Y-STR: Haplotype Frequency Estimation and Evidence Calculation

Master of Science Thesis

Mikkel Meyer Andersen

June 2010

Department of Mathematical Sciences, Aalborg University,
Denmark

Department of Mathematical Sciences
Fredrik Bajers Vej 7G
9220 Aalborg East
Denmark
Phone: +45 99 40 88 04
http://www.math.aau.dk

Title: Y-STR: Haplotype Frequency Estimation and Evidence Calculation

Semester: MAT6, from February 1, 2010 to June 1, 2010

Author: Mikkel Meyer Andersen

Supervisor: Poul Svante Eriksen

Circulation: 6

Pages: 138

Summary: Y-STR haplotype frequency estimation is important because it is required in order to calculate evidence. The loci on the Y-chromosome cannot be assumed to be independent as with on the autosomal STR, so the simultaneous probability does not factor to the product of the marginal probabilities. This means that a statistical model incorporating proper dependence must be created.

First an existing method, the frequency surveying approach, is described, and afterwards new models are developed. The new models considered are a new method called ancestral awareness and models based on existing methods such as kernel smoothing and model based clustering. Also a class of models, classification models, are developed. Examples of such models are classification trees, support vector machines, and ordered logistic regression.

Methods to assess the performance of the methods are developed and afterwards used to compare the models. It is found that classification trees is a good model, but it has the disadvantage of not using the prior knowledge such as the single step mutation model.

Besides frequency estimation, evidence calculations is also considered in this thesis.

Preface

Outline

Chapter 1 describes how Y-STR differs from autosomal STR on both the biological and statistical level. On the biological level, Y-STR has the clear advantage over autosomal STR that only male DNA is detected. This makes Y-STR applicable in e.g. rape cases where the large amounts of victim DNA does not pollute the smaller amounts of offender DNA when using Y-STR. On the statistical level, Y-STR differs from autosomal STR in the sense that the loci are not independent. The causes the frequency and evidence estimation to be non-trivial.

Chapter 2 analyses the data in an exploratory manner using factor analysis and principal component analysis. These methods do not give rise to usable dimension reduction, hence they are not pursued further in the rest of this thesis.

Chapter 3 describes several models to estimate Y-STR haplotype frequencies. In the beginning of this chapter, some desirable properties of such models are stated, e.g. model consistency. Then the frequency surveying approach from [Roewer et al., 2000] are described critically and some problems with the approach are pointed out. Afterwards it is argued that the the framework of graphical models is not sufficient if using Pearson's χ^2 to test for conditional independence, so other test of conditional independence are needed. In the last part of the chapter, three new models and a class of models are developed. The class of models are classification models where classification trees, support vec-

tor machines, and ordered logistic regression are examples of instances of this class. The three models developed are ancestral awareness, kernel smoothing, and model based clustering.

Chapter 4 describes methods for comparing the models. It is also argued that it is difficult to perform proper model control if the method is not a statistical model. The comparison methods used are how much unobserved probability mass the models predict based on [Robbins, 1968], comparing single and pairwise marginals using deviance, and comparing predicted and relative frequencies using deviance. The estimator introduced in [Robbins, 1968] are assessed through a simulation study.

Chapter 5 contains the results of the comparisons. The focus is on classification models because they exhibit such nice theoretical structure, and the marginals are investigated for the classification models only. The classification trees seem to be the best model, although they do not use a priori knowledge such as the single step mutation model. The support vector machines perform bad, but it is most likely due to a wrong kernel and parameters.

Chapter 6 describes how to calculate evidence and gives a couple of examples within this area as well. The articles [Wolf et al., 2005] and [Brenner, 2010] are considered. It is emphasised why it is so crucial to be able to estimate Y-STR haplotype frequencies.

Chapter 7 recapitulates the thesis and proposes several areas for further work.

Acknowledgements

This thesis is intended for readers having statistical knowledge corresponding to master's level. The forensic terms will to a great extent be explained when they are used for the first time. Only little prior biological knowledge is expected.

I wish to thank and express my deepest gratitude to my supervisor, Poul Svante Eriksen, associate professor at the Department of Mathematical Sciences at Aalborg University, Denmark, for providing invaluable feedback and always being very helpful and ready to discuss all sorts of issues and ideas. It has been a pleasure and truely rewarding to work with such a friendly and accomplished person.

I also wish to thank Niels Morling, director and head of The Department of Forensic Medicine, University of Copenhagen, for providing data and unravelling issues regarding forensics.

Appreciation also goes to Torben Tvedebrink, PhD student at the Department of Mathematical Sciences at Aalborg University, Denmark, for always being helpful, willing to explain all sorts of stuff, and contribute with valuable ideas during discussions.

A part of this thesis has been to attend and give a talk at 7^{th} International Y Chromosome User Workshop in Berlin, Germany, from April 22 to April 24, 2010. The slides for the talk is available with the supplementary material of this thesis (the location can be found in section 1.5). Thanks to both Svante and Torben for helping me prepare the talk.

In this thesis the statistical package [R Development Core Team, 2010] has been used. It will be referred to simply as `R`. Thanks to all the contributors for making this great software.

Lastly, I want to thank the Oticon Foundation, Denmark, for supporting this thesis with a scholarship.

Mikkel Meyer Andersen,
`mikl@math.aau.dk`

Resumé på Dansk (Abstract in Danish)

Autosomalt STR har gennem flere år været anvendt til DNA-analyser. Y-STR, STR foretaget på Y-kromosomet (som kun mænd har), er dog i visse sammenhænge på grund af nogle fordele i forhold til autosomalt STR blevet mere og mere populært de seneste år. Det kan eksempelvis være fordelagtigt at anvende Y-STR i voldtægtssager, hvor mængden af forbryder-DNA (fra mænd) er markant mindre end mængden af offer-DNA (fra kvinden). Ved at anvende Y-STR vil overrepræsentationen af offerets DNA ikke forstyrre de små mængder forbryder DNA.

Hvor en autosomalt STR DNA-type består af loci, der statistisk er uafhængige, er loci i en Y-STR DNA-type statistisk afhængige. Dette giver anledning til udfordringer, blandt andet i forbindelse med at estimere frekvensen af en Y-STR DNA-type i en befolkning, hvilket blandt andet anvendes til udregning af bevismæssig vægt under retssager. Intuitivt kan det motiveres ved følgende eksempel: hvis Y-STR DNA-typen fundet på et gerningssted stemmer overens med den mistænktes, er den mistænkte så skyldig? Hvis hver anden mand har den pågældende Y-STR DNA-type er det ikke stærkt bevis, men hvis derimod kun 1 ud af 10 milliarder mænd har Y-STR DNA-typen, er det derimod stærkt bevis. Derfor er det vigtigt at kunne estimere Y-STR DNA-typers frekvenser.

I tilfældet med autosomalt STR udnyttes den statistiske uafhængighed blandt loci, så den simultane sandsynlighed blot bliver produktet af marginalerne. For Y-STR kan dette ikke gøres, da loci ikke er uafhængige. Derfor skal der udvikles metoder til at estimere sandsynlighederne for Y-STR DNA-typer – også de Y-STR DNA-typer, man endnu ikke har observeret.

Hovedvægten af dette speciale er udvikling af sådanne metoder, hvilket sker i kapitel 3, med efterfølgende udvikling af metoder til at sammenligne modeller i kapitel 4 og resultaterne i kapitel 5.

Først kommer en kritisk gennemgang af metoden introduceret i [Roewer et al., 2000]. Herefter udvikles tre andre metoder og en klasse af metoder. De tre metoder er fælles forfader-metoden (afsnit 3.4), kerneudglatning (afsnit 3.6) og modelbaseret klyngedannelse (afsnit 3.7). Klassen af modeller er klassifikationsmodeller (afsnit 3.5) med hovedvægt på klassifikationstræer (afsnit 3.5.1).

Der afsluttes med introduktion til en teori omkring udregning af bevismæssig vægt (ofte blot kaldt *likelihood ratio* både på dansk og engelsk) i kapitel 6 baseret på bl.a. [Wolf et al., 2005].

I almindelighed giver klassifikationsmodellerne anledning til teoretiske attraktive modeller, der bl.a. muliggør effektiv og alsidig modelkontrol (bl.a. fordi man let kan simulere haplotyper efter deres sandsynlighed under modellen) og hvis struktur muligvis kan udnyttes til en mere effektiv udregning af bevismæssig vægt. Klassifikationstræer ser ud til at være et godt bud på et eksempel af klassifikationsmodelklassen med hensyn til enkelthed, ydelse og hastighed. En af manglerne ved klassifikationstræerne (og andre af klassifikationsmodellerne) er dog, at de ikke inkorporerer a priori viden om emnet (eksempelvis i form af *single step mutation*-modellen).

Som en del af afrundingen i kapitel 7 beskrives mulige emner, man kan arbejde videre med. Et af disse emner er, hvordan kvantitativt data kan anvendes til analyse af Y-STR miksturer. Et andet emne er at konstruere en statistisk model, der i højere grad aktivt anvender a priori viden; et forslag til en sådan er at udglatte kontingenstabeller ved at anvende *single step mutation*-modellen, og derefter anvende PC-algoritmen, hvor test for betinget uafhængighed eksempelvis kan foretages med Spearmans korrelationskoefficient for at udnytte ordningen i data i stedet for den traditionelle Pearsons χ^2-test for nominelle data.

