally spaced values for c in the range $(0, 200]$. We want to see how the likelihood varies for different values of c and n, with the knowledge that in fact $c = 100$, and identify at what point the function reaches a maximum. The results are presented in Figure 2.12. We see several things. First, the shape of the curve appears to differ relatively little from one sample size to another. In all cases, the curve appears concave, with a peak near the true value $c = 100$; however, the larger the sample the closer the maximum point is to the underlying parameter value of 100. It is also true that the curve is nearly flat at its maximum, indicating that values of c near the underlying parameter value produce a likelihood very close to that produced by the true value. Overall, we take the results as an encouraging sign: due to the concavity of the function, if our method can locate the value that maximizes the objective function this value should be close to the true value.

Having established the properties of the objective function, consider now a second test to determine our ability to accurately estimate c. We keep $\alpha = (0.2, 0.3, 0.5)$, consider c at the six different values 20, 30, 50, 100, 200, and 500 and consider sample sizes n equal to the five different values 10, 50, 100, 500, and 1000. For each of the $6 \cdot 5 = 30$ c/n pairs, we generate n IID Dirichlet random vectors assuming the value c in question, and we do each pair 1000 times.

We first group the results by c (regardless of the value of n for a particular trial) and plot the estimated value for c versus the true value. This is depicted in Figure 2.13. We see that for all values of c, the estimates are centered near the truth, as evidenced by each horizontal blue line indicating the true c value passing through the median of the corresponding boxplot. While the 25th to the 75th percentile interval is relatively narrow in each case, for each value of c the most extreme outliers appear to deviate from the true value by a factor of roughly three.

We then look at the same results in terms of the error, considering the results for each of the 30 different combinations. The results for the errors on an absolute basis are presented in Figure 2.14. We see that, consistent with intuition, that the *absolute* errors decrease as n increases and increase as c increases. Figure 2.15 displays the same results

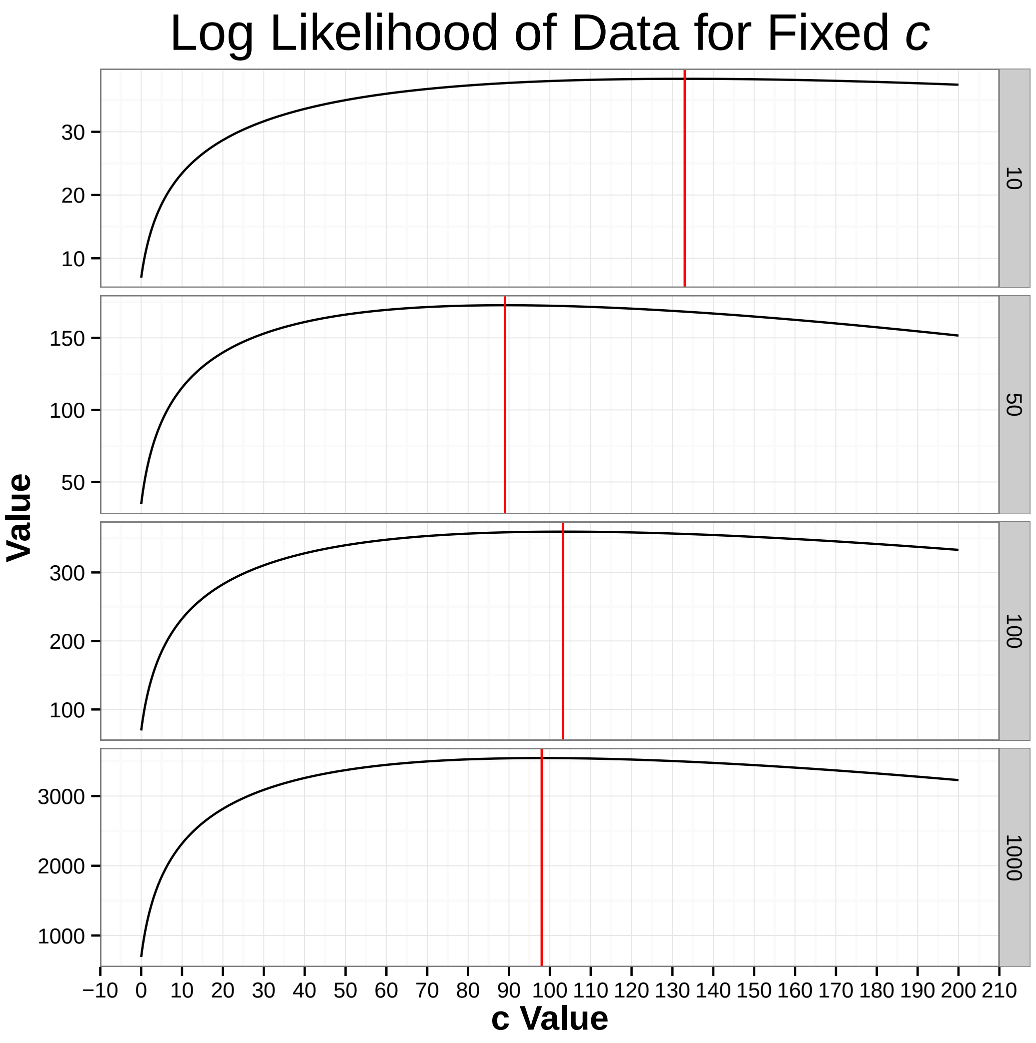


Figure 2.12: Log Likelihood of Data for Fixed c. Each plot is for a particular sample size and shows the value of the log likelihood for various values of c when the true $c = 100$. The red lines indicate the value of c at which the function is maximized. In all cases, the red line is near the true value of 100, although it is closer to this true value the greater the sample size.

on a relative basis. By "relative", we mean the error relative to the relevant true value of c. We see that the relative error decreases both with the sample size and c.

The conclusion of these tests is that the estimation method appears sound and will benefit from large sample sizes. The improvement in performance appears to be greatest

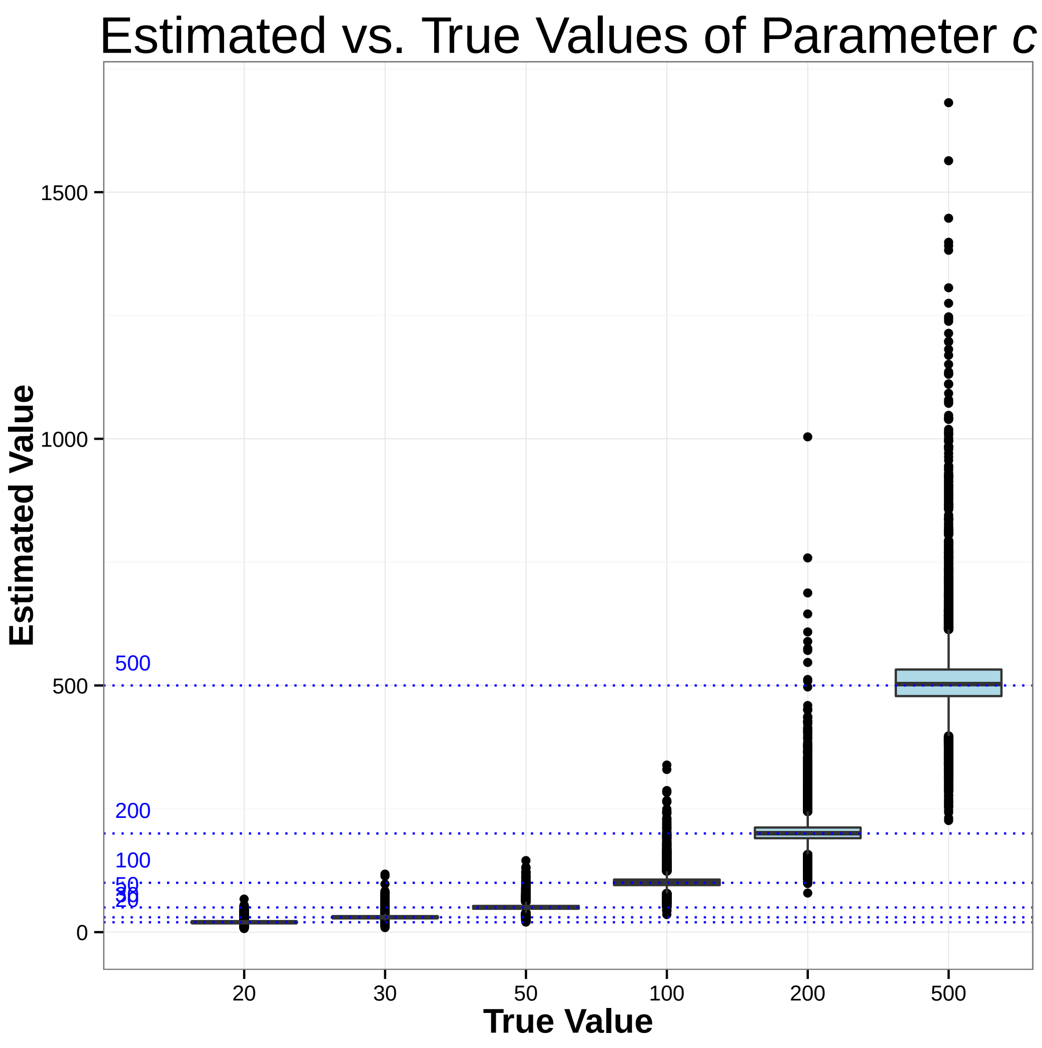


Figure 2.13: Estimated vs. True c Values by Sample Size - Absolute Scale. Each of the six boxplots shows the distribution of $10 + 50 + 100 + 500 + 1000 = 1660$ estimates for a single value of c using Dirichlet distributed data generated with the true c value. At the same time, the horizontal blue lines show the true value for c. For all values of c, the estimates are centered at the true value, as evidenced by the appropriate blue line passing over or close to the horizontal line in each box. As c increases, the outliers increase in *absolute* (as opposed to relative) value from the truth.

for lower values of n; once the sample size reaches 500 there is little improvement for higher sample sizes ($n = 1000$, in the tests performed).

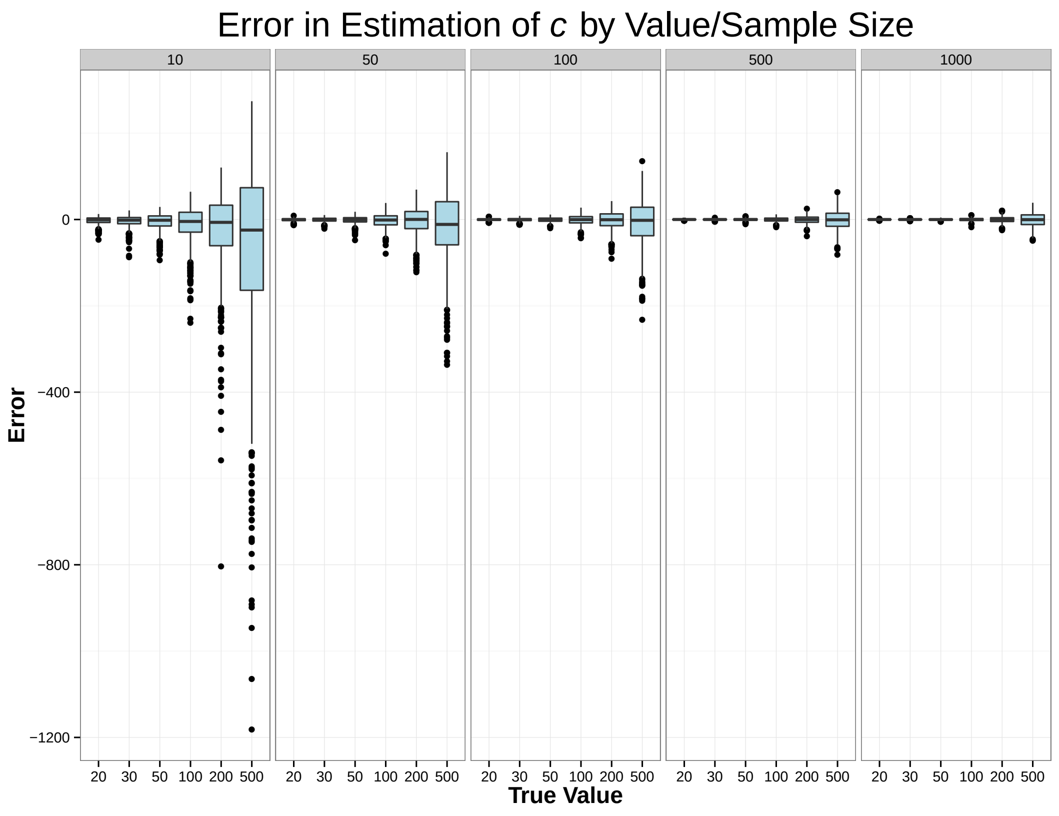


Figure 2.14: Error in Estimation of c by n and c - Absolute Scale. There are five main panels arranged horizontally, one per sample size. Within each panel, the distribution of absolute errors is shown for each of the six values of c. As the sample size increases, the errors decrease for all values of c. Similarly, within each sample size, the absolute errors increase as c increases.

2.8 Conclusion

The `LITSE` algorithm addresses the problem of identifying haplotype solutions using a data generating model with two primary components: a tree-based search for individual solutions and an EM component that uses all solutions jointly to propose probability-weighted solutions that refect the estimated parameters included in the data generating distribution.

The algorithm is divided into three phases. In the first, the SSI phase, the algorithm builds one tree for each observation/MOI combination considered. Here, runtime param-

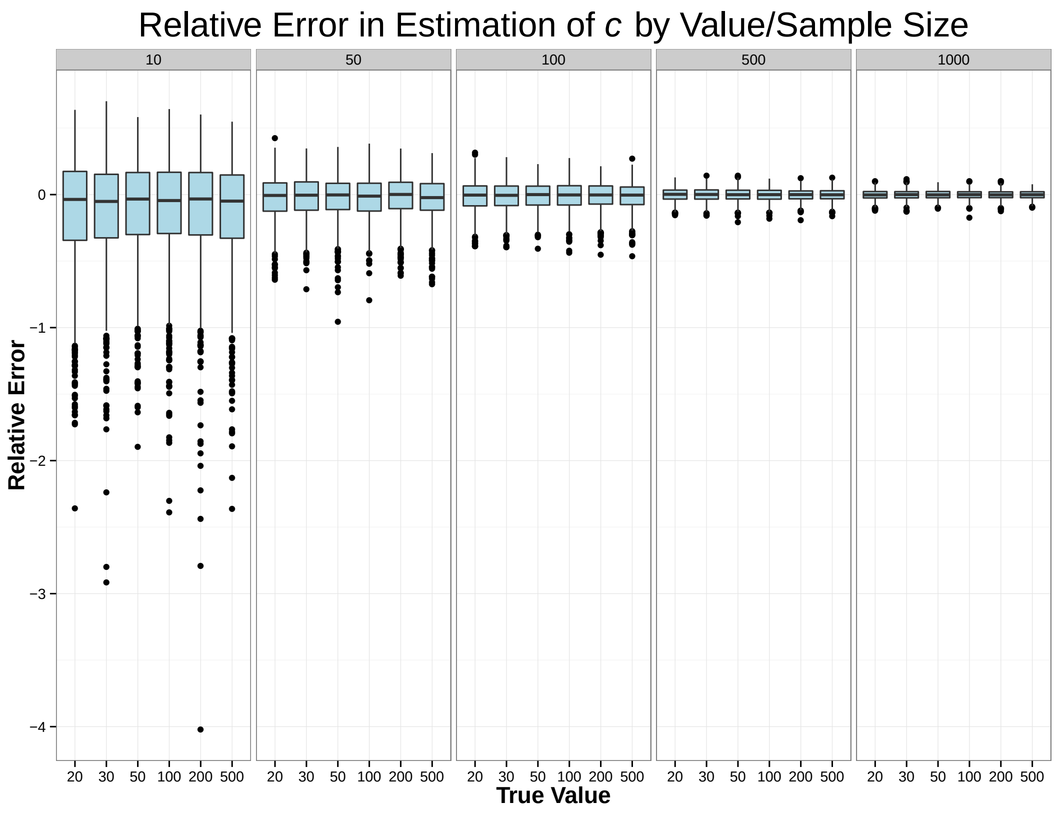


Figure 2.15: Error in Estimation of c by n and c - Relative Scale. There are five main panels arranged horizontally, one per sample size. Within each panel, the distribution of relative errors is shown for each of the six values of c. See the caption of Figure 2.14 for an explanation of the figure. On a relative basis. The relative error decreases both with the n and c.

eters control the size of the search space. The algorithm performs a greedy search using as a heuristic the interim error implied by the partial solution and the observed data. In the second phase, the algorithm addresses the missing data problem by proposing complete solutions for observations with missing locus data. The third and final component is an iterative, EM-based assessment of the probabilities of the solutions and the estimation of the underlying data-generating parameters.

2.9 Future Work

There are several areas in which the `LITSE` algorithm can be enhanced, both for runtime performance potential accuracy. Some key suggestions are listed below.

1. Parallelization. At present, the algorithm has been implemented in sequential form. That is, each observation is handled one at a time in a queue in the SSI procedure. Since the processing of separate observations is independent in this phase (this is *not* true in later phases), this processing could easily be parallelized. While the SSR procedure with its EM implementation cannot be entirely parallelized, the most significant step there – the re-estimation of the $\mathbf{c}$ parameter for each individual locus – can be parallelized as this is again independent. These two changes would likely lead to a significantly quicker implementation.
2. Dynamic choice of runtime parameters. Currently, the runtime parameters – `mer`, `smnc`, `hmnc`, etc. – are set once for the entire algorithm. A possible refinement is to adjust these parameters in real time as the algorithm processes each tree.
3. Incorporation with artifact detection. As mentioned earlier, currently the algorithm treats the preprocessing of the data, in particular the removal of artifacts, as a separate precursory process. In particular, the algorithm assumes that the stated observed alleles must be those in the actual solution. This assumption might be relaxed to take a probabilistic approach.
4. Bayesian approach. Currently, the algorithm assumes no prior knowledge of the distribution of any of the parameters. A Bayesian approach would impose a prior distribution on the parameters of interest and would let the data adapt the posterior distribution. Such a method would likely involve Markov Chain Monte Carlo methods, which have been used elsewhere in related research as discussed in §2.5.4.

Appendices

2.A Proofs

Proof of Proposition 2. Given the first partial derivative of the $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function with respect to $c^{(l)}$ as presented in equations (2.60) and (2.61), to demonstrate concavity it suffices to show that the second derivative is everywhere non-positive. As the first partial derivative has already been computed, using it as a starting point to compute the second partial derivative yields the following,

$$\begin{aligned}
\frac{\partial^2 Q(\Theta^{(t)}|\Theta^{(t-1)})}{\partial c^{(l)\ 2}} &= \frac{\partial}{\partial c^{(l)}}\left[\sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n}\Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\cdot\right.\\
&\quad\left.\left[F\left(c^{(l)}+|\mathcal{NZ}(\mathbf{u})|\right)+\sum_{i\in\mathcal{NZ}(\mathbf{u})}\left[u_i\log(x_i)-u_iF(c^{(l)}u_i+1)\right]\right]\right]\\
&= \sum_{n=1}^{N_{\text{samp}}}\sum_{\boldsymbol{u}\in\boldsymbol{U}^f_{n,\mathbf{y}}}\sum_{\mathbf{y}\in\mathcal{Y}^f_{n,\mathbf{h}}}\sum_{\mathbf{h}\in\mathcal{H}^f_{n,z}}\sum_{z\in\mathcal{Z}^f_n}\Pr\left[I_n(\boldsymbol{u},\mathbf{y},\mathbf{h},z)\middle|\Omega,\Theta^{(t-1)}\right]\cdot\\
&\quad\frac{\partial}{\partial c^{(l)}}\left[F\left(c^{(l)}+|\mathcal{NZ}(\mathbf{u})|\right)+\sum_{i\in\mathcal{NZ}(\mathbf{u})}\left[u_i\log(x_i)-u_iF(c^{(l)}u_i+1)\right]\right]
\end{aligned} \tag{2.62}$$

The probability in equation (2.62) is always positive. Therefore, the sign of this equation depends on the sign of the derivative component. This component equals,

$$\begin{aligned}
&\frac{\partial}{\partial c^{(l)}}\left[F\left(c^{(l)}+|\mathcal{NZ}(\mathbf{u})|\right)+\sum_{i\in\mathcal{NZ}(\mathbf{u})}\left[u_i\log(x_i)-u_iF(c^{(l)}u_i+1)\right]\right]\\
&=\Psi\left(c^{(l)}+|\mathcal{NZ}(\mathbf{u})|\right)-\sum_{i\in\mathcal{NZ}(\mathbf{u})}u_i^2\Psi(c^{(l)}u_i+1)
\end{aligned} \tag{2.63}$$

where $\Psi(\cdot)$ is the trigamma function, the derivative of the digamma function.

Applying the integral representation of the trigamma function [1],

$$\Psi(z) = -\int_0^1 \frac{t^{z-1}}{1-t}\log(t)dt, \tag{2.64}$$

to equation (2.63) produces,

$$\begin{aligned}
&\Psi\left(c^{(l)} + |\mathcal{NZ}(\mathbf{u})|\right) - \sum_{i\in\mathcal{NZ}(\mathbf{u})} u_i^2 \Psi(c^{(l)}u_i + 1) \\
&\quad = -\int_0^1 \frac{t^{c^{(l)}+|\mathcal{NZ}(\mathbf{u})|-1}}{1-t}\log(t)dt + \sum_{i\in\mathcal{NZ}(\mathbf{u})} u_i^2 \int_0^1 \frac{t^{c^{(l)}u_i}}{1-t}\log(t)dt \\
&\quad = \int_0^1 \frac{\sum_{i\in\mathcal{NZ}(\mathbf{u})} u_i^2 t^{c^{(l)}u_i} - t^{c^{(l)}+|\mathcal{NZ}(\mathbf{u})|-1}}{1-t}\log(t)dt \qquad (2.65)
\end{aligned}$$

Given that the range of the integral in equation (2.65) is from 0 to 1, the denominator of the integrand is always positive and $\log(t)$ is always negative. Thus to prove that the integral is non-positive, it suffices to show that the numerator $\left(\sum_{i\in\mathcal{NZ}(\mathbf{u})} u_i^2 t^{c^{(l)}u_i}\right) - t^{c^{(l)}+|\mathcal{NZ}(\mathbf{u})|-1}$ in the integrand is non-negative for $t \in (0, 1)$. To this end, note first that, since $0 < u_i \leq 1$ and $\sum_i u_i = 1$ by definition, $\sum_i u_i^2 \leq 1$. [35] Then,

$$\left(\sum_{i\in\mathcal{NZ}(\mathbf{u})} u_i^2 t^{c^{(l)}u_i}\right) - t^{c^{(l)}+|\mathcal{NZ}(\mathbf{u})|-1} \geq \sum_{i\in\mathcal{NZ}(\mathbf{u})} \left(u_i^2 t^{c^{(l)}u_i} - u_i^2 t^{c^{(l)}+|\mathcal{NZ}(\mathbf{u})|-1}\right) \qquad (2.66)$$

We need to show that the right hand side of equation (2.66) is non-negative. To prove this, it suffices to show that $u_i^2 t^{c^{(l)}u_i} \geq u_i^2 t^{c^{(l)}+|\mathcal{NZ}(\mathbf{u})|-1}$ for any $i \in \mathcal{NZ}(\mathbf{u})$. Since $u_i^2 > 0$ and $0 \leq t \leq 1$, this holds if $c^{(l)} + |\mathcal{NZ}(\mathbf{u})| - 1 \geq c^{(l)}u_i$. Compare the terms on either side. Since $u_i \leq 1$, we have that $c^{(l)} \geq c^{(l)}u_i$, and since $|\mathcal{NZ}(\mathbf{u})| \geq 1$, $|\mathcal{NZ}(\mathbf{u})| - 1 \geq 0$. Thus the result is established.

So the function $Q(\Theta^{(t)}|\Theta^{(t-1)})$ is concave with respect to $c^{(l)}$.

$\blacksquare$

Proof of Proposition 1. Given that m denotes the assumed MOI and n denotes the number of distinct observed alleles at locus l, we have by construction that $m \geq n$.

Any valid combination or permutation must have at least one representation of each observed allele, as depicted in Figure 2.16.

[35] The following is a quick proof of this relationship. Assume that $\sum_{i=1}^n x_i = 1$ with $x_i \geq 0\ \forall i$. Then $1 = \sum_{i=1}^n x_i \sum_{i=1}^n x_i = \sum_{i=1}^n \sum_{j=1}^n x_i x_j = \sum_{i=1}^n x_i^2 + \sum_{i\neq j}^n x_i x_j$. Since $x_i \geq 0\ \forall i$, $x_i x_j \geq 0, \sum_{i\neq j}^n x_i x_j \geq 0$ and therefore $\sum_{i=1}^n x_i^2 \leq 1$.

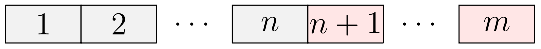


Figure 2.16: Valid Combination/Permutation. Each valid combination or permutation must have at least one representation of each observed allele, represented by blocks filled in gray. If $m > n$, the remaining blocks, indicated in red, can be occupied by any allele. Note that here, for convenience, the gray blocks are positioned contiguously. This need not be the case for a valid combination/permutation.

Combinations

For the value $\mathcal{C}(m, n)$, the number of valid combinations, once each observed allele has been represented at least once, the remaining (if any) $m - n$ elements are "free" to represent any combination of observed alleles. Thus the number of valid combinations equals the total number of combinations of size $m - n$ with all elements drawn from n different categories with replacement.

From [16, p.169], we have that the number of ways select x objects from y categories, allowing for repetition, is

$$\binom{x+y-1}{x}.$$

Applying this to the current problem, we substitute $m - n$ for x and n for y, giving

$$\mathcal{C}(m, n) = \binom{m-n+n-1}{m-n} = \binom{m-1}{m-n},$$

as claimed.

Permutations

The value $\mathcal{P}(m, n)$, the number of valid permutations, can be computed as follows.[36] Any valid permutation will have length m and be composed of categories from 1 to n. Since there are n different categories to choose from and the permutation is of length m, there are n^m possible permutations; the task is to determine the $\mathcal{P}(m, n) \leq n^m$ that are valid.

For each of $j \in \{1, 2, \ldots, n\}$, let $N(c_j)$ denote number of permutations where the j^{th} category is present in the permutation. More generally, let $N(c_{i_1} \ldots c_{i_l})$ denote the

[36]Recall that a valid permutation is one where each category in 1 to n is represented at least once.

number of permutations where categories $i_1, \ldots, i_l$ are present. Then

$$\mathcal{P}(m,n) = N(c_1 c_2 \ldots c_n). \tag{2.67}$$

That is, $\mathcal{P}(m,n)$ is the cardinality of the set of all permutations containing at least one representation from each category.

To assess the right hand side of equation (2.67), we invoke the Principle of Inclusion and Exclusion, as presented in [16, p. 157],

$$\begin{aligned} N_0 = N &- \sum_i N(a_i) + \sum_{i<j} N(a_i a_j) - \sum_{i<j<k} N(a_i a_j a_k) + \ldots \\ &+ (-1)^m \sum_{i_1<\cdots<i_m} N(a_{i_1} \ldots a_{i_m}) + \cdots + (-1)^r N(a_1 a_2 \ldots a_r) \end{aligned} \tag{2.68}$$

where N denotes the cardinality of the complete set, N_0 denotes the number of elements not meeting any criteria, $N(\cdot)$ is as defined above, m denotes the mth term in the sum, and r denotes the total number of categories (thus r is equivalent to n as used at present). Rearranging these terms and dropping the irrelevant N_0[37],

$$\begin{aligned} N(a_1 a_2 \ldots a_r) = (-1)^{r+1} \Bigg[N &- \sum_i N(a_i) + \sum_{i<j} N(a_i a_j) - \sum_{i<j<k} N(a_i a_j a_k) + \ldots \\ &+ (-1)^m \sum_{i_1<\cdots<i_m} N(a_{i_1} \ldots a_{i_m}) + \ldots \\ &+ (-1)^{r-1} \sum_{i_1<\cdots<i_{m-1}} N(a_{i_1} \ldots a_{i_{m-1}}) \Bigg] \end{aligned} \tag{2.69}$$

Applying equation (2.69) to (2.67), we get,

$$\begin{aligned} \mathcal{P}(m,n) = (-1)^{n+1} \Bigg[n^m &- \sum_i N(c_i) + \sum_{i<j} N(c_i c_j) - \sum_{i<j<k} N(c_i c_j c_k) + \ldots \\ &+ (-1)^j \sum_{i_1<\cdots<i_j} N(c_{i_1} \ldots c_{i_j}) + \ldots \\ &+ (-1)^{n-1} \sum_{i_1<\cdots<i_{n-1}} N(c_{i_1} \ldots c_{i_{n-1}}) \Bigg] \end{aligned} \tag{2.70}$$

Next, note that, by construction, all categories $1 \ldots, n$ are equivalently represented in the set of n^m permutations. Thus the value of each term $N(\cdot)$ in equation (2.70) is

[37]The term is irrelevant because all permutations by construction include at least one category

independent of the particular indices chosen. It suffices to compute the value for a single index or set of indices for each term.

Furthermore, the sums in equation (2.70) are simply the number of singletons, pairs, triplets, etc., of the n categories. Thus equation (2.70) can be restated as,

$$\begin{aligned}\mathcal{P}(m,n) = (-1)^{n+1}\Bigg[n^m - \binom{n}{1}N(c_i) + \binom{n}{2}N(c_ic_j) - \binom{n}{3}N(c_ic_jc_k) + \dots \\ + (-1)^j\binom{n}{j}N(c_{i_1}\dots c_{i_j}) + \dots \\ + (-1)^{n-1}\binom{n}{n-1}N(c_{i_1}\dots c_{i_{n-1}})\Bigg]\end{aligned} \tag{2.71}$$

For $N(c_i)$,

$$N(c_i) = n^m - (n-1)^m \ \forall i \tag{2.72}$$

That is, for any single category i, the number of permutations including that category equals the total number of permutations minus the number of permutations excluding it, with the latter quantity equal to the number of permutations of length m from the same set of categories with category i removed.

For $N(c_ic_j)$, the computation is similar to that above, but it is necessary to apply the Principle of Inclusion and Exclusion from equation (2.69). Letting $N(c_i \vee c_j)$ denote the number of permutations containing category c_i and/or c_j and $N(!c_i \wedge !c_j)$ denote the number of permutations containing neither category c_i nor c_j,

$$\begin{aligned}N(c_ic_j) &= -N(c_i \vee c_j) + N(c_i) + N(c_j) \\ &= -\left[n^m - N(!c_i \wedge !c_j)\right] + 2(n^m - (n-1)^m) \\ &= -\left[n^m - (n-2)^m\right] + 2(n^m - (n-1)^m) \\ &= n^m - 2(n-1)^m + (n-2)^m\end{aligned} \tag{2.73}$$

and similarly for $N(c_ic_jc_k)$,

$$\begin{aligned}N(c_ic_jc_k) &= N(c_i \vee c_j \vee c_k) - \left[N(c_i) + N(c_j) + N(c_k)\right] \\ &\quad + \left[N(c_ic_j) + N(c_jc_k) + N(c_ic_k)\right] \\ &= n^m - N(!c_i \wedge !c_j \wedge !c_k) - 3(n^m - (n-1)^m)\end{aligned}$$

$$\begin{aligned}
&+ 3\left[n^m - 2(n-1)^m + (n-2)^m)\right] \\
&= n^m - (n-3)^m - 3(n^m - (n-1)^m) \\
&\quad + 3\left[n^m - 2(n-1)^m + (n-2)^m)\right] \\
&= n^m - 3(n-1)^m + 3(n-2)^m - (n-3)^m
\end{aligned} \tag{2.74}$$

and for $N(c_ic_jc_kc_l)$,

$$\begin{aligned}
N(c_ic_jc_kc_l) &= -N(c_i \vee c_j \vee c_k \vee c_l) + [N(c_i) + N(c_j) + N(c_k) + N(c_l)] \\
&\quad - [N(c_ic_j) + N(c_ic_k) + N(c_ic_l) + N(c_jc_k) + N(c_jc_l) + N(c_kc_l)] \\
&\quad + [N(c_ic_jc_k) + N(c_jc_kc_l) + N(c_ic_kc_l) + N(c_ic_jc_l)] \\
&= -(n^m - N(!c_i \wedge !c_j \wedge !c_k \wedge !c_l)) + 4\left[n^m - (n-1)^m\right] \\
&\quad - 6\left[n^m - 2(n-1)^m + (n-2)^m)\right] \\
&\quad + 4\left[n^m - 3(n-1)^m + 3(n-2)^m - (n-3)^m\right] \\
&= -n^m + (n-4)^m + 4\left[n^m - (n-1)^m\right] \\
&\quad - 6\left[n^m - 2(n-1)^m + (n-2)^m)\right] \\
&\quad + 4\left[n^m - 3(n-1)^m + 3(n-2)^m - (n-3)^m\right] \\
&= n^m - 4(n-1)^m + 6(n-2)^m - 4(n-3)^m + (n-4)^m
\end{aligned} \tag{2.75}$$

From equation (2.69) and the previous results, the general form for $N(c_{i_1} \ldots c_{i_t})$ is,

$$N(c_{i_1} \ldots c_{i_t}) = \sum_{j=0}^{t} (-1)^j \binom{t}{j} (n-j)^m, \tag{2.76}$$

and combining equations (2.67) and (2.76),

$$\mathcal{P}(m,n) = \sum_{j=0}^{n-1} (-1)^j \binom{n}{j} (n-j)^m, \tag{2.77}$$

where we have used the fact that the summand for $j = n$ is always zero when $m > 0$.

■

2.B Illustration of Tree Construction

In this section, we discuss the construction of a tree computed by the `LITSE` algorithm for a simple observation. We focus on the tree itself, as the other components have already been discussed at length.

To keep the example simple, we assume only four loci are of interest and set `smnc` $= 2$. The true haplotype set is as follows, which under the assumption of $c = 1e9$ produces

ID	Loc$_1$	Loc$_2$	Loc$_3$	Loc$_4$	Prop
h_1	$A_{1,1}$	$A_{2,1}$	$A_{3,3}$	$A_{4,2}$	0.3356869
h_2	$A_{1,2}$	$A_{2,3}$	$A_{3,2}$	$A_{4,1}$	0.2832868
h_3	$A_{1,2}$	$A_{2,3}$	$A_{3,3}$	$A_{4,3}$	0.3810263

Table 2.11: Tree Illustration - True Haplotypes. The observed allele proportions will have been generated by the three specified haplotypes.

the following observed allele proportions,[38]

Loc$_1$	p	Loc$_2$	p	Loc$_3$	p	Loc$_4$	p
$A_{1,1}$	0.336	$A_{2,1}$	0.336	$A_{3,2}$	0.283	$A_{4,1}$	0.283
$A_{1,2}$	0.664	$A_{2,3}$	0.664	$A_{3,3}$	0.717	$A_{4,2}$	0.336
						$A_{4,2}$	0.381

Table 2.12: Tree Illustration - Observed Allele Proportions. Presented to three decimal places. Locus 4 is the longest locus and will start the tree construction process. Loci 1 and 2 have essentially the same configuration.

The corresponding tree is presented in Figure 2.17. As always, the algorithm starts with the initial assumption that the MOI equals the length of the longest locus, in this case 3. A root at level 0 is created before selecting any loci. For this observation, the nodes were processed in the following order: 4,1,2,3. Locus 4 was chosen first because it is the longest locus of the group. As always, when we branch to the first level we consider only *combinations* and not (yet) *permutations*. Since there are three distinct alleles represented at locus 4 and the current assumption is that the MOI is also three, there is only one combination (consisting of all three alleles) to consider; hence the single

[38]This is a very high value for c that corresponds to essentially no measurement error for any of the loci.

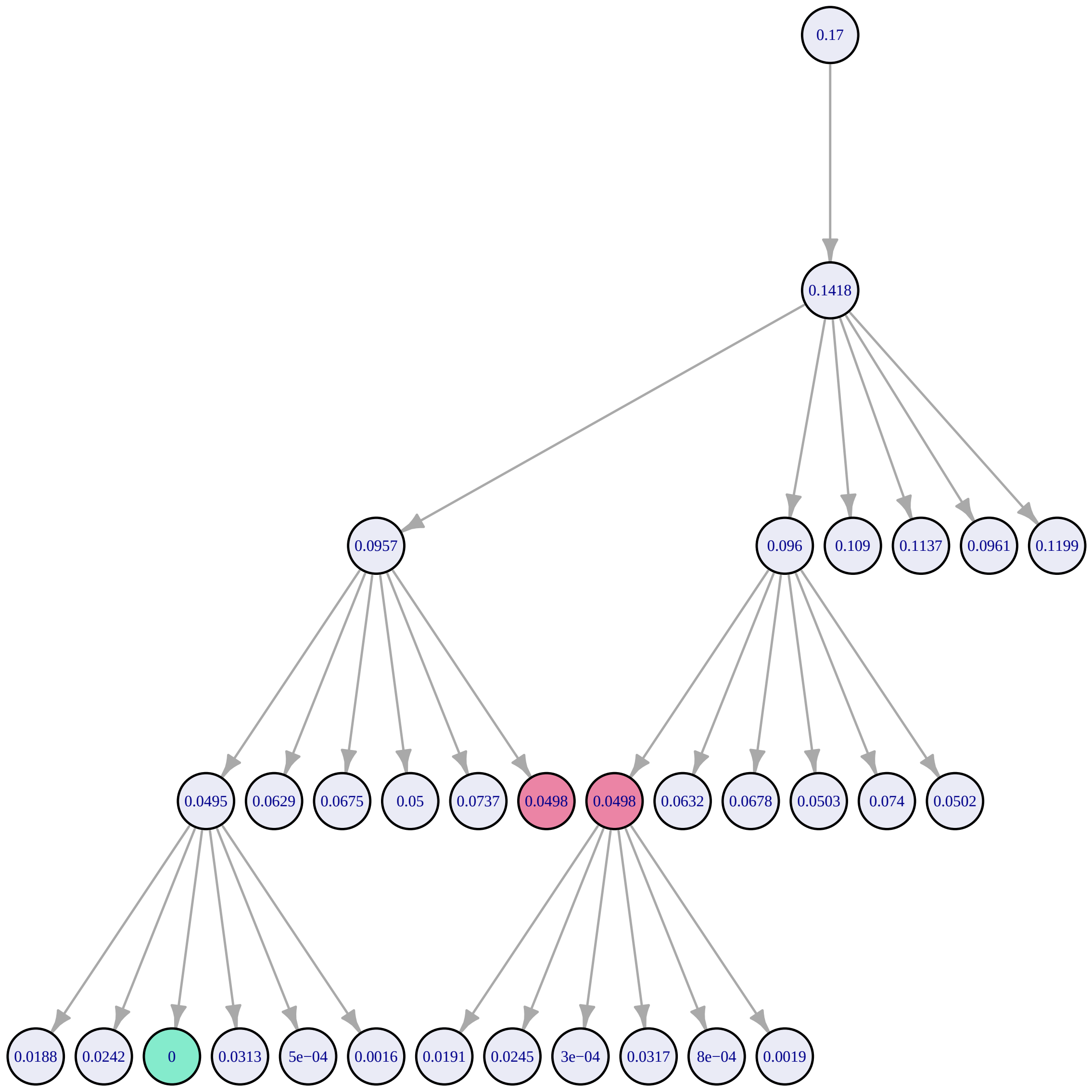


Figure 2.17: Sample Tree. A sample tree with five levels including a root node, implying a search over four loci. The number in each node is the error to four decimal places for the partial solution the node represents. In this tree, `smnc` = 2, hence at each level a maximum of two nodes have descendants. The node in green has an error of zero and corresponds to the correct solution. The two nodes in red represent an apparent tie only.

edge from the root to level 1. At level 1, since `smnc` = 2 and there is only one node, that node must be selected. From level 1 to level 2, we are considering permutations for a locus of two represented alleles, so there are six permutations to consider; hence the six outward edges. At level 2, we choose the two nodes representing the partial solutions with the lowest error for continuation. Similar to the previous step and for the same reason, we consider all six permutations of locus 2 when branching from level 2 to level 3; hence the six outbound edges from each of the two chosen nodes. Note that the set of six permutations is same for each node. At level 3, we again select the two nodes with the lowest error. In this case, the competition is close. The first node "wins" with an error of 0.0495 (rounded to four decimal places). More interesting is what happens to the second and third nodes, highlighted in red. The errors for these two nodes are 4.983703e-02 and 4.983717e-02, respectively. The first of these nodes clearly has a lower error, yet the difference is so small that the errors appear identical when presented to four decimal places in the tree. Finally, at the bottom, we construct the complete solutions. The node in green has an error of 2.670556e-10, or 0 to four decimal places. For this observation, this node corresponds to the correct solution.

We note a general feature of the tree: the errors decrease monotonically as we descend the tree. The reason for this is clear: at each level, we are filling in a previously blank locus and reach a better approximation for the observed allele proportions at that locus than beforehand. Prior to the completion, the construction of the signature matrix was left at 0 for the relevant locus, implying observed proportions of zero for *all* alleles at that locus. This was demonstrated in Example 2.23.

2.C Technical Implementation

The `LITSE` algorithm was implemented in two languages: `R` and `C++`.

The SSI step was largely coded in `C++`, as the tree searches are the most computationally intensive section of the algorithm. The `C++` code made extensive use of the Armadillo library of linear algebra data structures as routines. See [35] for details.

The remainder of the algorithm, including the SSR procedure and all wrapper functions, was coded in `R` . To enable the communication between `R` and `C++`, we used the `Rcpp` package in `R` . See Eddelbuettel [11] and Eddelbuettel and Franccois [12] for details.

Chapter 3

Simulation

In this chapter, we assess the performance of the `LITSE` algorithm using simulated data. We approach this assessment by performing two complementary exercises, each focusing on particular aspects of the algorithm:

1. Assess the performance of the algorithm in finding potential solutions per observation, including for observations with one or more missing loci values, before evaluating the likelihood of the implied haplotypes and MOI across all observations. This amounts to assessing the performance of the SSI and SSC steps, which appear as lines 3 and 4 in the summary Algorithm 1 and are documented in more detail in Algorithms 2 and 4, respectively. This simulation exercise is discussed in detail in §3.2.
2. Assess the performance of the entire algorithm using two criteria:
    a) The ability of the algorithm to assign high probabilities to the true solutions, including for observations with one or more missing loci values.
    b) The accuracy of the estimation of the underlying parameters Θ in the model, as discussed in detail in §2.7.4.

Recall that the `LITSE` algorithm is itself deterministic. Given a particular set of inputs, it will always produce the same results. Thus the randomness in both simulation exercises exists purely in the data generated.

3.1 Organization of Simulations

The two simulation exercises mentioned above were performed in a collection of separate simulation "runs". A run is defined as a set of individual "trials", each of which represents a particular configuration of parameters governing the data to be randomly generated in the trial and how the `LITSE` algorithm will process the data. Each trial differs from other trials by the joint selection of values for these parameters. The algorithm is then applied to the data generated in the trial and the results collected. These results are then pooled across all trials in the run for analysis.

The goal is to determine the runtime and accuracy performance of the `LITSE` algorithm and its sensitivity to the values of key parameters, both individually and in combination with other parameters.

3.1.1 Standard Steps in Each Trial

Regardless of the purpose of a particular run of which a trial is a member, there is a standard set of steps performed in each trial. In summary, first data are constructed following the data generating process outlined in §4.1.1 and then `LITSE` is applied to these data. The following is a description of these general steps (see Table 3.1 for a description of the simulation parameters mentioned):

1. Construct locus information details for `nl` different loci, each with `ll` different possible alleles. Randomly assign each possible allele a probability in the interval (0,1) such that for each locus the probabilities sum to 1.
2. Construct a set of `hpn` different haplotypes, where each haplotype is chosen by randomly selecting a single allele for each locus according to the probabilities for the alleles at that locus as determined in the previous step. This method bases itself on the previously discussed assumption of independence among loci. Each haplotype has its population frequency computed as the product of the probabilities of the alleles at its loci, with these initial frequencies normalized to sum to 1. This forms the $\boldsymbol{\pi}$ parameter vector used in the trial.
3. Determine the $\mathbf{p}$ parameter vector for the trial, the population frequencies for

the MOIs from 1 through `maxmoi`. Each element is first assigned a value on the uniformly distributed unit interval, with these values then normalized to sum to 1.

4. Generate `no` different observations for the trial, for each observation independently.
   a) First, for each observation select an MOI (i.e., Z_i) from the distribution determined by $\mathbf{p}$.
   b) Second, select MOI different haplotypes from the haplotype population created in the previous step using the population frequencies $\boldsymbol{\pi}$ generated. For each of these haplotypes, randomly assign observation proportions such that they sum to 1. These proportions are assigned without regard to $\boldsymbol{\pi}$. This is consistent with the previously discussed assumption that the proportions are independent from the frequencies. Once an individual has been infected with particular haplotypes, these haplotypes are assumed to multiply at different rates and their proportions in an individual say nothing about their overall population frequency.
   c) Third, compute observed allele proportions for the observation, as a function of the implied allele proportions determined by the chosen haplotypes in the preceding step and the value of the parameter $\mathbf{c}$ for the trial. For simplicity in data generation, within a trial $c_i = c_j, i \neq j$; that is, we assume that all loci have the same value for c. However, the `LITSE` algorithm still estimates the c values individually; no information from one locus is used to estimate the value for the other loci. In addition, once these values have been determined, randomly set to zero the allele proportions for a locus according to the parameter `br`. This parameter exists to control the degree to which locus information is missing; a value of zero indicates that all observations have allele proportions for each locus while high values represent high "gappiness" in the data.
5. Apply the `LITSE` algorithm to the data. This includes the following steps.
   a) For each observation individually, execute the SSI procedure (discussed in §2.7.2) to identify possible solutions. This will involve using the trial's settings for the following key runtime parameters: `mfl`, `smnc`, `hmnc`, `mer`, `am`. Each of these parameters controls how exhaustively the `LITSE` algorithm searches for

solutions, and each is discussed in more detail below.

b) Execute the SSC procedure (discussed in §2.7.3). That is, for all solutions with missing locus data, impute such data using other haplotypes. If there are no such solutions, because all observations had complete locus information, then this step is completed.

c) Using all observations collectively, perform the SSR procedure (discussed in §2.7.4) a maximum of `mi` times. Thus setting `mi` = 0 equates to turning off this step entirely.

3.1.2 Standard Parameters in Each Trial

The simulation parameters referenced in the preceding section are standard for all simulation runs and are described in Table 3.1.

In the remainder of the chapter, when a result is presented by value of parameter, the result is typically presented across all trials in which the parameter of interest took on a particular value and parameters not of interest took on any value. For example, in Figure 3.1, the results are presented by value of the parameter of interest (`nl`), and the individual points in the figure for each value of `nl` (in this case, faceted by two other variables) represent results for individual trials where `nl` took on the value specified (without regard to the value taken for any other parameter value – e.g., `no`, which is irrelevant to the content of the plot).

3.2 Simulation on Individually Determined Solutions

In this section, we present the properties and results for the section of the algorithm that identifies potential solutions for each observation. While this step is a predecessor to the iterative part of the algorithm whereby the probabilities of these solutions are computed and then refined, it is important that SSI procedure deliver accurate potential solutions as an input to the iterative refinement later.

Parameter	Description
`nl`	Number of loci present in each haplotype. Constant in trial; can vary across trials in same run.
`ll`	Locus length $\ell(l)$, the number of *possible* alleles at each locus. Constant in trial; can vary across trials in same run.
`hpn`	Number of distinct haplotypes in the haplotype population. This quantity identifies the number of different haplotypes assumed to exist in the haplotype population from which an individual may be infected. Constant in trial; can vary across trials in same run.
`no`	Number of observations in trial. Constant in trial; can vary across trials in same run.
`maxmoi`	Maximum MOI. Each observation draws an MOI from a distribution of MOIs, from 1 to `maxmoi`, according to a distribution captured by the parameter $\mathbf{p}$. Constant in trial; can vary across trials in same run.
`c`	True value for $\mathbf{c}$, the concentration parameter. Single value for all loci. Constant in trial; can vary across trials in same run.
`br`	"Blank rate", the percentage of all observation/locus combinations with missing locus information, simulating cases where such information is unavailable and must be inferred. Constant in trial; can vary across trials in same run.
`mfl`	Maximum floating level, lowest level at which rates are allowed to be recomputed. See Table 2.5. Constant across trials in a run.
`smnc`	Soft maximum node count. See Table 2.5. Constant across trials in a run.
`hmnc`	Hard maximum node count. See Table 2.5. Constant across trials in a run.
`mer`	Maximum error ratio. This value represents the maximum ratio permitted between the lowest error among all solutions for one MOI versus those for the next highest MOI. Its purpose is to detect when we can stop incrementing z before considering all `am` + 1 possible values for z. This parameter is discussed in more detail in §3.2.1. Constant across trials in a run.
`am`	Additional MOIs considered (above minimum). See Table 2.5. Constant in trial; can vary across trials in same run.
`mi`	Maximum number of iterations in the iterative Solution Set Refinement part of the algorithm. A value of zero implies no Solution Set Refinement step performed. Constant across trials in a run.
`inst`	Number of instances. Refers to number of trials in a run with the same configuration of above parameters.

Table 3.1: Simulation Trial Parameters.

This section comprises several subsections: a discussion of some of the properties of the algorithm and separate discussions of the runtime and accuracy performance.

3.2.1 Basic Properties of the SSI Procedure

Before presenting the performance of the algorithm in terms of runtime or accuracy, this section covers a few of the general relationships between some of the features of each run. Knowledge of these relationships will be useful in setting runtime parameters in later runs. Given their coverage later, runtime and accuracy are deliberately omitted from discussion in this section.

3.2.1.1 Configuration of Simulation

The parameter settings for this simulation are presented in Table 3.4. A value of 0 for

Parameter	Setting
`nl`	5,10,15,20
`ll`	6
`hpn`	10
`no`	10
`maxmoi`	4
`c`	200
`br`	0
`mfl`	100
`smnc`	5,10,15,20
`hmnc`	50
`mer`	Inf
`am`	2
`mi`	0
`inst`	20

Table 3.2: Simulation Settings: Basic Properties. Each parameter in the "Parameter" column was tested for each value in the corresponding entry in the "Settings" column. Every combination of the parameter settings is tested `inst` times.

`br` indicates that no solution will have missing data, thus the SSC step will be unnecessary. A value of 0 for `mi` indicates that the SSR step will not be performed either. The value of infinity for `mer` is such that this control is turned off. Similarly, the value of `hmnc` is sufficiently high that it has no effect in this simulation.

The total number of trials is thus equal to the product of the number of settings for each parameter times the value for `inst`. This means a total of

$$4 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 4 \times 1 \times 1 \times 20 = 320$$

trials, each having ten observations.

Table 3.3 presents a summary of the simulation run. The proportion of solutions used

Quantity	Value
Number of Observations	3200
Number of Trees	8392
Number of Trees per Observation	2.62
Number of Solutions	310527
Number of Solutions Analyzed for Proportion Evolution	3000

Table 3.3: Simulation Stats: Basic Properties. The number of trees is slightly below $3200 \cdot 3 = 9600$ because all solutions in some trees were deemed invalid; this occurred when all solutions for the tree were deemed invalid when an MOI higher than the true MOI was attempted. See §2.7.2.5 for details.

in the evolution analysis is relatively low because of the computational effort involved.

3.2.1.2 Determinants of Tree Size

The SSI procedure assembles a set of nodes in a tree structure; for each node, it computes a set of proportions using NNLS and an error versus the observed allele proportions. All else held equal, a tree with a high (low) number of nodes represents an extensive (abbreviated) search over possible solutions. There are three primary factors that are suspected to influence the size of the tree:

1. `nl`, the number of loci. The number of loci equals the number of levels in the search tree (excluding the root node level). Thus, assuming a constant tree "width" (number of nodes per level), the greater the number of loci, the greater the number of nodes. In addition, note that a constant tree width is possible only with the use of a trimming procedure, as explained in §2.7.2.4. In the absence of such a

procedure, the per level width of the tree will increase as the level increases, thus creating exponential growth in the number of nodes.

2. z, the MOI. For a fixed maximum locus length (which serves as the minimum possible MOI), the higher the value for the estimated z, the greater the number of permutations originating from each chosen node to its children.[1] This is described in §2.7.2.3. Recall that a given observation will typically have several MOIs attempted, as controlled by the parameter `am`. In such cases, the observation's trees will increase in size as the estimate of z increases.
3. `smnc` (and `hmnc`), the number of nodes chosen per level. These parameters are a direct attempt to control the size of the tree by limiting the number of nodes in a level permitted to have children in the following level. Thus all else held constant, a high (low) value will lead to a relatively large (small) tree.

In summary, we would expect an increasing relationship between tree size and each of these three factors. Figure 3.1 shows that this is the case. In particular, we see that the parameter `smnc` effectively controls the size of the tree, as evidenced by the significantly increasing range of tree size values as `smnc` increases (corresponding to different horizontal slice in the faceted plot). There is a clear positive relationship between tree size and all three variables, though for low values of z this can be hard to discern in the plot.

3.2.1.3 Number of Solutions

Next, considering that the proposed solutions for an observation will exist by construction as the leaves of the tree, it is natural to expect a positive relationship between tree size and number of solutions. Figure 3.2 shows the relationship. We see that there is no obvious relationship between the two variables. The fitted line appears flat across the various tree sizes.

Given the uncertain relationship documented above, we reconsider the drivers of the number of solutions. By definition, the solutions are the leaves in the search tree. The number of leaves is directly related to two things: the number of nodes at the previous level spawning children and the number of children per such node. The first quantity is

[1]More formally, their outbound branching order increases.

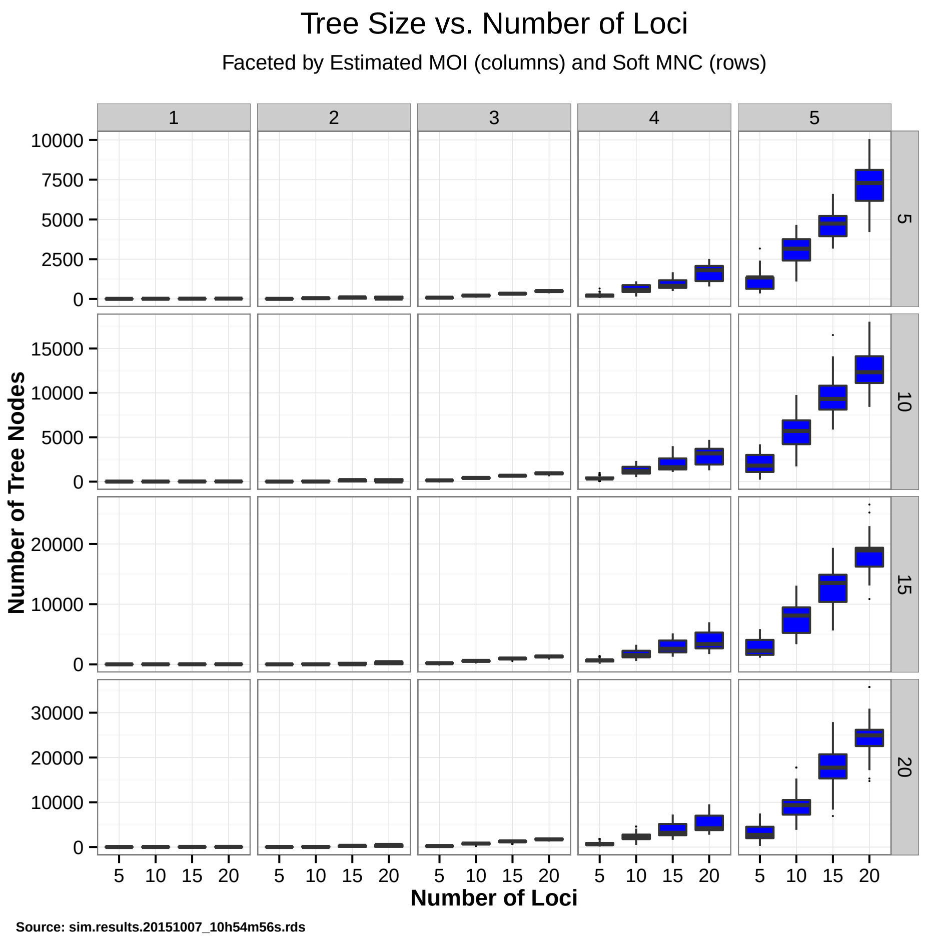


Figure 3.1: Size of Search Tree versus Number of Loci. The distribution of the tree sizes is presented in a separate box plot for each number of loci attempted, faceted by z values per column and `smnc` values per row. Boxplot structure: Each box captures all trials within the $25th$ and $75th$ percentiles, with the horizontal line representing the median. Trials outside these boundaries are represented by the vertical lines above and below each box when their distance from the box is more than 1.5 times the distance between the first and third quartiles; trials above this limit appear as individual points.

directly related to `smnc`. The second quantity is related to the estimated MOI and the number of represented alleles for the last locus to be processed. Since all loci are ordered from longest (with the most alleles) to the shortest (with the fewest alleles), this equates to considering the length of the shortest locus. Figure 3.3 shows the relationship. Here,

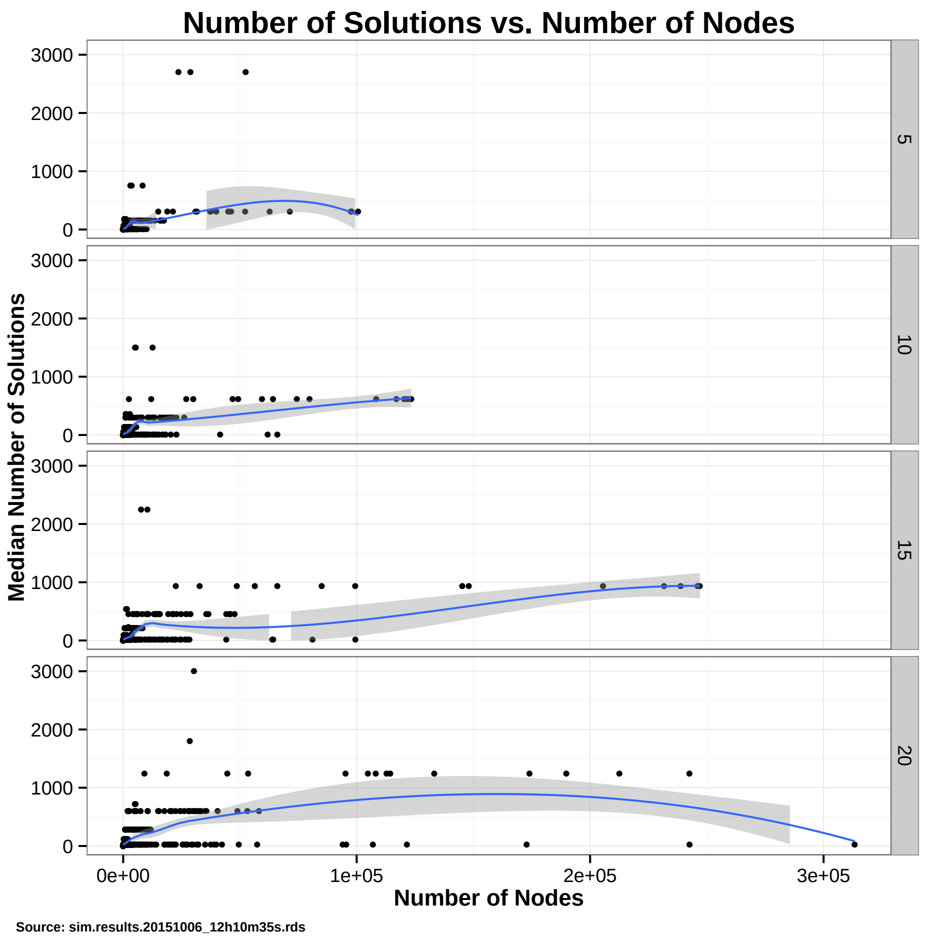


Figure 3.2: Number of Solutions vs. Number of Nodes. The figure demonstrates the effect of two factors, tree size and `smnc`, on the number of solutions produced for an observation over all parameter settings. The number of nodes is represented on the x-axis while each value of `smnc` is represented in a separate panel, with the panels vertically stacked by ascending `smnc`. Invalid solutions removed.

we see a much clearer relationship. As expected, there is a clear positive relationship between `smnc` and number of solutions. There is also a clear positive relationship between the other two variables of interest. The paucity of data points for a minimum locus length of 3 (i.e., the third row in the figure) is simply due to the rarity of this event: under the parameters used in the simulation, where the number of possible alleles for all loci equals 6, the number of haplotypes in the population is set to 20, and the true MOI is the range from 1 to 4, there is a significant skewness to lower values for this feature.

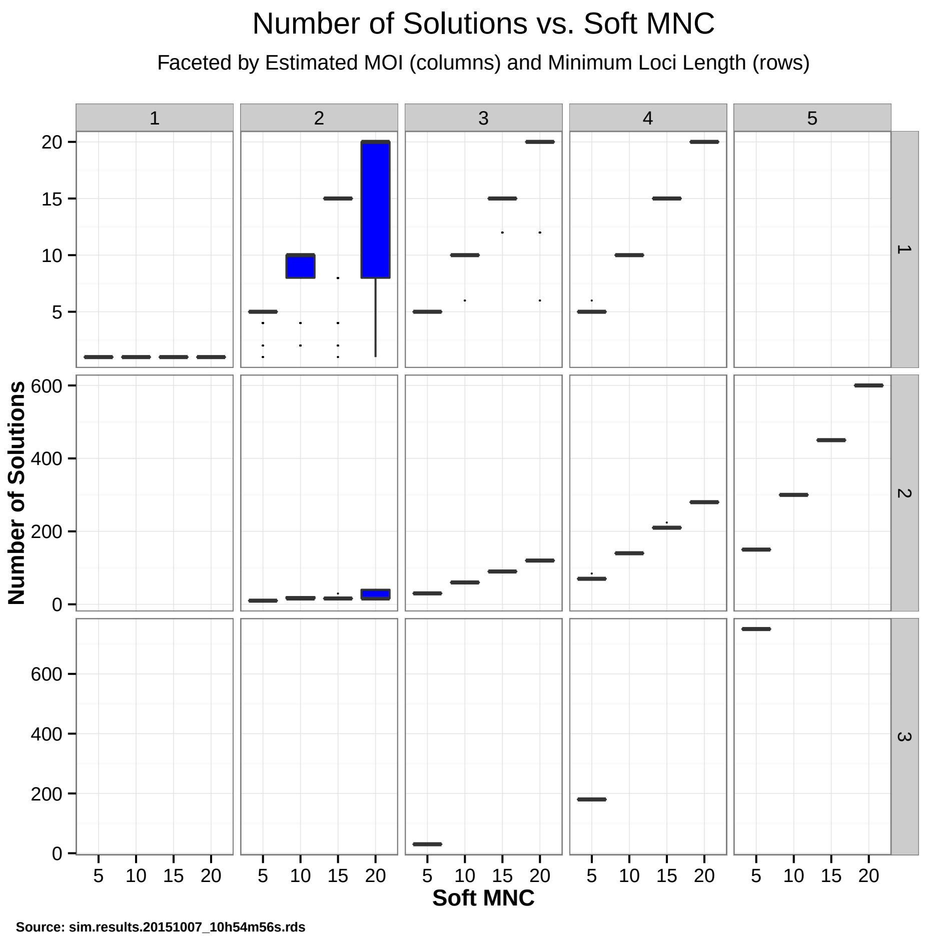


Figure 3.3: Number of Solutions vs. `smnc`. The figure demonstrates the effect of three factors, `smnc`, estimated MOI, and shortest locus length, on the number of solutions produced for an observation over all parameter settings. The parameter `smnc` is represented on the x-axis while the other two are represented in the facets. Invalid solutions removed.

3.2.1.4 Evolution of Proportions

Also of interest is the evolution of the proportions for a solution as it passes down the tree, as depicted in Figure 3.4. Knowing the natural evolution will be useful in determining at what tree level, if any, to force the algorithm to estimate the proportions for the haplotypes. In other words, determine a good value for the parameter `mfl`. The figure depicts how the estimated haplotype proportions change as the solutions pass through

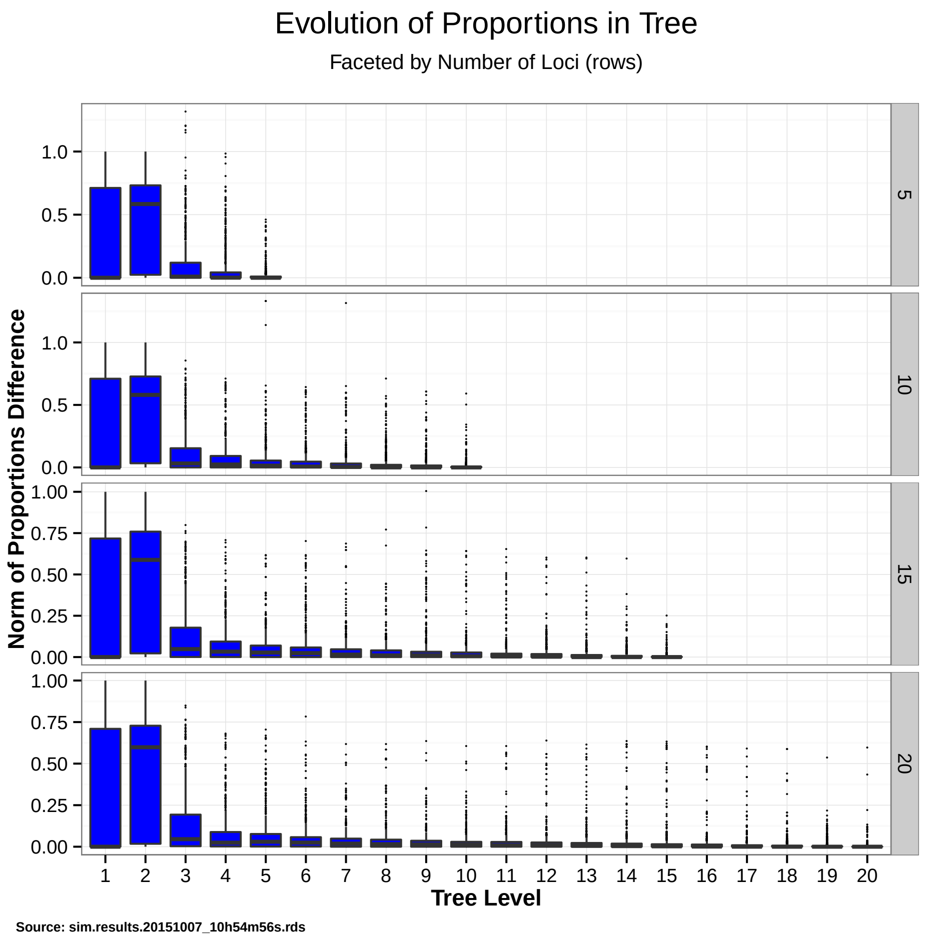


Figure 3.4: Evolution of Proportions. For a sample of solutions in this simulation, this figure presents the $L2$ norm of the difference between the proportions from one level to another. Each point in this figure represents such a difference at a particular level in the tree for some solution, where the results have been faceted by number of loci. For an explanation of the box plot structure, see Figure 3.1.

the tree. We know that by construction at level 0 (the root), the proportions have not yet been estimated. In cases where the first locus block equals the assumed MOI, the proportions are estimated starting with the first locus and are fixed only after the last locus in the first locus block; otherwise, the proportions are estimated starting with the second locus block and are fixed only after the last locus in the second block. Recall that in this simulation the value of `mfl` has been set to 100, far above the number of loci for any trial. Thus it has no effect on the results.

We see that for all number of loci tested, the great majority of the change in proportions takes place in the first two levels. As a general rule, the volatility in the estimation of the proportions decreases as the algorithm descends levels in the tree. Given this result in an unrestricted case, we can consider setting the parameter `mfl` to a value in the single digits with the comfort that we will not be impeding the refinement of the proportions at lower levels.

3.2.2 Runtime Analysis

This section investigates the sensitivity of the runtime of the Solution Set Identification-step to a variety of factors. By "runtime", we mean wall time, or the duration (measured in seconds) of the procedure, either for all observations or per observation (where that observation has particular characteristics).

The runtime is important to consider because we envision cases where there is a tradeoff between speed and accuracy.

To ensure consistency, all trials were run on the same processor type. This simulation was run using an r3.8xlarge instance at Amazon Web Services, which uses Intel Xeon E5-2670 v2 2.50GHz processors.

3.2.2.1 Configuration of Simulation

This time, the parameter settings are as presented in Table 3.4, implying a total of

$$4 \times 1 \times 1 \times 4 \times 1 \times 1 \times 1 \times 4 \times 4 \times 1 \times 1 \times 1 \times 1 \times 4 = 1024$$

trials, each having either five, ten, fifteen or twenty observations and each computing up to three trees (since `am` $= 2$) per observation. The parameters that vary in the configurations are those that will be tested for their effect on the runtime. As before, the values for `mer`, `br`, and `mi` are not relevant.

Parameter	Setting
`nl`	10,15,20,30
`ll`	6
`hpn`	10
`no`	5,10,15,20
`maxmoi`	4
`c`	200
`br`	0
`mfl`	3,5,10,15
`smnc`	5,10,15,20
`hmnc`	50
`mer`	Inf
`am`	2
`mi`	0
`inst`	4

Table 3.4: Simulation Settings: SSI Runtime. Each parameter in the "Parameter" column was tested for each value in in the corresponding entry in the "Settings" column. Each combination of the parameter settings is tested `inst` times.

Table 3.5 presents a summary of this simulation run, which is clearly larger than its predecessor.

Quantity	Value
Number of Observations	12800
Number of Trees	38400
Number of Trees per Observation	3.00
Number of Solutions	4671339
Average Tree Size (Number of Nodes)	19621

Table 3.5: Simulation Stats: SSI Runtime. The number of trees equals $12800 \cdot 3 = 38400$ because each tree was deemed to have at least one valid solution. See §2.7.2.5 for details.

3.2.3 Overview

This section briefly presents a summary of all the runtimes for tree constructions for all observations in all simulation trials in this run. The distribution of the runtimes is presented in Figure 3.5. The corresponding quantiles are presented in Table 3.6.

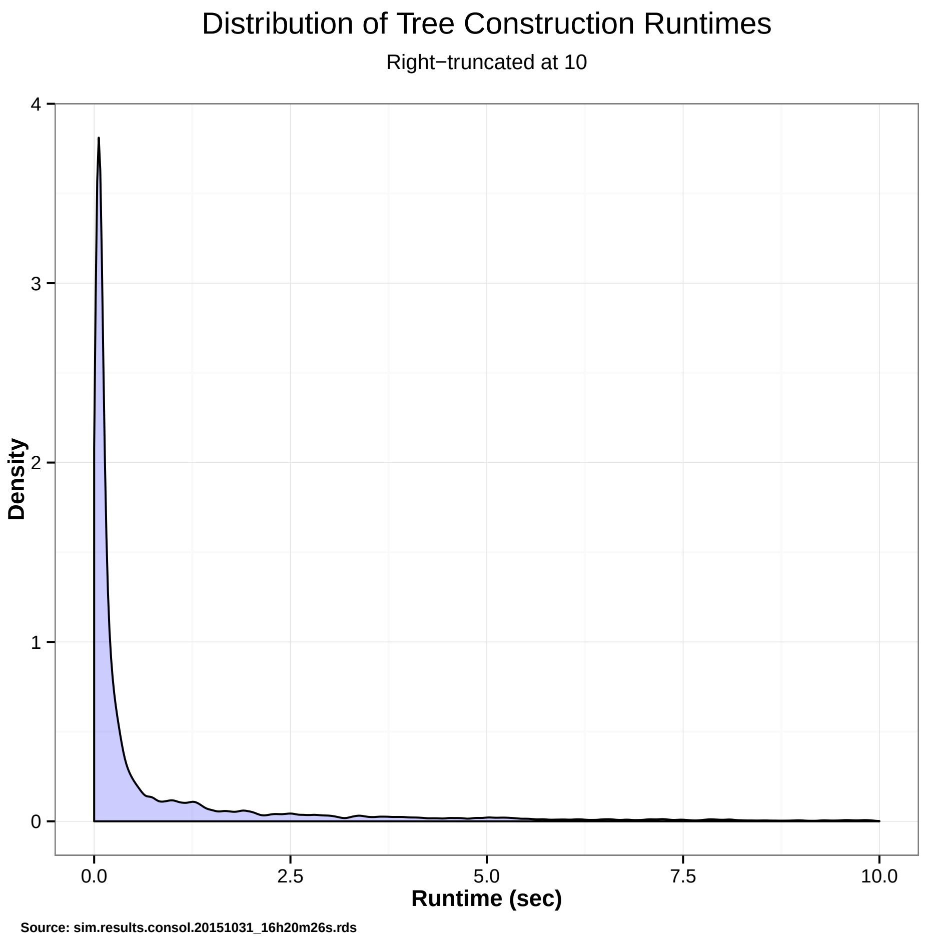


Figure 3.5: Distribution of Tree Search Runtimes. . The distribution of the runtimes is centered well under one second but with a long righthand tail. In this figure, runtimes exceeding 10 seconds have been excluded to better depict the distribution for the vast majority of trees.

We see that the median of runtimes is approximately 175 milliseconds, though there is a great amount of variety in this quantity. The remainder of this section investigates what drives these differences.

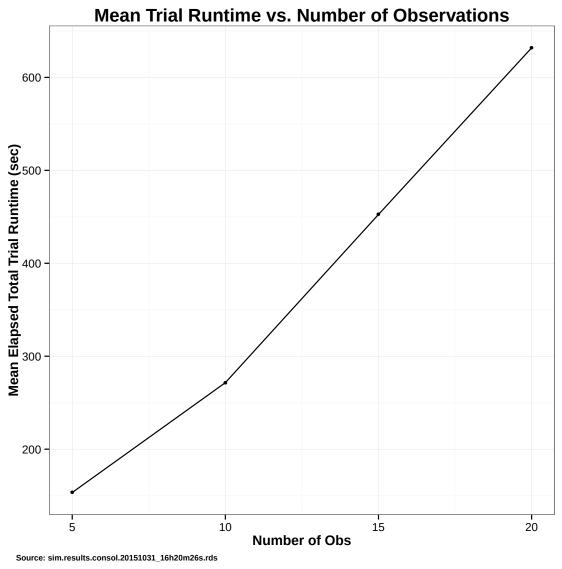
3.2.3.1 Sensitivity to Number of Observations

We first consider the sensitivity of the SSI procedure to the number of observations processed. This relationship should be linear, on the assumption that the average com-

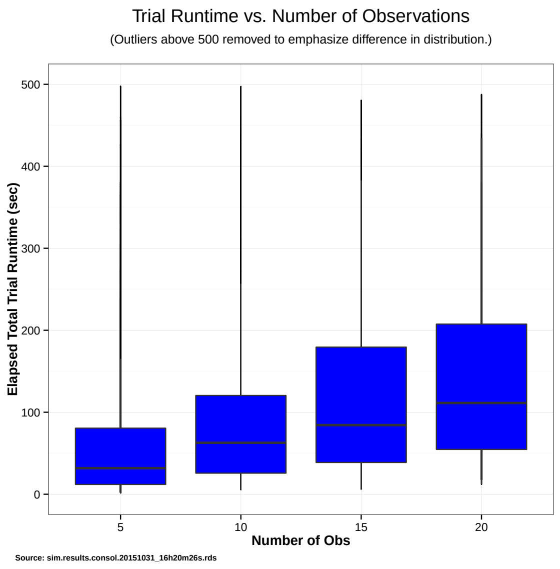
Quantile	EquivPercent	Value
0	0%	0.0230
1	10%	0.0380
2	20%	0.0490
3	30%	0.0670
4	40%	0.1030
5	50%	0.1750
6	60%	0.3330
7	70%	0.9300
8	80%	2.7142
9	90%	12.5066
10	100%	1420.7420

Table 3.6: Deciles of Tree Construction Runtimes. The unit in the value column is seconds.

plexity of an observation remains constant. Figure 3.6 reveals that this is the case. We



(a) Mean Runtime



(b) Runtime Distribution

Figure 3.6: Runtime of Tree Searches versus Number of Observations. The left figure plots the mean trial runtime versus number of observations in a trial. On the right, the distribution of the run times is presented in a separate box plot for each number of observations attempted. For an explanation of the box plot structure, see Figure 3.1.

note, however, that the means depicted in Figure 3.6(a) are significantly higher than the medians depicted in 3.6(b); this is due to some outliers significantly above the medians.

3.2.3.2 Sensitivity to Tree Size and Number of Loci

Next, we consider the sensitivity of the runtime to the number of loci and the tree size. Figure 3.7 displays the runtime per tree as a function of the tree size. We would expect

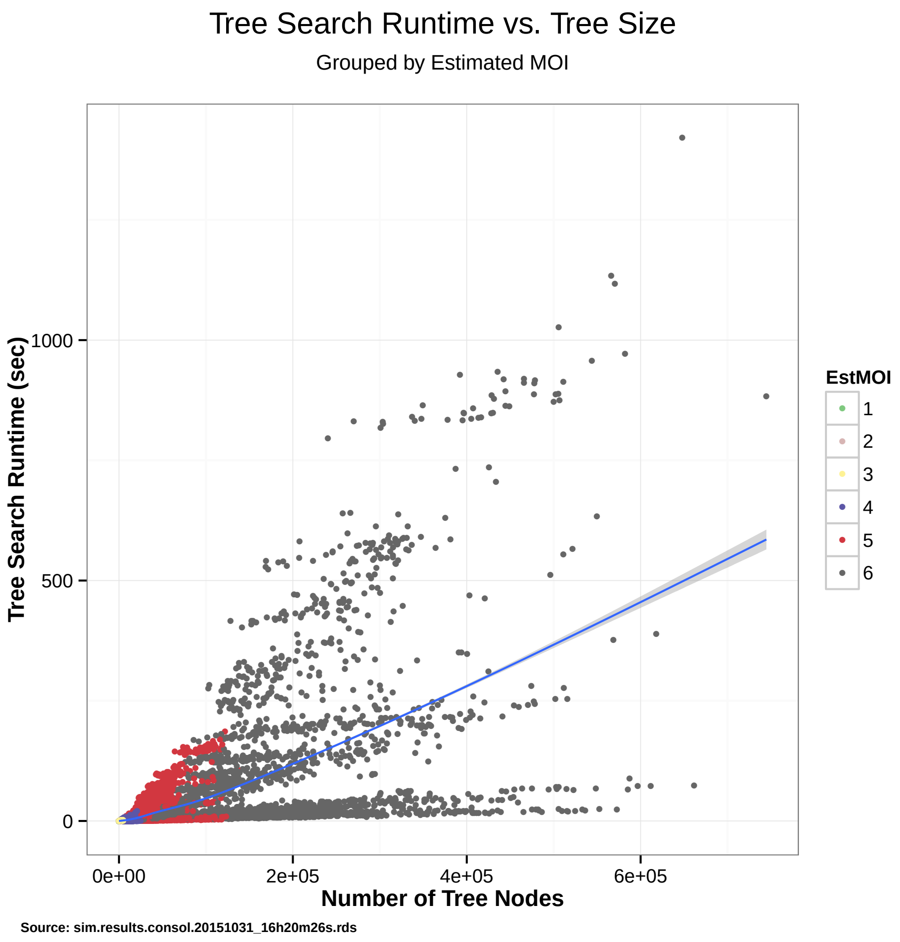
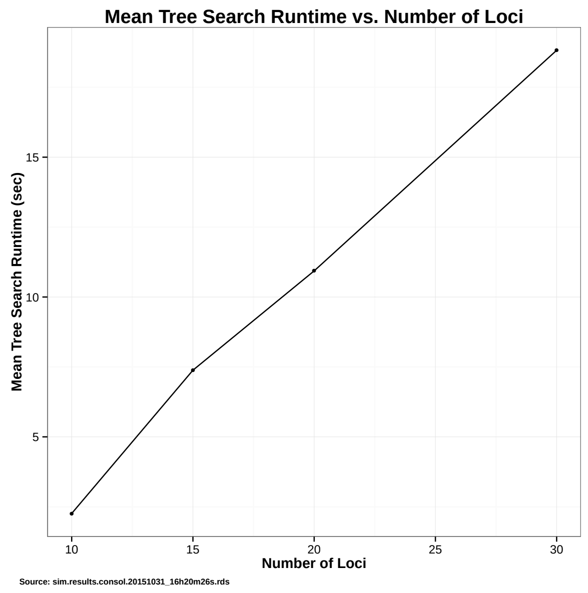


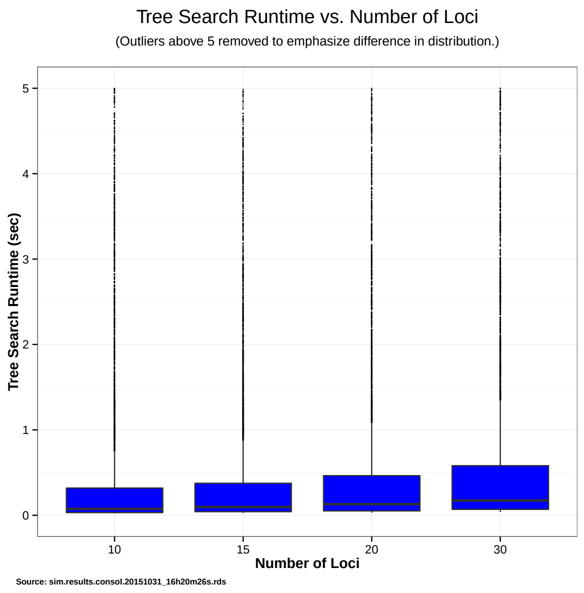
Figure 3.7: Runtime of Tree Searches versus Tree Size. Each point represents a single tree search. The points are color-coded by estimated MOI.

the runtime of the SSI procedure to be positively related to the number of nodes in the tree. The figure shows that there is indeed a positive relationship between tree size and runtime and it is roughly linear, though the dispersion increases as the number of tree nodes increases. The points naturally cluster around estimated MOI, which is a strong driver of tree size.