Contents

1 Introduction **15**

1.1 Biological Framework 15

1.2 Motivation 17

1.3 Data 19

1.3.1 Subpopulations 20

1.4 Notation 20

1.5 Supplementary Material 21

2 Exploratory Data Analysis **23**

2.1 Principal Component Analysis 23

2.2 Factor Analysis 24

2.3 Remark 28

3 Estimating Haplotype Frequencies **33**

3.1 Desired Properties 33

3.1.1 Consistent Models 34

3.1.2 Requirements 34
3.2 Frequency Surveying 34
3.2.1 Bayesian Inference 35
3.2.2 Estimating Prior Parameters 36
3.2.3 Model Control 36
3.2.4 Verifying the Method 37
3.2.5 Comments on the Method 39
3.2.6 Improvement 44
3.2.7 Generalising the Method 44
3.3 Graphical Models 45
3.4 Ancestral Awareness 47
3.4.1 How to Choose the Ancestral Set 48
3.4.2 Fixed Size of the Ancestral Set 48
3.4.3 Threshold of the Proportion of Haplotypes Left Given the Ancestral Set 52
3.4.4 Mutation 53
3.5 Classification 53
3.5.1 Classification Trees 55
3.5.2 Ordered Logistic Regression 59
3.5.3 Support Vector Machines 64
3.6 Kernel Smoothing 70
3.7 Model-Based Clustering 71

4 Methods for Comparison of the Models 75

4.1 Observed Probability Mass 75
4.1.1 Verification Through Simulation 78
4.2 Marginals 84

4.2.1 Deviance 84
4.2.2 How to Find Marginals 86
4.2.3 Normalised Marginal Approximation 89
4.2.4 Bootstrapping 90
4.3 Deviance Comparing Predicted with Relative Frequencies . . 91
4.4 Multinomial Distribution 92

5 Results for Comparison of the Models 93
5.1 Implementation 93
5.1.1 Classification Models 94
5.1.2 Kernel Smoothing 96
5.2 Properties of the Datasets 96
5.3 Classification Models 97
5.3.1 Unobserved Probability Mass 97
5.3.2 Single Marginals 97
5.3.3 Pairwise Marginals 102
5.3.4 Assessing Validity of Normalising Predicted Marginals 103
5.3.5 Deviance Comparing Predicted with Relative Frequencies 106
5.4 Frequency Surveying 106
5.5 Ancestral Awareness 107
5.6 Kernel Smoothing 107
5.7 Model Based Clustering 108
5.8 Comparing Models 111

6 Evidence 115
6.1 Two Contributors 115
6.2 n Contributors 119

6.2.1 Approximation with Known Error Bound 122
6.3 The κ-model 124

7 Recapitulation **127**
7.1 Further Work 129
7.1.1 Statistical Model Incorporating Genetic Knowledge . . 129
7.1.2 Y-STR Mixtures 129
7.1.3 Subpopulations 129
7.1.4 Extended Models 130
7.1.5 Signal Processing 130

A Correlation Matrices **135**

CHAPTER 1

Introduction

This chapter introduces some fundamental terms, both biologically and statistically. In addition to this, practical things such as where to find supplementary material, e.g. R-code, is placed in this chapter.

First some basic biology will be explained, and afterwards a section about why the problems investigated in this thesis arise and why it is important to solve these problems.

1.1 Biological Framework

This thesis is set within the field of forensic statistics, more specifically the application of Y-STR, which is only a small branch of the area. This section will only explain the biological parts briefly, please see [Butler, 2005] or [Butler, 2009] for an elaborate reference in this area.

Y-STR has a "sibling" called (just) STR which is an abbreviation for short tandem repeat. STR is a standardised way to extract a DNA-profile from biological material such as blood. Often, like in most crime cases, the amount of DNA-material is limited, and if so, the DNA-material is first amplified using PCR (polymerase chain reaction). This is a complicated chemical process and gives rise to some interesting problems that can also be treated statisti-

cally, e.g. estimating allelic drop-in/-out rate and modelling artefacts of the PCR-process. One parameter to set in the PCR-process is how many cycles to use, because it has impact on the phenomena just mentioned. This topic is outside the scope of this thesis, and will not be treated further.

When the PCR-process is done, hopefully there is enough DNA-material to perform the next step. At predefined positions called the loci (one locus) on the 22 autosomes (autosomale chromosomes) pairs, i.e. the chromosomes not having anything to do with the gender, the number of times a pattern of adjacent nucleotides (the nucleotides are A, T, G, and C and they make up the basepairs A-T and G-C) – normally a sequence of four basepairs – repeats itself are counted. This number is called the allele. In principle the length of the pattern can vary from two and upwards, but often a length of four is used. To clarify, a locus is one certain position on a chromosome. For example, if the DNA-sequence on a locus is

$$\underbrace{\begin{array}{cccc} A & G & T & T \\ | & | & | & | \\ T & C & A & A \end{array}}_{\text{pattern}} \underbrace{\begin{array}{cccc} A & G & T & T \\ | & | & | & | \\ T & C & A & A \end{array}}_{\text{pattern}} \underbrace{\begin{array}{cccc} A & G & T & T \\ | & | & | & | \\ T & C & A & A \end{array}}_{\text{pattern}}$$

the allele on this locus is 3. A lot of the biology on how this works in practice is omitted. Please refer to the literature given in the beginning of this section for further details.

Because each autosomal chromosome pair consists of two chromosomes, there are two alleles for each locus. If these alleles are the same, the person is said to be homozygot on that particular locus, and if the alleles are different the person is said to be heterozygot on that particular locus. Some of the problems arising are that an allele can drop-out such that a person looks like a homozygot on a locus, but is in fact a heterozygot. Similarly, noise – also amplified during the PCR-reaction – can cause drop-ins such that a homozygot person suddenly looks like a heterozygot. These problems are not treated in this thesis, but merely mentioned. Please refer to the literature given in the beginning of this section for details.

Y-STR is a bit different: instead of typing the autosomal chromosome pairs, only the Y-chromosome is typed. Remember that besides the 22 chromosome pairs, humans have two sex chromosomes. A female has two X-chromosomes and a male has one X-chromosome and one Y-chromosome. This means that Y-STR can only be done for males, but this can be used in some interesting situations. These will be discussed in section 1.2. Because only one of the chromosomes is typed, there is only one allele for each locus. Since only men have Y-chromosomes, it has also other advantages, because DNA from women does not interfere with Y-STR. More on this topic will come in section 1.2. Because there is only one allele at each locus, DNA-types based on Y-STR are often referred to as Y-STR haplotypes or simply haplotypes from

the Greek prefix "haplo" meaning one-fold or single.

The number of loci used is different depending on, among other things, which kit is used in the process and what (inter)national standards dictate. A minimal haplotype is often defined to contain 9 loci in order to be able to discriminate properly, and the number of loci is increasing as new loci are identified and included. An example of a 10 loci Y-STR haplotype is

$$\texttt{DYS19} = 13, \texttt{DYS389I} = 13, \texttt{DYS389II} = 29, \texttt{DYS390} = 22, \texttt{DYS391} = 10,$$
$$\texttt{DYS392} = 6, \texttt{DYS393} = 13, \texttt{DYS437} = 14, \texttt{DYS438} = 11, \texttt{DYS439} = 12$$

where DYS$*$ is the loci and the numbers are the alleles. Normally haplotypes are written

$$(13, 13, 29, 22, 10, 6, 13, 14, 11, 12)$$

such that it is defined that the first component (or dimension) is DYS19, the second DYS389I, and so on.

In the rest of this thesis, the focus will be on Y-STR unless otherwise mentioned.

1.2 Motivation

In crime cases where the victim is a female and the attackers are males, as for example with rapes, the amount of male DNA is limited compared to the amount of female DNA. This type of cases are difficult to analyse using traditional STR, but using Y-STR instead makes the analysis easier and more reliable. But it also introduces a need for slightly different statistical theory than developed for STR.

The general setup of the use of Y-STR statistics in a court room is similar to the one for autosomal STR. Often the hypothesis in a trial is

H_p : The suspect left the crime stain (prosecutor's hypothesis).

H_d : Some other person left the crime stain (defender's hypothesis).

or more generally,

H_p : The suspect left the crime stain together with n additional contributors.

H_d : Some other person left the crime stain together with n additional contributors.

If E is the evidence found at the crime scene, the likelihood ratio is

$$LR = \frac{P(E|H_p)}{P(E|H_d)}.$$

From here the similarities of evidential handling between Y-STR and autosomal STR end. A short and specific example for Y-STR will now be presented. Say that the evidence consists of the trace $T = (\{1,2\},\{2\})$ (simplified to point out the principle) and that the suspect's Y-STR haplotype is $\boldsymbol{h}_s = (1,2)$. Then if one additional contributor is assumed, the contributor's Y-STR haplotype has to be $\boldsymbol{h}_1 = (2,2)$ in order to explain the trace. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ and define

$$\begin{aligned}\boldsymbol{a} \oplus \boldsymbol{b} &= (\{a_1, b_1\}, \{a_2, b_2\}, \ldots, \{a_1, b_n\}) \\ T \ominus \boldsymbol{a} &= (\{b_1\}, \{b_2\}, \ldots, \{b_n\})\end{aligned}$$

such that $T = \boldsymbol{h}_s \oplus \boldsymbol{h}_1$ and $\boldsymbol{h}_1 = T \ominus \boldsymbol{h}_s$. Say that $(\boldsymbol{h}_1, \boldsymbol{h}_2)$ is consistent with the trace T if $\boldsymbol{h}_1 \oplus \boldsymbol{h}_2 = T$, which is denoted $(\boldsymbol{h}_1, \boldsymbol{h}_2) \equiv T$. In this example, $T = (\{1,2\},\{2\})$ is shorthand notation for $T = (\{1,2\},\{2,2\})$.