Concerning the number of loci, Figure 3.8 displays the relationship. As expected



(a) Mean Runtime

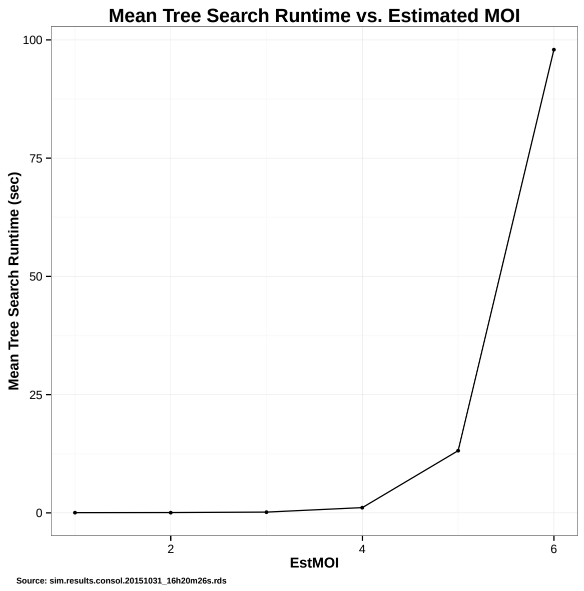


(b) Runtime Distribution

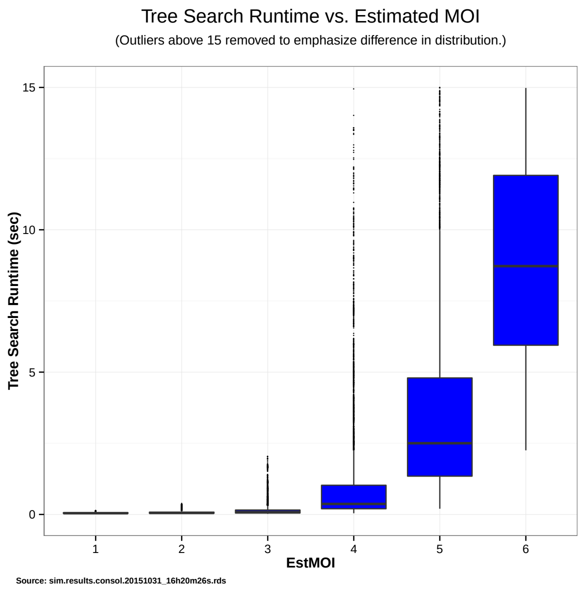
Figure 3.8: Runtime of Tree Searches versus Number of Loci. The left figure plots the mean runtime per observation versus number of loci. On the right, the distribution of the runtimes is presented in a separate box plot for each number of loci attempted. For an explanation of the box plot structure, see Figure 3.1.

there is an increasing, linear relationship between the number of loci and the runtime, a reflection of the fact that the number of loci directly determines the number of levels in the tree.

Next, we consider the estimated MOI, z. Figure 3.9 displays this relationship in the simulation. Here, we see a sharply exponential relationship between the two. This is driven by the fact that as the estimated MOI increases for a given observation, the number of valid permutations increases exponentially, as explained in Proposition 1 in §2.7.2.4, each of which is applied to each of the chosen nodes. Note that this effect is not influenced by the `smnc` parameter, which controls only the number of nodes chosen.



(a) Mean Runtime



(b) Runtime Distribution

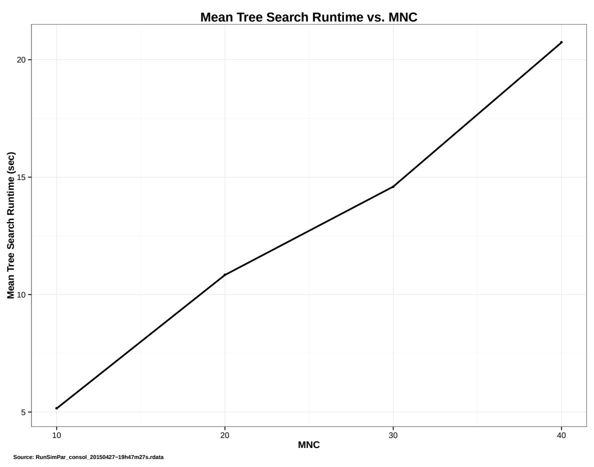
Figure 3.9: Runtime of Tree Searches versus Estimated MOI. The left figure plots the mean runtime versus z. On the right, the distribution of the runtimes is presented in a separate box plot for each value of z attempted. For an explanation of the box plot structure, see Figure 3.1.

This analysis demonstrates that there is a significant price in terms of runtime when considering higher possible MOIs for a set of observations.

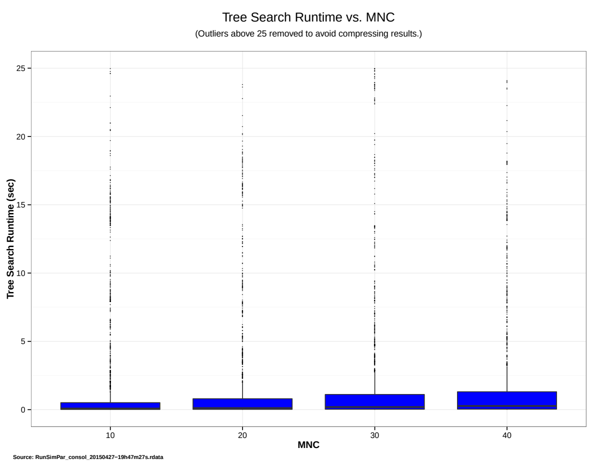
3.2.3.3 Sensitivity to Parameters `smnc` and `mfl`

Finally, we now consider the effect of two user-selected parameters on the running time of the algorithm. Each parameter is explicitly intended to control the number of computations executed in searching for solutions and thus have a direct effect on the running time.

The first parameter is `smnc`. Recall that this parameter defines the number of nodes chosen at a particular level that have children at the next level in the tree. Figure 3.10 displays the relationship. Here, we see an almost perfect, positive linear relationship between the parameter `smnc` and the tree search runtime. Clearly, there is a price to pay in runtime when setting `smnc` to a relatively high level. Again, the means on the left are



(a) Mean Runtime



(b) Runtime Distribution

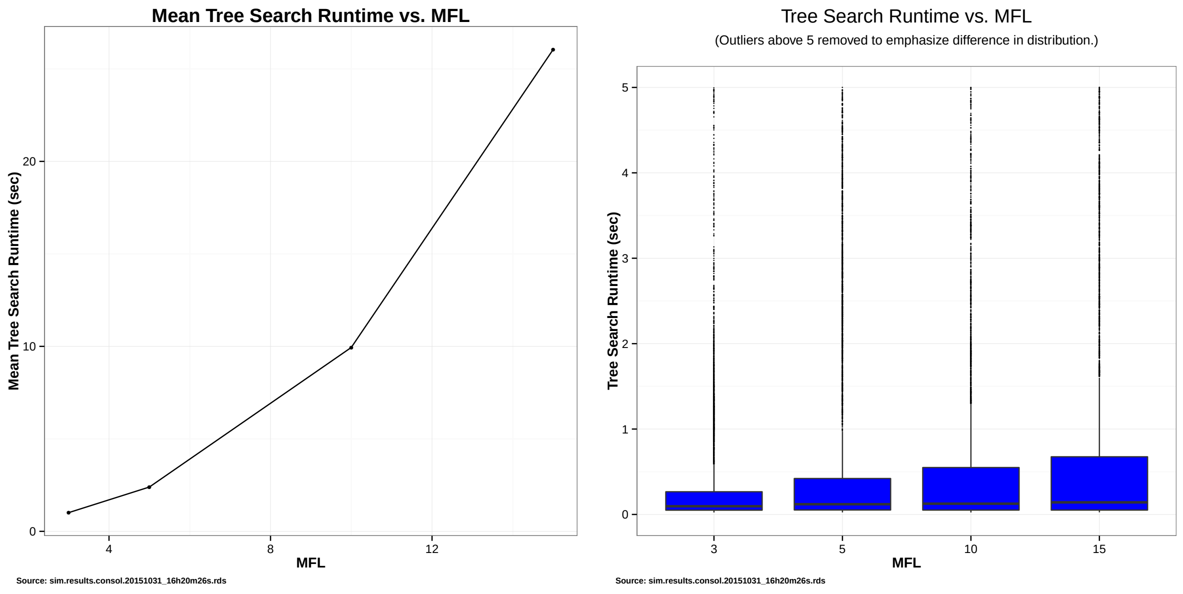
Figure 3.10: Runtime of Tree Searches versus `smnc`. The left figure plots the mean runtime versus `smnc`. On the right, the distribution of the runtimes is presented in a separate box plot for each `smnc` implemented. For an explanation of the box plot structure, see Figure 3.1.

higher than the medians due to outliers. There is an open question regarding to what extent a low value for this parameter will eliminate the consideration of good solutions in certain cases; this is explored in the next section.

The second parameter is `mfl`. Recall that this parameter sets the maximum number of levels at which the estimated proportions for each solution are allowed to “float” or be recomputed. Once this limit has been reached, the proportions are fixed for the remaining levels of the tree. Figure 3.11 displays the relationship.

The significance of fixing the proportions for a solution is that once this has been done at a particular level, that will be the last level at which the algorithm will use NNLS to estimate the proportions based on all loci filled to that point. For all future levels, the proportions are fixed and the only computation is determining the error. As this latter computation is less computationally expensive than NNLS, setting `mfl` to a low level is a choice that will accelerate the completion of the tree with a possible loss in the accuracy of the proportions. However, as discussed in §3.2.1.4, when `mfl` $= \infty$ and is thus not enforced, the change in proportions from one level to another quickly tapers off.

The conclusion of this section is that the `smnc` and `mfl` parameters can be effectively



(a) Mean Runtime **(b)** Runtime Distribution

Figure 3.11: Runtime of Tree Searches versus `mfl`. The left figure plots the mean runtime versus `mfl`. On the right, the distribution of the runtimes is presented in a separate box plot for each `mfl` implemented. For an explanation of the box plot structure, see Figure 3.1.

used to reduce the running time of the `LITSE` algorithm.

3.2.4 Accuracy Analysis

In this section, we investigate the accuracy of the SSI routine. How "good" are the solutions this routine proposes? Can it find always find the correct solution for an observation and what factors influence its ability to do so? Because the `LITSE` algorithm will typically suggest *many* solutions for the same observation, among which we would expect the true solution to be included, these questions can be rephrased as follows: How reliable is the `LITSE` ranking of solutions by error? How accurate are the solutions that `LITSE` ranks near the top? And what factors does this accuracy depend on?

Recall that in the SSI step, there is no concept of the *probability* of a solution being correct. The goal of the step is to identify the most accurate solutions – among all possible solutions – whose probabilities will be determined in the SSR step that follows.

Nevertheless, in this section we view the SSI step as an end in itself and assess its performance.

This section is restricted to cases where all observations have complete information (i.e., allele proportions for each locus). This condition will be relaxed in §3.3.

The section is organized as follows. As before, we first specify the configuration of the simulation run. Second, we investigate the detection of the MOI among all solutions for a given observation. Third, we define several measures of accuracy used in later sections and, fourth, we investigate the accuracy of the solutions produced. To put `LITSE`'s performance in perspective, we compare it to that of `IMH`, which is described in §2.5.4.1.

3.2.4.1 Configuration of Simulation

The parameter settings for this simulation are presented in Table 3.7.

Parameter	Setting
`nl`	10,20,30
`ll`	3,10
`hpn`	10,20,30
`no`	10,15,20
`maxmoi`	4
`c`	100,200,1000,1e9
`br`	0
`mfl`	5
`smnc`	10,15,20,30
`hmnc`	50
`mer`	Inf
`am`	2
`mi`	0
`inst`	1

Table 3.7: Simulation Settings: SSI Accuracy. Each parameter in the "Parameter" column was tested for each value in in the corresponding entry in the "Settings" column. Each combination of the parameter settings is tested `inst` times.

The total number of trials is thus equal to the product of the number of settings for each parameter times the value for `inst`. This means a total of

$$3 \times 2 \times 3 \times 3 \times 1 \times 4 \times 1 \times 1 \times 4 \times 1 \times 1 \times 1 \times 1 \times 1 = 864$$

trials, each having ten, fifteen, or twenty observations.

Including the value 3 among the set for `ll` will tend to encourage the "doubling up" of alleles at each locus, since each haplotype will have relatively few alleles from which to choose. The range of values for `c` are intended to cover cases with high (100), medium, and low (1e9) measurement error. As mentioned earlier, `br` is left at 0 since the significance of missing data is best explored in §3.3. The value of `hmnc` is sufficiently high that it has no effect in this simulation. Once again, the values for `mer` and `mi` are such that they are irrelevant in this run.

Table 3.8 presents a summary of the simulation run, which is slightly larger than that for analyzing the runtime.

Quantity	Value
Number of Observations	12960
Number of Trees	29834
Number of Trees per Observation	2.30
Number of Solutions	4687177
Average Tree Size (Number of Nodes)	19316

Table 3.8: SSI Accuracy: Simulation Stats. The number of trees is slightly below $12960 \cdot 3 = 38880$ because all solutions in some trees were deemed invalid; this occurred when all solutions for the tree were deemed invalid when an MOI higher than the true MOI was attempted. See §2.7.2.5 for details.

3.2.4.2 Detecting the True MOI

As a first test of the algorithm, we investigate how `LITSE` might detect the correct MOI for an observation when we have several possible MOIs from which to choose.

Figure 3.12 depicts the distribution of the observations in the simulation run by true MOI and the maximum locus length. Recall that the former will always be greater than or equal to the latter. The cases where the true MOI is strictly greater than the maximum locus length are those where all loci in the observation feature "doubling up" of alleles; that is, in the true solution each locus has one or more allele repeats. Reviewing the figure, it is apparent that for true MOIs of 1 or two, the maximum locus length equals

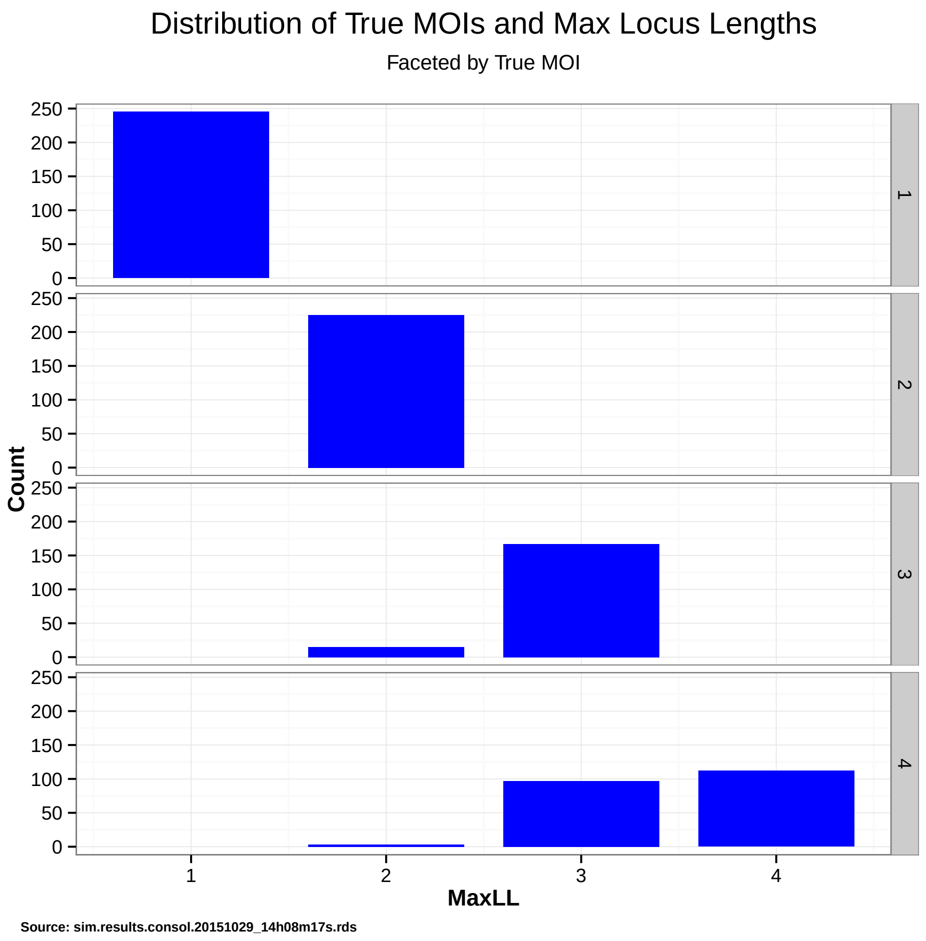


Figure 3.12: Distribution of True MOIs in Simulation. Each observation in the simulation will have a true MOI and a maximum locus length. The figure shows the distribution of the true MOI/max locus length combinations in the simulation. Combinations where the maximum locus length is strictly less than the true MOI represent cases of "doubling up".

the true MOI in all cases. This is logical; for true MOIs of 1, there is only one haplotype and thus all loci lengths (and thus the maximum) must equal 1. For true MOIs greater than 1, the maximum locus length must always be greater than 1 since the haplotypes must differ at at least one locus. However, it is possible, as witnessed in this simulation, to have a maximum locus length equal to 2 even when the true MOI is greater than 2. In the simulation, there is a wide cross section of combinations present.

Recall that the SSI procedure will typically try a total of `am` + 1 different MOIs for an observation, starting at the minimum possible MOI, which is the maximum locus length.[2] The exception is when the solutions for an attempted MOI are all invalid or there is insufficient improvement in the error (as checked using the `mer` parameter), at which point the algorithm ceases to check new MOIs for the observation. In this simulation, since `mer` $= \infty$, meaning that any improvement (or even deterioration) at one MOI compared to its predecessor is accepted, an MOI and its successors are excluded only if all solutions for the MOI are deemed invalid. Using this approach, it is critical that the `LITSE` algorithm retain the true MOI for an observation and reject as many invalid MOIs as possible. Figure 3.13 shows the results. The algorithm has retained all true MOIs, as indicated by solid green ("retain") bars in the "E" column in each cell, with a cell representing the results for a particular combination of the parameter c and true MOI. This is a positive result. We also see that the algorithm has retained every MOI less than the true MOI.[3] For MOIs greater than the true one, we see several things. First, when the true MOI is one, all attempts using a higher MOI are rejected; this is because any solution with an MOI greater than one will by necessity have duplicate haplotypes, making it invalid. Second, when the value of the c parameter is high, indicating very little measurement noise, such cases are rejected about 50% of the time for true MOIs of 2, 3, and 4. Ideally, we would have rejected all cases where the attempted MOI was different from the true one; thus this first pass classification of MOIs is suboptimal.

Related to the above, a complicating factor is that the errors of the solutions may on average decrease as the estimated MOI increases; intuitively, this is because as the number of assumed haplotypes increases, the algorithm has more flexibility in fine-tuning the construction of specific haplotypes to mirror the observed allele proportions (assuming that such a solution remains valid). Any such property will contaminate the computation of the estimation of the MOI for the observation, as represented in equation (2.51), since inflated values for the MOI will tend to carry better fits with the observed data and thus higher probabilities for $f_{\mathbf{X}^{(l)}|\mathbf{U}^{(l)}}(\mathbf{x}^{(l)}, \mathbf{u}^{(l)})$, defined in equation (2.18). This effect is actually beneficial for the MOIs below the true one; they will be downweighted in the

[2]`am` $= 2$ in this simulation.

[3]This result is in fact implied by the previous result. For the algorithm to evaluate and retain the true MOI, it must have evaluated and retained any lower MOIs along the way.

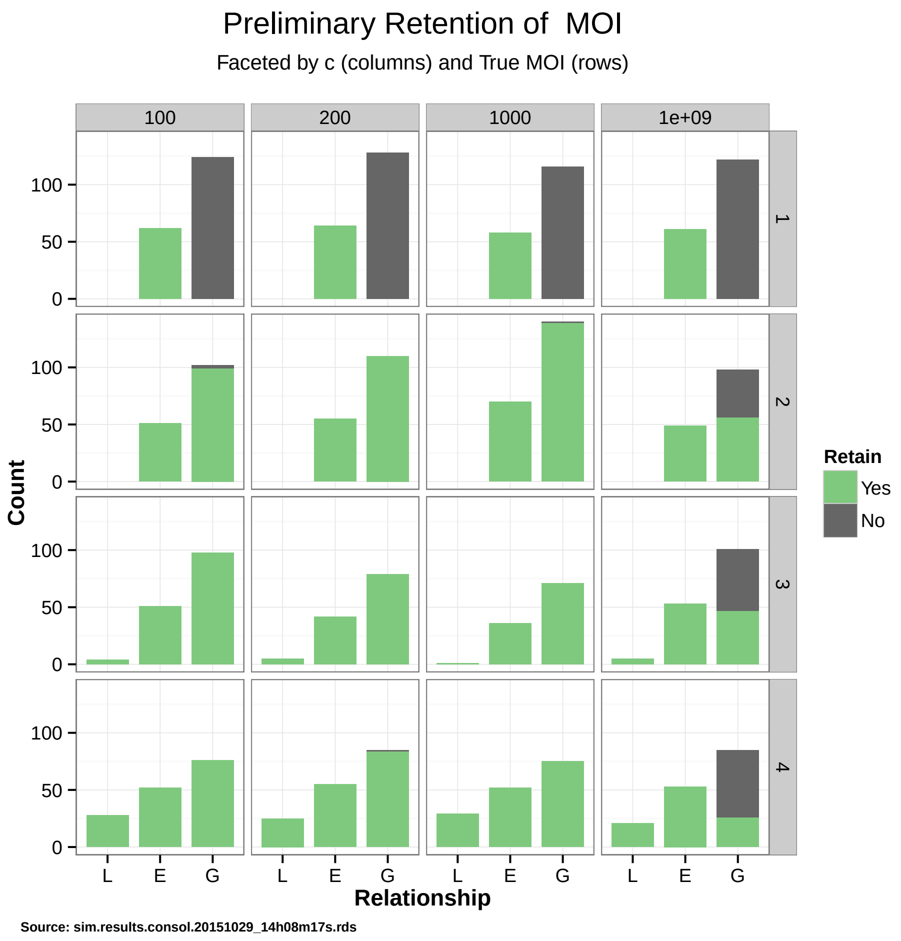


Figure 3.13: Preliminary Classification of MOI. This figure shows the outcome (categorization) of the initial filtering for each MOI considered for each observation. Each MOI for each observation is either retained or rejected. The MOIs are grouped according to whether they are L(ess) than, E(qual) to, or G(reater) than the true MOI.

likelihood equation. It is detrimental for those above the true MOI, and an additional method will be proposed to detect such cases.

This behavior is depicted below for all valid MOI instances.[4] Figure 3.14 shows how the minimum error for each observation in the current simulation varies as the estimated MOI increases when we separate the results by the concentration parameter `c` and the true MOI, with a single line linking the results per observation. The results are somewhat

[4]In this simulation, since `mer` $= \infty$, an MOI is omitted only if all solutions are deemed invalid.

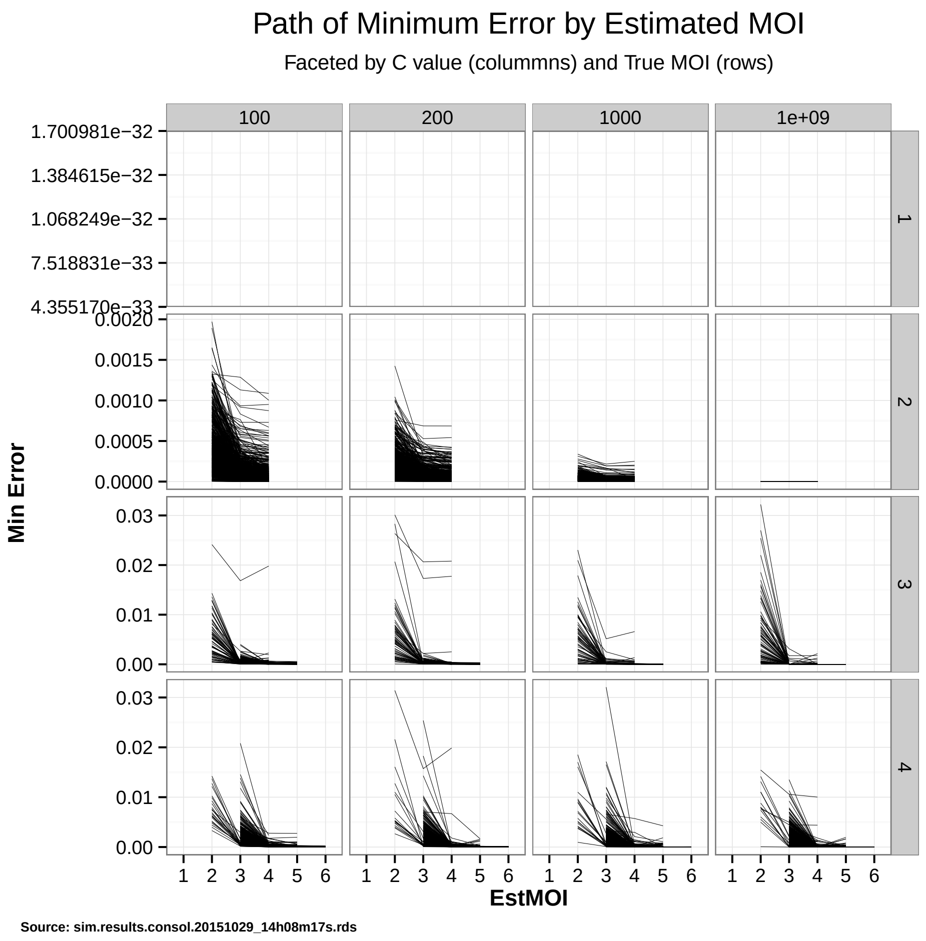


Figure 3.14: Minimum Error by Estimated and True MOI. For each observation and for each estimated MOI for that observation, we record the minumum error among all solutions at that estimated MOI. We then plot, per observation, the path of this minimum error over the three estimated MOIs used for that observation. The results are faceted by true MOI (vertically) and the value of c (horizontally).

difficult to extract from the figure because in each pane there are many lines close to one another. Nevertheless, in most cases, the minimum error declines as the estimated MOI increases, particularly for higher MOIs.

Figure 3.15 presents the same data in a simpler format. Here, the data have been normalized and averaged to more easily view the typically pattern of the minimum error over different values for the estimated MOI. We again see that the minimum error typically

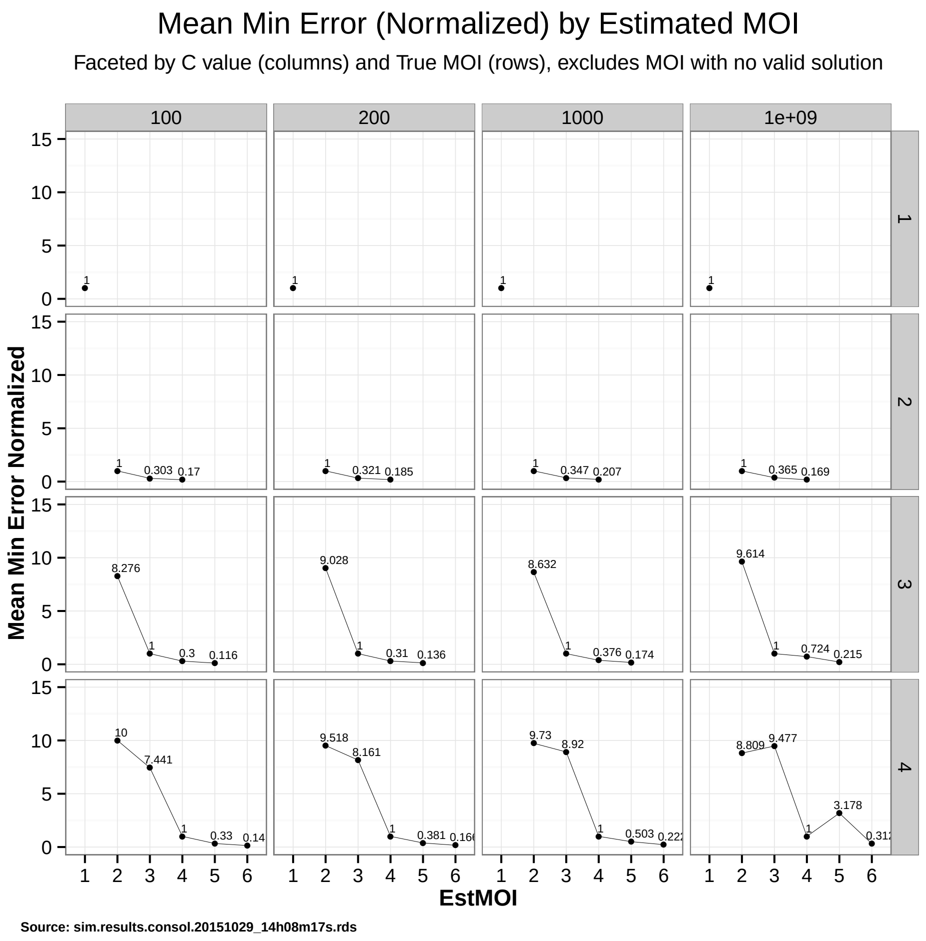


Figure 3.15: Mean Normalized Minimum Error by Estimated and True MOI. For each observation and for each estimated MOI for that observation, we record the minumum error among all solutions at that estimated MOI. We then normalize these averages by setting the value at the true MOI equal to 1 and expressing the values for the other MOIs as a ratio to 1. Finally, we then plot the path of this normalized average over the three estimated MOIs. The results are faceted by true MOI (vertically) and the value of c (horizontally).

decreases as the estimated MOI increases, yet this effect tends to flatten out after the true MOI has been reached. For cases with essentially no error, corresponding to $c =$1e9, the effect is even greater: the minimum error bottoms out at the correct MOI and rises again for higher MOIs. The reason for this behavior is that when the `LITSE` algorithm finds the correct solution, in the absence of any variance this correct solution implies no error relative to the observed allele proportions.

The primary result here is that as the estimated MOI increases, the improvement tends to flatten out or disappear altogether. To this end, we propose that the `LITSE` algorithm use a control parameter – which we term the *maximum error ratio* or `mer` – to discard what are likely to be solutions from inflated estimates of the MOI.

The parameter `mer` is used as follows. For each observation and all solutions for that observation at a particular MOI, the `LITSE` algorithm identifies the minimum error. As it progresses through the set of MOIs, it computes this figure for the latest MOI. When the ratio between this figure at one MOI and that of its predecessor is greater than `mer`, the `LITSE` algorithm discards the results from that MOI and stops considering higher MOIs. Otherwise, the results are retained and the algorithm moves on to the next MOI, assuming that there are MOI left to try given the value set for `am`. Informally put, we view this latter situation as a case where there is sufficient evidence that the true MOI may be at least this latest MOI attempted.

Example 3.1 (Use of `mer`). Assume that `mer` has been set to 0.5 and `am` has been set to 2. These parameters will hold for all observations.

Consider an observation whose maximum locus length is 3. Given this, we know that the true MOI is at least 3 and possibly greater. Given the value for `am`, this means we will consider a maximum of three MOIs: 3, 4, and 5 for that observation.

The algorithm will first compute all solutions for an assumed MOI of 3. Assume that the lowest error among all solutions is $\epsilon_{\min,3}$. The algorithm will then compute all solutions for an assumed MOI of 4. If the lowest error among all solutions for this assumed MOI, $\epsilon_{\min,4}$, is greater than $\texttt{mer} \cdot \epsilon_{\min,3} = 0.5\epsilon_{\min,3}$, then the results are discarded and we keep only the results for the assumed MOI of 3. Otherwise, we retain the results for the assumed MOI of 4 and then compute solutions for an MOI of 5. We perform the same check again. If $\epsilon_{\min,5} > 0.5\epsilon_{\min,4}$, then we discard the results (but still retain those for MOIs of 3 and 4). Otherwise, we retain the results for the MOI of 5 as well. Since $3 + \texttt{am} = 3 + 2 = 5$, we have completed the highest MOI we will attempt and do not attempt higher MOIs regardless of the results of this last check.

□

What value to use for `mer` in practice remains an outstanding question. Figure 3.16 plots the percentage of times the algorithm stops at the correct MOI for an observation in the simulation over varying values of `mer`. The value of `mer` that produces the highest

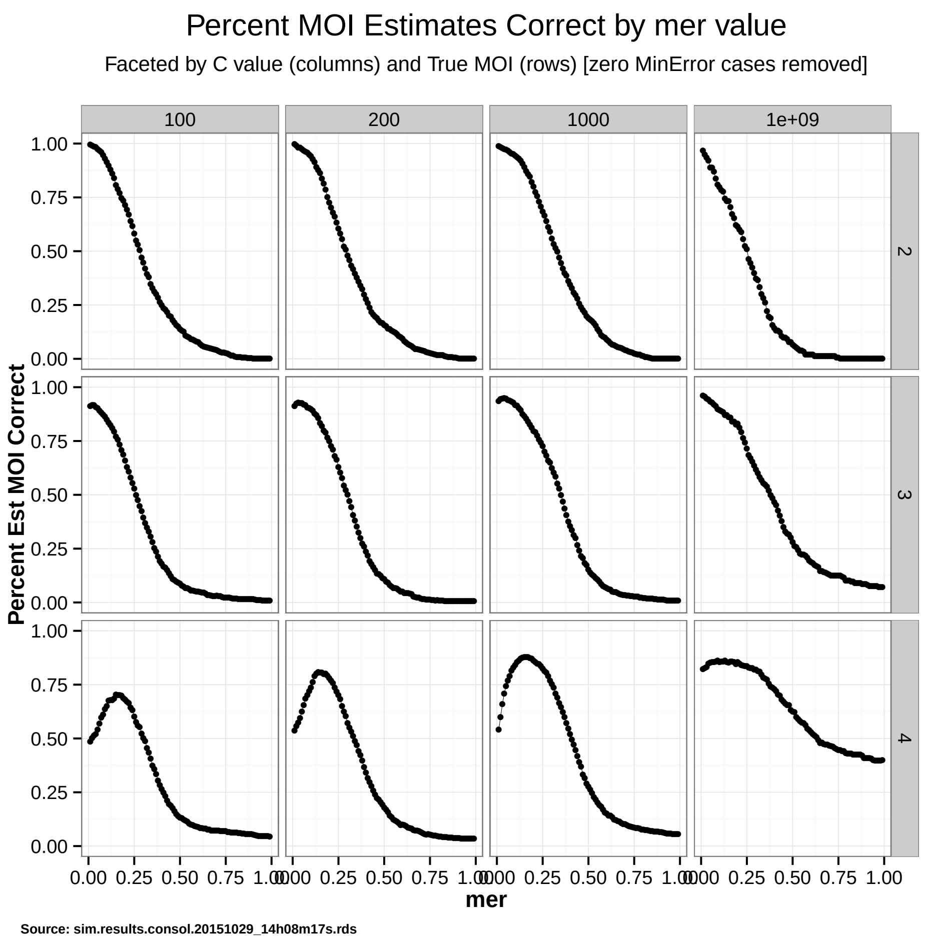


Figure 3.16: Percent MOIs Correct by `mer`. Each panel shows the percentage of observations for which the stopping MOI was determined correctly (i.e., the algorithm stopped at the correct MOI) for each value of `mer` tested. The results are faceted by true MOI (vertically) and the value of c (horizontally).

accuracy varies somewhat depending on the true MOI – for true MOIs of 2 and 3 it appears that a value close to zero is optimal while for a true MOI of 4 a value of 0.2 appears best. The value of 0.20 appears to be a good compromise and it is this value that will be implemented when testing the SSR procedure in §3.3. In other words, a proposed MOI will be considered only if its lowest error is at least 80% lower than that of the preceding MOI.

3.2.4.3 Measures of Accuracy

Before analyzing the accuracy of the solutions in §3.2.4.4, we first define several measures of accuracy that will be used to assess the `LITSE` algorithm.

In all cases, we will compute the accuracy of an individual *proposed* solution vis-à-vis the corresponding true solution for that observation.

For the following definitions, let $\widehat{H} = \{\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_{\text{EstMOI}}\}$ be a proposed solution set of haplotypes and let $H = \{h_1, h_2, \ldots, h_{\text{TrueMOI}}\}$ be the true solution against which $H(\hat{\mathbf{y}})$ will be evaluated.

3.2.4.3.1 Correct Solution

A solution is deemed to be correct if and only if it matches the MOI and haplotype composition of the true solution, where the "true solution" is the set of haplotypes that actually generated the observed allele proportions. Note that a solution can be correct even its proposed proportions are incorrect.

3.2.4.3.2 Recall

As a general metric, recall measures the percentage of true positives that are classified as such using a particular method like `LITSE`. Applying this principle to our problem, we define recall as,

$$\textit{Recall}(\widehat{H}) := |\widehat{H} \cap H|/|H| \tag{3.1}$$

In words, equation (3.1) is defining recall as the number of haplotypes found in *both* the proposed and true haplotype sets divided by the number of haplotypes in the true solution. Since $|H|$ is the number of haplotypes in the true solution, it is simply equal to what we have been calling the true MOI. By construction, $\textit{Recall}(\widehat{H}) \in [0, 1]$.

Note that the haplotype proportions – either true or estimated – play no role in the computation of recall.

Example 3.2 (Recall). First, let the true solution for an observation be

$H =$

Num	L1	L2	L3	L4	L5	Prop
1	1A	2B	3A	4D	5C	0.34
2	1B	2A	3A	4C	5E	0.15
3	1B	2B	3C	4D	5C	0.51

Here, $|H| = 3$; i.e., the true MOI equals 3. The row ordering of the three haplotypes has been determined by the haplotypes' column-wise definitions; that is, first by the column for locus 1, then for locus 2, and so on.

While setting the order of the haplotypes in the table is a somewhat arbitrary choice, performing this ordering using locus column entries from left to right makes it easier to search for a specific haplotype in either a true or proposed solution.

Now consider the following candidate solutions.[5]

First, let

$\widehat{H}_1 =$

Num	L1	L2	L3	L4	L5	Prop
1	1A	2B	3A	4D	5C	0.30
2	1B	2A	3A	4C	5E	0.17
3	1B	2B	3C	4D	5C	0.53

Here, we see that the proposed haplotypes are exactly the same as those in the true solution. This implies that every haplotype in H has a match in $\widehat{H}_1$. While the proportions are not exactly correct, this is irrelevant for the recall calculation. Thus

$$\mathit{Recall} = \frac{3}{3} = 1.$$

Next, let

$\widehat{H}_2 =$

Num	L1	L2	L3	L4	L5	Prop
1	1A	2B	3A	4D	5C	0.68
2	1B	2A	3C	4C	5E	0.32

We notice immediately that the proposed MOI is lower than the correct MOI. Thus there is no possibility of achieving a perfect recall. The first haplotype in H has a match in

[5]While all of these candidate solutions are feasible, we ignore their relative likelihoods. They are simply presented as illustrations. They represent a subset of all the candidate solutions that would be proposed by LITSE.

$\widehat{H}_2$ while the second and third do not, so

$$Recall = \frac{1}{3}.$$

Next, let

$\widehat{H}_3 =$

Num	L1	L2	L3	L4	L5	Prop
1	1A	2B	3A	4D	5C	0.30
2	1B	2A	3A	4C	5E	0.16
3	1B	2B	3C	4D	5C	0.52
4	1B	2B	3C	4D	5E	0.02

This time, the proposed MOI is greater than the correct MOI. Each of the three haplotypes in H is matched by an entry in $\widehat{H}_3$. Thus,

$$Recall = \frac{3}{3} = 1.$$

Note that $\widehat{H}_3$'s recall was not penalized by suggesting an invalid haplotype in position 4. However, a penalty will be realized in the precision calculation below.

Finally, let

$\widehat{H}_4 =$

Num	L1	L2	L3	L4	L5	Prop
1	1A	2A	3A	4D	5C	0.10
2	1A	2B	3A	4D	5C	0.30
3	1B	2A	3A	4C	5E	0.16
4	1B	2B	3C	4D	5E	0.44

Once again, the proposed MOI is greater than the correct MOI. This time, the first haplotype in H is matched by the second in $\widehat{H}_4$ and the second haplotype in H is matched by the third in haplotype in $\widehat{H}_4$. The third haplotype has no match in $\widehat{H}_4$. Thus,

$$Recall = \frac{2}{3}.$$

□

3.2.4.3.3 Precision

As a general metric, precision measures the percentage of proposed positives that are correct. Applying this principle to our problem, we define precision as,

$$Precision(\widehat{H}) := |\widehat{H} \cap H| / |\widehat{H}| \tag{3.2}$$

By construction, $\mathit{Precision}(\widehat{H}) \in [0,1]$.

Example 3.3 (Precision). Using the same true solution and proposed solutions from Example 3.2, we get the following results.

For $\widehat{H}_1$, all the proposed haplotypes have matches in H. So $\mathit{Precision}(\widehat{H}_1) = 3/3 = 1$.

For $\widehat{H}_2$, the first of the proposed haplotypes has a match while the second does not. So $\mathit{Precision}(\widehat{H}_2) = 1/2$.

For $\widehat{H}_3$, the first three proposed haplotypes have matches while the fourth does not. Thus $\mathit{Precision}(\widehat{H}_3) = 3/4$. Indeed, before performing any comparisons, we know that a proposed solution with an excessive estimated MOI must have a precision less than 1 since at least one proposed haplotype must be incorrect.

For $\widehat{H}_4$, only the second and third of the proposed haplotypes were correct. So $\mathit{Precision}(\widehat{H}_3) = 2/4 = 1/2$.

□

On the basis of the previous two examples, we make the following two points regarding recall and precision:

1. A solution is correct if and only if both its recall and precision equal 1.
2. The proposed solution $\widehat{H}_4$ demonstrates a case of both low recall and precision. Indeed, a proposed solution could have a recall and precision both equal to 0.

3.2.4.3.4 Proportional Recall

Note that in the definition of *Recall* in equation (3.1), a true haplotype must be matched across all loci to be considered "detected" in candidate solution. In cases where the number of loci is relatively large, say 20-30, this means a true haplotype will be considered absent from a proposed solution if only a single locus is mismatched despite a very high proportion of locus matches. As a result, the "strict" recall *Recall* may understate the quality of a proposed solution.

A separate metric, *PropRecall*, exists to address this issue. It is defined as follows,

$$\mathit{PropRecall}(\widehat{H}) = \frac{\sum_{i=1}^{|H|}(\#\text{loci shared by } h_i \text{ and closest haplotype in } \widehat{H})/L}{|H|} \tag{3.3}$$

where L is the number of loci. By construction, $\mathit{PropRecall}(\widehat{H}) \in [0,1]$.

Given its definition,

$$\mathit{PropRecall}(\widehat{H}) \geq \mathit{Recall}(\widehat{H}),$$

since the latter can be viewed as a more restrictive version of the former, where the summand in the numerator is counted only on a perfect match.

From equation (3.3) it is clear that we will need to compare each haplotype in H against each haplotype in $\widehat{H}$ (or at least until we find a perfect match, at which point there is no sense in continuing comparisons for that haplotype).

Example 3.4 (Proportional Recall). Using the same true solution and proposed solutions from Examples 3.2 and 3.3, we note that $L = 5$ and obtain the following results.

For $\widehat{H}_1$, because of the relationship between *Recall* and *PropRecall* and the finding that $\mathit{Recall}(\widehat{H}_1) = 1$, it follows that $\mathit{PropRecall}(\widehat{H}_1) = 1$.

For $\widehat{H}_2$, we need to assess each of the three true haplotypes against the two proposed haplotypes. We already know from Example 3.2 that the first haplotype in H has a perfect match; thus the corresonding summand will equal 1 in the denominator in equation (3.3). For the second haplotype in H, the number of matching loci alleles versus the two proposed haplotypes is 1 and 4, respectively; thus the summand for the second haplotype is $4/5 = 0.8$. For the third haplotype, the number of matching loci alleles versus the two proposed haplotypes is 3 and 2, respectively; thus the summand for the second haplotype is $3/5 = 0.6$. Combining these results, $\mathit{PropRecall}(\widehat{H}_2) = (1 + 0.8 + 0.6)/3 = 0.8$.

For $\widehat{H}_3$, similar to $\widehat{H}_1$ and for the same reasons, $\mathit{PropRecall}(\widehat{H}_3) = 1$.

For $\widehat{H}_4$, we already know from Example 3.2 that the first two haplotypes in H have a perfect match; thus their corresonding summand will equal 1 in the denominator in equation (3.3). For the third haplotype in H, the number of matching loci alleles versus the four proposed haplotypes is 2, 3, 1, and 4, respectively. Thus the summand for the second haplotype is $4/5 = 0.8$. For the third haplotype in H, the number of matching loci alleles versus the two proposed haplotypes is 3 and 2, respectively. Thus the summand for the third haplotype is $3/5 = 0.6$. Combining these results, $\mathit{PropRecall}(\widehat{H}_4) = (1 + 1 + 0.8)/3 = 0.93$.

$\square$

This example illustrates an important point for *PropRecall*: haplotypes in a proposed solution can in theory be aligned against more than one haplotype in the true set of haplotypes.

3.2.4.3.5 Proportional Precision

The precision analog to *PropRecall*is defined as follows,

$$\mathit{PropPrecision}(\widehat{H}) = \frac{\sum_{i=1}^{|\widehat{H}|}(\text{\#loci shared by } \hat{h}_i \text{ and closest haplotype in } H)/L}{|\widehat{H}|} \tag{3.4}$$

As for the preceding metrics, by construction $\mathit{PropPrecision}(\widehat{H}) \in [0, 1]$. We have an analogous relationship between *PropPrecision*and *Precision*, namely,

$$\mathit{PropPrecision}(\widehat{H}) \geq \mathit{Precision}(\widehat{H}),$$

for the same reason underlying the relationship between *PropRecall* and *Recall*.

Example 3.5 (Proportional Precision). Continuing with the same true solution and proposed solutions from Examples 3.2 and 3.3, we obtain the following results.

For $\widehat{H}_1$, due to the relationship between *Precision* and *PropPrecision* and the finding that $\mathit{Precision}(\widehat{H}_1) = 1$, it follows that $\mathit{PropPrecision}(\widehat{H}_1) = 1$.

For $\widehat{H}_2$, we need to assess each of the two proposed haplotypes against the three true haplotypes. We know from Example 3.3 that the first haplotype in $\widehat{H}$ has a perfect match against the first haplotype in H; thus the corresonding summand will equal 1 in the denominator in equation (3.4). For the second haplotype in $\widehat{H}_2$, the number of matching loci alleles versus the two proposed haplotypes is 1 and 4, respectively. Thus the summand for the third haplotype is $4/5 = 0.8$. Combining these results, $\mathit{PropPrecision}(\widehat{H}_2) = (1 + 0.8)/2 = 0.9$.

For $\widehat{H}_3$, the first three haplotypes have perfect matches in H and thus have a corresponding summand of 1. For the fourth, the number of positions matched against the three true haplotypes is 2, 2, and 4, respectively; thus the corresponding summand is $4/5 = 0.8$. So $\mathit{PropPrecision}(\widehat{H}_3) = (1 + 1 + 1 + 0.8)/4 = 0.95$.

For $\widehat{H}_4$, we already know from Example 3.3 that haplotypes numbered 2 and 3 in $\widehat{H}_4$ have a perfect match; thus their corresonding summand will equal 1. For the first haplotype in $\widehat{H}_4$, the number of matching loci alleles versus the three true haplotypes is 4, 1, and 2, respectively. Thus the summand for the first haplotype is $4/5 = 0.8$. For the fourth haplotype in $\widehat{H}_4$, the number of matching loci alleles versus the two proposed haplotypes is 2, 2, and 4, respectively. Thus the summand for the fourth haplotype is $4/5 = 0.8$. Combining these results, $\mathit{PropPrecision}(\widehat{H}_4) = (0.8 + 1 + 1 + 0.8)/4 = 0.9$.

□

3.2.4.4 Accuracy of Solutions

In this section, we investigate the accuracy of the SSI procedure. There are two primary questions that we attempt to answer:

1. What is the relationship between a solution's rank and its accuracy, as defined in §3.2.4.3? On what factors does this relationship depend?
2. What is the relationship between a solution's rank and its likelihood (for now, in an informal sense) of being the correct solution if we assume that we know the true MOI? On what factors does this relationship depend?

Regarding the second question above, in reality, the MOI will need to be estimated. Section §3.3 will loosen this assumption by applying the technique in §3.2.4.2 for this estimation. In addition, while the number of solutions will vary from observation to observation and is partially dependent on the runtime parameter settings, typically only the top solutions for any observation have a chance of being correct.

To put the remainder of this section in perspective, first consider the profile of all errors for the solutions generated in this simulation run, as depicted in Figure 3.17, broken down by the error rate in the observed data as dictated by the parameter $\mathbf{c}$. We see a positive skewness in the distribution with a concentration near zero and a long tail to the right, a pattern more significant for higher values of $\mathbf{c}$. The distribution is presented in alternative format in Table 3.9.

3.2.4.4.1 Relationship between Rank and Measures of Accuracy

In this section, we investigate how a solution's error, as defined in §2.7.2.4.1, and accuracy, as defined in §3.2.4.3 interrelate.

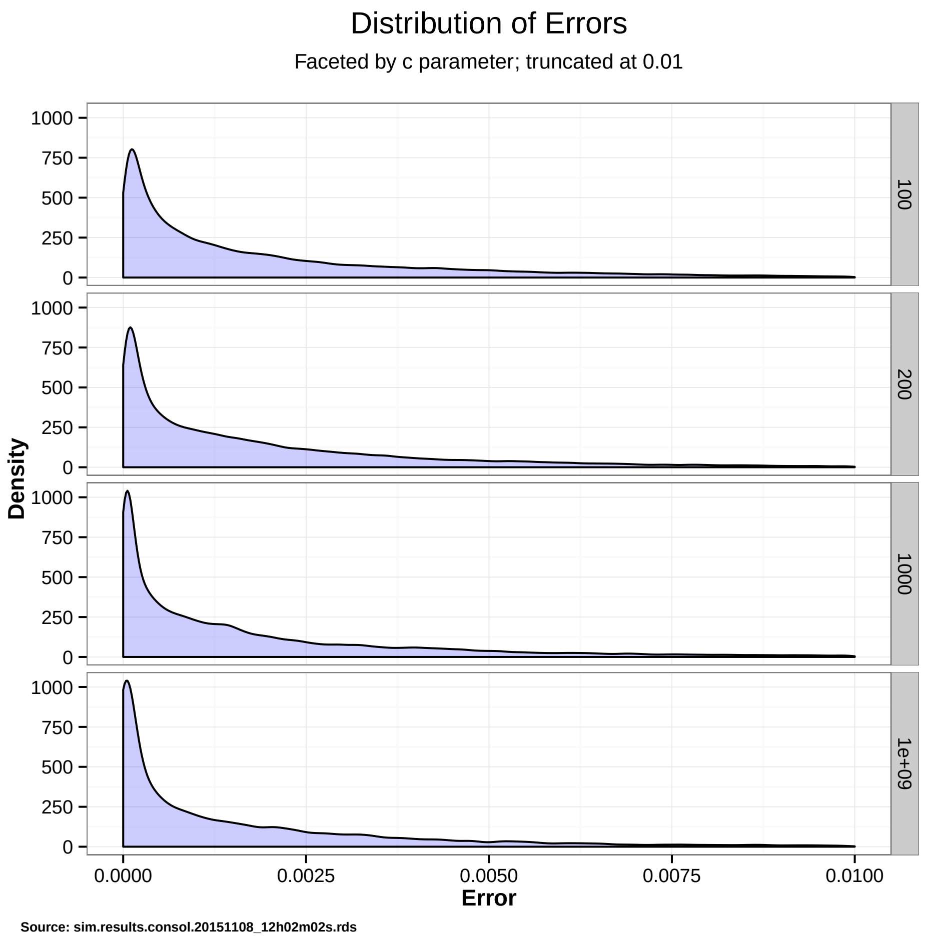


Figure 3.17: Error Distribution. The figure represents a density plot of the errors for all solutions.

Quantile	EquivPercent	Value
0	0%	0.0000
1	20%	0.0002
2	40%	0.0006
3	60%	0.0016
4	80%	0.0034
5	100%	0.0911

Table 3.9: Quantiles of Errors.

Recall that within the SSI procedure, error is the only criterion by which we can evaluate a solution.[6] Given the ultimate aim to discover and focus on solutions that are "accurate", it is important that a meaningful relationship exist between the relative error (i.e., rank) and correct solution. To investigate this, we consider each of the four accuracy measures in turn.

To separate the effects of choosing the correct MOI versus choosing an accurate solution *within* the correct MOI, in the following discussion we assume that the correct MOI has been identified. These two effects will be allowed to interact in section §3.3.

First consider Figure 3.18, which investigates the relationship between rank and recall when the correct MOI is attempted. The figure is divided into 16 cells or panels, each corresponding to a combination of values for `c` and `smnc`. The horizontal axis represents the rank of a solution for an observation among all solutions for that observation at the same MOI; we treat ranks 1 through 4 separately and combine rank 5 with all higher ranks. In each cell, for each of the five bins, the distribution among the realized recall values is displayed in a stacked bar chart, with darker values representing superior performance. The recall values are discrete in this case and are a function of the true MOIs that were used in this run. As reported in Table 3.7 for the parameter `maxmoi`, these were 1 through 4. An MOI of 1 has possible recall values of 0 or 1; for 2, the possible values are 0, 0.5, and 1; for 3, they are 0, 0.333, 0.667, and 1; and for 4, they are 0, 0.25, 0.5, 0.75, and 1. The union of these possible values thus represents the complete set of recall values that will be witnessed over all solutions in the simulation. Because of the structuring of the simulation, which performs the runs on the Cartesian product of all tested parameter values, each cell and each bin can witness any of the possible recall values.

There are several comments to make regarding Figure 3.18. First, there is a clear inverse relationship between rank and recall. This is evident in the direct relationship between rank and the skewness towards low recall values, which take on lighter shades in the figure. This relationship appears to hold for all panels in the figure, and is consistent with our expectations. It is also evidence in support of the `LITSE` algorithm, which uses error to guide the construction of solutions with high recall (among other properties such

[6]When considering the overall algorithm, in particular the EM component, this will change.

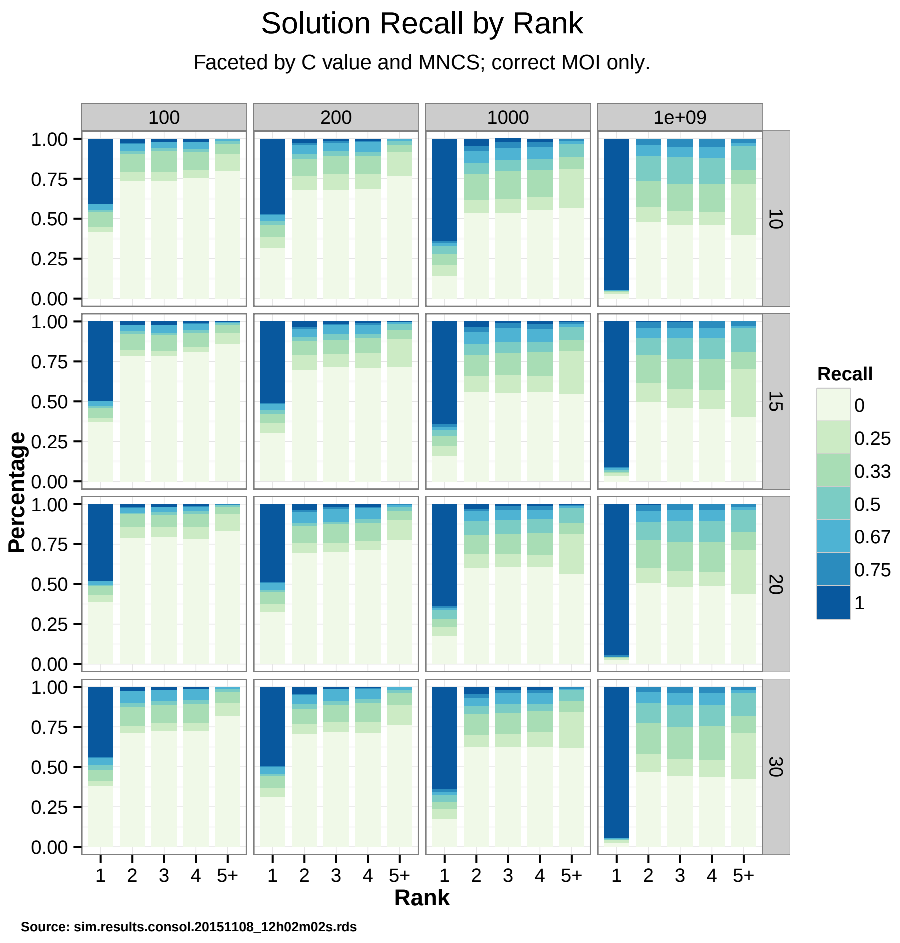


Figure 3.18: Rank vs. Recall. The results are faceted by `smnc` (vertically) and the value of c (horizontally). Each panel shows a bar chart for the combination. Each bar chart shows the distribution of the recall results for each of the bins.

as precision). Second, as the variance of the allele proportions decreases, corresponding to higher values of **c** as one moves rightward across the columns in the figure, the recall of the solutions in any bin increases. This is again as expected: when the variance of the observed proportions is low (high), meaning they deviate little (greatly) from the truth, they are a good (poor) guide to the true haplotype construction. Finally, there is no apparent improvement in recall performance as `smnc` increases. We know from §3.2.1.3 that as `smnc` increases so does the number of solutions. It may be the case that higher values of `smnc` are simply permitting the inclusion of primarily low recall solutions.

Overall, the results can be viewed positively and consistent with expectations regarding the relationship between rank and recall.

Second, Figure 3.19 presents analogous results for precision for the same simulation run. Because we have restricted the solutions considered to those with the correct MOI,

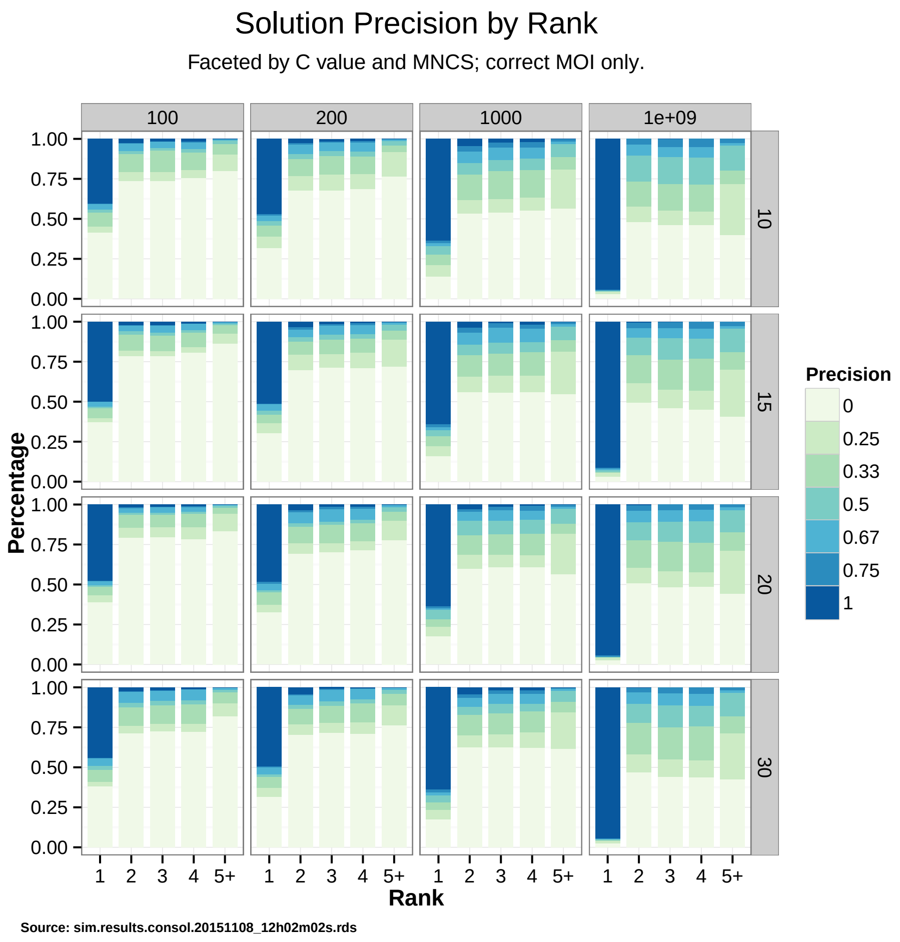


Figure 3.19: Rank vs. Precision. The results are faceted by `smnc` (vertically) and the value of c (horizontally). Each panel shows a bar chart for the combination. Each bar chart shows the distribution of the precision results for each of the bins.

it can be shown that recall and precision are identical in this case.[7] Nevertheless, we show the computed results as empirical evidence of this relationship.

[7]Since all solutions have the correct MOI, it follows that $|H| = |\hat{H}|$ for all solutions $\hat{H}$. Therefore, the denominator in equations (3.1) and (3.2) are the same in this case. Since the numerators are always identical, the two quantities are equal.

Next, Figure 3.20 investigates the relationship between rank and proportional recall. Here, while proportional recall is still a discrete variable, the number of possible values is much greater than with the "pure" recall depicted in Figure 3.18. Nevertheless, we present the results in the same format and ssee a similar pattern. As the variance

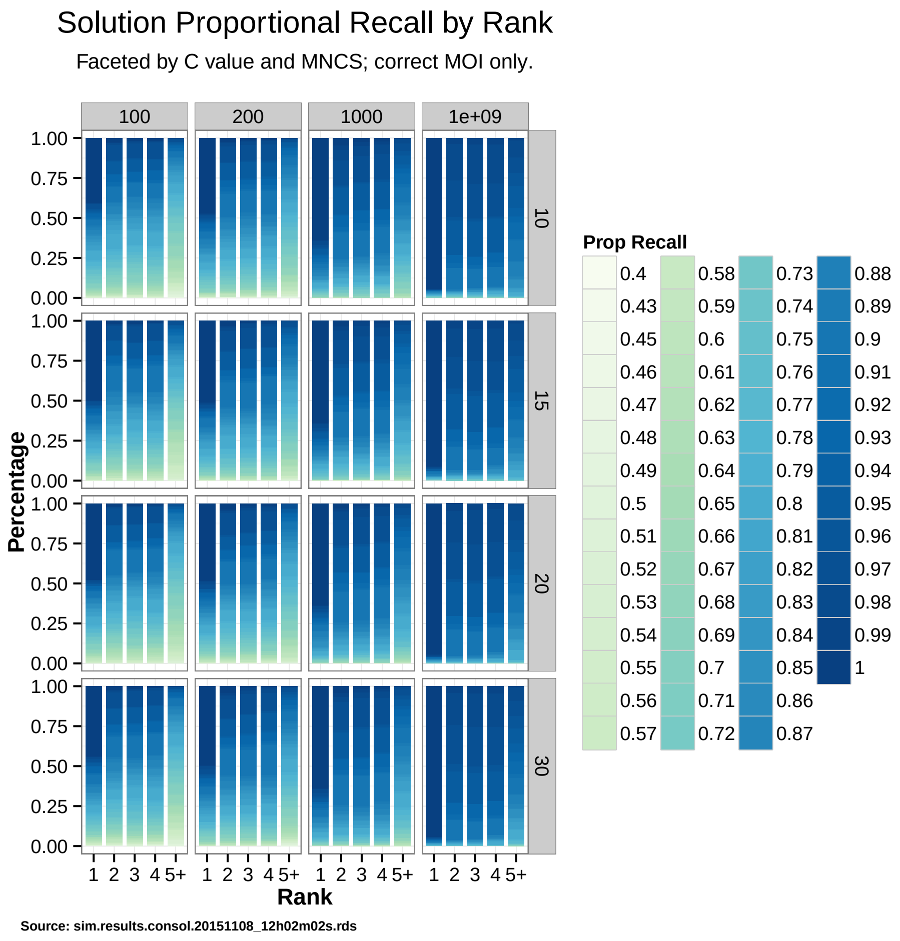


Figure 3.20: Rank vs. Proportional Recall. The results are faceted by `smnc` (vertically) and the value of c (horizontally). Each panel shows a bar chart for the combination. Each bar chart shows the distribution of the proportional recall results for each of the bins.

decreases, the distribution of proportion recall skews rightward for all ranks.

Finally, Figure 3.21 depicts the relationship between error and proportional precision. The results are identical to those for proportional recall for the same reason as the equality between recall and precision when the estimated MOI is correct.

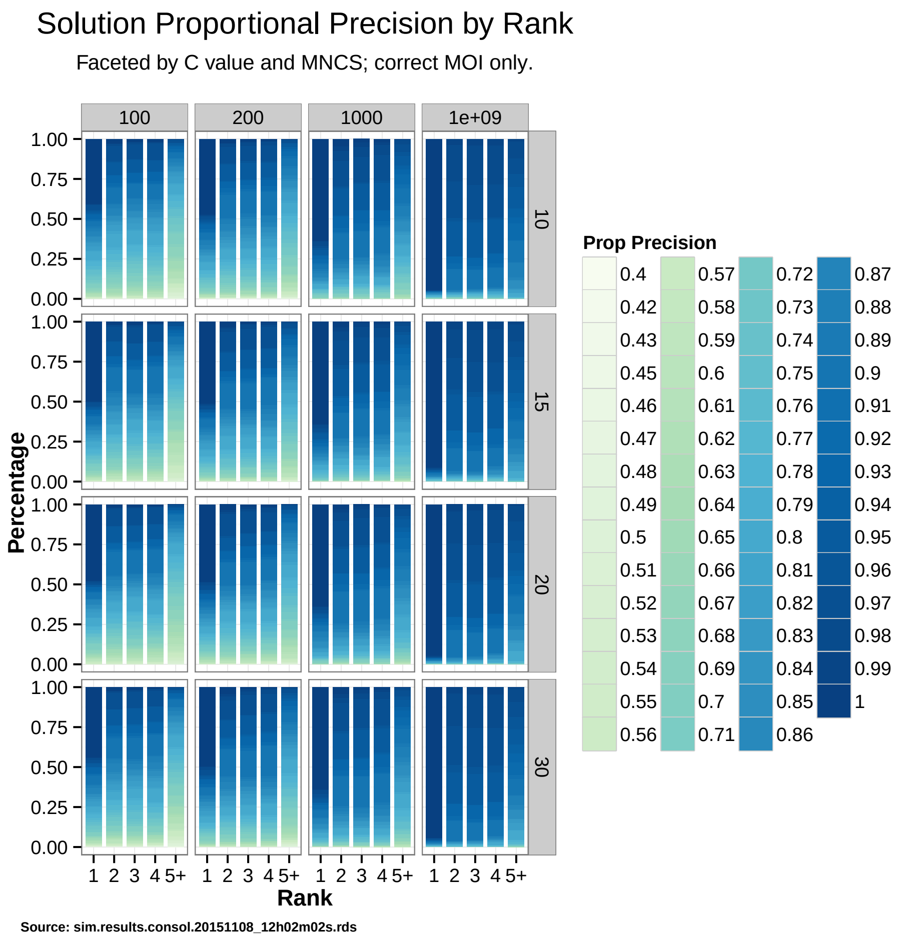


Figure 3.21: Rank vs. Proportional Precision. The results are faceted by `smnc` (vertically) and the value of c (horizontally). Each panel shows a bar chart for the combination. Each bar chart shows the distribution of the proportional precision results for each of the bins.

Overall, we view the results of this analysis as confirmatory of the basic approach of the `LITSE` algorithm. There is a clear inverse relationship between a solution's error and its accuracy. This is particularly true for cases where the variance is very low, where the results are essentially perfect.

3.2.4.4.2 Relationship between Rank and Correct Solution

In this section, we investigate how a solution's rank relates to the general frequency (or "likelihood") with which the solution is correct. Thus we attempt to determine where in a series of solutions the correct solution lies.

To this end, Figure 3.22 depicts the split between the observations for which the algorithm did and did not identify the correct solution somewhere among the list of all solutions, and Table 3.10 presents the precise percentages for the same results. We see

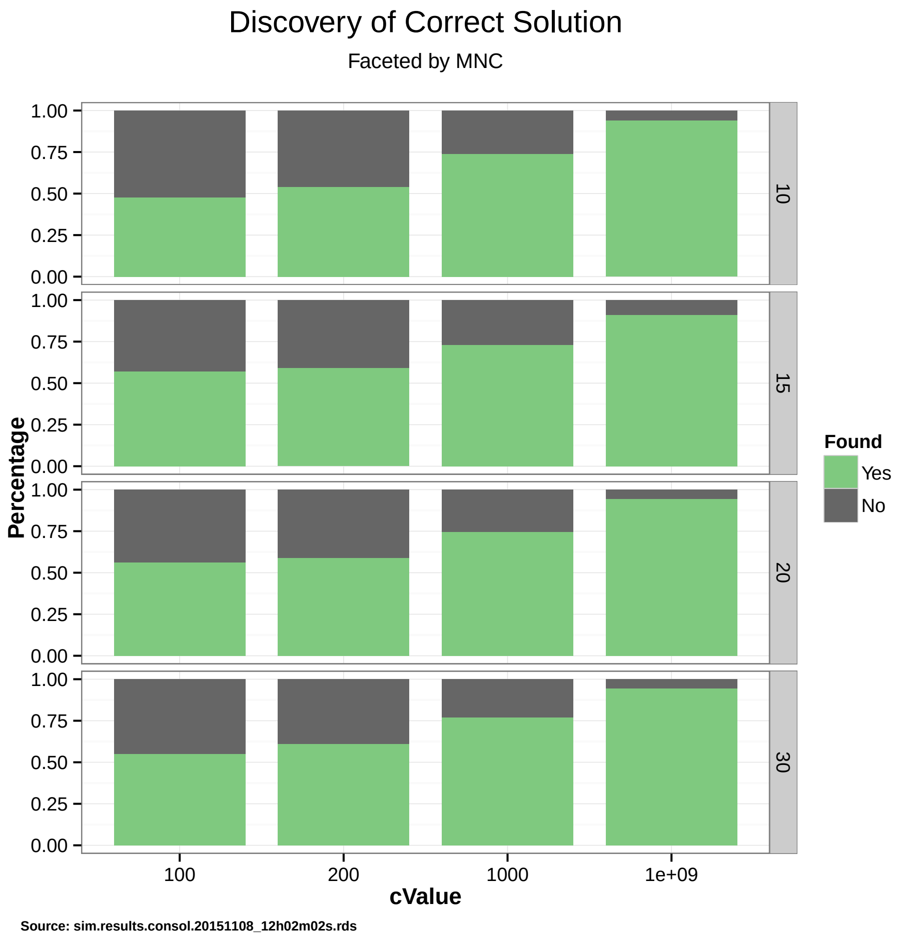


Figure 3.22: Discovery of Correct Solution. Each panel shows the distribution of the split. The results are presented by value of c (horizontally) and are faceted by `smnc` (vertically).

that both variables influence the ability of the algorithm to include the correct solution. In particular, the ability of the algorithm to find the correct solution improves as the error decreases. As the `smnc` increases, there is a slight improvement in the discovery, particularly for the case with low error. These results are consistent with expectations and previous results, though the low degree of improvement with higher values of `smnc` is

`smnc`	$c = 100$	$c = 200$	$c = 1000$	$c = 1e + 09$
10	47.9	54.0	73.8	94.2
15	57.0	59.3	73.2	91.2
20	56.0	58.8	74.6	94.4
30	55.1	61.0	77.0	94.3

Table 3.10: Percentage of Observations with Correct Solution Discovered.

surprising.

For those observations for which the correct solution was found, Figure 3.23 depicts the position or rank of the correct solution. Each rank is colored separately, with the colors starting at dark blue (for rank 1, i.e., the solution with the lowest error) passing through turquoise and finishing with light green. There are several comments regarding these results. First, we see that as the error decreases, the degree to which the correct solution lies past 1 decreases as well. At one extreme, for $c = 100$, for only 60-70% of the cases (depending on `smnc`) is the rank 1. At the other extreme, for $c = 1e9$, in all cases the rank is 1. This is as expected; low variance cases imply that the observed allele proportions are a good guide to determining the correct solution. Second, as `smnc` increases, the range of the rank of the correct solutions is generally wider, as evidenced by taller non-blue strips in the bar charts of the lower panels; the exception is the column for $c = 1e9$, where as mentioned the rank is always 1 and there is no perceivable difference among the `smnc` values.

3.2.4.4.3 Typical Paths to Correct Solution

The preceding discussion investigated the rank of the correct solution at the bottom level of the tree but did not investigate how the algorithm arrived at the solution while passing through the tree. As a final analysis of the performance of the SSI routine, we look at the paths typically taken through the tree to get to the correct answer when it is included among all solutions. A path through the tree is the sequence of the ranks, one for each level from the root to the leaf level, of each node through which the solution passed. For example, if a solution at the bottom level is arrived at by passing through nodes each of

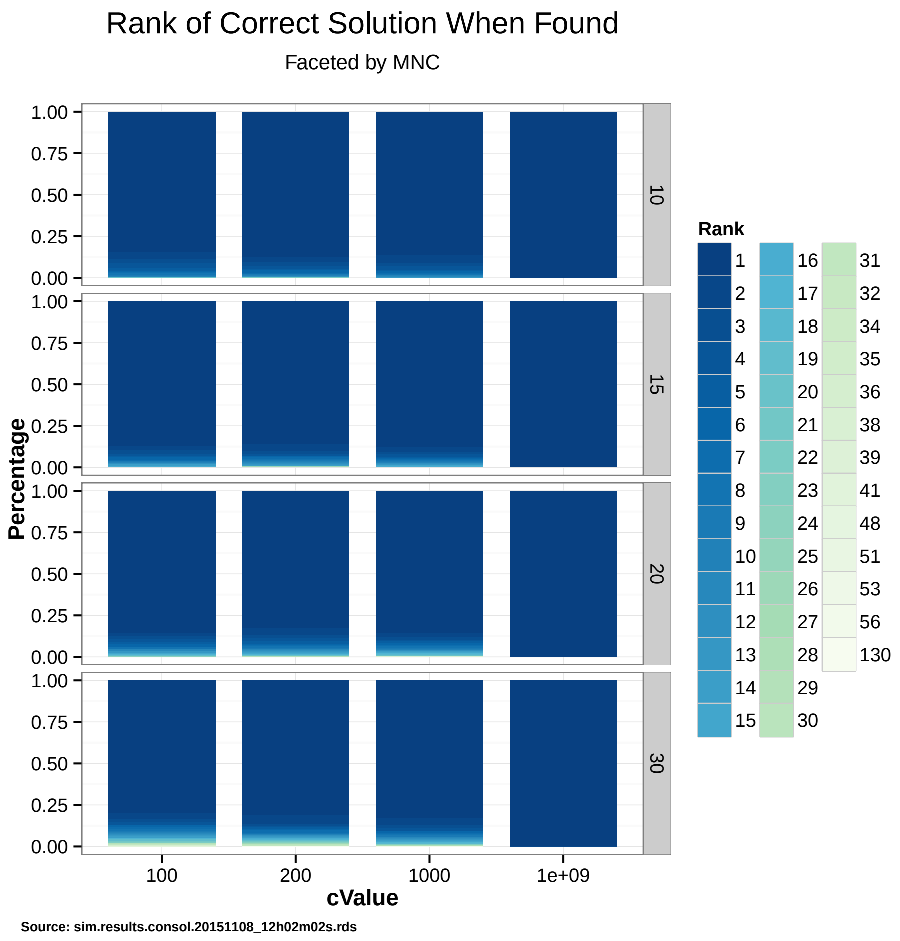


Figure 3.23: Rank of Correct Solution. Each panel shows the distribution of the rank in the cases where the correct solution was found. The results are presented by value of c (horizontally) and are faceted by `smnc`(vertically).

which ranked first in terms of error at its respective level, then the path is a sequence of 1's of length equal to the number of levels in the tree. This analysis will shed some light on the extent to which the algorithm needs to deviate from a greedy path (i.e., always take the node with the lowest error) to the bottom to include the correct solution.

In this discussion, we refer to a node as having "low" rank when the numerical value of the rank (and hence the relative error) is small. Thus low (high) rank nodes are those that appear the most (least) attractive when constructing a tree.

For the set of observations represented in Figure 3.23, a sample of the set of paths leading to the correct solution is included in Table 3.21 in the Appendix and a complete list can be found at `www.stat.berkeley.edu/~curt/litse/uniquepaths.html`. As seen on that webpage, there are 388, 245, and 182 unique paths for $N_l = 10, 20, 30$, respectively, found among the correct solutions in the simulation.

To investigate when paths deviate from a greedy one, first consider Figure 3.24. In

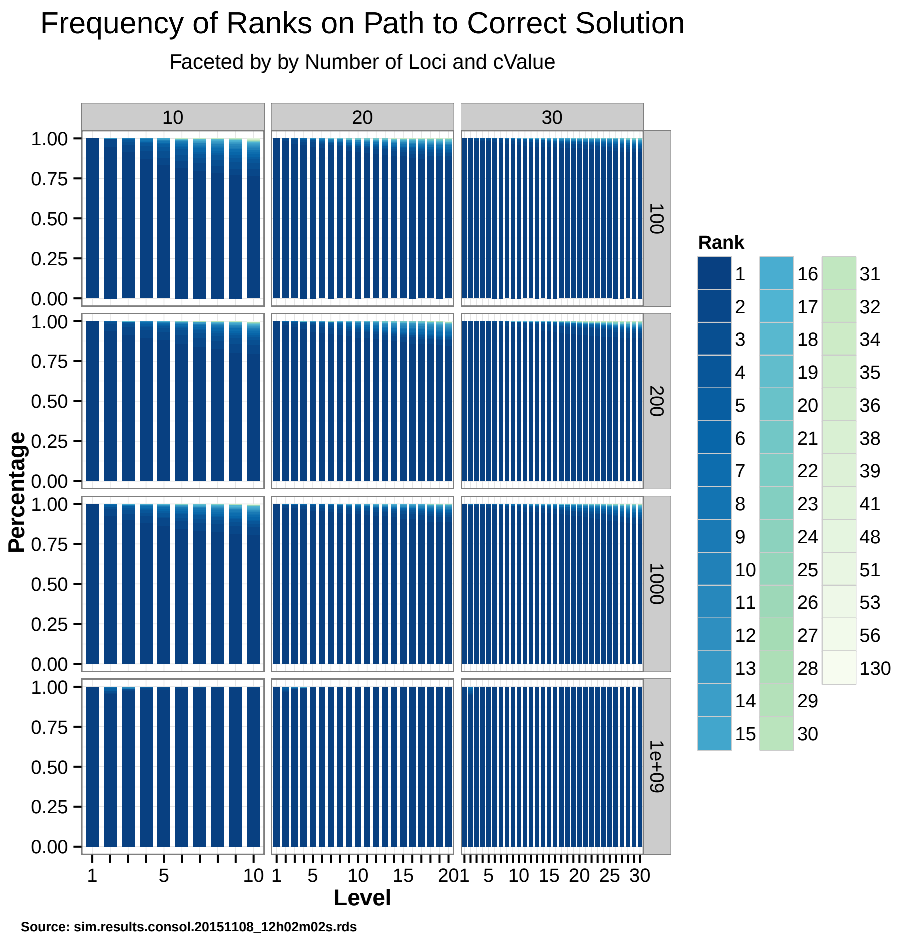


Figure 3.24: Frequency of Ranks. Each panel shows the distribution of the ranks at each level in the tree for all paths leading to a correct solution for an observation. The ranks are color coded on the right. The results are presented by number of loci (horizontally) and c (vertically).

this figure, each of the ranks represented in the data is given a separate color on the spectrum from dark blue (for rank 1) passing through turquoise and ending at light green (for high ranks). A complementary view of the same data is presented in Figure 3.25.