Then

$$\begin{aligned}LR &= \frac{P(E|H_p)}{P(E|H_d)} \\ &= \frac{P(\boldsymbol{h}_s, T \ominus \boldsymbol{h}_s)}{\sum_{(\boldsymbol{h}_1,\boldsymbol{h}_2)\equiv T} P(\boldsymbol{h}_s, \boldsymbol{h}_1, \boldsymbol{h}_2)} \\ &= \frac{P(\boldsymbol{h}_s)\, P(T \ominus \boldsymbol{h}_s)}{P(\boldsymbol{h}_s) \sum_{(\boldsymbol{h}_1,\boldsymbol{h}_2)\equiv T} P(\boldsymbol{h}_1)\, P(\boldsymbol{h}_2)} \\ &= \frac{P(T \ominus \boldsymbol{h}_s)}{\sum_{(\boldsymbol{h}_1,\boldsymbol{h}_2)\equiv T} P(\boldsymbol{h}_1)\, P(\boldsymbol{h}_2)} \\ &= \frac{P(\boldsymbol{h}_1 = (2,2))}{P(\boldsymbol{h}_1 = (1,2))\, P(\boldsymbol{h}_2 = (2,2)) + P(\boldsymbol{h}_1 = (2,2))\, P(\boldsymbol{h}_2 = (1,2))}\end{aligned}$$

by assuming that haplotypes are independent.

To calculate this LR a method of calculating haplotype probabilities is needed.

Because traditional STR uses loci spread out on the 22 autosomale chromosome pairs, it is generally accepted, see e.g. [Roewer et al., 2000], to assume statistical independence between the loci. This makes it quite easy to estimate the frequency of a DNA-type because the simultaneous probability is just the product of the marginal probabilities.

This is however not the case with Y-STR because the loci are on the same chromosome. So it is no longer appropriate to assume independence between the loci. This is easily shown by considering the correlation matrices for three datasets in appendix A. The datasets will be introduced in section 1.3. This immediately creates the problem of how to estimate Y-STR haplotype frequencies, which is used for e.g. calculating LR as mentioned.

1.3 Data

In this thesis, three datasets with Y-STR haplotypes have been used:

- *berlin* contains 7 loci haplotypes from citizens in Berlin, Germany (excluding DYS385a/b, please see below for explanation), provided by Lutz Roewer, lutz.roewer@charite.de

- *dane* contains 10 loci haplotypes from Danes (excluding DYS385a/b, please see below for explanation), published in [Hallenberg et al., 2004]

- *somali* contains 10 loci haplotypes from Somalis (excluding DYS385a/b, please see below for explanation), published in [Hallenberg et al., 2005]

Please refer to table 1.1 for number of observations, haplotypes, and singletons (haplotypes only observed once). In table 1.2, table 1.3, and table 1.4 a count table is presented for the datasets *berlin*, *dane*, and *somali*, respectively.

A common locus called DYS385a/b is present in all of the datasets. It is actually two loci, DYS385a and DYS385b, which is why it has two alleles instead of one. The problem with this locus (or these loci) is that it is not possible to distinguish between the two. So if DYS385a/b $= (11, 12)$ then either $(\texttt{DYS385a}, \texttt{DYS385b}) = (11, 12)$ or $(\texttt{DYS385a}, \texttt{DYS385b}) = (12, 11)$. This means that these loci cannot be treated as the other loci in the dataset. Because of this, DYS385a/b is often simply ignored like in [Roewer et al., 2000]. In this thesis, DYS385a/b is also ignored. One could argue that valuable information is thrown away, and maybe instead just use the sum of the alleles, but then the same sum does not necessarily correspond to the same alleles. So because no straightforward solution seems to solve the problem of identification, the locus (or loci) is ignored.

In *berlin*, five observations (row 551 to 555) have been removed because they had multiple alleles on one or several loci without any explanation of why.

In *dane* the allele 10.2 is observed only once in the entire dataset, so this is changed to 10 because neither *somali* nor *berlin* has such decimal alleles. The decimal means that the pattern did not repeat itself 11 times entirely, which is why it has been changed to 10.

The correlation matrices for *berlin*, *dane*, and *somali* can be found in appendix A. Here it is seen that there is a significant correlation between several pairs of loci.

	Observations	Haplotypes	Singletons
berlin	652	333	238
dane	185	136	112
somali	201	70	56

Table 1.1: Information about the datasets *berlin*, *dane*, and *somali*.

1	2	3	4	5	7	8	9	10	13	14	20	21	26	27
238	51	20	4	4	2	4	2	1	1	1	1	2	1	1

Table 1.2: How many times haplotypes have been observed in the dataset *berlin*. The first column indicates how many haplotypes have only been observed once. The second column indicates how many haplotypes have only been observed twice. And so on. Omitted columns means that no haplotype has been observed that particular number of times.

1.3.1 Subpopulations

Some haplotypes are more common in some parts of the world than in the rest; there is a geographical variation of which haplotypes are frequent. Actually there is quite a huge difference among different populations, see e.g. [Andersen, 2009a, p. 17] where a dendogram of pairwise AMOVA-distances (as defined in [Excoffier et al., 1992]) is presented.

In this thesis subpopulation effects are disregarded by using datasets from the same subpopulation. This is not necessarily a large limitation, because it is often – but not always – known which population an offender stems from.

1.4 Notation

It is assumed that the database is complete without missing data such that all the haplotypes consist of the same loci. If our haplotypes consists of loci $L_1, L_2, \ldots, L_r$ each taking values in $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_r$, respectively, then the set of all possible haplotypes is assumed to be

$$\mathcal{H} = \mathcal{L}_1 \times \mathcal{L}_2 \times \cdots \times \mathcal{L}_r,$$

where $\times$ denotes the Cartesian product.

An immediate consequence of this is that an allele has to be observed before we know it exists. So if we on `DYS19` has observed alleles $13, 14, \ldots, 17$, then in that particular dimension our set is those alleles only and it is not possible to estimate a probability of observing another allele.

Normally a database of haplotypes are represented as a matrix where each row is a haplotype and each column corresponds to a locus. When we have

1	**2**	**3**	**4**	**5**	**8**	**9**
112	15	4	1	2	1	1

Table 1.3: How many times haplotypes have been observed in the dataset ***dane***. Please refer to explanation in table 1.2.

1	**2**	**3**	**4**	**5**	**9**	**16**	**18**	**28**	**42**
56	4	2	1	1	2	1	1	1	1

Table 1.4: How many times haplotypes have been observed in the dataset ***somali***. Please refer to explanation in table 1.2.

a database of observations, the idea is basically the same, but it can happen that some males have the same haplotype. This can be dealt with either by adding the haplotype several times (one row for each person) such that some rows are equal. This is called extended representation. Another way is to add an extra column representing the number of times each haplotype has been observed. This is called compact representation. Note that the haplotype database corresponds to either deleting the column with observation count from the compact observation database or by removing duplicate rows in the extended observation database.

Assume that we have observed n haplotypes $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathcal{H}$. Now denote our observations $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{N_+} \in \mathcal{H}$ where $N_+ = \sum_{i=1}^{n} N_i$ and N_i denotes how many times the i'th haplotype $\boldsymbol{x}_i$ has been observed.

Let M be the number of singletons, i.e. the number of haplotypes observed only once.

1.5 Supplementary Material

Supplementary material like R-scripts (approximately 4000 lines in total) etc. can be found at

`http://people.math.aau.dk/~mikl/thesis`

where also links to [Andersen, 2009a] and its supplementary material together with [Andersen, 2009b] can be found.

A mirror has been set up at

`http://www.mikl.dk/math/thesis`

just in case the other one is unavailable.

CHAPTER 2

Exploratory Data Analysis

In this chapter an exploratory analysis of the data is made. This is done via dimension reduction because the data is high-dimensional. Two types of dimension reduction are introduced.

In section 2.1 principal component analysis is described, and in section 2.2 factor analysis is described. These areas will not get that much attention but are described briefly because they are two important techniques. Another technique of dimension reduction is principal Hessian directions, but this will not be described any further.

2.1 Principal Component Analysis

This section is based on [Venables and Ripley, 1997] and [Prendergast and Kabaila, 2009]. The latter is lecture notes where the multivariate analysis-part is based on [Johnson and Wichern, 2001].

PCA (principal component analysis) is a method to find linear combinations of the variables causing the greatest variance under the constraint that the length of the linear projection vector must be one.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)^\top$ be the variables with mean $\boldsymbol{\mu}$. PCA can either be

done on the covariance matrix or correlation matrix. Let

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p > 0$$

denote the eigenvalues of either matrix. Now consider p linear combinations of the original p variables where the i'th is given by

$$y_i = \boldsymbol{a}_i^\top \boldsymbol{x} = \sum_{j=1}^{p} a_{ij} x_j$$

where $\|\boldsymbol{a}_i\| = 1$ for all i.