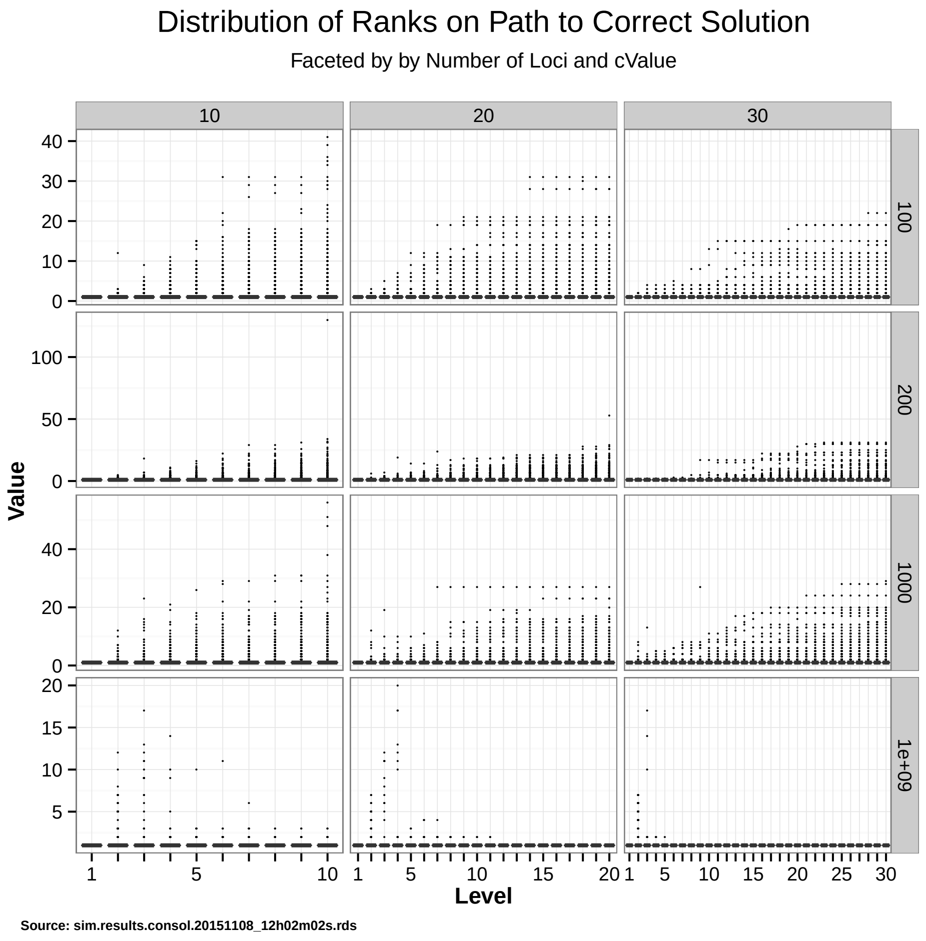


Figure 3.25: Distribution of Ranks. Each panel shows the distribution in boxplot form of the ranks observed at each level in the tree for all paths leading to a correct solution for an observation. The results are presented by number of loci (horizontally) and c (vertically).

We see from both figures that the vast majority of paths pass through low rank values exclusively, as indicated by the predominance of dark blue in each of the bars in the first figure and the concentration near 1 in the second. In cases where the path to a correct solution deviates significantly from the low ranks, evidenced by greater light green shading for such levels in the first figure and more outliers in the second, the location of this deviance appears to depend on the value for **c**. When the variance is lowest, i.e., where $c = 1e9$, the few cases where the path deviated from the lowest rank path occur near the top of the tree rather than at the bottom. For high variance (low **c**), the deviation appears to occur more frequently near the bottom of the tree.

3.2.4.4.4 Comparison to `IMH` Algorithm

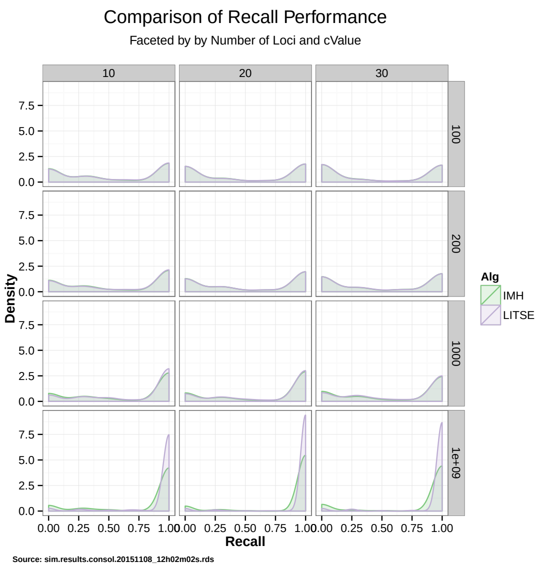
In the final section for the SSI simulation exercise, we look at `LITSE`'s relative performance.

To evaluate this, we apply the `IMH` algorithm to the same set of simulated data and compare the results. As outlined in §2.5.4 and discussed in detail in [6], the `IMH` algorithm shares the same goal as `LITSE` and features four variants. It uses the same input information, with the exception that two variants assume the existence of allele frequencies. Since all four variants showed nearly identical accuracy in simulations in [6] and allele frequencies are not always available, we choose the first `IMH` variant[8] as the comparison method.

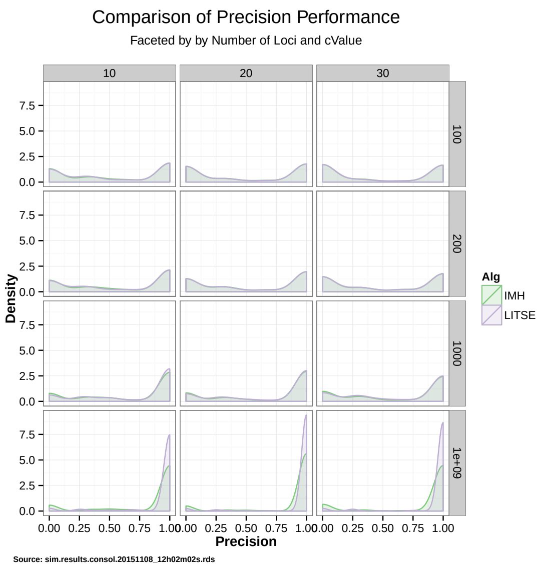
Recall that a major weakness of the `IMH` method is that it assumes the correct solution has an MOI equal to the longest locus for an observation. Therefore, it will never identify the correct solution when this is not the case.

Since the `IMH` algorithm produces only one solution per observation, we compare this solution to the top ranked solution (which is not always correct) from `LITSE` for all four levels of variance. Figure 3.26 and Table 3.11 show the results for the four measures of accuracy over various combinations of `smnc`(which is relevant only for `LITSE`) and **c** (which is relevant for both). We see that in nearly all cases (precision when `c` $= 100$ being the only exception), `LITSE` outperforms `IMH`, although the improvement can be modest. This explains why the Figure's results for the first two values of `c` appear to overlap each other. The improvement of `LITSE` is particularly noteworthy when `c` $= 1e9$. Both algorithms perform better as the variance decreases, but as evidenced in the Figure, the density for high values of recall and precision are significantly more concentrated around 1 for `LITSE` than for `IMH`. Neither algorithm appears sensitive to the number of loci in question.

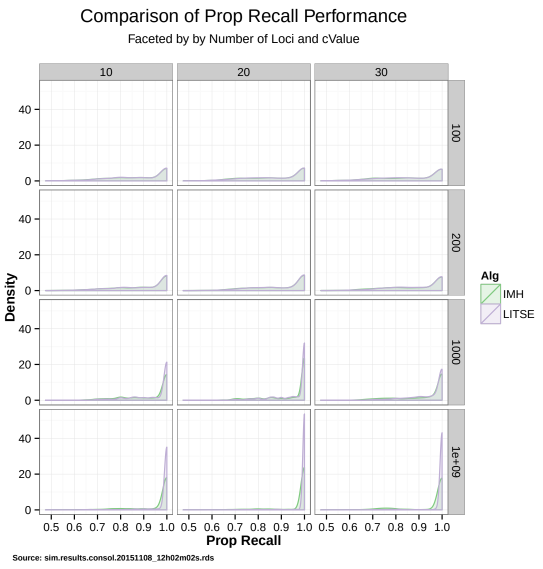
[8]Estimate haplotype proportions using only maximum locus block, no allele frequencies used.



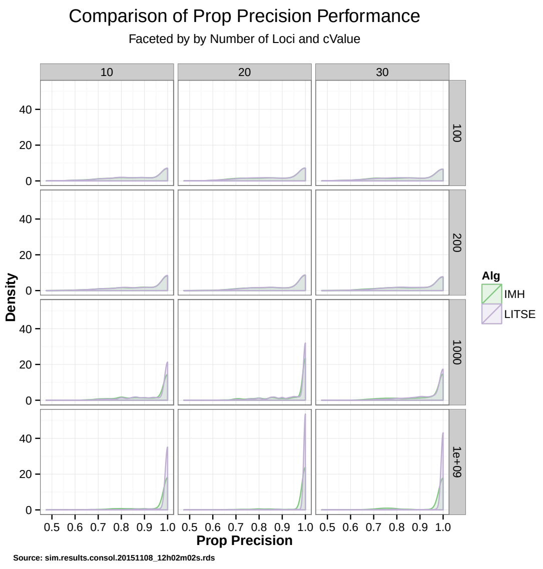
(a) Recall



(b) Precision



(c) Prop Recall



(d) Prop Precision

Figure 3.26: Performance Comparison - `LITSE` vs. `IMH`. The four subplots show the distribution of the performance of each algorithm for varying combinations of **c** and `smnc`. In all cases, only solutions known to have the correct MOI are included. For `IMH`, there is one such solution per observation; for `LITSE`, only the top-ranked solution is assessed.

		IMH		LITSE	
Measure	**c**	Mean	Median	Mean	Median
Recall	100	0.515	0.333	0.519	0.500
	200	0.563	0.667	0.569	0.667
	1000	0.696	1.000	0.719	1.000
	1e+09	0.863	1.000	0.950	1.000
Precision	100	0.520	0.500	0.519	0.500
	200	0.568	0.667	0.569	0.667
	1000	0.700	1.000	0.719	1.000
	1e+09	0.871	1.000	0.950	1.000
Prop Recall	100	0.891	0.950	0.895	0.950
	200	0.906	0.978	0.912	0.983
	1000	0.939	1.000	0.953	1.000
	1e+09	0.968	1.000	0.992	1.000
Prop Precision	100	0.893	0.950	0.894	0.950
	200	0.908	0.978	0.912	0.983
	1000	0.941	1.000	0.953	1.000
	1e+09	0.970	1.000	0.991	1.000

Table 3.11: Mean and Median Accuracy for `LITSE` and `IMH`. The table displays the mean and median performance for all four measures of accuracy across all settings for **c**.

Next, we assess the relative ability of each algorithm to detect the true solution. Figure 3.22 displayed the percentage of cases in which `LITSE` detected the correct solution and Figure 3.23 demonstrated that in such cases the correct solution is nearly always ranked first among all potential solutions. As such, we perform a comparison similar to that performed earlier in this section. Once again, we restrict the results to observations where the greatest locus length equals the true MOI; only for such cases does the `IMH` algorithm have a chance of detecting the correct solution. We then compare the frequency with which the top-ranked `LITSE` solution for the correct MOI is the correct solution versus the corresponding frequency for `IMH`. Table 3.12 displays the results. Consistent with the previous results, the performance of the two algorithms is essentially the same. As before, each algorithm performs best with lower variance and a lower number of loci.

As mentioned at the outset, the assessments in this section are biased in favor of `IMH` for two reasons:

	LITSE			IMH		
c	10	20	30	10	20	30
100	0.567	0.531	0.500	0.566	0.531	0.500
200	0.638	0.563	0.517	0.640	0.563	0.520
1000	0.767	0.746	0.681	0.768	0.750	0.687
1e+09	1.000	1.000	1.000	1.000	1.000	0.999

Table 3.12: Solution Detection for LITSE and IMH. Each figure is the percentage of observations (whose maximum locus length equals its MOI) detected by either `IMH` or the first ranked solution of `LITSE`.

1. They are restricted to cases where `IMH` assumes the correct MOI.
2. They consider only the *top* ranked solution from `LITSE`, even though it has been demonstrated that the algorithm can detect correct solutions using solutions with lower rankings.

Regarding the first condition, this will fail to be the case whenever the longest locus is strictly less than the true MOI; in other words, when there is "doubling up" across all loci. Section §3.3 will perform a more natural comparison, where both algorithms will be faced with an observation and will be tested for their ability to detect both the correct MOI and the correct solution within that MOI.

3.2.5 Conclusion

The `LITSE` algorithm's SSI procedure serves to identify which MOIs will be considered for an observation and, within such MOIs, identifies potential solutions using a greedy search. It operates on each observation individually, before feeding the individual solutions detected to the EM component of the algorithm, where they will be analyzed jointly.

For the detection of the correct MOI, we typically witness a "hockey stick" behavior in terms of the minimum error over the MOIs attempted in increasing order, with a bend at the correct MOI. The `mer` parameter is designed to detect this bend and end the attempt of higher MOIs once this bend has been achieved.

Within an MOI, the SSI procedure's accuracy and ability to detect the correct solution improve with

1. a decreasing variance/error, as modeled by the parameter $\mathbf{c}$, and/or
2. a decreasing number of loci `nl`

These features will be investigated further in the next section.

3.3 Simulation on Entire Algorithm

Having investigated the performance of the algorithm's SSI phase, we now consider the performance of the entire algorithm. This implies increasing the scope to include the SSR phase as well. Recall that in this phase, the probabilities of the individual solutions are estimated for each observation through the use of an implementation of the EM algorithm.

In section §3.2, the focus was on the estimation of the solution for each observation separate from all other observations. This reflected the nature of the SSI phase, in which each observation is treated separately when identifying potential solutions using the tree-based search. The set of potential solutions is fixed in the SSI phase. In the SSR phase, each solution is assigned a probability, which is a function of both its fit with the observed data and the estimated probability of its (1) MOI (using a model stipulating a particular distribution for this quantity among all infected individuals) and (2) component haplotypes (using a model specifying the haplotype frequency in the population of all haplotypes). These last two probabilities are computed using information from *all* observations; thus the accuracy of the estimate of the probability of a single solution is dependent on both the other potential solutions for the same observation and the potential solutions identified for all other observations in the same trial. This interdependence must be considered when structuring the simulations.

The benefits of using information across observations extend to the imputation of missing locus data. Recall from section §2.7.3 that the algorithm borrows haplotype information from other observations when imputing missing locus information for a given observation.

Recall from §2.7 that the outputs of the `LITSE` algorithm are an estimate for each of the parameters $\mathbf{c}$, $\mathbf{p}$, and $\boldsymbol{\pi}$ and a set of proposed solutions, each with a probability, for each observation. This simulation attempts to answer the following questions:

1. How well does the algorithm identify the true MOI of a solution? On what does this depend?
2. Is the probability assigned to a solution a good indicator of its (unknown) recall and precision, in both classic and proportional form? What does this depend on?
3. Similarly, is a solution's probability truly indicative of its chance of being the correct solution?

3.3.1 Configuration of Simulation

The parameters used in this simulation are the same as in previous simulation runs. However, the configuration will differ somewhat from that used for the SSI phase. Because this simulation is intended to test the functioning of the entire algorithm, it is larger than its predecessors in terms of number of parameter value combinations and thus number of trials.

The configuration is presented in Table 3.13. We deliberately vary the following parameters to test the algorithm's sensitivity to them. All of these parameters have been set in previous runs, though sometimes to values that renders them irrelevant for a simulation trial.

1. `nl`, the number of loci. The results in §3.2.4.4 suggested that the accuracy of the algorithm suffers slightly as the number of loci increases.
2. `ll`, the number of possible alleles per locus. This will be deliberately set to include low values (in particular, 3) to ensure that some cases represent complete doubling up of alleles across all loci.[9]
3. `hpn`, the number of different strains in the haplotype population. On the one hand, it might be expected that the estimate of haplotype population frequencies will be more accurate when there are few strains, particularly given a fixed number of observations. Yet at the same time, the number of different strains must be high enough to approximate a multinomial distribution.[10]

[9]In particular, the situation where the true MOI exceeds the value `ll` guarantees that all loci will double up, since the maximum locus length is bounded above by `ll`.

[10]Recall that it is assumed that the selection of MOI different haplotypes to infect an individual

Parameter	Setting
`nl`	10,20,30
`ll`	3,10
`hpn`	20,30,40
`no`	20,30,50,100
`maxmoi`	4
`c`	200,1000,1e9
`br`	0,0.01,0.05
`mfl`	5
`smnc`	20
`hmnc`	50
`mer`	0.2
`am`	2
`mi`	30
`qvd`	0.005
`inst`	1

Table 3.13: Simulation Settings: Overall. Each parameter in the "Parameter" column was tested for each value in in the corresponding entry in the "Settings" column. Each combination of the parameter settings is tested `inst` times.

4. `no`, the number of observations in a trial. Because the algorithm shares information across observations, we would expect its accuracy to improve as the number of observations increases.
5. `c`, the concentration parameter. This parameter controls the degree of variance, or error, around true allele proportions; high (low) values imply low (high) variance. It was demonstrated in the previous section that the accuracy of the algorithm improves as the value of `c` increases.
6. `br`, the rate at which loci information is blank. On a 0 to 1 scale, this equates to the probability that a given locus for a given observation has missing allele information. Previously, this parameter was set to 0, thus ensuring that all observations had complete information. Here, we allow it to vary from 0 to 0.1. We expect that the

is governed by the multinomial distribution. Yet the multinomial distribution assumes sampling with replacement, an impossibility in this case given that each haplotype can be chosen. Thus the true distribution is hypergeometric, which converges to the multinomial as (in this implementation) the number of haplotypes passes to infinity.

accuracy of the algorithm will suffer as the value of this parameter rises.

Previously, these parameters were largely ignored, with the exception of the expected finding that the number of observations influenced linearly the runtime of the algorithm.

The following parameters will be set to single values, based in part on results witnessed in previous simulation runs.

1. `maxmoi`, the maximum true MOI allowed for an observation. This will be set to 4, as in previous simulation runs, for all trials in this run. Note that this is the *maximum* MOI possible. In each trial, a distribution of MOIs from 1 through 4 will be created and the algorithm will estimate this distribution.
2. `mfl`, the maximum float level. To decrease runtimes, this value will be set to 5 for all trials. It was demonstrated earlier that this haplotype proportions vary little after level 5 in the tree, and ending the NNLS routine underlying this estimation will increase the speed of the algorithm.
3. `smnc`, the soft maximum number of nodes chosen. This will be set to 20. Previous results suggested that there are minimal benefits from increasing this value, especially for low variance trials.
4. `hmnc`, the hard maximum number of nodes chosen. This will be set to 50, as before.
5. `mer`, the maximum error ratio used to detect the correct MOI. As discussed in section §3.2.4.2, this will be set to 0.2 for all trials.
6. `am`, the number of additional MOIs considered above the minimum. This will be set to 2, as before.
7. `mi`, the number of iterations in the SSR phase. This now becomes relevant and will be set to 30, rather than zero as was the case in the previous simulation where it was intended to avoid any SSR iterations.
8. `inst`, the number of instances per parameter combination. This will be set to 5, an increase from previous runs.
9. `qvd`, the minimum change in the value of the $Q(\Theta^{(t)}|\Theta^{(t-1)})$ function, as stated in equation (2.33), from one iteration to the next that will permit the EM component to perform a new iteration (up to `mi`). Thus `qvd` and `mi` jointly form the stopping

criteria for the EM component.[11] This value will be set to 0.005, a suitably small value based on an informal observation of the path of the value of $Q(\Theta^{(t)}|\Theta^{(t-1)})$ during exploratory runs.

As before, the configuration implies a certain number of trials, in this case,

$$3 \times 2 \times 3 \times 4 \times 1 \times 3 \times 3 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 = 648$$

trials, each having from 20, 30, 50 or 100 observations. Table 3.14 presents a summary of the simulation run.

Quantity	Value
Number of Observations	12780
Number of Trees	16324
Number of Trees per Observation	1.28
Number of Solutions	720587
Average Tree Size (Number of Nodes)	2406

Table 3.14: Simulation Stats: Overall Accuracy. The number of trees is below $12780 \cdot 3 = 38340$ because some trees were deemed invalid and others were dropped because they failed the `mer` check; this occurred when all solutions for the tree were deemed invalid when an MOI higher than the true MOI was attempted. See §2.7.2.5 for details.

3.3.2 Procedure of Simulation

The procedure followed in the simulation is similar to that for the SSI phase. Each trial represents the application of the algorithm to a set of data from a particular environment and with a particular group of settings.

Given a value for `hpn`, the simulation routine randomly creates this number of distinct haplotypes to form a population and draws from this population when assigning haplotypes to an individual. Each haplotype is randomly given a population frequency such that the frequencies across the `hpn` haplotypes sum to 1.

As stated above, in a trial the parameter `maxmoi` determines the maximum true MOI that an observation in that trial can take. Given `maxmoi`, the simulation trial randomly

[11] $Q(\Theta^{(t)}|\Theta^{(t-1)})$ serves as a proxy for the likelihood of the overall data given the parameter estimates.

generates a true $\mathbf{p}$ vector for the distribution of MOIs such that $p_1 = 0$ and $\sum_i p_i = 1$. When generating observations in a trial, the MOI for each individual is randomly assigned using the $\mathbf{p}$ vector. This approach is taken because one of the tasks of the simulation is to determine the extent to which LITSE can correctly estimate $\mathbf{p}$. In addition, for trials where $\texttt{br} > 0$, randomly chosen loci will be dropped from an observation's set of observed allele proportions according to the parameter's setting; for example, if $\texttt{br} = 0.05$, then each locus has 1/20 chance of being dropped.[12]

Once the observations are generated for a trial, the procedure is at first identical to that followed for the SSI simulation. Namely, each observation is subject to a series of tree searches, one per accepted MOI, to generate possible solutions. Once determined, the solutions remain fixed for the remainder of the algorithm. At this point, the SSR phase begins, during which the probability for each solution and estimates for the parameters $\mathbf{c}$, π, and $\mathbf{p}$ are initially computed and then refined in each iteration. In previous runs, these parameters were used to generate data only; here, they are used to generate data and then the data-induced estimates computed by `LITSE` are then checked back to the true values to assess the estimates' accuracy.

3.3.3 Illustration of a Single Simulation Trial

Before presenting the results of the simulation, it is useful to illustrate the details of a single run. The concepts in this illustration will then be used in discussion the general results over all trials later in the section.

For the illustration, consider the trial with the configuration in Table 3.15. This trial

$\texttt{nl} = 10$	$\texttt{ll} = 10$	$\texttt{hpn} = 20$	$\texttt{no} = 20$
$\texttt{maxmoi} = 4$	$\texttt{c} = 1e9$	$\texttt{br} = 0$	$\texttt{mfl} = 5$
$\texttt{smnc} = 20$	$\texttt{hmnc} = 50$	$\texttt{mer} = 0.2$	$\texttt{am} = 2$
$\texttt{mi} = 30$	$\texttt{qvd} = 0.005$		

Table 3.15: Configuration of Illustration Trial.

[12]The simulation ensures that it is never the case that the same locus is dropped for all observations in the run.

was chosen because the characteristics of its components (e.g., `hpn`) are limited enough that it will be reasonably straightforward to visualize what the algorithm is doing.

For this illustrative trial, first consider the evolution of the proxy for the likelihood function, the function $Q(\Theta^{(t)}|\Theta^{(t-1)})$. This is the primary criterion by which the algorithm judges whether additional iterations are improving on the estimation exercise. It is expected that this value will monotonically increase with the iteration number, with the rate of increase declining such that the increase is imperceptible after some point. The evolution for the trial of interest is presented in Figure 3.27. The result is as expected: a

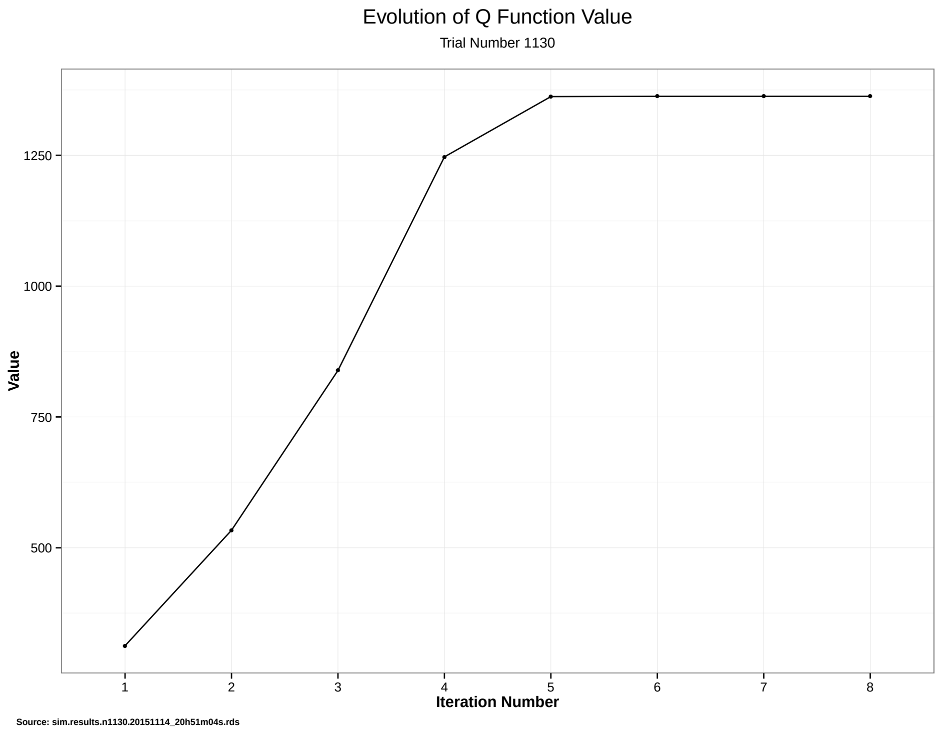


Figure 3.27: Evolution of Q Function in Illustration Trial.

curve that steeply rises initially and then flattens as the iteration number increases. In this case as for all trials, the setting `mi` $= 30$ was in force; yet the procedure stopped at $i = 8$, indicating that the stopping criterion `qvd` $= 0.005$ terminated the iterations.

Next, consider the estimates of the parameters $\mathbf{p}$, $\boldsymbol{\pi}$, and $\mathbf{c}$, which will evolve over the iterations. Regarding $\mathbf{p}$, the parameter values of the true MOI are presented in Table 3.16. Since `maxmoi` $= 4$, the maximum true MOI across all observations was 4. As for

MOI	p
1	0.41
2	0.09
3	0.47
4	0.03

Table 3.16: Illustration Trial - True MOI Parameter p

all trials, the setting of the **p** parameter vector was done randomly. In this particular trial, MOIs of 1 and 3 have the highest parameter setting; in other trials, this will be different.

The evolution of the estimation of **p** is presented in Figure 3.28. The algorithm will

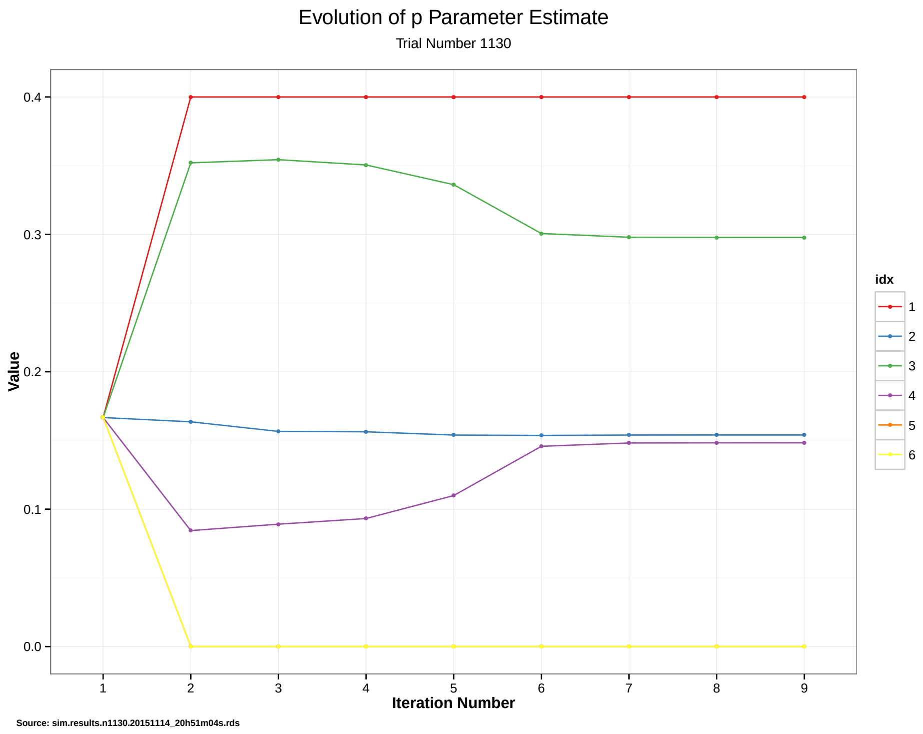


Figure 3.28: Evolution of **p** Estimate in Illustration Trial. The estimates for all feasible MOIs begin the same and then diverge to different values after the first round.

initially assign an equal frequency to each feasible MOI of the entire run. Since there was an observation with a maximum locus length of 4 and `am` $= 2$, this means that each of the MOIs from 1 through 6 will get an initial estimate of $1/6$. After the first iteration, when the potential solutions for each observation are considered in re-estimating $\mathbf{p}$, the estimated frequency for $\mathbf{p}_i, i = 5, 6$ are set to zero while the others remain strictly positive. In this case, the estimates are quite accurate although estimates for 2 and 4 are slightly inflated at the expense of 3.

Regarding $\boldsymbol{\pi}$, the true frequencies for the `hpn` $= 20$ haplotypes generated are presented in Table 3.17. The ID for each haplotype, when present in the table, is its ID among

ID	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	Freq
	1	9	1	4	4	5	6	8	8	5	0.006
	1	10	3	1	6	9	9	10	9	2	0.030
50	2	2	2	7	8	5	5	2	6	5	0.108
97	2	2	5	4	4	6	4	2	8	2	0.294
265	2	6	6	8	9	6	5	2	1	5	0.044
345	2	9	8	8	7	2	4	8	8	7	0.027
359	2	9	10	7	4	2	1	2	5	7	0.027
415	3	2	4	4	4	8	5	5	3	8	0.037
	3	2	4	9	4	2	4	4	7	7	0.010
431	3	6	5	9	2	8	9	2	3	9	0.030
453	3	8	1	8	7	5	1	1	10	5	0.026
522	3	8	8	8	4	8	8	10	1	7	0.076
	3	10	10	3	6	1	1	8	3	2	0.046
	4	6	8	9	3	8	4	7	9	1	0.025
542	4	6	10	4	8	8	2	8	5	2	0.030
570	7	7	10	7	6	6	5	3	10	2	0.025
573	7	8	5	3	2	1	2	5	10	7	0.005
591	7	10	5	3	6	8	5	5	8	1	0.065
630	10	4	9	1	6	1	8	6	5	2	0.046
648	10	10	2	7	4	6	2	6	8	5	0.046

Table 3.17: Illustration Trial - True Haplotype Frequencies The ID column refers to unique ID among all haplotypes considered. Haplotypes without an ID were not considered in the trial.

a total of 648 distinct haplotypes identified as components of the potential solutions across all 20 observations. Those haplotypes without an ID the table were not proposed by the algorithm as part of any solution. While this is initially concerning, further inspection reveals that only fifteen of the twenty haplotypes were actually used in one or more of the twenty observations in the trial with the remaining five omitted. It is precisely these five that lack an ID.[13] Of the 648 suggested haplotypes, each began with an estimated population frequency of $1/648 = 0.00154$. One of the tasks of the SSR phase is to correctly estimate the population frequencies of the haplotypes during its iterations. Figure 3.29 presents the evolution of this estimation across all haplotypes. Given the

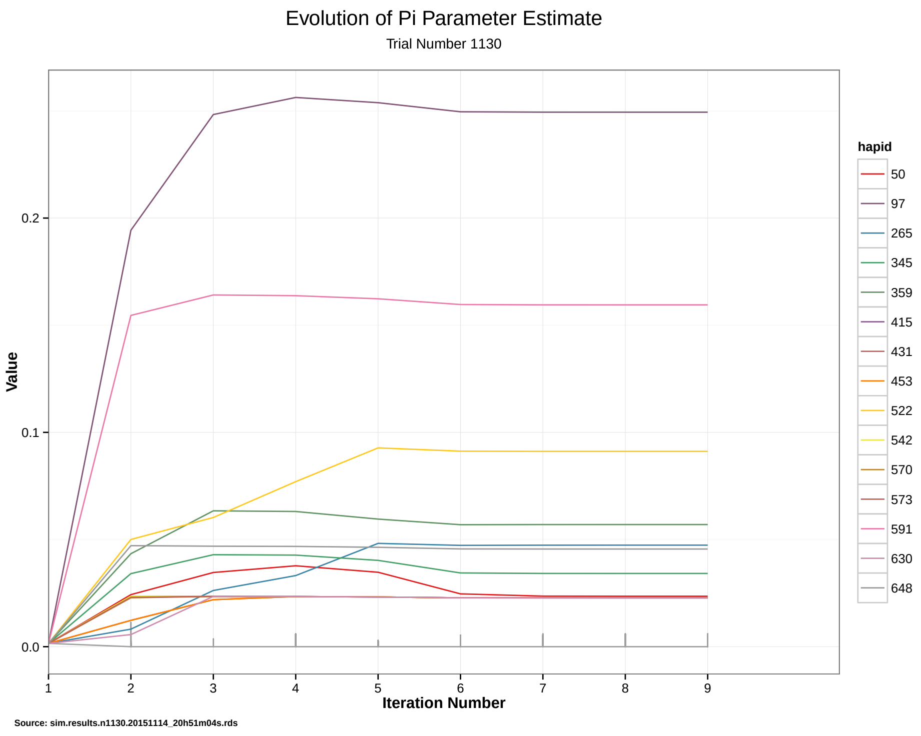


Figure 3.29: Evolution of π Estimate in Illustration Trial. The evolution of all haplotypes is plotted. The fifteen true haplotypes in color converge to non-zero values. The non-true haplotypes in gray converge quickly to zero, as evidenced by overlapping lines running to zero by the second iteration.

high number of haplotypes, the figure color codes only the fifteen true haplotypes while leaving the other 939 in gray. At first, all haplotypes begin with the same estimated value (the aforementioned 0.00154). Thereafter, all fifteen true haplotypes (with lines overlapping in some cases) rise above this initial value at iteration number 2 with only the true haplotypes having a non-negligible estimated proportion by iteration 6. Compared

[13]We note that the five haplotypes are among those with the lowest frequencies in the population. In addition, there were only twenty observations in the entire trial; a trial with more observations would have a greater chance of including a broader range of true haplotypes.

to their true frequencies, the haplotypes appear in approximately the right order, though there appears to be a bias towards the middle: high frequency haplotypes such as #97 are underestimated while low frequency haplotypes such as #573 are overestimated. This may be a reflection of the relatively small sample size and the use of the multinomial distribution for what is a sampling without replacement.[14]

The final parameter to estimate is the vector `c`, with one entry per locus. Recall that in this trial $c^{(l)} = 1e9$ for all loci $l = 1, \ldots, \texttt{nl}$. Each iteration of the EM component reestimates the value of each c_i by finding the zero of the first derivative documented in equation (2.60); there is no initial estimate. The results of the estimation are presented in Figure 3.30. The pattern is similar for all loci: the estimate remains low for the first three

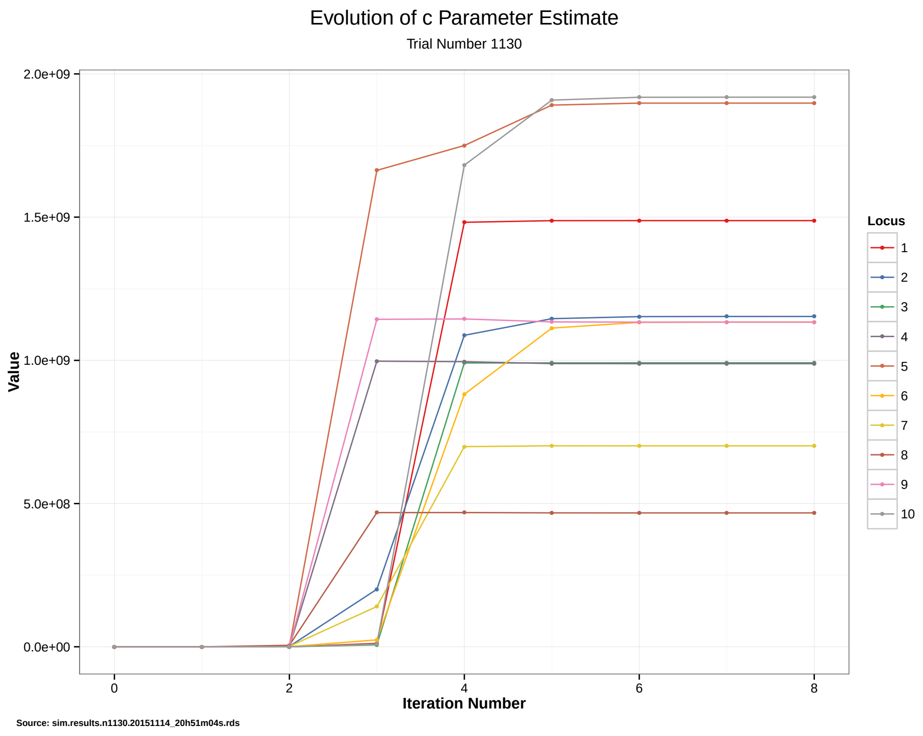


Figure 3.30: Evolution of **c** Estimate in Illustration Trial.

or four iterations before jumping to close to the true value. Thereafter, the estimates appear to change little. The mean of the final estimates appears close to the true value

[14]See §2.6.1.2.

of 1e9, with the great majority of the estimates falling within 75% to 150% of the true value.

Regarding the actual solutions proposed for the observations in the trial, it is useful to first examine the position of the true solution among all solutions for the same observation in terms of conditional probability, which was formally defined in (2.43). This corresponds to the value in equation (2.42) for any solution. The expectation is that the correct solution will appear near the top among all solutions for the observation and present with a value close to 1. This probability is a function of several components – the likelihood for the true MOI, the likelihood for the true haplotypes, and the likelihood for the fit with the observed data – and all three will need to be competitive among all solutions for the true solution to rank highly. The results for the trial run in question are presented in Figure 3.31. Each of the 20 observations had the correct solution among

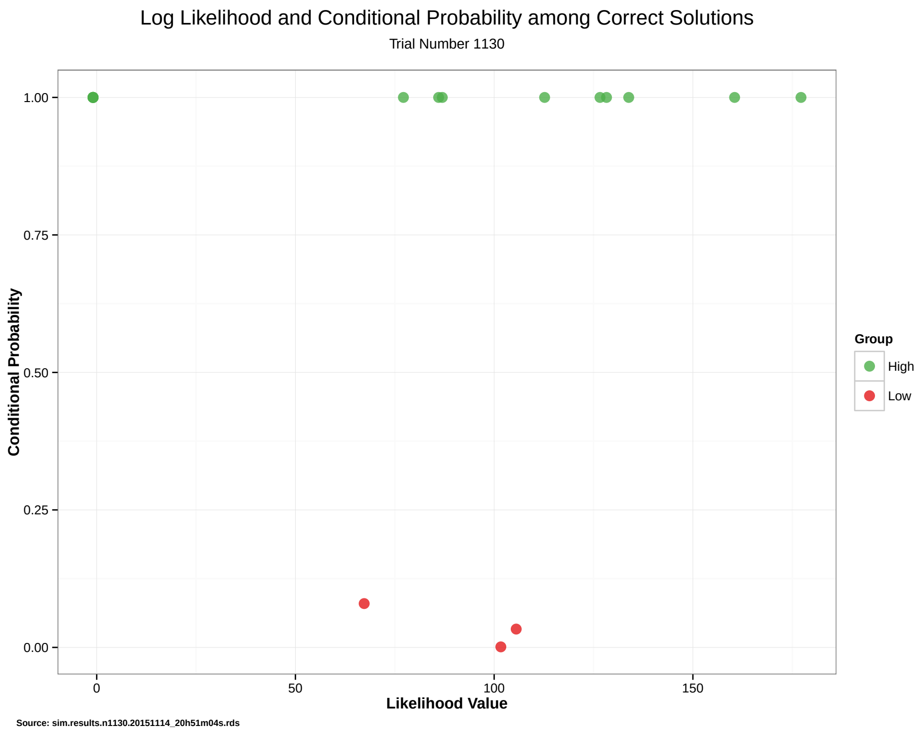


Figure 3.31: Distribution of Conditional Probability for Correct Solution.

all solutions proposed, but we see that in the three cases the probability assigned to the correct solution was significantly less than one. In all three cases, the correct solution scored highly in terms of likelihood as captured in the numerator of equation (2.43); yet at the same time the algorithm accepted the next higher MOI (3 in one case and 4 in two cases) because the ratio of minimum errors was in the range 0.096 to 0.157, below `mer` = 0.2. There were additional solutions at these MOIs that fit the data equally well, resulting in a relatively high value for the denominator of (2.43) for the observation. Loosely speaking, the relative probability for the correct solution was “drowned out” by other good solutions (which ideally would have been excluded through the use of the `mer` parameter).

3.3.4 Runtime

Section §3.2.2 analyzed the sensitivity of the runtime of the solution identification component of LITSE to various factors including number of loci, number of observations, etc.

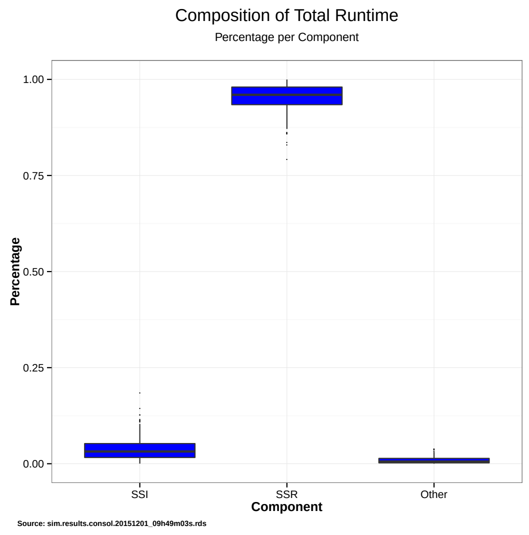
This section compares the relative component runtimes of the entire LITSE algorithm and investigates the sensitivity of the overall algorithm's runtime to the number of observations and the number of EM iterations.

Figure 3.32 presents a number of figures depicting the nature of runtime of the algorithm over various parameters.

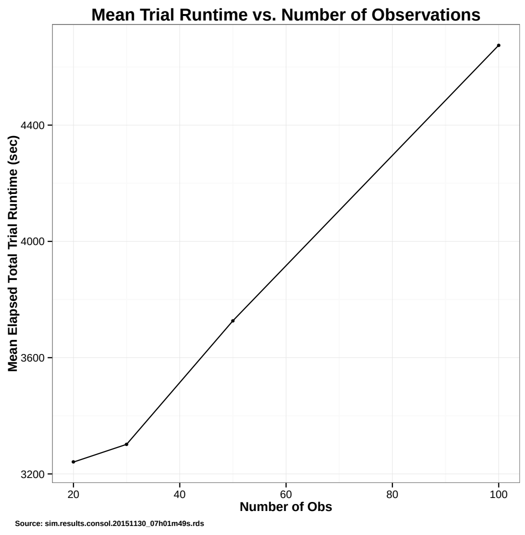
3.3.4.1 Runtime Composition

It is first interesting to consider the proportion of total runtime each major component of the algorithm represents. There are two major components of the algorithm: the SSI routine and the SSR routine (which includes the EM procedure). In addition, the SSC routine is a third routine relevant only when the algorithm detects missing information.

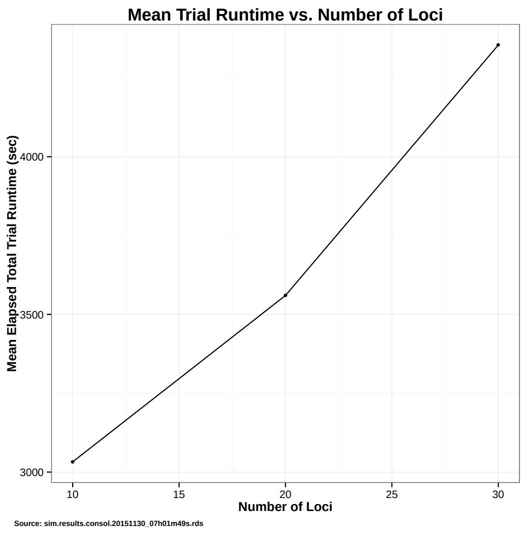
We see in Figure 3.32(a) that the SSR routine represents by far the greatest proportion of runtime. A further analysis reveals that this is due to the M step in which the parameters are reestimated. In particular, it is the reestimation of the c parameter, which is done iteratively due to the lack of a closed form solution, that accounts for the majority of this time.



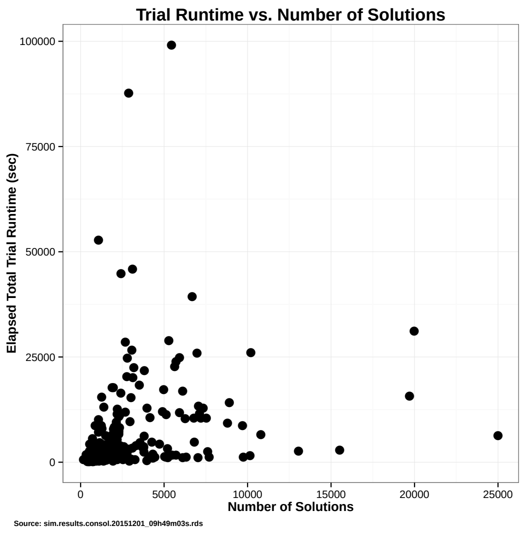
(a) Composition of Total Algorithm Runtime



(b) Number of Observations



(c) Number of Loci



(d) Number of Solutions

Figure 3.32: Runtime Sensitivity to Factors. All figures are presented at the level of a simulation trial.

3.3.4.2 Sensitivity to Runtime Environment

Next, we look at several key environmental factors and confirm their influence on the runtime of the entire algorithm. The second through fourth panels of Figure 3.32 present the results for all three factors investigated.

3.3.4.2.1 Sensitivity to Number of Observations

As before, we would expect the runtime to increase linearly with the number of observations in the trial. Figure 3.32(b) demonstrates that this is indeed the case.

3.3.4.2.2 Sensitivity to Number of Loci

Similarly, we would expect the runtime to increase linearly with the number of loci in the trial. This is for two reasons. In the SSI phase, the number of loci will determine the depth of the tree; in the SSR phase, the number of loci represents the number of calculations the M step must perform each round to reestimate the c parameters. Figure 3.32(c) demonstrates that this relationship indeed holds.

3.3.4.2.3 Sensitivity to Number of Solutions

The relationship here is a bit more complicated. We would expect a positive relationship between the two, since a high number of solutions across all observations in a trial will suggest a long runtime for two reasons: larger trees in the SSI phase and a greater number of solutions over which to maximize the parameter c in the SSR phase. Figure 3.32(d) suggests that there is only a loose positive relationship between these two variables; this appears particularly true for higher levels along the x axis, where the data points are sparser.

3.3.5 Accuracy

In this section, we look at the accuracy of the solutions produced by the `LITSE` algorithm. In particular, we now have probabilities associated with solutions, which was not the case when we looked at accuracy in §3.2.4.

We also look at the accuracy of the underlying parameters to the model – $\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}$ – which are estimated iteratively in the SSR phase.

3.3.5.1 Estimation of Solutions

Section §3.2.4 established the relationship between the *rank* of a solution and its accuracy (as measured by recall, precision, and their proportional versions). We now look at the relationship between the *likelihood* of the solution and its accuracy.

There are two additional differences in the present assessment versus §3.2.4. First, we are now assessing solutions across several MOIs, whereas previously solutions were assessed only within the same MOI. Second, we have introduced the possibility of missing data, which is modeled using the parameter `br`. The assessment will cover both aspects.

We first look at the distribution of the computed likelihoods for all solutions, as depicted in Figure 3.33 and Table 3.18. We see that the vast majority of solutions have

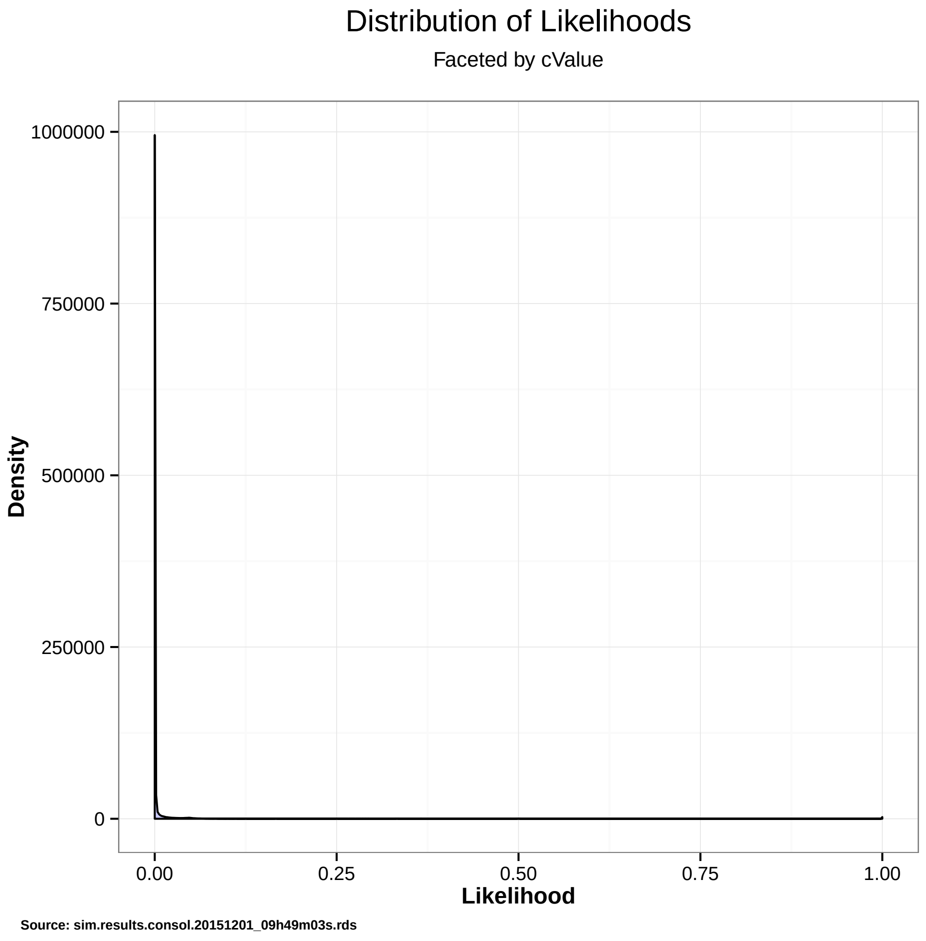


Figure 3.33: Distribution of All Likelihoods.

	Quantile						
	0.2	0.5	0.75	0.9	0.95	0.99	1
Value	3.97E-229	3.32E-13	1.03E-05	5.67E-04	7.46E-03	6.30E-02	1.00E+00

Table 3.18: Quantiles of Likelihoods. For the series of values in the header row, the "Value" row shows the relevant quantile.

a very low likelihood, less than 0.01. In the far right of Figure 3.33 there appears to be a slight uptick, indicating a small group of solutions with very high likelihood.

3.3.5.1.1 Inclusion of Correct Solution

The first question we ask is similar to that first posed in §3.2.4.4.2; namely, to what extent does the `LITSE` algorithm include the correct solution among all solutions? We have already answered this question in section §3.2.4.4.2 for the case where there are no missing data across observations. Here, we extend this analysis to cases where some degree of observed locus proportions is missing. We define this "blank percentage" for an observation as the percentage of the loci for which such information is missing. Thus if we are considering 30 loci and an observation has all locus information, its blank percentage is 0; if it is missing information for 2 loci, its percentage is $2/30 = 0.0667$.

Figure 3.34 shows the results, faceted by variance. This figure presents the percentage of all observations with a particular blank percentage that had a correct solution among all solutions identified. As expected, as the blank percentage increases, the average discovery rate decreases, though there are some kinks in the curve. Once the blank percentage reaches the value of approximately 0.2, for example 6 loci out of 30 with missing data, the correct solution is no longer found. Somewhat surprisingly, the results appear relatively constant across all values of `c`; one might have expected the performance to have been noticeably better for trials with low variance.

3.3.5.1.2 Relationship between Likelihood and Accuracy Measures

Finally, we consider the relationship between likelihood and the four accuracy measures introduced in §3.2.4.3. Figure 3.35 shows results for all four measures of accuracy. The approach to constructing the plots was as follows. A sample of 100,000 solutions was

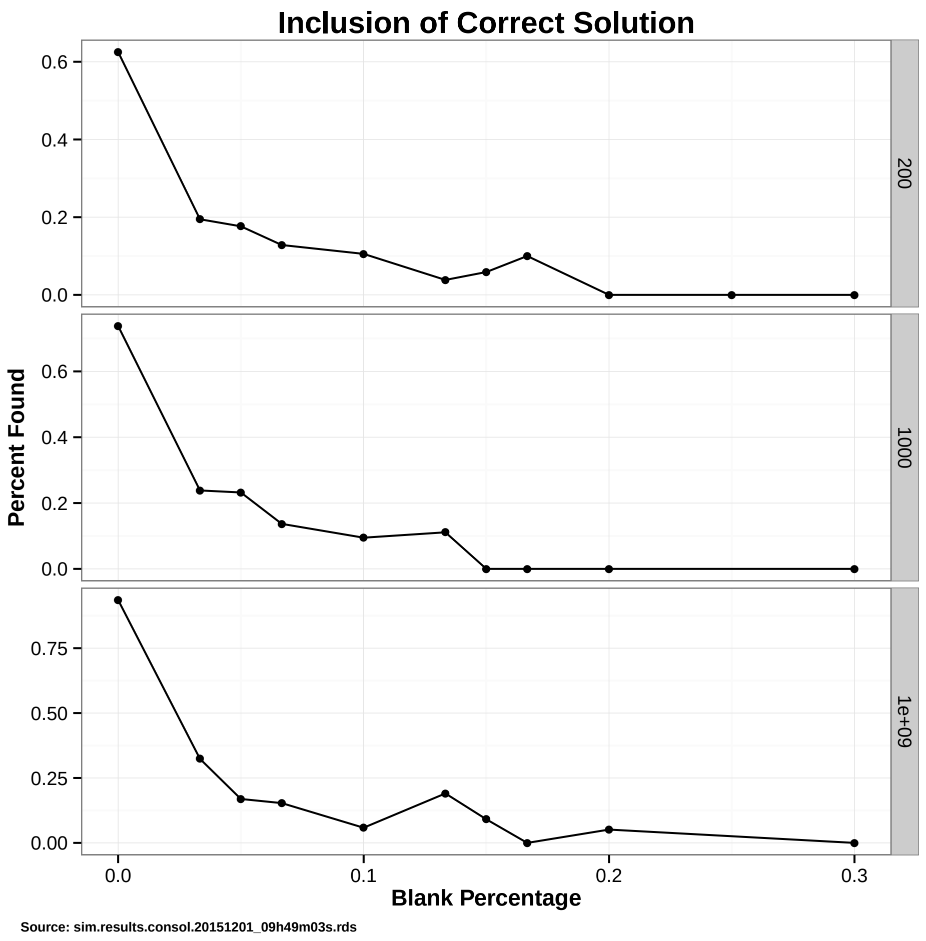
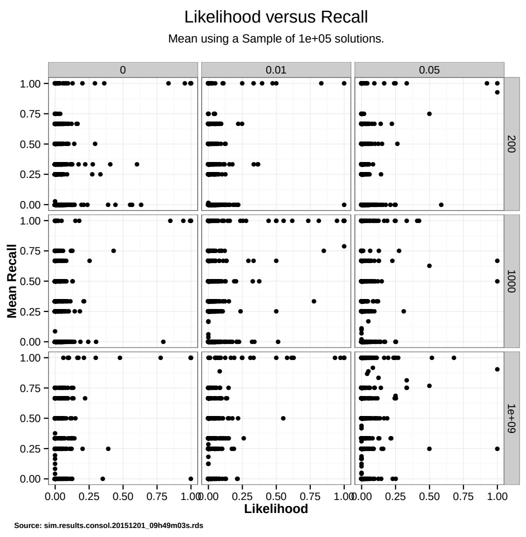


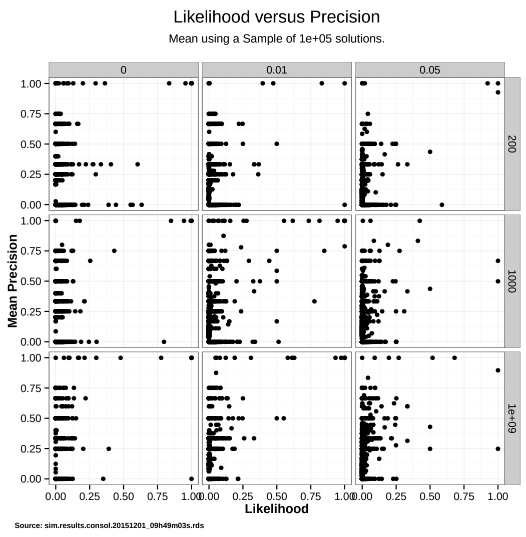
Figure 3.34: Inclusion of Correct Solution.

drawn from the total pool of all solutions (over all observations in all simulation trials) as documented in Table 3.14. The sample is meant to be representative of all solutions but small enough to feasibly be plotted using the `ggplot` library in R. The mean of each of the four accuracy measurements is then computed for every observed likelihood in the sample. Each mean is then plotted in the relevant subfigure. Within any of the four subfigures, there are nine panels, each including the points for a particular combination of `br` (a measure of the "gappiness" in the data, with higher values indicating more frequent cases of missing locus data) along the horizontal dimension and `c` (a measure of variance) along the vertical dimension.

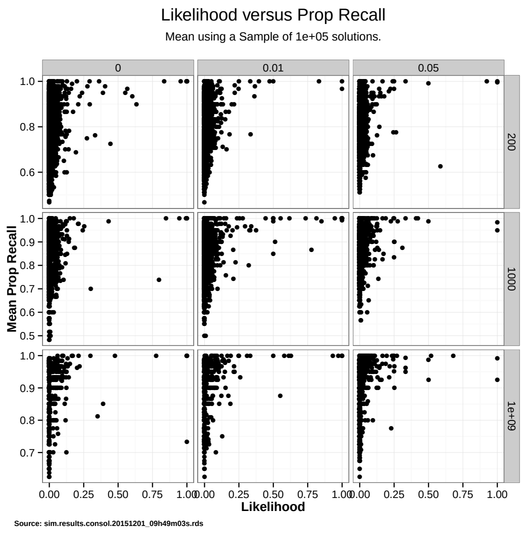
In each panel in each subfigure, a natural pattern to be expected is an upward sloping



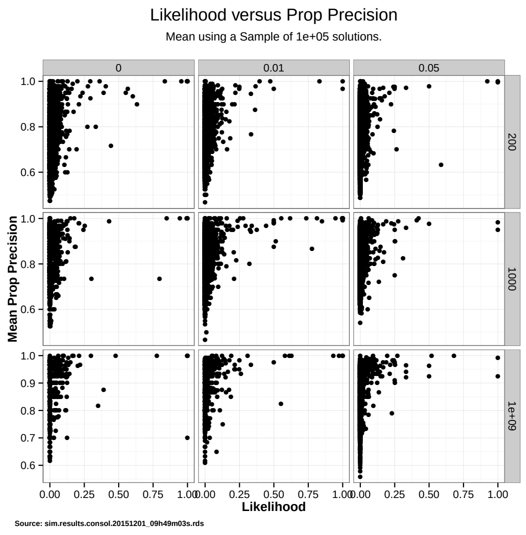
(a) Recall



(b) Precision



(c) Proportional Recall



(d) Proportional Precision

Figure 3.35: Likelihood and and Accuracy Measures. Each of the four subfigures focuses on one of the measures of accuracy. All four are derived from the same random sample of size 100,000 from the total set of solutions (as documented in Table 3.14). A sample was taken because of the impracticality of plotting all solutions in a figure.

set of points from the bottom left corner to the top right corner. Such a pattern would indicate that on average low (high) likelihood points score poorly (well) in terms of the relevant accuracy measure. The question is to what extent we see such a pattern depending on the measure, gappiness, and variance in question. In subfigures 3.35(a) and 3.35(b), corresponding to recall and precision, this pattern is somewhat more evident when the c is high (variance is low) but with no obvious trend when moving from low gappiness to higher gappiness. Overall, the results in these first two subfigures do not suggest a strong relationship between likelihood and the measure in question.

The situation with the third and fourth subfigures, corresponding to the proportional versions of these measures, is much more encouraging. Here, we see the desired pattern in all of the panels. The strong concentration of points near the left axis of each panel is simply a manifestation of the heavy concentration of low likelihood solutions as documented in Figure 3.33. This effect does not appear to depdend on either the gappiness or variance.

Overall, we conclude that there is evidence that the likelihood of a solution is a good indicator of its proportional recall and precision.

3.3.5.2 Estimation of Parameters

This section analyzes the performance of the `LITSE` algorithm in terms of the estimation of the parameters – $\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}$ – that govern the distributions of the component variables in the model. These parameters were introduced sequentially in §2.6 and summarized in §2.6.1.7.

In any trial run, each of the parameters will be estimated iteratively as outlined in §2.7.4.2. The main concern here is the extent to which the estimations converge to the true underlying parameter values and what this depends on.

3.3.5.2.1 Number of Iterations

By way of background, it is interesting to view the distribution of the number of iterations the SSR routine performs. Recall that a hard limit of 30 has been imposed on the number of iterations, with the procedure finishing earlier if the difference in the value of the Q function falls below a threshold.

Figure 3.36 shows the distribution of the number of iterations performed across the trials in the simulation run. We see that the majority of the trials end in five to eight

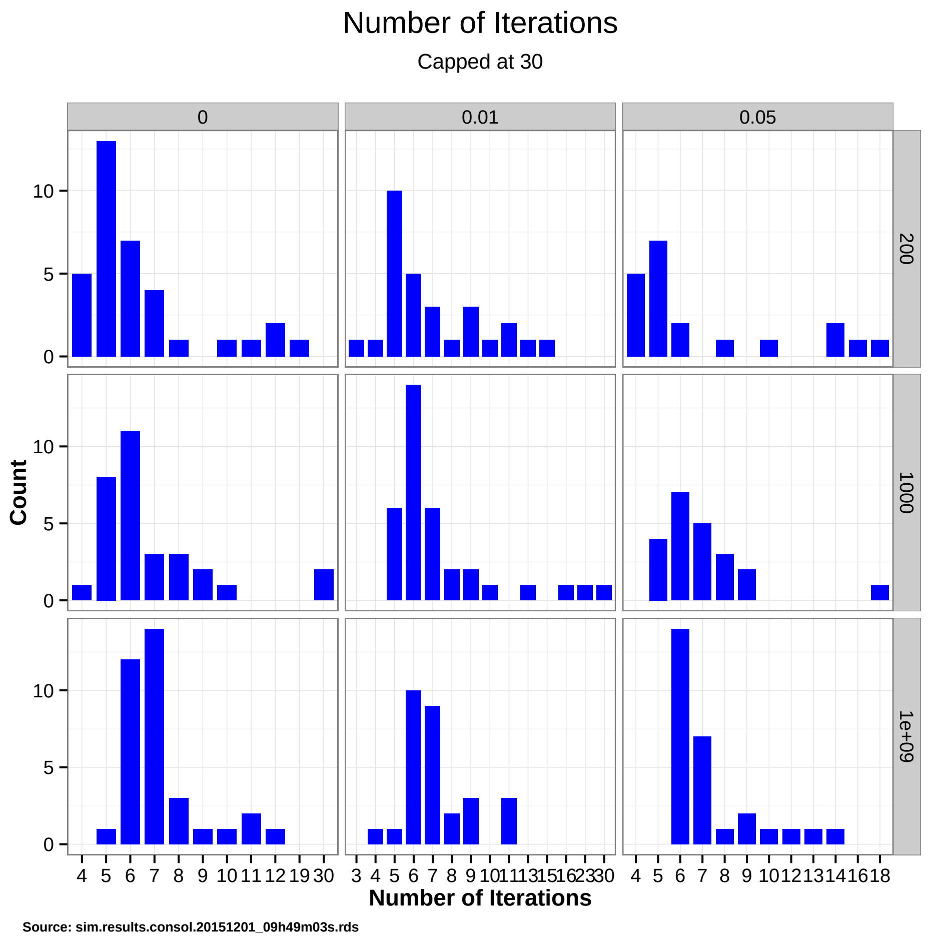


Figure 3.36: Number of Iterations. The distribution of the number of iterations in the SSR (EM) procedure.

iterations, with a very few taking the full 30. There is no noticeable difference by blank rate `br` or variance `c`.

3.3.5.2.2 Measure of Similarity

All three parameters, in either true or estimated form, are vectors. As such, it is convenient to adopt a scalar-valued metric to assess the degree to which an estimated vector and true vector agree. We propose using the L2 norm of the difference between the true and estimated versions of the same vector,

$$D(\rho_t, \rho_e) = \|\rho_t - \rho_e\|_2 \tag{3.5}$$

where ρ_t is the true vector of interest and ρ_e is an estimate.

Using such a metric, we compute the value in (3.5) at each iteration for each trial and present the paths below.

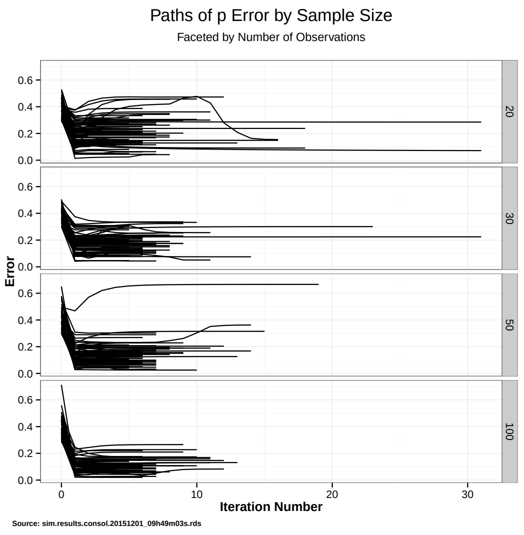
An alternative measure of similarity, the Renkonen measure, is presented in §3.D. This method is a common choice to measure the similarity in proportion vectors.

Note that among the parameters of interest, $\mathbf{p}$ and $\boldsymbol{\pi}$ are proportion vectors while the c vector is not.

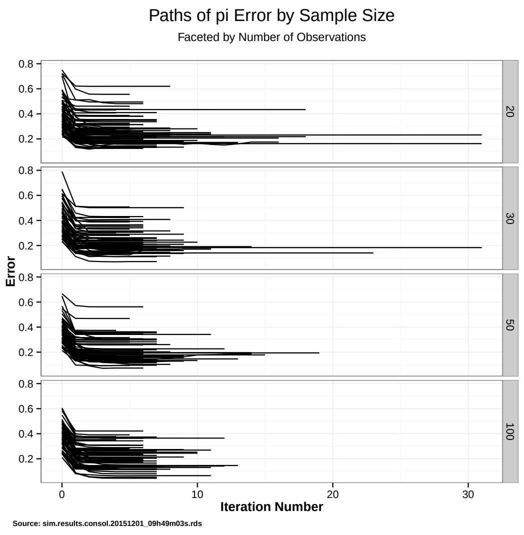
3.3.5.2.3 Performance of Algorithm

Using the metric defined in equation (3.5), we now consider the performance for each of the three parameter estimates. Figure 3.37 presents the results in three subfigures. In each case, the results are faceted by the total number of observations in the trial. We are interested in determining to what extent this factor influences the typical path of the error for a parameter. In each case, the iteration number is on the horizontal axis and the error, as measured in equation (3.5), is on the vertical axis. The vertical scale has been fixed in each subfigure to facilitate the comparison between the different facets.

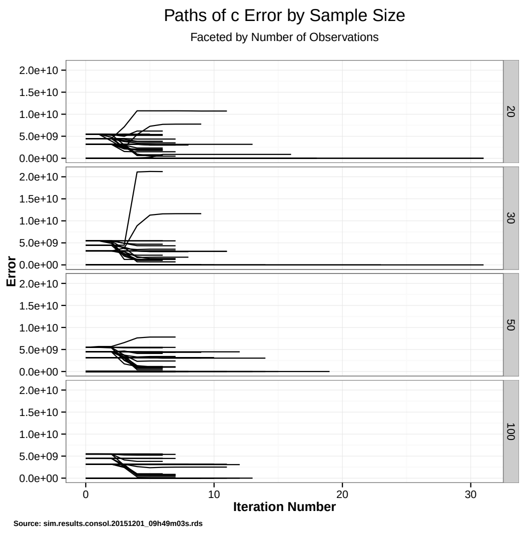
In viewing the subfigures, it is apparent in three ways that the algorithm benefits from larger sample sizes. First, we see that the number of iterations needed increases as the number of observations decreases; this is apparent when we look at the length of the horizontal lines from facet to facet. Second, as indicated by the vertical height of the lines, the overall level of error is lower for larger sample sizes; the lower number of iterations does not come at the expense of higher errors. Third, the pattern of error evolution over the iterations increases in stability as the number of observations increases. In several isolated cases, in particular for the parameter $\mathbf{p}$, we see a wandering path before convergence. In other cases, particularly for c, we see the error increase from its initial level before stabilizing. We see no such erratic behavior for larger samples. These results are intuitive: the observations serve as samples from a distribution, and we would expect the performance of our estimator to improve in line with the sample size.



(a) Parameter $\mathbf{p}$



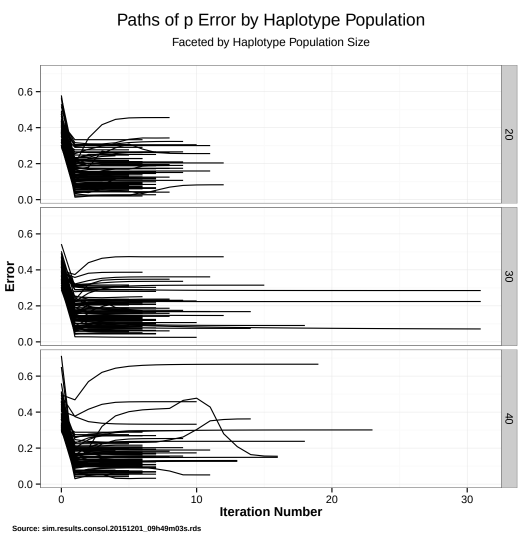
(b) Parameter $\boldsymbol{\pi}$



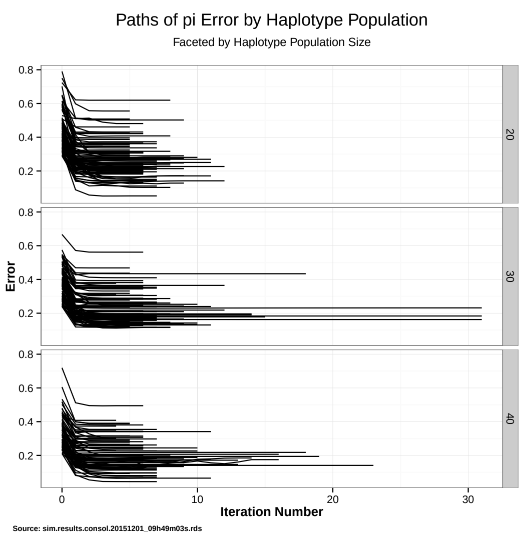
(c) Parameter c

Figure 3.37: Path of Estimate Errors by Number of Observations. Each line represents the path for a single trial for the parameter in question.

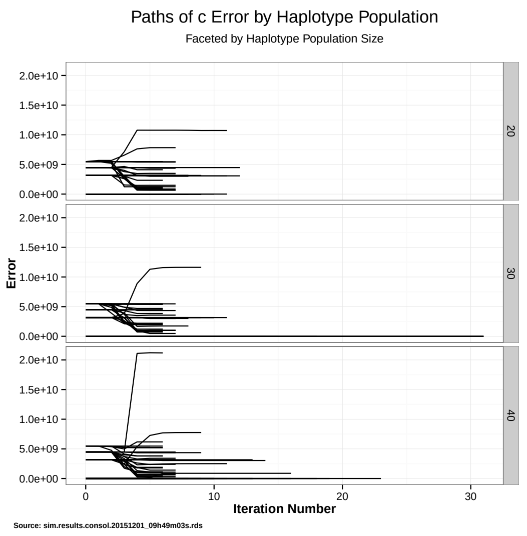
Finally, we consider the same analysis varying the haplotype population size. Figure 3.38 presents the results in the same format. This time, we do not see a distinctive pattern with regard to the haplotype population size. There is no clear trend as the size increases, and we see some erratic behavior at each of the levels of this variable. There



(a) Parameter $\mathbf{p}$



(b) Parameter $\boldsymbol{\pi}$



(c) Parameter c

Figure 3.38: Path of Estimate Errors by Haplotype Population Size. Each line represents the path for a single trial for the parameter in question.

does not appear to be evidence that the population size makes a difference in terms of the accuracy of the estimation.

3.4 Conclusion on Simulation Results

This chapter performed a series of simulations, each structured to assess the accuracy and runtime performance of the `LITSE` algorithm with respect to a series of parameters. The exercises in this chapter support the following conclusions,

1. The algorithm performs best in terms of accuracy with
   a) low levels of variance, as measured by the `c` variable,
   b) large sample sizes, as measured by the `no` variable, and
   c) low levels of gappiness, as measured by the `br` variable.
2. In addition, the algorithm's runtime is positively related with a number of factors including,
   a) "environmental" features characterizing the run such as,
      i. the number of loci as measured by the parameter `nl`, and
      ii. the number of observations as measured by `no`

      with both appearing to have a linear effect, and
   b) runtime parameters explictly designed to control the balance between an exhaustive search and an acceptable runtime including,
      i. the maximum nodes chosen per level, as captured by the parameters `smnc` and `hmnc`, and
      ii. the maximum level at which the proportions are allowed to float, as controlled by the `mfl` parameter.

The algorithm performed well against its closest competitor, the `IMH` algorithm, in an artificial situation where all observations had a true MOI equal to the longest locus length. In reality, as demonstrated in the simulation where haplotypes were drawn randomly, it is expected that many observations will have true solutions where this condition does not hold. A comparison of `LITSE` and `IMH` in such an environment would likely show a superior performance on the part of `LITSE`.

Finally, one of the greatest challenges the algorithm faces is to identify the correct MOI for an observation. The algorithm addresses this by monitoring the relative improvement in fit as additional MOIs are attempted and terminating the search when the improvement falls below a user-defined level for the parameter `mer`, set to 0.2 in this simulation exercise.

In the next section, we perform a similar test of the algorithm, but this time with real data generated by the Greenhouse Lab at UCSF.

Appendices

3.A Identification of Characteristic c^l Values for Simulation

In this section, we discuss the derivation of plausible values for $\mathbf{c}^l$ to be used in the simulations.

Recall from §2.6.1.6.2, and in particular equation (2.25), that the concentration parameter c^l controls the variance of the observed proportions at locus l about their true proportions (as determined by the underlying haplotypes and their proportions).

Since we want to make the simulated data used in the assessment of `LITSE` to be as realistic as possible, we consider a set of real lab data where we have both observed and true proportions for several loci across many observations. This dataset – *dataLab*– was sourced by the Greenhouse Laboratory at the University of California, San Francisco. Table 3.19 presents a summary of the data.

Feature	Value
Number of Observations	132
Number of Loci	9
Loci Names	"TA81","TA60", "lyAlpha","TA1", "PFG377", "Ara2", "TA87","TA40", "1Pftpk2"

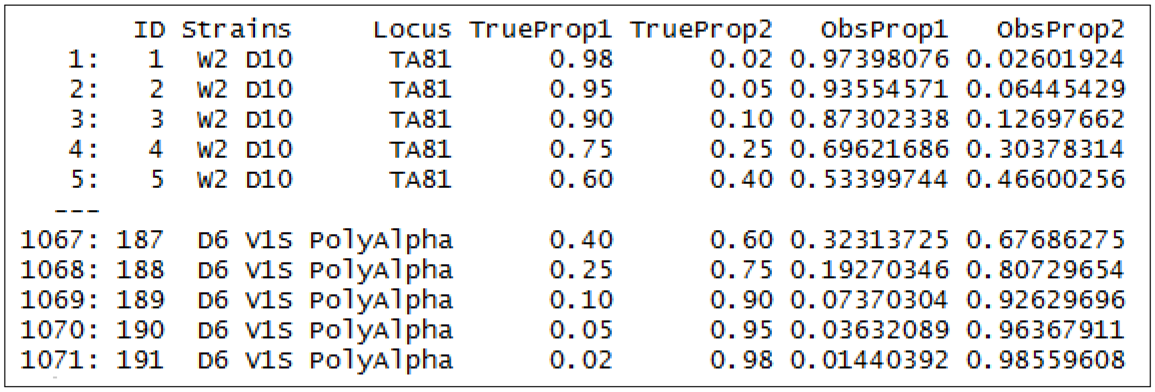
Table 3.19: Summary of *dataLab*. The data count 132 different observations across 9 different loci.