The idea is to choose the $\boldsymbol{a}_1$ such that y_1 has the greatest variance (this makes sense because $\|\boldsymbol{a}_i\| = 1$). Afterwards $\boldsymbol{a}_2$ should be chosen such that y_2 has the greatest variance and is uncorrelated with y_1. Continuing we get that $\boldsymbol{a}_i$ is chosen such that $\mathbf{Var}[y_i]$ is maximised and

$$\mathbf{Cov}[y_1, y_i] = \mathbf{Cov}[y_2, y_i] = \cdots = \mathbf{Cov}[y_{i-i}, y_i] = 0.$$

The y_i's are called the principal components.

It can be shown that $\boldsymbol{a}_i = \boldsymbol{e}_i$ where $\boldsymbol{e}_i$ is the normalised i'th eigenvector with the corresponding eigenvalue λ_i. Note that the covariance and correlation matrix are both square and symmetric with real entries, so they are self-adjoint, hence the eigenvalues are real and the eigenvectors are orthogonal.

When the covariance or correlation matrix is estimated from data, then $\boldsymbol{e}_i$ is estimated by $\hat{\boldsymbol{e}}_i$, and the i'th principal component y_i is estimated by the sample principal component, denoted by $\hat{y}_i$ or SPC_i.

A scree plot is often used to decide how many principal components to use. A scree plot can either be based on the eigenvalues or how much variance of the total variance each component contributes with.

The scree plots for the PCA on ***berlin*** can be seen in figure 2.1, for ***dane*** in figure 2.2, and for ***somali*** in figure 2.3.

Plots of the first SPC (sample principal component) against the second can for ***berlin*** be seen in figure 2.4, for ***dane*** in figure 2.5, and for ***somali*** in figure 2.6.

Especially ***dane*** looks interesting with the three groups. On the other hand, the PCA on ***berlin*** and ***somali*** do not seem relevant.

2.2 Factor Analysis

This section is based on [Venables and Ripley, 1997, section 13.5] and the help for the `factanal`-function in `R`.

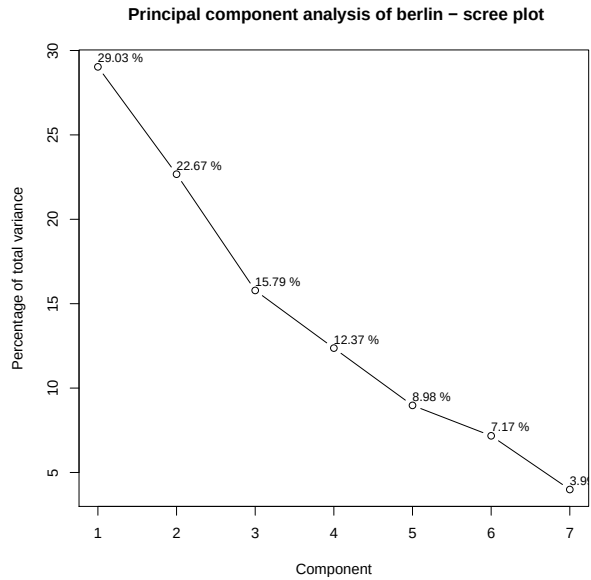

Figure 2.1: Scree plot of the principal component analysis of *berlin*.

Just like PCA (principal component analysis), FA (factor analysis) finds linear combinations of the data. These linear combinations are called factors. The two methods, PCA and FA, resemble each other, but they are build on different statistical models. Another difference is that in FA the desired number of factors must be specified in advance, whereas in PCA one often uses scree plots after analysis has been performed to select a certain – reduced – number of variables to use. An explanatory approach can avoid specifying the exact number of factors in advance and instead start by choosing to have one factor and test for sufficiency. If sufficiency is rejected, then use two factors and so on.

The factors often correspond to some underlying properties, e.g. is FA used in psychology where studies try to determine underlying factors such as intelligence (and often several other factors) by a number of variables. FA can for example be used to analyse surveys designed to identify certain properties of persons.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)^\top$ be the variables with covariance matrix Σ. For $k < p$ factors $\boldsymbol{f} = (f_1, f_2, \ldots, f_k)^\top$, the FA-model assumes that

$$\boldsymbol{x} = \Lambda \boldsymbol{f} + \boldsymbol{e},$$

where Λ is a $k \times p$-matrix of factor loadings, the components of $\boldsymbol{f}$ have variance one and are uncorrelated, and $\boldsymbol{e}$ are independent with unknown

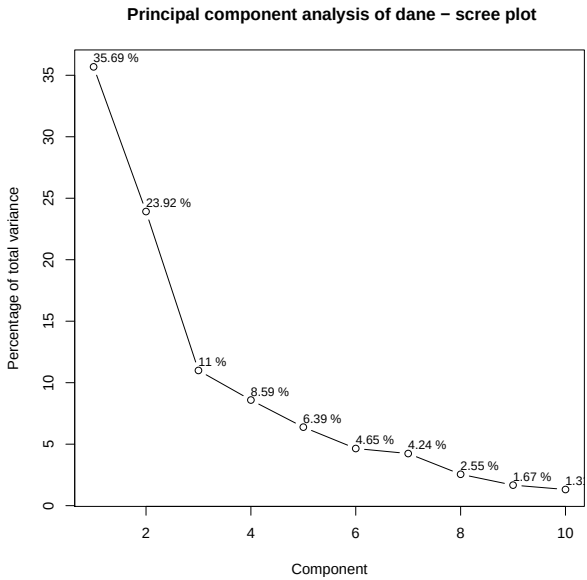

Figure 2.2: Scree plot of the principal component analysis of ***dane***.

diagonal covariance matrix Ψ. The variances of the components in $\boldsymbol{e}$ is called uniquenesses.

The conditions stated are equivalent to assuming that the covariance matrix Σ can be expressed as

$$\Sigma = \Lambda\Lambda^\top + \Psi.$$

If G is an arbitrary orthogonal $k \times k$ matrix, then the factors $G^\top \boldsymbol{f}$ with loadings matrix ΛG does not change the model for Σ, because

$$(\Lambda G)(\Lambda G)^\top + \Psi = \Lambda G G^\top \Lambda^\top + \Psi = \Lambda I \Lambda^\top + \Psi = \Sigma.$$

Choosing a G corresponds to choosing a basis for the subspace $\Lambda\boldsymbol{f}$, and is referred to as rotation. Choosing a rotation is often done in order to interpret the data in the best possible way.

The matrix G does not necessarily have to be orthogonal, but then the model changes and so does the interpretation. As an example, the rotation `promax` in the `factanal`-function in `R` can cause the factors to be correlated (this may be sensible in the case where factors is a person's different skills, but in general one has to be careful). Interpretation is often done by looking at the loadings matrix. The default rotation in the `factanal`-function in `R` is `varimax`.

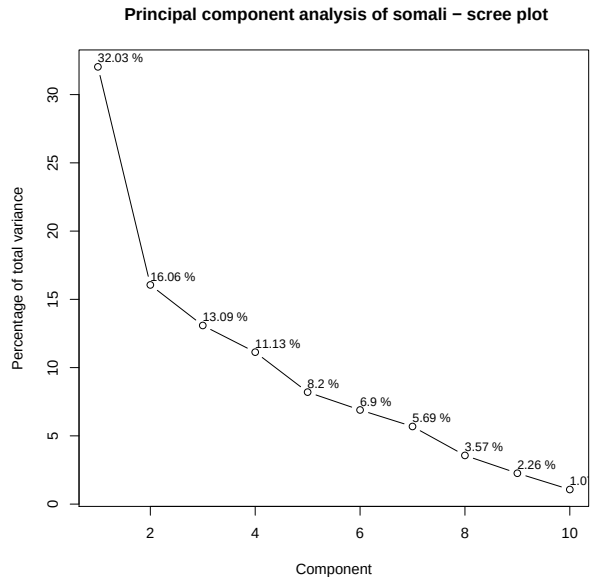

Figure 2.3: Scree plot of the principal component analysis of *somali*.

The degrees of freedom in the model is in [Venables and Ripley, 1997, p. 405] found to be

$$s = \frac{(p-k)^2 - (p+k)}{2}$$

In order to ensure that only one solution to the problem exists, we need to have $s > 0$, or equivalently

$$(p-k)^2 > p + k.$$

Solving this quadratic equation shows that given data dimension p, a certain maximum of factors

$$k_{max} = \left\lfloor \frac{1}{2} + p - \frac{\sqrt{1+8p}}{2} \right\rfloor \tag{2.1}$$

can be chosen in order to ensure uniqueness, where $\lfloor \cdot \rfloor$ is the operation taking the closest integer smaller than or equal to the expression.

For each of the datasets, a FA is tried with a number of factors from 1 to the equation given in (2.1). The `factanal`-function computes a p-value of the null hypothesis that the given number of factors is sufficient. Because the factor analysis here is used as exploratory, a model is automatically rejected if the p-value is smaller than 0.05. A model is also automatically rejected if the degrees of freedom equals 0.