The aim is to estimate a value for c^l for $l = 1, \ldots, 9$. That is, for *dataLab*, the task is to find a single c^l for each locus l that explains the variance between the implied allele proportions and the observed allele proportions. To this end, we compute a maximum likelihood estimate for each c^l separately using the likelihood equation for that locus alone, based on (2.25), as follows,

$$\hat{c}^l = \underset{c \in (0,\infty)}{\text{argmax}} \prod_{j \in dataLab(l)} \frac{1}{B(c\boldsymbol{\alpha} + 1)} \prod_{i=1}^{d} o_i^{c\alpha_i}, \tag{3.6}$$

where l denotes the locus in question, $dataLab(l)$ denotes the subset of rows in the dataset relevant to the locus l, and $d = 2$ since by construction for all loci there will be exactly two non-zero alleles. As always, o_i denotes an observed allele proportion and α_i denotes the corresponding expected allele proportion, to be discussed in more detail below. Note that equation (3.6) uses a slight simplification of equation (2.25), since we know by construction that both the observed and true proportions sum to 1.[15]

An extract of *dataLab* is included in Figure 3.39 as a snapshot of the relevant variable used in the R programming environment. The columns "TrueProp1" and "TrueProp2" contain the proportions of the haplotypes. Since there is no overlap in the alleles of the haplotypes, these values equal the true allele proportions as well. The columns "ObsProp1" and "ObsProp2" contain the actual proportions observed after certain pre-processing.



```
       ID Strains     Locus TrueProp1 TrueProp2   ObsProp1   ObsProp2
   1:   1  W2 D10      TA81      0.98      0.02 0.97398076 0.02601924
   2:   2  W2 D10      TA81      0.95      0.05 0.93554571 0.06445429
   3:   3  W2 D10      TA81      0.90      0.10 0.87302338 0.12697662
   4:   4  W2 D10      TA81      0.75      0.25 0.69621686 0.30378314
   5:   5  W2 D10      TA81      0.60      0.40 0.53399744 0.46600256
  ---
1067: 187  D6 V1S PolyAlpha      0.40      0.60 0.32313725 0.67686275
1068: 188  D6 V1S PolyAlpha      0.25      0.75 0.19270346 0.80729654
1069: 189  D6 V1S PolyAlpha      0.10      0.90 0.07370304 0.92629696
1070: 190  D6 V1S PolyAlpha      0.05      0.95 0.03632089 0.96367911
1071: 191  D6 V1S PolyAlpha      0.02      0.98 0.01440392 0.98559608
```

Figure 3.39: Snapshot of Lab Dataset *dataLab*. This figure presents the first five and last five rows of the data table of allele proportions, true and observed, as generated in the Greenhouse Lab. In all cases, the MOI of an observation equals 2. The column "ID" indicates the observation number, "Strains" indicates the names of the strains/haplotypes, "Locus" indicates the locus in question, "TrueProp1" and "TrueProp2" indicate the true proportions of the two alleles present at the locus, and "ObsProp1" and "ObsProp2" indicate the corresponding observed proportions. Note that while the maximum value of "ID" is 191, some numbers have been deleted in the sequence, leaving 132 observations in total.

There are 1071 rows for 132 unique observations, and since the latter does not divide the former, we know immediately that the sample sizes will vary somewhat from locus to locus.

[15]In other words, we drop the case structure of the pdf.

When we compute the estimates for c^l for each of the nine loci, we get the results presented in Table 3.20. The range of estimated values for c^l is roughly 200-250. The

	Locus	Count	c
1	Ara2	130	204.39
2	PFG377	101	241.51
3	PfPK2	132	205.47
4	PolyAlpha	130	202.68
5	TA1	131	205.44
6	TA40	120	247.89
7	TA60	77	248.51
8	TA81	123	236.93
9	TA87	127	221.62

Table 3.20: The distribution of the counts is fairly even, with one locus noticeably below the others. Nevertheless, no locus has an inappropriately low number on the basis of which to estimate its c parameter. The range of c value estimates will be used as a guideline during the simulation.

simulation exercise will ensure that the range of c values envelopes this range to ensure the reasonability of the parameter values.

3.B Supplementary Figures

3.B.1 SSI Performance for Top Solutions

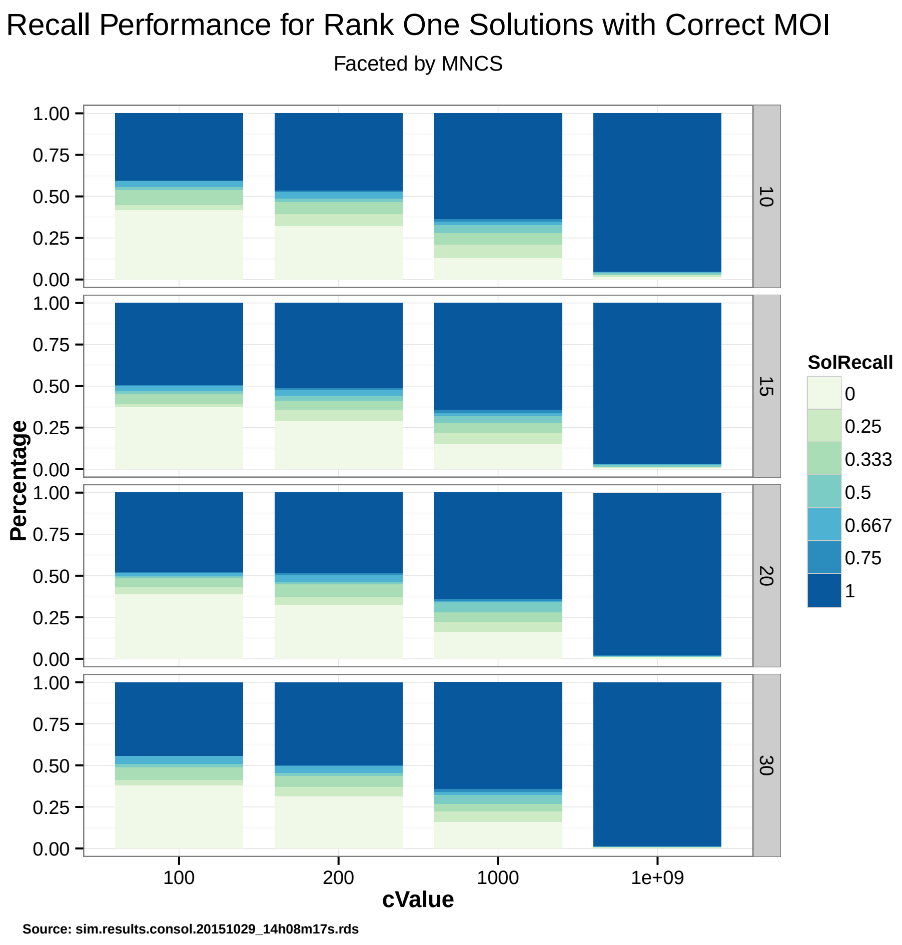


Figure 3.40: Recall/Precision Performance for Rank One Solutions of Correct MOI. Constructed using only solutions whose estimated MOI matched the true MOI. When the estimated MOI equals the true MOI, the recall and precision are equal for a solution. The results are faceted by `smnc`(vertically) and the value of c (horizontally). Each panel shows a bar chart showing the distribution of the recall/precision results for that panel.

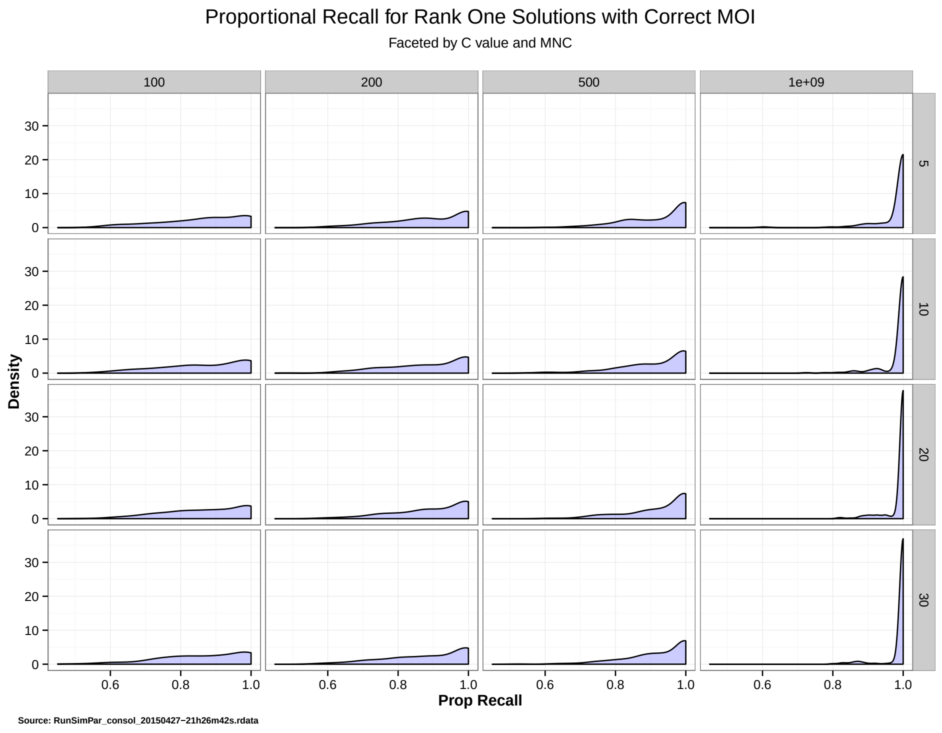


Figure 3.41: Proportional Recall/Precision Performance for Rank One Solutions of Correct MOI. Constructed using only solutions whose estimated MOI matched the true MOI. When the estimated MOI equals the true MOI, the recall and precision are equal for a solution. The results are faceted by `smnc`(vertically) and the value of c (horizontally). Each panel shows a density plot showing the distribution of the proportional recall/precision results for that panel.

3.C Supplementary Tables

3.C.1 Unique Paths to Correct Solution

NumLoci	ID	1	2	3	4	5	6	7	8	9	10
10	1	1	1	1	1	1	1	1	1	1	1
10	2	1	1	1	1	1	1	1	1	1	2
10	3	1	1	1	1	1	1	1	1	1	3
10	4	1	1	1	1	1	1	1	1	1	5
10	5	1	1	1	1	1	1	1	1	1	8
10	6	1	1	1	1	1	1	1	1	1	12
10	7	1	1	1	1	1	1	1	1	1	13
10	8	1	1	1	1	1	1	1	1	1	16
10	9	1	1	1	1	1	1	1	1	1	32
10	10	1	1	1	1	1	1	1	1	2	2
10	11	1	1	1	1	1	1	1	1	2	3
10	12	1	1	1	1	1	1	1	1	2	6
10	13	1	1	1	1	1	1	1	1	3	3
10	14	1	1	1	1	1	1	1	1	3	4
10	15	1	1	1	1	1	1	1	1	3	15
10	16	1	1	1	1	1	1	1	1	4	4
10	17	1	1	1	1	1	1	1	1	5	5
10	18	1	1	1	1	1	1	1	1	5	21
10	19	1	1	1	1	1	1	1	1	6	51
10	20	1	1	1	1	1	1	1	1	7	7
10	21	1	1	1	1	1	1	1	1	9	9
10	22	1	1	1	1	1	1	1	1	11	11
10	23	1	1	1	1	1	1	1	1	11	23
10	24	1	1	1	1	1	1	1	1	15	31
10	25	1	1	1	1	1	1	1	2	2	2
10	26	1	1	1	1	1	1	1	2	2	4
10	27	1	1	1	1	1	1	1	2	2	9
10	28	1	1	1	1	1	1	1	2	3	3
10	29	1	1	1	1	1	1	1	2	4	4
10	30	1	1	1	1	1	1	1	3	3	3

Table 3.21: Sample of Thirty Paths to Correct Solution. The table displays the first 30 unique paths for the case where the number of loci equals 10. For the complete set of unique paths for all numbers of loci investigated, see `www.stat.berkeley.edu/~curt/litse/uniquepaths.html`.

3.D Alternative Measure of Similarity

All three parameters, in either true or estimated form, are vectors. As such, it is convenient to adopt a scalar-valued metric to assess the degree to which an estimated vector and true vector agree. As introduced in §2.5.4 and used in related research, one such metric is the Renkonen similarity measure [33] for proportional vectors,

$$S_R := \sum_{i=1}^{m} \min(p_{i,1}, p_{i,2}) \tag{3.7}$$

where m is the total number of entries in each vector and $p_{i,j}$ is the proportion of the i^{th} entry in the j^{th} vector. In our implementation, we must make sure that both the true and estimated vectors have the same occurrence of entries and that these align. For example, in the case of $\boldsymbol{\pi}$, the two vectors must agree on the exact sequence of all haplotypes represented in each vector. If one vector has a non-zero entry for a haplotype, the other must have an entry at the same location, with value 0 if appropriate.

Since by definition we must have $p_{i,j} \geq 0$ and $\sum_i p_{i,j} = 1$, it follows that $S_R \in [0, 1]$. The extreme case of 0 indicates no overlap at all in the proportions and thus a maximum dissimilarity; conversely, the extreme of 1 indicates perfect alignment and the maximum similarity. As such, a positive outcome in the estimation is to witness the S_R increase with each iteration.

Chapter 4

An Application to Real Data

Having tested the `LITSE` algorithm on simulated data in Chapter §3, in this chapter we demonstrate the application of the algorithm to a real world dataset from the Greenhouse lab at UCSF.

In this exercise, a number of observations were generated by mixing known malarial haplotypes in pre-determined proportions and then applying the usual platform to in turn read the observed allele proportions for each locus and each observation.

Part of this process involves adjusting for "artifacts" present in raw allele proportion data. This step is described briefly in §2.4.2 in this dissertation. As the `LITSE` algorithm is intended for use *after* such artifacts have removed, the input to this exercise was adjusted allele proportions using an approach adopted by the Greenhouse Lab.

Since this exercise is not a simulation, we do not have the ability to generate data according to a large set of parameter combinations. However, it gives a flavor for how the algorithm may be used in practice.

The remainder of this chapter describes the data in question, presents the results of the application of the `LITSE` algorithm, and closes with a conclusion on the exercise.

4.1 Data Description

This section describes the data used in the exercise. Details of the data can be found in the Appendix §4.A and on the `LITSE` website, as described in more detail below.

4.1.1 Data Generation

The data were generated by identifying nine different strains and then creating observations by mixing either two or three strains in pre-specified proportions.

This implies that, as was the case in the simulation exercise in Chapter §3, all true haplotypes are known and the true solution is known for each observation.

Contrary to the situation in Chapter §3, we do not model directly model any of the three parameters $\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}$. Each of these parameters will be estimated by the `LITSE` algorithm, but these estimates will not be compared against their true value (since this is unknown).

4.1.2 Data Preprocessing and Cleansing

Recall from §2.4.2 that the data observed for an individual observation is a set of real-valued peak heights for each locus using a fluourescence assay after amplification. Figure 4.1 presents a simple illustration of the data for one locus and one observation. In this illustration, there are four distinct peaks, with each peak corresponding to a potential allele reading for a particular base pair count (on the x axis). The heights of the peaks are indicative of the proportions of the alleles, but are subject to two revisions:

1. Remove all artifacts. An artifact is defined as an observed peak for an allele that does not actually exist for the observation. The identification and removal of artifacts is performed before the application of the `LITSE` algorithm and is beyond the scope of this dissertation. In the illustration, for example, the second peak may well be deemed an artifact and removed from the data, implying that there are in fact only three true alleles expressed for that observation/locus pair.
2. Debias proportions. With the artifacts removed, only peaks deemed to represent true alleles remain. However, it has been established that the relative peak height

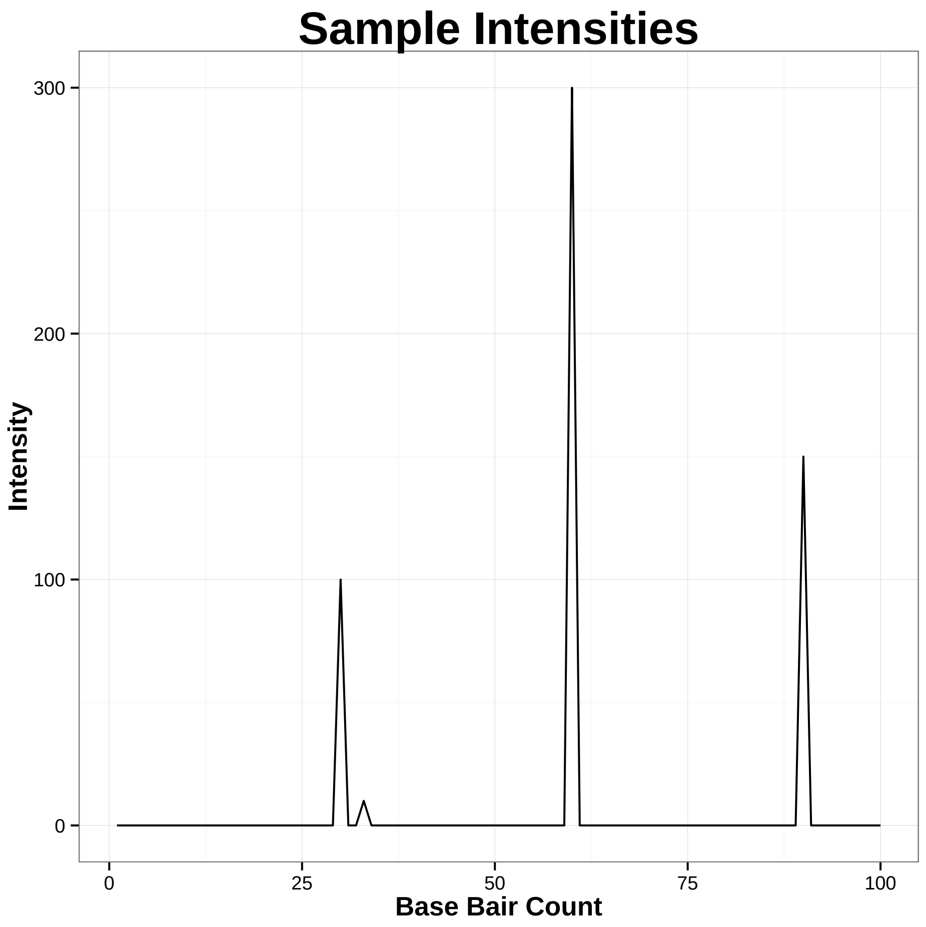


Figure 4.1: Sample Intensity Readings. There are four peaks in the sample illustration, the second of which might be deemed an artifact due to its low value and close proximity to a much greater peak to the immediate left.

for an allele is not a perfect indicator of the allele's proportion. Depending on the locus, there is frequently a bias for either low base pair count or high base pair count alleles. This is discussed in more detail in the Appendix.

4.1.3 Data Summary

With the corrections described in §4.1.2 completed, we now have a set of checked and bias-adjusted observations.

A summary of the data environment is presented in Table 4.1 below. While the total

Feature	Number	Note
Number of Haplotypes	216	See Table 4.5 in Appendix.
Number of Loci	24	See Table 4.6 in Appendix.
Number of Observations	176	See `www.stat.berkeley.edu/~curt/litse/obsdata.html`.
		See `www.stat.berkeley.edu/~curt/litse/obswhaps.html`.

Table 4.1: Data Summary. The highlights of the data used in this application exercise.

number of observations is 176, their completeness (or "gappiness") varies significantly from observation to observation. Table 4.2 profiles the gappiness in the sample, with each column header specifying a particular level of completeness (e.g., exactly n complete loci) and the entry in the "Count" row specifying the number of observations with that level of completeness. Given the total number of observations, we see that only a minority

	Number of Loci Complete																
	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Count	1	2	1	3	3	4	5	4	9	3	7	7	20	19	35	35	18

Table 4.2: Observation Totals by Completeness. The table presents the total count of observations for each level of completion. A higher number indicates greater completion. For example, there are 3 observations with exactly 11 complete loci.

are 100% complete (i.e., are included in the far right figure).

Figure 4.3 below presents the same information along the other axis; that is, it shows for each locus how many of the observations have complete allele information for that locus. The loci are presented in three columns and are sorted from most frequently complete to least. For example, we see that the locus AS7 is the most complete among all loci while the least common loci have coverage of less than half of all observations.

For a detailed display of each observation and for which particular loci the observation has allele information, please view the table at `www.stat.berkeley.edu/~curt/litse/obsdata.html`.

Locus	Count	Locus	Count	Locus	Count
AS7	175	AS12	154	PfPK2	144
AS32	169	AS2	154	AS25	140
AS1	166	AS21	154	TA40	139
B7M19	162	Ara2	152	AS31	137
TA81	162	AS20	149	TA87	137
AS4	161	AS3	149	AS11	136
AS8	159	AS34	145	AS15	132
AS13	156	PFG377	144	TA1	71

Table 4.3: Locus Totals by Completeness. The table presents the count of observations with complete allele information for each locus. A higher number indicates greater completion.

4.2 Approach

The approach of this exercise is straitforward: we run the `LITSE` algorithm once on the data in question with the parameters as outlined in Table 4.4 and analyze the results. The parameters listed in the table are fewer than in the simulation sections of Chapter §3 because only runtime parameters – i.e., those that determine how the algorithm will function in terms of breadth of search or number of iterations – need to be specified. The exercise in this chapter can be viewed as a single trial in a simulation run, where the environmental features, e.g., number of observations, loci, and haplotypes, have been determined by the data and have been presented in §4.1.3.

Parameter	Setting
`mfl`	100
`smnc`	30
`hmnc`	50
`mer`	0.2
`am`	2
`mi`	30

Table 4.4: Application Parameter Settings. These are the settings used in single run in §4. Only runtime parameters need to be set since other features of the run, e.g., number of observations, are determined by the data themselves.

The present exercise differs from the simulations in that we test only a single set of observations, with a fixed set of loci and haplotypes. In addition, we have no control over the data generating parameters $\mathbf{p}, \boldsymbol{\pi}, \mathbf{c}$. We do, however, know the correct solution for each of the observations since the observations were constructed with known haplotype mixtures.

4.3 Results

In this section, we describe the results of applying the method to the data. Given that we cannot assess the accuracy of the parameter estimates $\hat{\mathbf{p}}, \hat{\boldsymbol{\pi}}, \hat{\mathbf{c}}$, we focus on the accuracy of the solutions for the observations in question.

Recall that, as documented in Table 4.2, the observations vary widely in terms of their loci coverage, with some being complete and some missing a full two thirds of all loci. We would naturally expect the accuracy of the estimates to decline as the gappiness increases.

4.3.1 Inclusion of Correct Solution

The first question we ask is to what extent the algorithm is able to include the correct solution among all proposed solutions for an observation. Figure 4.2 presents the results for every level of gappiness witnessed in the data. The completeness is presented on the x-axis and the percentage of times the solution was included in the set of solutions is presented on the y-axis. We see that the results are roughly as expected, indicating that the increasing completeness (or decreasing gappiness) found on the left hand side of the x-axis does indeed help in including the correct solution. Note that these results are complicated by the sparsity of the data, as indicated in Table 4.2; for example, there is only a single example of an observation with only eight loci complete.

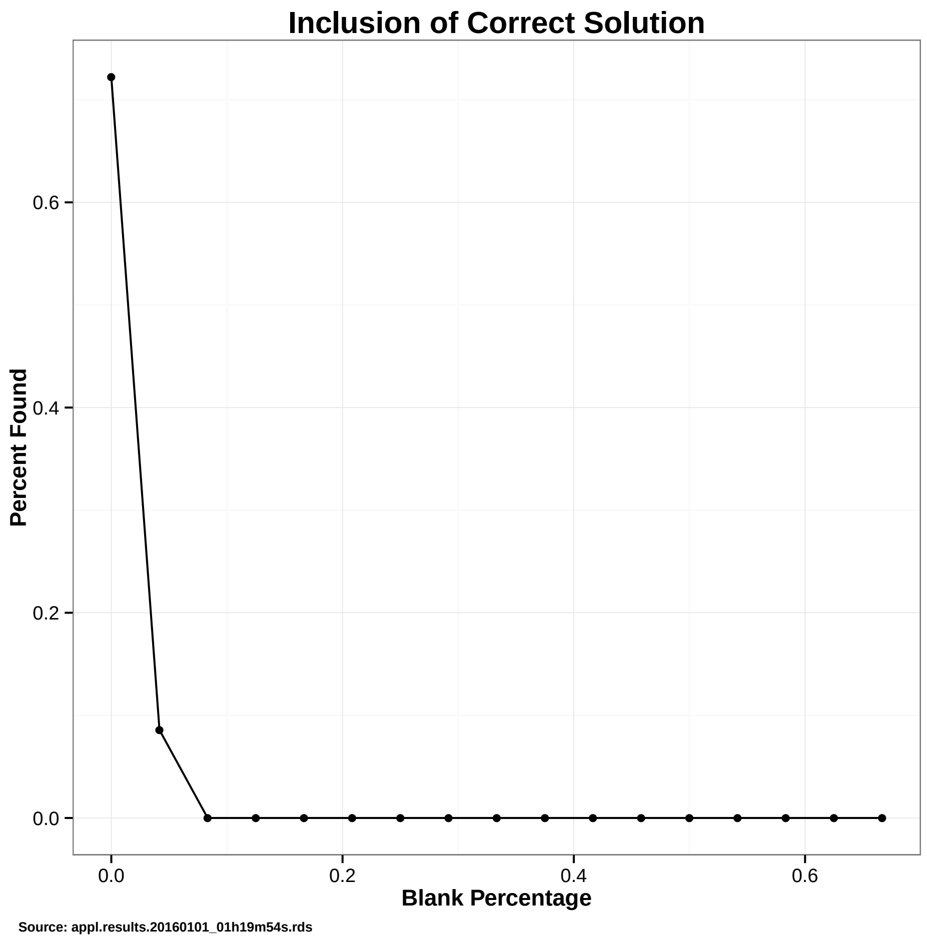
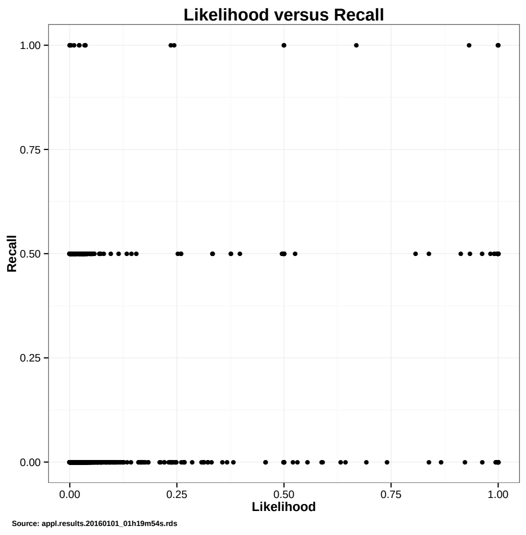


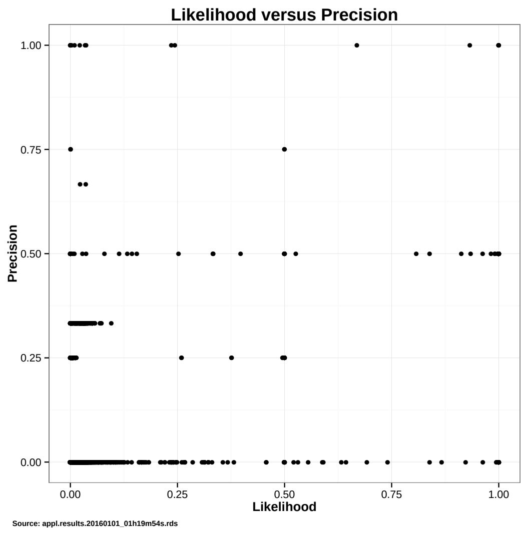
Figure 4.2: Inclusion of Correct Solution. The figure plots the average inclusion rate as a function of the completeness of the loci information.

4.3.2 Accuracy of Top Solution

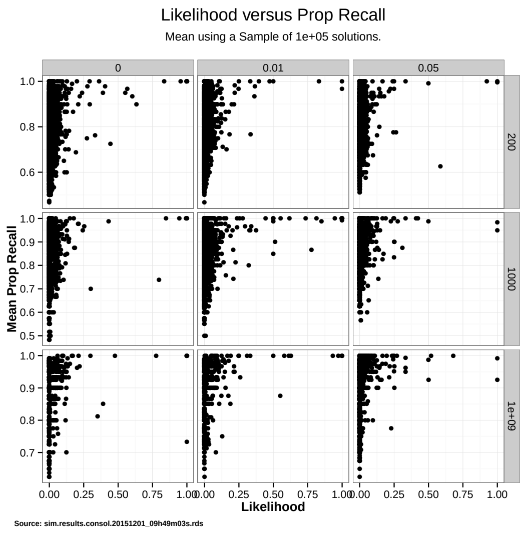
The primary question we ask is how the solutions performed in terms of the four accuracy measures introduced in §3.2.4.3. This is depicted in Figure 4.3. We see that the the algorithm performs well on these data in terms of the four measures of accuracy. In particular, for the two proportional measures of performance there is a clear direct relationship between a solution's likelihood and its accuracy.



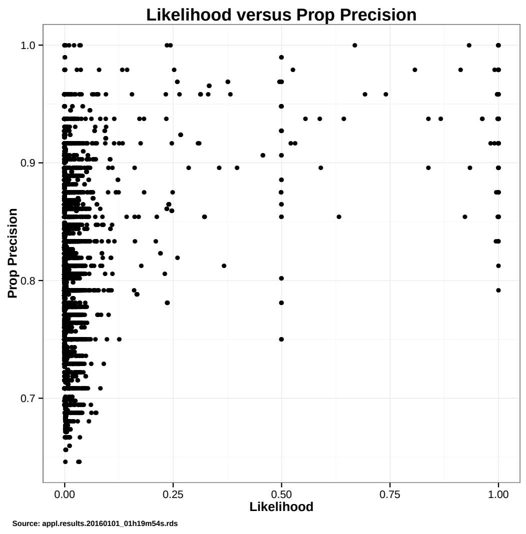
(a) Recall



(b) Precision



(c) Proportional Recall



(d) Proportional Precision

Figure 4.3: Likelihood and and Accuracy Measures. Each of the four subfigures focuses on one of the measures of accuracy.

4.4 Conclusion

In this chapter, we briefly applied the `LITSE` algorithm to an example of real data obtained from controlled mixtures. There was a wide variety of gappiness in the data, and one of the considerations was the extent to which the estimates for an observation suffered if the observation featured a high degree of gappiness.

The algorithm performed well and the results confirmed the assumption that an observation's estimate will increase in accuracy as the gappiness of its allele proportion readings increases.

The intention is to perform significantly more exhaustive testing on additional laboratory data and present this in one or more papers using the website described in §5.

The following appendix includes tables describing the dataset in more detail.

Appendices

4.A Supplementary Tables

	W2	D6	HB3	V1S	U519	FCR3	D10	U497	U659
AS1	173	173	173	173	173	173	176	173	176
AS2	198	198	195	195	189	195	195	195	195
AS3	173	167	170	161	173	176	161	176	170
AS4	156	156	156	159	159	159	159	159	159
AS32	257	242	257	242	226	257	257	242	242
AS7	171	171	171	171	171	171	171	174	171
AS8	197	197	200	197	200	197	219	197	197
AS11	162	165	162	162	162	162	162	162	162
AS12	160	163	163	163	160	160	163	163	163
AS13	183	186	186	186	183	183	183	186	186
AS15	120	123	121	127	135	121	133	121	127
AS20	165	162	168	165	165	165	165	165	162
AS21	155	155	155	152	158	155	152	158	155
AS34	184	187	184	184	184	184	184	187	187
AS25	99	109	90	96	109	102	90	119	103
B7M19	155	155	155	138	155	155	155	155	155
AS31	190	193	199	196	202	190	196	199	196
Ara2	143	143	137	146	134	137	137	140	146
PfPK2	170	179	203	173	176	173	173	173	182
TA1	171	174	180	171	183	168	177	168	171
TA87	112	109	109	115	100	109	100	100	106
TA81	127	127	133	121	124	121	121	130	121
PFG377	103	103	106	103	103	103	97	103	103
TA40	265	280	256	250	247	250	273	267	270

Table 4.5: Application Haplotypes. The haplotypes/strains used and their composition, one strain per column and one locus per row.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21	A22	A23	A24	A25	A26	A27	A28	A29	A30	A31	A32	A33
AS1	161	164	167	170	173	176	179	182	185	188																							
AS2	181	184	186	189	192	195	198	201	204	207	211	217																					
AS3	148	151	154	158	161	164	167	170	173	176	179	182	185	188	190	193	196																
AS4	147	150	153	156	159	162	165	168	171	174	177	179																					
AS32	190	193	196	201	211	221	223	226	230	233	236	239	242	245	248	251	254	257	260	263	266	269	272	278									
AS7	152	155	158	161	164	168	171	174	177	180	184	187	193	197																			
AS8	171	174	177	181	184	189	191	194	197	200	203	206	209	215	219	222																	
AS11	141	144	147	150	153	156	159	162	165	168	171	173	176	180	183	186																	
AS12	139	144	148	151	155	157	160	163	166	169	173	175	179	183	185																		
AS13	162	165	168	171	174	177	180	183	186	189	192	195	198	201	205																		
AS15	108	111	114	117	120	124	127	130	133	136	139	142	145	148	150	153	156																
AS20	143	146	149	152	155	158	162	165	168	171	174	177	180	183	186	189	192	195	198	202	205	208											
AS21	146	149	152	155	158	161	164	168	171	174	177	180	183	186	190	193																	
AS34	151	154	157	160	163	166	169	172	175	178	181	184	187	190	193	196	200	203															
AS25	66	69	72	75	78	81	84	87	90	93	96	100	103	106	109	112	115	119	122	125	128	131	134	137	140	143	146	149					
B7M19	135	138	140	143	146	149	152	155	158	161	163	166	170	172	175	179	183	186	189														
AS31	147	149	153	155	158	162	164	167	170	173	176	178	181	184	187	190	193	196	199	202	205	209	212	215	218	221	224	227	230	233			
Ara2	118	121	125	128	131	134	137	140	143	146	149	152	155	158	161	164																	
PfPK2	142	145	148	152	155	158	161	164	167	170	173	176	179	182	185	188	191	194	197	200	203	206	209	212	215	218	222	225	228	231	234		
TA1	132	134	137	140	143	147	150	153	156	159	162	165	168	171	174	177	180	183	187	189	192	196	199	202	205	208	211	214	217	220	223	226	229
TA87	82	85	88	91	94	97	100	103	106	109	113	116	119	121	124	127	131	134	137	140													
TA81	104	106	109	112	115	118	121	124	127	130	134	137	139	143	146	149	152	155	158	161	164												
PFG377	88	91	94	97	100	103	106	109	112	115	118	121	124	131																			
TA40	201	205	208	211	214	217	220	223	226	229	232	235	238	241	244	247	250	253	256	259	261	265	267	270	274	277	279	282	286	289	292		

Table 4.6: Application Loci. The loci used and their possible alleles, one locus per row. The far left column is the locus name.

Locus	α	β	SSR (Pre)	SSR (Post)
Ara2	-0.068	0.087	7.638	4.349
AS1	0.008	0.017	0.928	0.968
AS11	-0.010	0.003	4.648	4.633
AS12	-0.088	0.042	7.092	7.216
AS13	0.006	-0.036	2.500	2.494
AS15	-0.025	0.022	12.801	12.803
AS2	-0.119	0.113	3.790	2.648
AS20	0.007	-0.213	13.598	7.256
AS21	-0.076	-0.069	6.353	5.559
AS25	-0.018	-0.013	5.935	5.347
AS3	0.038	0.004	3.156	3.118
AS31	-0.038	-0.026	8.924	8.629
AS32	-0.084	0.024	3.707	2.451
AS34	0.029	0.008	2.614	2.630
AS4	0.097	-0.287	16.321	12.443
AS7	-0.173	-0.204	2.369	0.466
AS8	-0.012	-0.010	1.533	1.378
B7M19	0.025	-0.011	0.939	0.746
PFG377	0.003	0.048	4.622	4.174
PfPK2	0.025	0.010	6.261	5.914
TA1	0.013	0.092	23.085	22.143
TA40	0.113	0.003	4.466	4.395
TA81	0.063	0.006	3.711	3.682
TA87	0.140	0.086	11.612	4.490

Table 4.7: Debiasing Model for Loci. Each locus is fit with a linear model to debias individual entries based on their allele position.

4.B Supplementary Figures

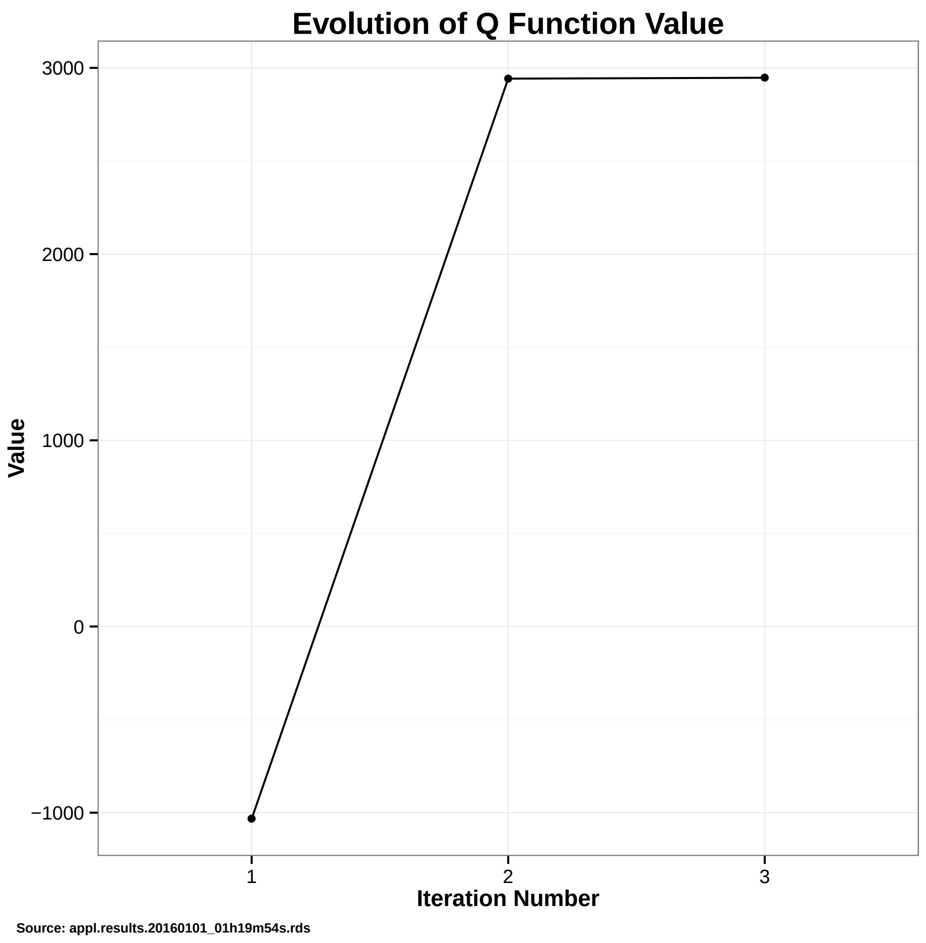


Figure 4.4: Evolution of Q Function Value. The figure plots the value of the Q function in each of the iterations. The function is quickly maximized and the number of iterations is minimal.

Chapter 5

Implementation

The LITSE application has been made available to any user through the use of a dedicated website. In addition, an `R` package is envisioned for those users wishing to use LITSE offline. Both are discussed further in this chapter.

5.1 LITSE Website

The LITSE application can be found at the following website:

`http://gandalf.berkeley.edu:3838/curt/LITSE`

The website is based on the `shiny` technology provided by the organization RStudio [34]. As such, it provides an appealing user interface while using the same code as that deployed in §3.

Following the standard format for a `shiny` implementation, the website can be divided into two parts: a left-hand side panel, which is always visible, stating the steps to be taken and a right-hand side set of six tabbed panels. These panels correspond to the five steps to be taken in any use of the website plus a help page explaining the format of the data to be submitted.

5.1.1 Input Formats

Consistent with §2, there are two sets data to be input:

1. a description of the loci environment, and
2. the observed allele proportions

Each set of data is submitted to the website in the form of a csv file, which follows a specific tabular structure.

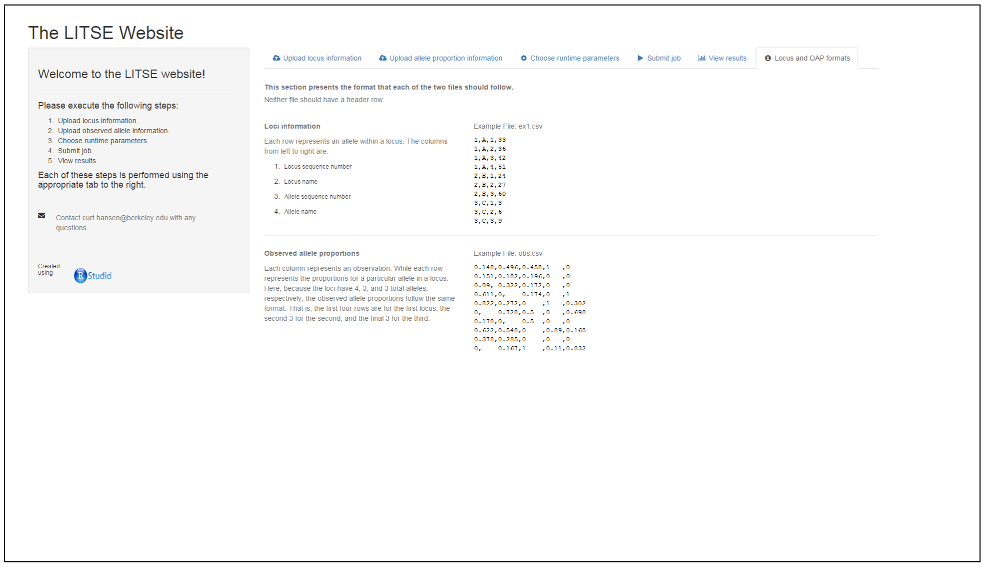
The loci environment is described through a csv file with four columns, one each for the locus id (a sequence number), locus name (a character string), allele if (within the locus), and allele name (a character string, perhaps an indication of the repeat count at the relevant locus). There is one row per allele within a locus. Since locus information existing above the component allele information is included in the first two columns, the values in these two columns will be repeated for all rows belonging to the locus.