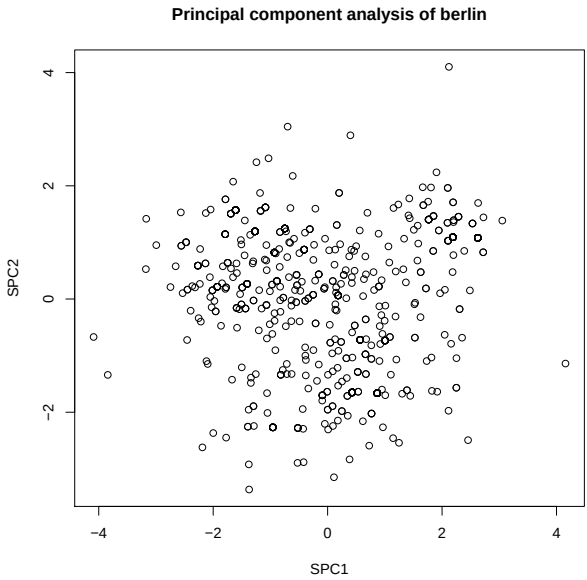

Figure 2.4: SPC1 vs. SPC2 of the principal component analysis of *berlin*.

The result of this exploratory approach is that no model can be fitted for *berlin*. Two models with 4 and 5 factors can be fitted for *dane*. Only one model with 5 factors can be fitted for *somali*.

Factor 1 vs. factor 2 is plotted for the different fits. For *dane*, these plots can be seen in figure 2.7 for 4 factors and figure 2.8 for 5 factors. For *somali*, the plot for 5 factors is in figure 2.9.

2.3 Remark

Because of the poor performance of the methods on the data, they will not be used further.

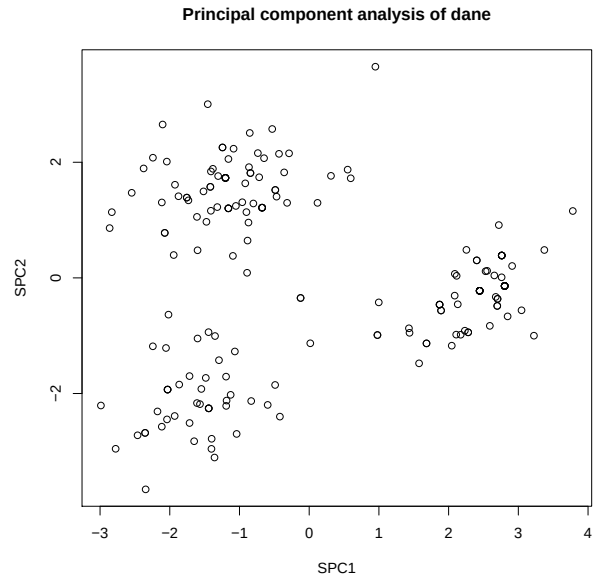

Figure 2.5: SPC1 vs. SPC2 of the principal component analysis of *dane*.

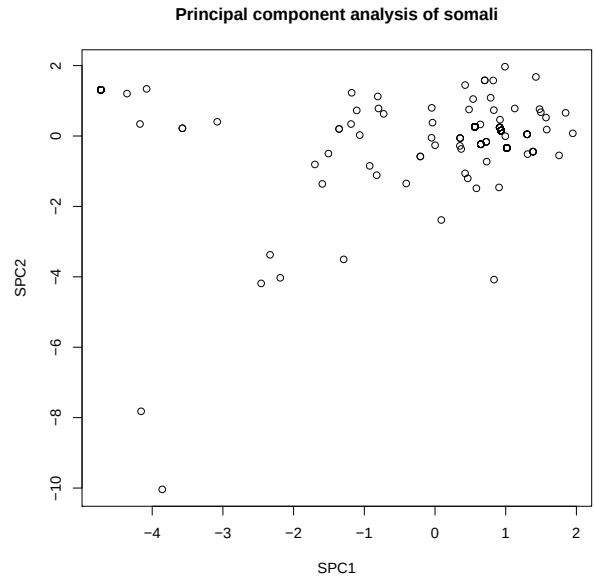

Figure 2.6: SPC1 vs. SPC2 of the principal component analysis of *somali*.

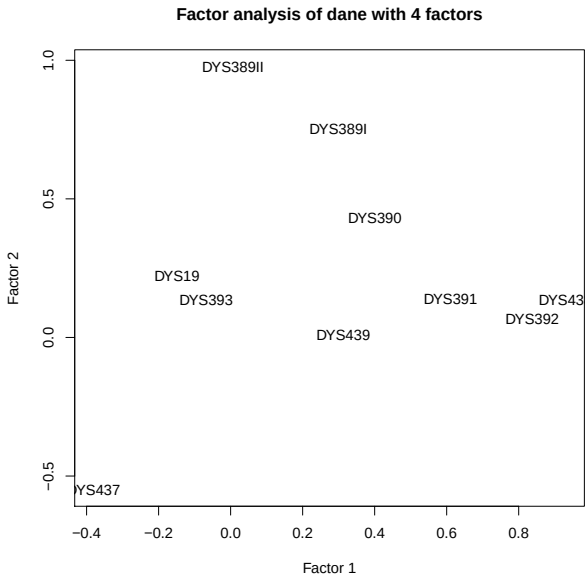

Figure 2.7: Factor analysis of ***dane*** with 4 factors

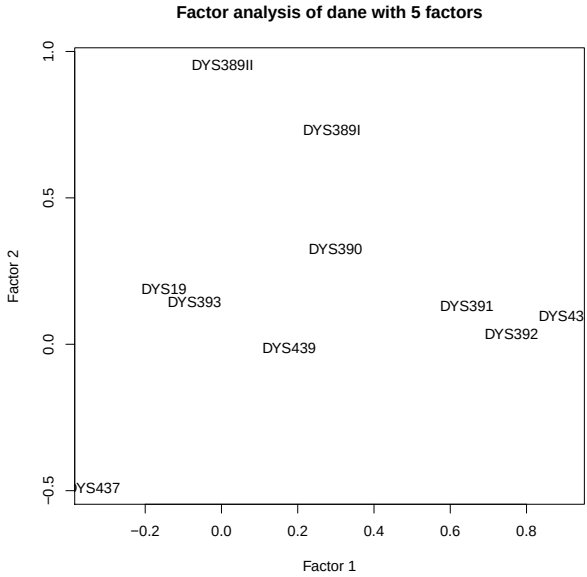

Figure 2.8: Factor analysis of ***dane*** with 4 factors

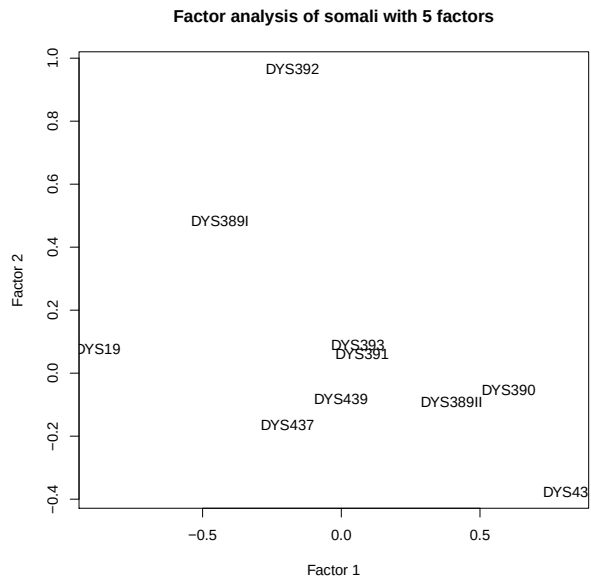

Figure 2.9: Factor analysis of *somali* with 5 factors

CHAPTER 3

Estimating Haplotype Frequencies

In this chapter different methods of how to estimate haplotype frequencies, including unobserved haplotypes, will be presented.

The focus will be on haplotypes from the same subpopulation, because then the subpopulation effect does not have to be taken into account. See section 1.3.1 for further info on subpopulations.

Being able to estimate haplotype frequencies is crucial in several cases, e.g. when calculating evidence which was shortly introduced in section 1.2 and will be further investigated in chapter 6.

Because a haplotype as such can be viewed as a multinomial sample, it would be tempting to use a multinomial model. But by doing so, a lot of information about the ordering between variables is thrown away because the multinomial distribution does not consider how the variables are ordered. Because of this, a multinomial model approach is not further pursued.

3.1 Desired Properties

Before starting to develop models, some desired properties of such a model are described.

3.1.1 Consistent Models

Models for estimating haplotype frequencies can be divided into two groups: consistent and inconsistent models.

Let $\mathcal{H}$ be a set of different haplotypes (refer to section 1.4 for the definition of $\mathcal{H}$). Given a model $\mathcal{M}(\mathcal{H})$ and a haplotype $h \in \mathcal{H}$, denote by $P_{\mathcal{M}(\mathcal{H})}(h)$ the probability of a haplotype calculated under the model $\mathcal{M}(\mathcal{H})$. Then a model $\mathcal{M}(\mathcal{H})$ is said to be consistent if

$$\sum_{h \in \mathcal{H}} P_{\mathcal{M}(\mathcal{H})}(h) = 1.$$

If a model is not consistent it is said to be inconsistent. Whether a model is consistent or not should be verified theoretically.