The observed allele proportions have a fixed number of rows, one row per locus/allele combination. Thus this file will have the same number of rows as the loci environment file. The number of columns is equal to the number of observations in the sample, with each column carrying all the data for a single observation. The i, j entry in the table is simply the observed proportion for that allele for that column's observation. As such, the structure of each column is a stacked version of the proportion set `props` described in §2.4.1.2.

These formats are explained in the rightmost tab on the website, as shown in Figure 5.1. The user can click on this tab to be reminded of the necessary format.

Recall that even though the data can be viewed as a table, it is submitted to the website in the form of a csv file; there is no need (or ability) to submit the data as an `R` dataframe. Nevertheless, dataframes are easily converted to and from csv files.

As the structure of the files is predetermined, the submitted csv files should not contain headers. All rows in the files are deemed to be data. There is currently no validation performed on the files input through the website.



Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

Created using RStudio

This section presents the format that each of the two files should follow.

Neither file should have a header row.

Loci information

Each row represents an allele within a locus. The columns from left to right are:

1. Locus sequence number
2. Locus name
3. Allele sequence number
4. Allele name

Example File: ex1.csv

```
1,A,1,33
1,A,2,36
1,A,3,42
1,A,4,51
2,B,1,24
2,B,2,27
2,B,3,60
3,C,1,3
3,C,2,6
3,C,3,9
```

Observed allele proportions

Each column represents an observation. While each row represents the proportions for a particular allele in a locus. Here, because the loci have 4, 3, and 3 total alleles, respectively, the observed allele proportions follow the same format. That is, the first four rows are for the first locus, the second 3 for the second, and the final 3 for the third.

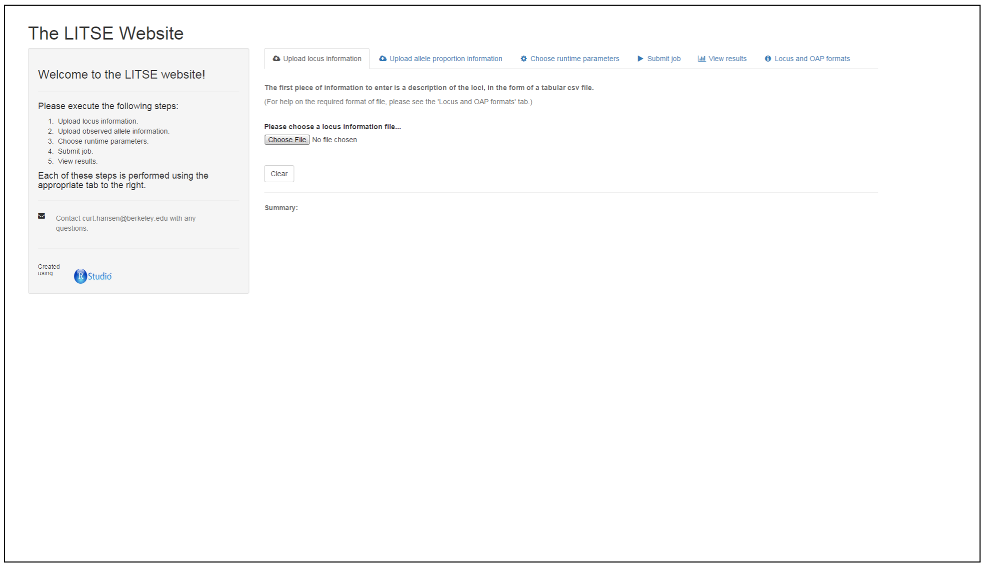
Example File: obs.csv

```
0.148,0.496,0.458,1   ,0
0.151,0.182,0.196,0   ,0
0.09, 0.322,0.172,0   ,0
0.611,0,    0.174,0   ,1
0.822,0.272,0    ,1   ,0.302
0,    0.728,0.5  ,0   ,0.698
0.178,0,    0.5  ,0   ,0
0.622,0.548,0    ,0.89,0.168
0.378,0.285,0    ,0   ,0
0,    0.167,1    ,0.11,0.832
```

Figure 5.1: LITSE Website - Help. The site features a sidebar and a series of tabs along the top. These tabs are used to enter the required information and view the results. The help section shows the format of the required data.

5.1.2 Entering Loci Information

The locus information is entered by clicking on the leftmost tab on the website, as depicted in Figure 5.2.



Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

Created using RStudio

The first piece of information to enter is a description of the loci, in the form of a tabular csv file.

(For help on the required format of file, please see the 'Locus and OAP formats' tab.)

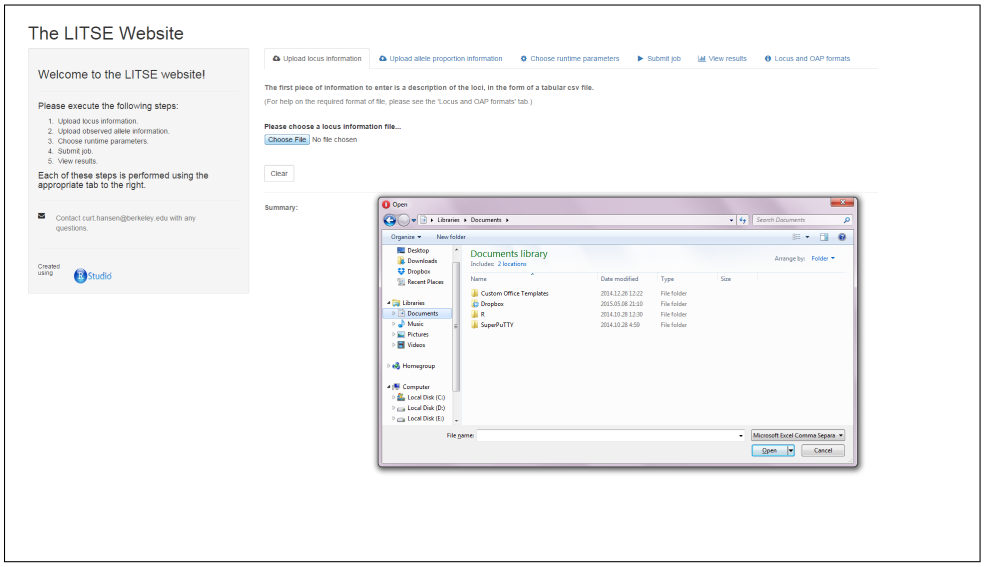
Please choose a locus information file...

No file chosen

Summary:

Figure 5.2: LITSE Website - Locus Information.

Clicking on the $\boxed{\text{Upload file}}$ button opens a dialog box, as depicted in Figure 5.3, allowing the user to select a pre-constructed file.



The LITSE Website

Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

Created using RStudio

The first piece of information to enter is a description of the loci, in the form of a tabular csv file.

(For help on the required format of file, please see the 'Locus and OAP formats' tab.)

Please choose a locus information file...

Choose File No file chosen

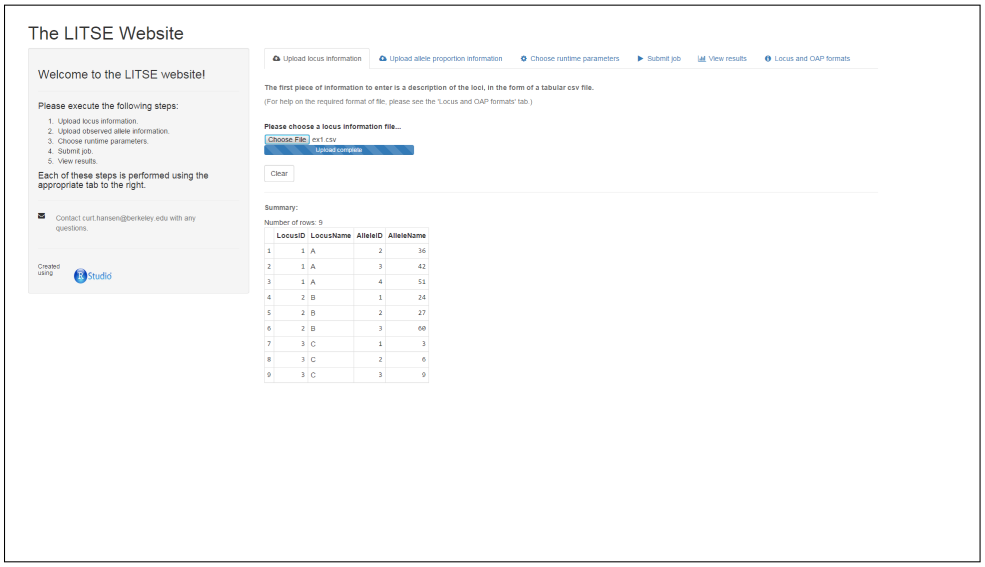
Clear

Summary:

Name	Date modified	Type	Size
Custom Office Templates	2014.12.26 12:22	File folder	
Dropbox	2015.05.08 21:10	File folder	
R	2014.10.28 12:30	File folder	
SuperPuTTY	2014.10.28 4:59	File folder	

Figure 5.3: LITSE Website - Choose File.

Once the file has been chosen, a summary of its contents are displayed near the bottom of the screen for confirmation, as depicted in Figure 5.4.



The LITSE Website

Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

Created using RStudio

The first piece of information to enter is a description of the loci, in the form of a tabular csv file.

(For help on the required format of file, please see the 'Locus and OAP formats' tab.)

Please choose a locus information file...

Choose File ex1.csv

Upload complete

Clear

Summary:

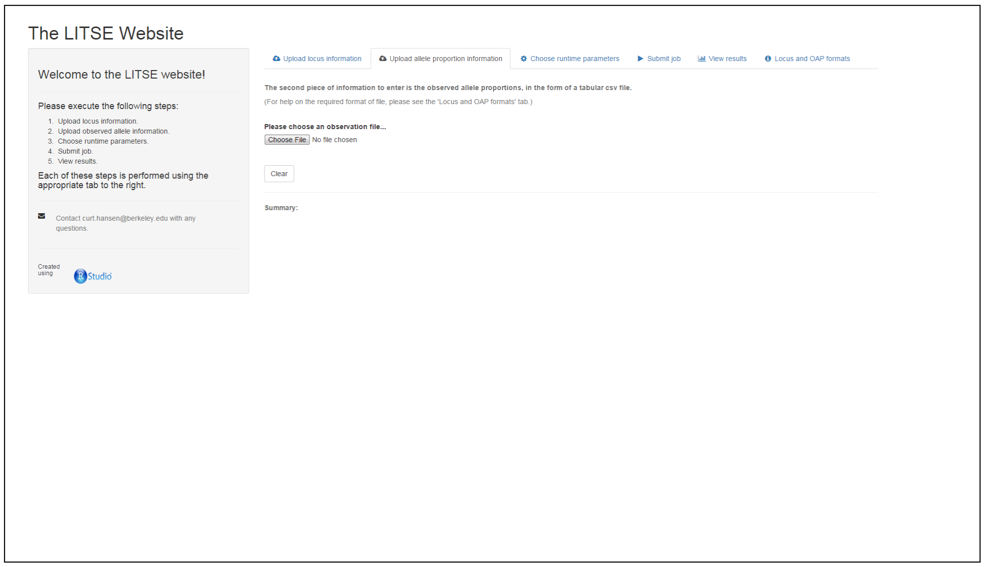
Number of rows: 9

	LocusID	LocusName	AlleleID	AlleleName
1	1	A	2	36
2	1	A	3	42
3	1	A	4	51
4	2	B	1	24
5	2	B	2	27
6	2	B	3	60
7	3	C	1	3
8	3	C	2	6
9	3	C	3	9

Figure 5.4: LITSE Website - Choose Loci File. Once the file has been chosen, a preview of the file is displayed for confirmation.

5.1.3 Entering Observed Allele Information

The next step is to enter the observed allele proportions on the second panel, as depicted in Figure 5.5. The format of the panel is identical to the previous one, including a similar



Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

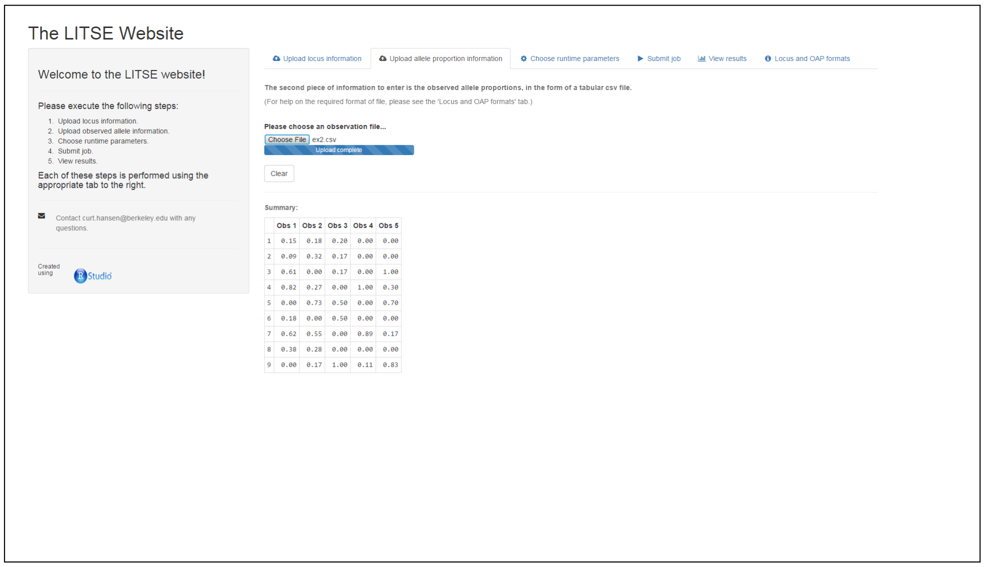
The second piece of information to enter is the observed allele proportions, in the form of a tabular csv file.

(For help on the required format of file, please see the 'Locus and OAP formats' tab.)

Please choose an observation file...

Figure 5.5: LITSE Website - Observed Allele Information.

summary function as depicted in Figure 5.6.



Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

The second piece of information to enter is the observed allele proportions, in the form of a tabular csv file.

(For help on the required format of file, please see the 'Locus and OAP formats' tab.)

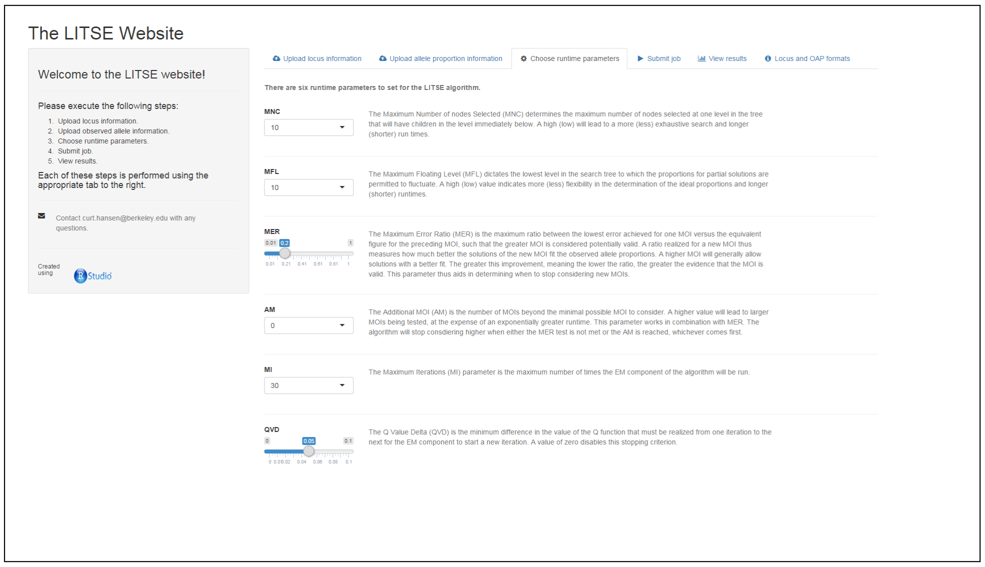
Please choose an observation file...

	Obs 1	Obs 2	Obs 3	Obs 4	Obs 5
1	0.15	0.18	0.20	0.00	0.00
2	0.09	0.32	0.17	0.00	0.00
3	0.61	0.00	0.17	0.00	1.00
4	0.82	0.27	0.00	1.00	0.30
5	0.00	0.73	0.50	0.00	0.70
6	0.18	0.00	0.50	0.00	0.00
7	0.62	0.55	0.00	0.89	0.17
8	0.38	0.28	0.00	0.00	0.00
9	0.00	0.17	1.00	0.11	0.83

Figure 5.6: LITSE Website - Choose Observations File. Once the file has been chosen, a preview is displayed for confirmation.

5.1.4 Entering Run Parameters

The third step is to enter the run parameters – `smnc`, `mfl`, `mer`, `am`, `mi`, and Δq_{min} – discussed in detail in §3. These are set in the third panel, as shown in Figure 5.7. Because the values that these parameters can take are limited, either by the data (e.g.,


The LITSE Website

Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

Created using RStudio

Upload locus information | Upload allele proportion information | Choose runtime parameters | Submit job | View results | Locus and OAP formats

There are six runtime parameters to set for the LITSE algorithm.

MNC 10

The Maximum Number of nodes Selected (MNC) determines the maximum number of nodes selected at one level in the tree that will have children in the level immediately below. A high (low) will lead to a more (less) exhaustive search and longer (shorter) run times.

MFL 10

The Maximum Floating Level (MFL) dictates the lowest level in the search tree to which the proportions for partial solutions are permitted to fluctuate. A high (low) value indicates more (less) flexibility in the determination of the ideal proportions and longer (shorter) runtimes.

MER 0.01 0.2 1

The Maximum Error Ratio (MER) is the maximum ratio between the lowest error achieved for one MOI versus the equivalent figure for the preceding MOI, such that the greater MOI is considered potentially valid. A ratio realized for a new MOI thus measures how much better the solutions of the new MOI fit the observed allele proportions. A higher MOI will generally allow solutions with a better fit. The greater this improvement, meaning the lower the ratio, the greater the evidence that the MOI is valid. This parameter thus aids in determining when to stop considering new MOIs.

AM 0

The Additional MOI (AM) is the number of MOIs beyond the minimal possible MOI to consider. A higher value will lead to larger MOIs being tested, at the expense of an exponentially greater runtime. This parameter works in combination with MER. The algorithm will stop consdiering higher when either the MER test is not met or the AM is reached, whichever comes first.

MI 30

The Maximum Iterations (MI) parameter is the maximum number of times the EM component of the algorithm will be run.

QVD 0 0.05 0.1

The Q Value Delta (QVD) is the minimum difference in the value of the Q function that must be realized from one iteration to the next for the EM component to start a new iteration. A value of zero disables this stopping criterion.

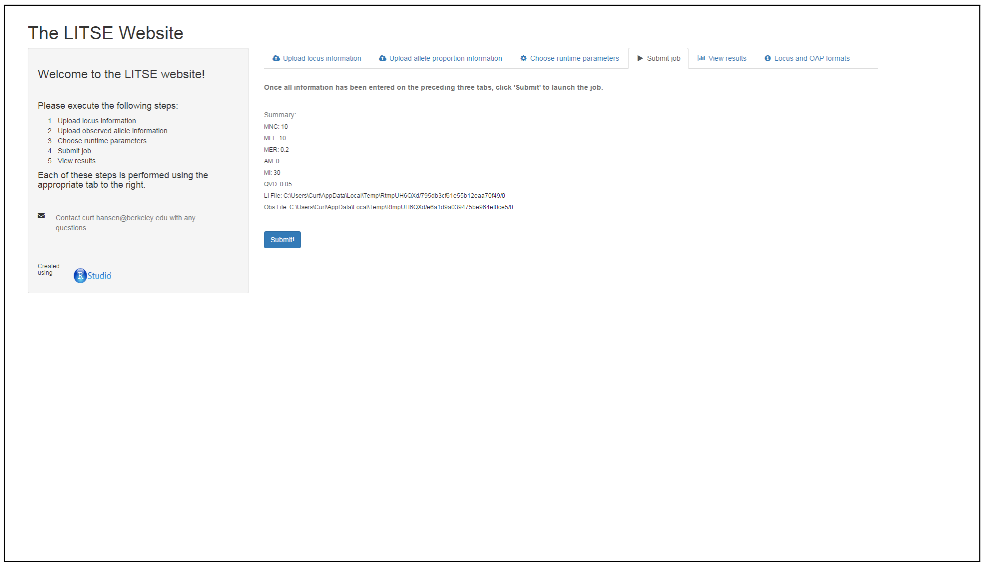
Figure 5.7: LITSE Website - Parameter Settings. Six parameters must be entered. See §3 for more information on each parameter. A default is given for each.

`mfl`is limited to the number of loci) or by computational capacity (e.g., `mer`and especially `am`), the user is forced to choose from a predetermined range of values. These are presented either in the form of list boxes or sliders.

In addition, each parameter has a default setting consistent with what was identified to be a good choice in §3. There is frequently a trade-off between comprehensiveness and computational efficiency.

5.1.5 Submitting the Job

With these three groups of inputs entered, the user can submit the job. This is done on the fourth panel, as depicted in Figure 5.8. A brief summary of the data entered is displayed along with a Submit button.



The LITSE Website

Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

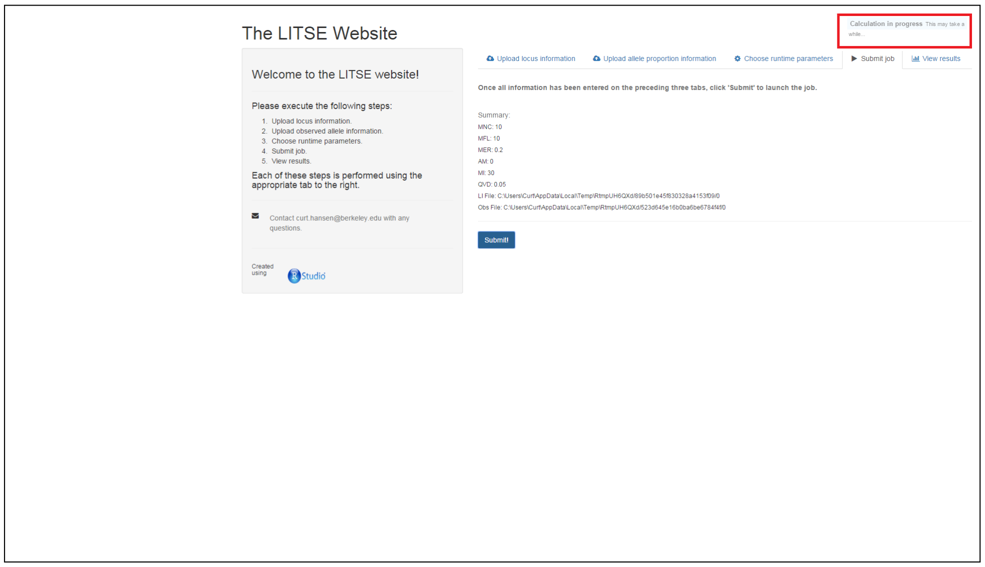
Created using RStudio

Once all information has been entered on the preceding three tabs, click 'Submit' to launch the job.

Summary:
MNC: 10
MFL: 10
MER: 0.2
AM: 0
MI: 30
QVD: 0.05
LI File: C:\Users\Curt\AppData\Local\Temp\RtmpUH6QXd/795db3cf61e55b12eaa70f49/0
Obs File: C:\Users\Curt\AppData\Local\Temp\RtmpUH6QXd/e6a1d9a039475be964ef0ce5/0

Figure 5.8: LITSE Website - Submit Job. The panel displays a summary of the settings and a `Submit` button.

As the algorithm processes the data, a status indicator in the top right corner of the webpage, as depicted in Figure 5.9, indicates the progress of the program. It is the



The LITSE Website

Calculation in progress This may take a while...

Upload locus information
Upload allele proportion information
Choose runtime parameters
Submit job
View results

Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

Created using RStudio

Once all information has been entered on the preceding three tabs, click 'Submit' to launch the job.

Summary:
MNC: 10
MFL: 10
MER: 0.2
AM: 0
MI: 30
QVD: 0.05
LI File: C:\Users\Curt\AppData\Local\Temp\RtmpUH6QXd/89b501e45f830328a4153f09/0
Obs File: C:\Users\Curt\AppData\Local\Temp\RtmpUH6QXd/523d645e16b0ba6be6784f4f/0

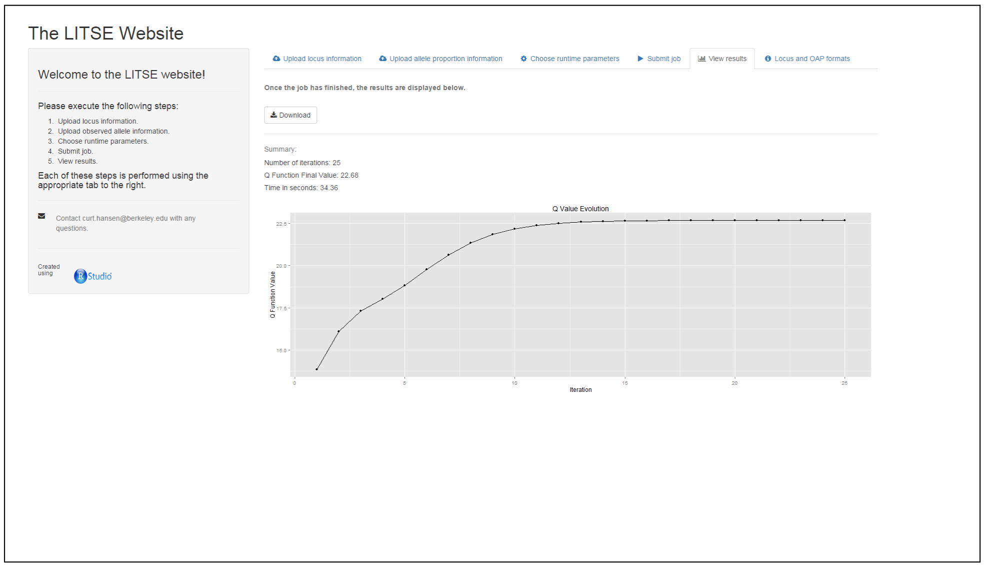
Figure 5.9: LITSE Website - Progress Indicator.

intention to enhance reposition this widget in the center of the panel and to include more detailed information on the current location of the program in any particular run. For example, is the program still in the tree search (Solution Set Identification) phase?

Which observation is it processing? If it is in the refinement phase, which iteration is it on?

5.1.6 Viewing the Results

Once the program has finished, the results are available on the fifth panel. A brief summary of the statistics is included along with a plot of the value of the Q function (see §2.7.4.2.3) over the iterations (in this case, 25). The panel is pictured in Figure 5.10. In addition, the user can download in csv format a table of the results per observation,


The LITSE Website

Welcome to the LITSE website!

Please execute the following steps:

1. Upload locus information.
2. Upload observed allele information.
3. Choose runtime parameters.
4. Submit job.
5. View results.

Each of these steps is performed using the appropriate tab to the right.

Contact curt.hansen@berkeley.edu with any questions.

Once the job has finished, the results are displayed below.

Summary:
Number of iterations: 25
Q Function Final Value: 22.68
Time in seconds: 34.36

Figure 5.10: LITSE Website - View Results. The high level results are displayed in the panel. The user can download the detailed results (e.g., predictions per observation) through the use of the `Download` button.

including all solutions identified and the probability of each.

5.2 LITSE Package

A LITSE package with documentation is planned for release in 2016. Due to the current `Rcpp`code and use of shared objects for the `C++`section of code, it may be easiest to limit the first release of the package to Linux. In particular, the current implementation uses compiled `.so` shared object files that will need to be provided in `.dll` equivalents for Windows. It remains to be be determined to what extent the standard package building infrastructure facilitates this.

5.3 Conclusion

The LITSE algorithm has been made available to all interested users through a simple website. It is intended to make some near term enhancements to the website to include more informative progress information and permit additional results to be downloaded through a series of different download options.

A LITSE `R` package can be developed to enable offline users the same access. As parts of the LITSE implementation operate only under Linux, a Windows version will require migrating this code.

Appendix A

Notation

Type	Appearance	Example
scalar variable	upper case, non-bold	Z
scalar variable Value	lower case, non-bold	z
scalar constant	lower case, non-bold	c
matrix constant	upper case, non-bold	S
vector variable	upper case, bold	$\mathbf{X}$
vector variable Value	lower case, bold	$\mathbf{x}$
matrix element value	upper case, non-bold, subscripted	X_{ij} (ijth entry)
matrix column	upper case, non-bold, subscripted	$X_{\cdot j}$ (jth column)
matrix row	upper case, non-bold, subscripted	$X_{i\cdot}$ (ith row)

Table A.1: Variable and Constant Naming Conventions.

Symbol	Definition
$a := b$	a defined as b
$a \triangleq b$	value of a set to b
$\text{sign}(x)$	sign function $\text{sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & o.w. \end{cases}$
$\mathbb{1}(\cdot)$	indicator function

Table A.2: Operators and Functions.

Symbol	Description	Reference
N_{samp}	number of samples/observations	page 33
N_l	number of loci	page 33
N_{haps}	number of potential haplotypes	page 33
$\ell(l)$	number of alleles at locus l	page 33
S^l	signature matrix for locus l	page 38

Table A.3: Key Constants.

Symbol	Description	Reference
Z	MOI	page 33

Table A.4: Key Variables.

See also Table 2.3 on page 52.

Symbol	Description	Reference
$\mathbf{p}$	parameter vector for multinomial Z	page 33
π	haplotype frequencies	page 34
c^l	concentration parameter for locus l	page 43

Table A.5: Key Parameters.

See also Table 2.4 on page 53.

Bibliography

[1] Milton Abramowitz and Irene Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, 10 edition, 1972.

[2] John Aitchinson. A concise guide to compositional data analysis. `http://www.leg.ufpr.br/lib/exe/fetch.php/pessoais:abtmartins:a_concise_guide_to_compositional_data_analysis.pdf`.

[3] John Aitchinson and Jim Kay. Possible solutions of some essential zero problems in compositional data analysis. `http://ima.udg.es/Activitats/CoDaWork03/paper_Aitchison_and_Kay.pdf`.

[4] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4): 77–84, 2012.

[5] Center for Disease Control. URL `www.cdc.gov`. Accessed: February 2015.

[6] Paul Chaffee, Inna Gerlovina, Bryan Greenhouse, Curt Hansen, and Hana Lee. Inference of malarial haplotypes. Stat245D class project, May 2009.

[7] Sandra J. Cheesman, Jacobus C. de Roode, Andrew F. Read, and Richard Carter. Real-time quantitative PCR for analysis of genetically mixed infections of malaria parasites: technique validation and applications. *Molecular & Biochemical Parasitology*, 131:83–91, 2003. doi:10.1016/S0166-6851(03)00195-6.

[8] Andrew G. Clark. Inference of haplotypes from pcr-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7:111–122, 1990.

[9] Gabor Csardi. igraph The network analysis package. http://igraph.org. Accessed: April 2015.

[10] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39 (1):1–38, 1977.

[11] Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. ISBN 978-1-4614-6867-7.

[12] Dirk Eddelbuettel and Romain Franccois. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL http://www.jstatsoft.org/v40/i08/.

[13] Laurent Excoffier and Montgomery Slatkin. Maximum-Likelihood Estimation of Molecular Haplotype Frequencies in a Diploid Population. *Molecular Biology and Evolution*, 12:921–927, 1995. doi:0737-4038/95/1205-0021.

[14] Malcolm Gardner, Owen White, Matthew Berriman, Richard W Hyman, Jane M Carlton, Arnab Pain, and Karen E Nelson. Genome sequence of the human malaria parasite Plasmodium falciparum. *Nature*, 10 2002. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3836256/.

[15] Bryan Greenhouse. Reconstructing multilocus genotypes from malaria parasites in mixed infections. Presentation, February 2013.

[16] John Harris, Jeffry Hirst, and Michael Mossinghof. *Combinatorics and Graph Theory*. Springer, 2008.

[17] Ian Hastings. Malhaplofreq: A computer program for estimating malaria allele and haplotype frequences from their prevalences in blood samples using Maximum Likelihood methodology. Notes Version 1.1 for Programme Version 1.1.1.

[18] Ian Hastings and Thomas Smith. Malhaplofreq: A computer programme for estimating malaria haplotype frequencies from blood samples. *Malaria Journal*, 7, 2 2008. doi:10.1186/1475-2875-7-130.

[19] Ian M. Hastings, Christian Nsanzabana, and Tom A. Smith. A Comparison of Methods to Detect and Quantify the Markers of Antimalarial Drug Resistance. *American Journal of Tropical Medicine and Hygienes*, 83(3):489–495, 2010. doi:10.4269/ajtmh.2010.10-0072.

[20] Paul Hunt, Richard Fawcett, Richard Carter, and David Walliker. Estimating snp proportions in populations of malaria parasites by sequencing: Validation and applications. *Molecular & Biochemical Parasitology*, 143:173–182, 2005. doi:10.1016/j.molbiopara.2005.05.014.

[21] Darren Kessner, Tom Turner, and John Novembre. Maximum Likelihood Estimation of Frequencies of Known Haplotypes from Pooled Sequence Data.

[22] Bonnie Kirkpatrick. Estimating haplotype frequencies from genotypes of pooled dna. Technical report, University of California, Berkeley, 2007. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-180.html`.

[23] Charles Lawson and Richard Hanson. *Solving Least Squares Problems*. Classics in Applied Mathematics. SIAM, 1995.

[24] Xiaohong Li, Andrea Foulkes, Recali Yucel, and Stephen Rich. An expectation maximization approach to estimate malaria haplotype frequencies in multiply affected children. *Statistical Applications in Genetics and Molecular Biology*, 7, 2007. doi:10.2202/1544-6115.1321.

[25] Jiming Liu, Bo Yang, William Cheung, and Guojing Yang. Malaria transmission modeling: a network perspective. *Infectious Diseases of Poverty*, 1(11), 2012.

[26] Quan Long, Daniel C. Jeffares, Qingrun Zhang, Kai Ye, Viktoria Nizhynska, Zemin Ning, Chris Tyler-Smith, and Magnus Nordborg. PoolHap: Inferring Haplotype Frequencies from Pooled Samples by Next Generation Sequencing. *PLoS One*, 5 (12), 2011. doi:10.1371/journal.pone.0015292.

[27] Wolfram MathWorld. Ternary diagram. `http://mathworld.wolfram.com/TernaryDiagram.html`.

[28] Matti Pirinen, Sangita Kulathinal, Dario Gasbarra, and Mikko J. Sillanpää. Estimating population haplotype frequencies from pooled DNA samples using PHASE algorithm. *Genetics Research*, 90:509–524, 2008. doi:10.1017/S0016672308009877.

[29] Plasmodium vivax Assembly and Gene Annotation. URL `http://protists.ensembl.org/Plasmodium_vivax/Info/Annotation/`. Accessed: April 2015.

[30] Mattia Prosperi and Marco Salemi. QuRe: software for viral species reconstruction from next-generation sequencing data. *BMC Bioinformatics*, 28(1), 2012. doi:10.1093/bioinformatics/btr627.

[31] Mattia Prosperi, Luciano Prosperi, Alessandro Bruselles, Isabella Abbate, Gabriella Rozera, Donatella Vincent, Maria Carmela Solmone, Maria Rosaria Capobianchi, and Giovanni Ulivi. Combinatorial analysis and algorithms for quasispecies reconstruction using next-generation sequencing. *BMC Bioinformatics*, 12(5), 2011. doi:10.1186/1471-2105-12-5.

[32] Shannon R.E. Quade, Robert C. Elston, and Katrina A.B. Goddard. Estimating haplotype frequencies in pooled DNA samples when there is genotyping error. *BMC Genetics*, 6(25), 2005. doi:10.1186/1471-2156-6-25.

[33] Olavi Renkonen. Statisch-ökologische Untersuchungen über die terrestrische Käferwelt der finnischen Bruchmoore. *Annales Zoologici Societatis Zoologicae-Botanicae Fennicae 'Vanamo'*, 6:1–231, 1938.

[34] RStudio, Inc. *shiny: Easy web applications in R*, 2015. URL: `http://shiny.rstudio.com`.

[35] Conrad Sanderson. Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments. Technical report, 2010. URL `http://www.leg.ufpr.br/lib/exe/fetch.php/pessoais:abtmartins:a_concise_guide_to_compositional_data_analysis.pdf`. Technical Report.

[36] Ilya Shmulevich and Edward R. Dougherty. *Genomic Signal Processing*. Princeton University Press, 2007.

[37] Lydia Sinapova. Sorting Algorithms: QuickSort. `http://faculty.simpson.edu/lydia.sinapova/www/cmsc250/LN250_Weiss/L16-QuickSort.htm`.

[38] M Stephens and P. Donnelly. A comparison of Bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73:1162–1169, 2003.

[39] M Stephens and P. Scheet. Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. *American Journal of Human Genetics*, 76:449–462, 2005.

[40] M Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68: 978–989, 2001.

[41] Christopher Stover. Ternary diagram. `http://mathworld.wolfram.com/TernaryDiagram.html`. Accessed: Feb 2015.

[42] Shannon L. Takala, David L. Smith, O. Colin Stine, Drissa Coulibaly, Mahamadou A. Thera, Ogobara K. Doumbo, and Christopher V. Plowe. A high-throughput method for quantifying alleles and haplotypes of the malaria vaccine candidate Plasmodium falciparum merozoite surface protein-1 19 kDa. *Malaria Journal*, 5(31), 2006. doi:10.1186/1475-2875-5-31.

[43] Aimee Taylor, Jennifer Flegg, Samuel Nsobya, Adoke Yeka, Moses Kamya, Philip Rosenthal, Grant Dorsey, Carol Sibley, Philippe Guerin, and Chris Holmes. Estimation of malaria haplotype and genotype frequencies: a statistical approach to overcome the challenge associated with multiclonal infections. *Malaria Journal*, 13 (102), 2014. doi:http://www.malariajournal.com/content/13/1/102.

[44] Mark van der Laan and Katherine Pollard. A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. *Journal of Statistical Planning and Inference*, 117:275–303, 2003.

[45] Shirley S. Wang. A Warning from the Front Lines of Malaria Research. *Wall Street Journal*, February 10, 2015.

[46] DA Warrell and HM Gilles. *Essential Malariology*. Arnold, 4 edition, 2002.

[47] Sean Webster. Audio demixing with decorrelation, cross cancellation, normalization, and regularization. Online notes.

[48] World Health Organization. World Malaria Report, 2014. `http://www.who.int/malaria/publications/world_malaria_report_2014/en/`.

[49] Yaning Yang, Jingshan Zhang, Josephine Hoh, Fumihiko Matsuda, Peng Xu, Mark Lathrop, and Jurg Ott. Efficiency of single-nucleotide polymorphism haplotype estimation from pooled DNA. *PNAS*, 100(12):7225–7230, 2003. doi:10.1073–pnas.1237858100.

[50] Osvaldo Zagordi, Lukas Geyrhofer, Volker Roth, and Niko Beerenwinkel. Deep sequencing of a genetically heterogeneous sample: local haplotype reconstruction and read error correction. *Journal of Computational Biology*, 17(3):417–28, 2010.

[51] Osvaldo Zagordi, Arnab Bhattacharya, Nicholas Eriksson, and Niko Beerenwinkel. ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinformatics*, 12(119), 2011. doi:10.1186/1471-2105-12-119.