3.1.2 Requirements

Estimation should be based on a statistical model with the following properties:

1. The structure of the model should not depend on the data. This ensures that the structure is essentially the same no matter of the data. Only parameters to the model can change with the data.
2. The model should be consistent. Please refer to section 3.1.1 for details.

One way of doing this is to model the simultaneous probability mass function directly.

3.2 Frequency Surveying

In [Roewer et al., 2000] a new method for estimating haplotype frequencies is presented. First this method will be outlined, and afterwards some comments will be made.

The notation in [Roewer et al., 2000] is slightly different than the one used in this thesis. To be consistent with [Roewer et al., 2000], their notation will be adopted for this section only.

Let N be the total number of observations in the Y-STR database, M the number of different haplotypes, and N_i the number of times haplotype i is observed (noting that $i = 1, 2, \ldots, M$). So N correspond to N_+ and M to n in our notation.

3.2.1 Bayesian Inference

[Roewer et al., 2000] states (referring to a classical population genetics theory) that if f_i denotes the frequency of a haplotype, and it is considered as a random variable, then it can be assumed a priori that

$$f_i \sim \text{Beta}\,(u_i, v_i)\,,$$

i.e. f_i is Beta-distributed with first shape parameter u_i and second shape parameter v_i.

The parameters u_i and v_i depend on a number of unknown factors in the case with Y-STR (mutation rates, population size etc.), so [Roewer et al., 2000] estimates these parameters through an exponential regression model. This will be described in section 3.2.2, but first the general idea of [Roewer et al., 2000] will be described.

If we assume a likelihood model given by

$$N_i - 1 | f_i \sim \text{Binomial}\,(N-1, f_i)$$

where the -1 is from the fact that [Roewer et al., 2000] takes the first observation of a haplotype merely as an indication of existence. A posteriori we by Bayes Theorem then have that

$$\begin{aligned} P(f_i|N_i - 1) &= \frac{P(N_i - 1|f_i)P(f_i)}{P(N_i - 1)} \\ &= \frac{P(N_i - 1|f_i)P(f_i)}{\int_0^1 P(N_i - 1|f)P(f)\ \mathrm{d}f} \quad \text{(continuous Law of Total Probability)} \end{aligned}$$

for $f \sim \beta(u, v)$ for some u and v. It can be shown that

$$f_i|N_i - 1 \sim \text{Beta}\,(u_i + N_i - 1, v_i + N - N_i)$$

i.e. the posterior distribution of $f_i|N_i - 1$ is also Beta with parameters $u_i + N_i - 1$ and $v_i + N - N_i$.

In [Krawczak, 2001], the case where the requested haplotype has not been observed earlier, is discussed. The conclusion is to adjust the parameters in the posterior distribution slightly. This is caused by assuming the likeliheed $N_i|f_i \sim \text{Binomial}\,(N, f_i)$ such that $f_i|N_i \sim \text{Beta}\,(u_i + N_i, v_i + N - N_i)$ a posteriori.

If a suspect has a known haplotype i and has not been included in the database count, then this extra information should be used to adjust the posterior to $f_i|N_i \sim \text{Beta}\,(u_i + N_i + 1, v_i + N - N_i)$ as argued by [Krawczak, 2001].

3.2.2 Estimating Prior Parameters

To estimate u_i and v_i for the prior

$$f_i \sim \text{Beta}\,(u_i, v_i)\,,$$

let d_{ij} denote the molecular distance (the L^1 or Manhattan norm) between the i'th and j'th haplotype. This distance corresponds to the number of single step mutations necessary to get from one haplotype to another. First the weighted distance for the i'th haplotype is calculated as

$$W_i = \frac{1}{N} \sum_{i \neq j} \frac{N_j}{d_{ij}} \tag{3.1}$$

for $i = 1, 2, \ldots M$. Note that $\sum_{i \neq j} N_j = N - N_i \neq N$, but according to [Veldman, 2007], who also mentions this, the implementation in `http://www.yhrd.org` has been changed to use $\frac{1}{N-N_i}$ instead of $\frac{1}{N}$.

Now these W_i's are sorted by their numerical value and divided into $G = 15$ groups based on the value of W_i. For each group, the mean of the W_i's in the group is calculated. Also, for each group, the mean and variance of the $\hat{f}_i = \frac{N_i - 1}{N - M}$ corresponding to the W_i's in the group is calculated. Now two exponential regression models are fitted, namely

$$\mu(W) = \beta_1 + \exp\left(\beta_2 W + \beta_3\right) \quad \text{and} \quad \sigma^2(W) = \beta_4 + \exp\left(\beta_5 W + \beta_6\right).$$

This is easily done in `R` as shown in listing 3.1 using the non-linear least squares fitting implemented in the `nls`-function.

```
1 mu.model <- nls(mu ~ beta1 + exp(beta2 * W + beta3), start=list(beta1 =
      0.005, beta2 = 15, beta3 = -10), trace = TRUE)
```

Listing 3.1: Exponential regression in `R`

When the parameters have been fitted, a μ and σ^2 can be found for all M different haplotypes using W_i for $i = 1, 2, \ldots, M$. These μ_i and σ_i^2 can be transformed to the u_i and v_i parameters by using [Roewer et al., 2000, Equation (1), p. 35], namely

$$u_i = \frac{\mu_i^2(1 - \mu_i)}{\sigma_i^2} \quad \text{and} \quad v_i = u_i \left(\frac{1 - \mu_i}{\mu_i} \right).$$

Hence we know the distribution of f_i a priori.

3.2.3 Model Control

Given the database, let K_j denote the number of haplotypes observed $j + 1$ times, such that K_0 denotes the number of singletons etc.

Then [Roewer et al., 2000, p. 36] states that

$$\mathbf{E}\left[K_j\right] = \sum_{i=1}^{M} \int_0^1 P(N_i - 1 = j | f_j) \varphi(f_i) \, \mathrm{d}f$$

with $\varphi(f_i) = P(f_i)$ with parameters u_i and v_i.

To make this clear, let $\varphi_i(f)$ denote the density for a Beta distribution with parameters u_i and v_i evaluated in f. Then

$$\mathbf{E}\left[K_j\right] = \sum_{i=1}^{M} \int_0^1 P(N_i - 1 = j | f) \varphi_i(f) \, \mathrm{d}f.$$

3.2.4 Verifying the Method

Unfortunately the data used in [Roewer et al., 2000] is not available, so *berlin*, *dane*, and *somali* have been used to verify the method. The definition of W_i from (3.1) has been used.

In the process of implementing the described method, it became clear that the exponential regression models

$$\mu(W) = \beta_1 + \exp\left(\beta_2 W + \beta_3\right) \quad \text{and} \quad \sigma^2(W) = \beta_4 + \exp\left(\beta_5 W + \beta_6\right) \tag{3.2}$$

carried some trouble along. With these datasets it happened that either $\beta_1 < 0$ or $\beta_4 < 0$, implying that some of the estimated μ and σ became negative. So the procedure to test this method is to first use the regression models from (3.2) (which might fail), and then afterwards with $\beta_1 = \beta_4 = 0$, e.g. only with parameters $\beta_2, \beta_3, \beta_5, \beta_6$:

1. Calculate W_i for all i
2. Divide the W_i's into 15 groups
3. Estimate μ and σ^2 for the f_i's in each group to build the regression model
4. Build suitable regression model
5. If $\beta_1 < 0$ or $\beta_4 < 0$, stop
6. Estimate a μ_i and σ_i^2 for each W_i using the regression models
7. If any $\mu_i < 0$ or $\sigma_i^2 < 0$, stop
8. Finalise the analysis

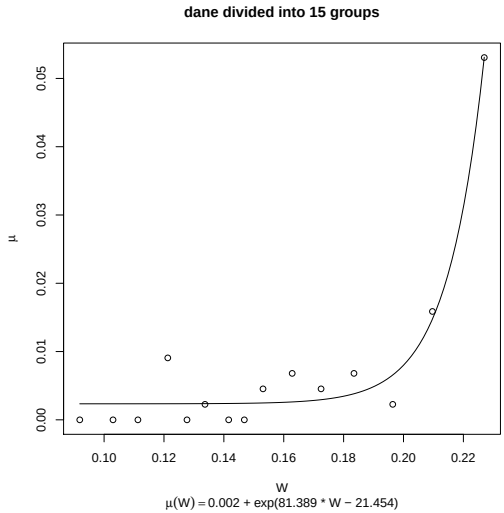

Figure 3.1: Estimated mean for ***dane*** using the model $\mu(W) = \beta_1 + \exp(\beta_2 W + \beta_3)$.

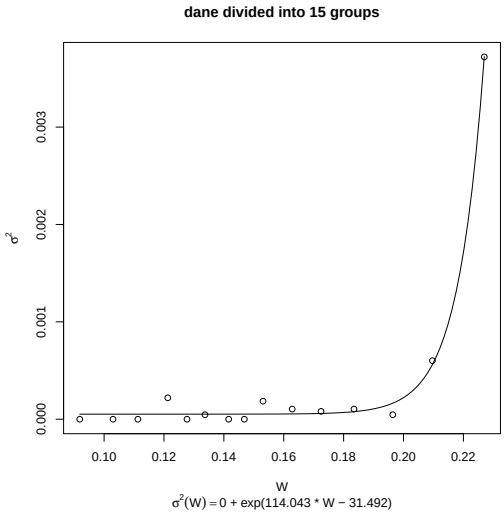

Figure 3.2: Estimated variance for ***dane*** using the model $\sigma^2(W) = \beta_4 + \exp(\beta_5 W + \beta_6)$.

	β_1	β_2	β_3	β_4	β_5	β_6
dane	0.002	81.389	-21.454	$5.194 \cdot 10^{-5}$	114	-31.49

Table 3.1: Parameters for the full regression model

	β_2	β_3	β_5	β_6
berlin	34.442	-12.972	21.217	-13.662
dane	66.503	-18.039	107.286	-29.945
somali	14.328	-9.286	10.290	-9.328

Table 3.2: Parameters for the reduced regression model

This procedure resulted in the conclusion that only *dane* could be fitted using the regression model stated in (3.2). And the fit does not seem comforting, refer to figure 3.1 and figure 3.2 for the fitted models for μ and σ^2, respectively.

When setting $\beta_1 = \beta_4 = 0$, all the datasets could make a fit, and the results are almost similar, but *berlin* made the best fit, so only the plots for *berlin* in figure 3.3 and figure 3.4 for μ and σ^2, respectively, is included. It looks okay for μ and – without exaggerating – less okay for σ^2. The other plots are automatically generated by running the R-code `surveying.R` in the supplementary material (refer to section 1.5), and it is encouraged to have a look at them.

The parameters for the full regression model and the reduced regression models can be found in table 3.1 and table 3.2, respectively.

In regards to model control, the expected counts versus the observed counts can for *dane* using the reduced regression model with $\beta_1 = \beta_4 = 0$ be found in figure 3.5, and for *berlin* using the full regression model in figure 3.6. Again, the other cases are quite similar and is automatically generated running the `surveying.R` as mentioned earlier.

3.2.5 Comments on the Method

In regards to model control, not much was done in the article. They only compared the observed and expected K_j's like described. For other ways of comparing models to estimate haplotype frequencies, refer to chapter 4, although a lot of these methods require a true statistical model which this is not. For the results of the relevant comparison methods introduced in chapter 4, see section 5.4.

In order to actually perform model checks, a posterior estimate for the haplotype frequency would often be needed. An evident choice for this would be

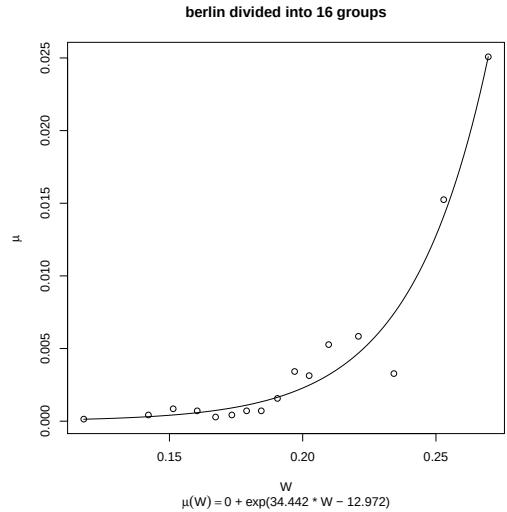

Figure 3.3: Estimated mean for ***berlin*** using the model $\mu(W) = \exp(\beta_2 W + \beta_3)$.

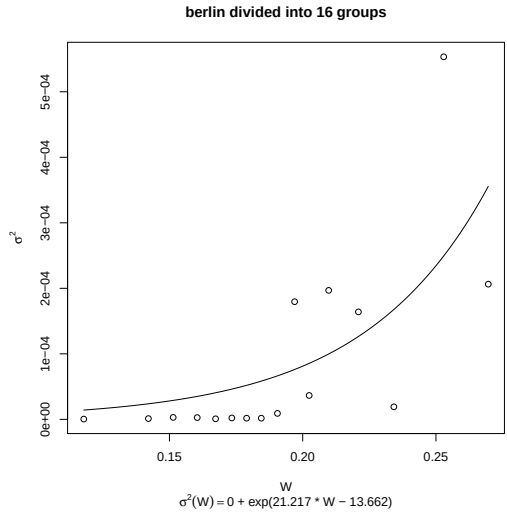

Figure 3.4: Estimated variance for ***berlin*** using the model $\sigma^2(W) = \exp(\beta_5 W + \beta_6)$.

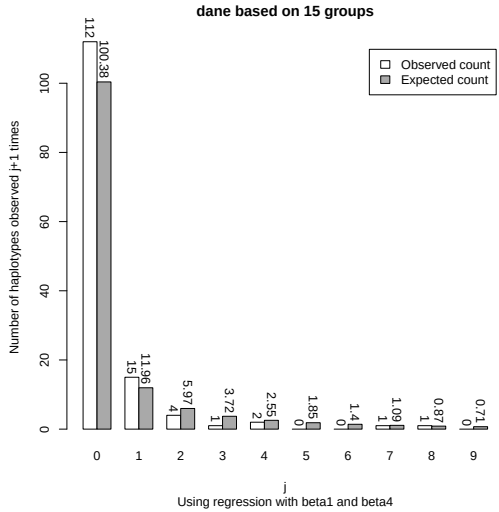

Figure 3.5: Expected vs. observed counts of singletons, doubletons etc. for ***dane*** using the reduced regression model $\mu(W) = \exp(\beta_2 W + \beta_3)$ and similar for $\sigma^2(W)$.

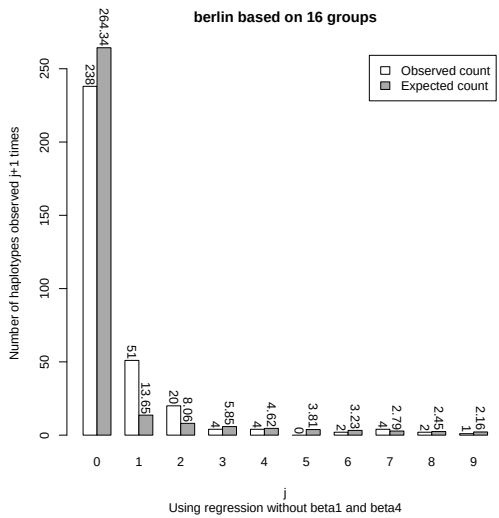

Figure 3.6: Expected vs. observed counts of singletons, doubletons etc. for ***berlin*** using the full regression model.

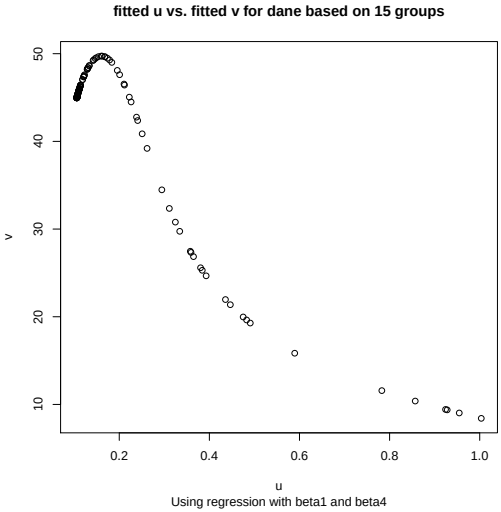

Figure 3.7: The connection between the estimated u's and v's using the full regression model. It is clear that the connection is very systematic.

the mean value for the posterior distribution

$$f_i|N_i - 1 \sim \text{Beta}\left(u_i + N_i - 1, v_i + N - N_i\right),$$

hence

$$\tilde{f}_i = \frac{u_i + N_i - 1}{u_i + N_i - 1 + v_i + N - N_i} = \frac{u_i + N_i - 1}{u_i + v_i + N - 1}$$

because the mean value for a $\text{Beta}(\alpha, \beta)$ is $\frac{\alpha}{\alpha+\beta}$.

One could also argue that the method is more like an ad-hoc method than a general statistical model. Especially because the estimation of μ and σ^2 based on W_i seems somewhat artificial and the model structure changes for different datasets. Also notice that the correlation structure of this approach is very complex. Furthermore, only one value, W_i, is used to estimate the parameters, u_i and v_i, so certain dependency must be expected, but this implicit assumption is not accounted for. This is quite clear from the regression formulas and is also obvious when plotting the u_i's against the v_i's as seen in figure 3.7 for ***dane*** using the full model and in figure 3.8 for the reduced model with $\beta_1 = \beta_4 = 0$. The other datasets give similar plots, which can be verified running the `R`-code `surveying.R` as mentioned earlier. Running this code generates the plots automatically.

Based on the facts just presented, the method must be said to be doubtful. A lot of the problems arise because the estimation is not based on a statistical

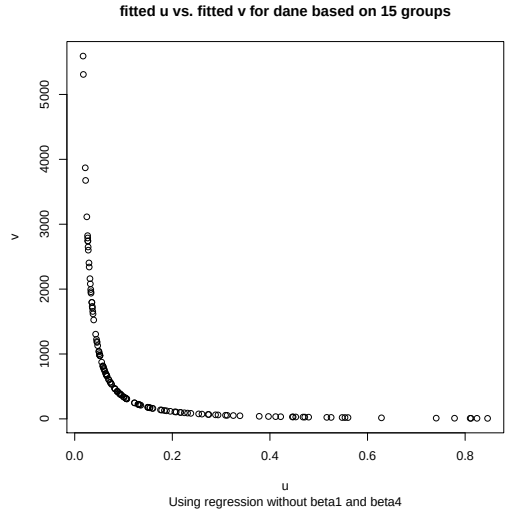

Figure 3.8: The connection between the estimated u's and v's using the regression model with $\beta_1 = \beta_4 = 0$. It is clear that the connection is very systematic.

model as described in section 3.1.2. On the other hand, the method should be acknowledged for incorporating generic prior knowledge.

Implementation Details of `http://www.yhrd.org`

At the 7$^{\text{th}}$ International Y Chromosome User Workshop in Berlin, Germany, April 2010, Sascha Willuweit (one of the persons behind `http://www.yhrd.org`) mentioned a couple of changes between their implementation at `http://www.yhrd.org` and the original article:

- They also only use the reduced regression models, i.e. without intercepts β_1 and β_4
- The number of groups are determined by fitting several regressions and choosing the one with the lowest coefficient of determination

Further details for selecting the minimum number of groups was however not mentioned.

3.2.6 Improvement

As mentioned in section 3.2.5, only the W_i has been used to fit two parameters. Using W_i is an obvious choice because it incorporates the first moment of the allele differences on each locus, but maybe it would be beneficial to also use the second moment. Recall that d_{ij} is the Manhattan distance, i.e.

$$d_{ij} = \sum_{k=1}^{r} d_{ijk} \quad \text{for} \quad d_{ijk} = |x_{ik} - x_{jk}|$$

where x_{ik} is the k'th locus at the i'th haplotype. Then

$$W_i = \frac{1}{N} \sum_{i \neq j} \frac{N_j}{d_{ij}} = \frac{1}{N} \sum_{i \neq j} \frac{N_j}{\sum_{k=1}^{r} d_{ijk}}.$$

Similar, one could introduce

$$Z_i = \frac{1}{N} \sum_{i \neq j} \frac{N_j}{\sum_{k=1}^{r} \left(d_{ijk} - \frac{d_{ij}}{r} \right)^2}$$

and then try to fit μ_i's and σ_i's by multiple regression using a grid of W_i and Z_i values. In order for this to work, a lot of observations are required, because the 15 groups of the W_i's only correspond to a 4×4-grid, which is way too coarse. If requiring 15 groups of each W_i and Z_i, the grid would have size $15 \cdot 15 = 225$, so in order to just get ten observations in each, at least 2250 observations are needed.

Because the requirement for the number of observations cannot be fulfilled for the datasets *berlin*, *dane*, nor *somali*, this improvement has not been tried in practice, although it would be interesting to see how it would perform compared to just using the W_i's.

3.2.7 Generalising the Method

Some of the problems mentioned might be solved by formulating the method a bit more generally.

Let $\boldsymbol{f} = (f_1, f_2, \ldots, f_k)$ be the vector of frequencies for all possible haplotypes, and use similar notation for other vectors.

Assume a priori, that

$$\boldsymbol{f} \sim \text{Dirichlet}\,(\boldsymbol{\alpha})$$

and that the likelihood is

$$\boldsymbol{N}|\boldsymbol{f} \sim \text{Multinomial}\,(N_+, \boldsymbol{f})\,.$$

Then the posterior becomes

$$\begin{aligned}\boldsymbol{f}|\boldsymbol{N} &\sim \text{Dirichlet}\left(\alpha_1 + N_1, \ldots, \alpha_k + N_k\right) \\ &= \text{Dirichlet}\left(\alpha_1 + N_1, \ldots, \alpha_n + N_n, \alpha_{n+1}, \ldots, \alpha_k\right)\end{aligned}$$

such that, without loss of generality, the indices $1, 2, \ldots, n$ are the observed haplotypes and $n+1, n+2, \ldots, k$ are the unobserved haplotypes.

The marginal posterior distribution for the i'th haplotype is

$$\begin{aligned}f_i|N_i &\sim \text{Beta}\left(\alpha_i + N_i, \sum_{j=1}^{k}\left(\alpha_j + N_j\right) - \left(\alpha_i + N_i\right)\right) \\ &= \text{Beta}\left(\alpha_i + N_i, \alpha_+ - \alpha_i + N_+ - N_i\right)\end{aligned}$$

where $\alpha_+ = \sum_{i=1}^{k} \alpha_i$.

To obtain actual frequency estimates, one possibility is to use the mean

$$\mathbf{E}\left[f_i|N_i\right] = \frac{\alpha_i + N_i}{\sum_{j=1}^{k}\left(\alpha_j + N_j\right) - \left(\alpha_i + N_i\right) + \left(\alpha_i + N_i\right)} = \frac{\alpha_i + N_i}{\alpha_+ + N_+}$$

of the marginal posterior distribution. Doing so gives

$$\sum_{i=1}^{k} \mathbf{E}\left[f_i|N_i\right] = \left(\alpha_+ + N_+\right)^{-1} \sum_{i=1}^{k}\left(\alpha_i + N_i\right) = 1$$

i.e. a consistent model.

If we want to incorporate prior knowledge for all possible haplotypes, such as choosing prior parameters α_i's based on molecular distances, α_+ might be problematic to calculate for large k.

Sampling haplotypes according to their probabilities under this generalised formulation is still not straightforward.

3.3 Graphical Models

The framework of graphical models would be an obvious choice for a model to estimate haplotype frequencies. A lot of theory exists and graphical models fulfil the requirements described in section 3.1. In an earlier project, [Andersen, 2009a], this approach was investigated further by learning the model based on data. The learning approaches were both score based (BIC and AIC) and structure based (the PC algorithm). For references on the learning algorithms, refer to [Jensen and Nielsen, 2007], but be aware of the errors

in the description of the PC algorithm as described in [Andersen, 2009a]. Another reference for the PC algorithm is [Kalisch and Buhlmann, 2007].

The learning algorithms did however not perform satisfactory because of the many zeros in the datasets we are dealing with. In the PC algorithm one reason for this is that Pearson's χ^2-tests was used to test for conditional independence (the term d-separation is sometimes used), and this test is unreliable for tables with many zeros. Another problem with this test is that it does not exploit the ordering of the data. To use the PC algorithm on haplotype data, other kinds of tests must be used. Exact tests – both exhaustive tests and MCMC-approaches – are computational infeasible because the size of the set of possible contingency tables, with the given marginals, is simply too large. But tests such as Spearman's rank-order correlation coefficient [Hollander, 2006] would maybe be applicable, because it takes ordering into account. [Hollander, 2006] also mentions other dependence tests.

The PC algorithm finds the skeleton iteratively. Another approach, called the RAI algorithm, has been proposed by [Yehezkel and Lerner, 2009], which resembles a recursive version of the PC algorithm:

> *While other [constraint-based] algorithms d-separate structures and then direct the resulted undirected graph, the RAI algorithm combines the two processes from the outset and along the procedure. By this means and due to structure decomposition, learning a structure using RAI requires a smaller number of CI tests of high orders. This reduces the complexity and run-time of the algorithm and increases the accuracy by diminishing the curse-of-dimensionality.*
>
> [Yehezkel and Lerner, 2009]

Both the PC algorithm and the RAI algorithm allows testing for independence by whatever test the user wants, so it would definitely be interesting to try both the PC algorithm and the RAI algorithm with more reasonable tests for the data at hand.

An attempt to implement the RAI in `R` was made, but some implementation issues came up. E.g. in step 3 of the algorithm, "C. Ancestor sub-structure decomposition" the algorithm is "For $i = 1$ to k, call RAI[$n + 1$, GAi , Gex , Gall]." but it is not specified how to use the result of the call like e.g. GAi = RAI[$n + 1$, GAi, Gex, Gall].

Because of this, one of the authors, Raanan Yehezkel, was contacted and asked about this problematic step of the algorithm and to obtain the reference implementation (unfortunately it was not publicly available) based on The Bayes Net Toolbox for Matlab [Murphy, 2001] which could be used

to resolve other implementation issues. A good dialogue was initiated, and Raanan acknowledged the problem and the proposed solution with step 3 in the algorithm mentioned above and offered to both run the algorithm on the data and send the reference implementation. Both proposals were accepted. Unfortunately Raanan could not find the reference implementation, but instead explained how to run the algorithm manually only doing the conditional independence tests on the computer. This approach was however not pursued.

In conclusion, none of the standard learning algorithms yielded a satisfactory model.

3.4 Ancestral Awareness

In this section a new method for estimating Y-STR haplotype frequencies will be introduced.

First the basic idea will be introduced. Recall the notation from section 1.4, such that the loci is $L_1, L_2, \ldots, L_r$, taking values in $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_r$, respectively.

Let $\mathcal{I} \subseteq \{1, 2, \ldots, r\}$ and note that

$$P(L_1 = a_1, L_2 = a_2, \ldots, L_r = a_r) = P\left(\bigcap_{j \notin \mathcal{I}} L_j = a_j \;\middle|\; \bigcap_{i \in \mathcal{I}} L_i = a_i\right) \times P\left(\bigcap_{i \in \mathcal{I}} L_i = a_i\right).$$

The basic idea is to find $\mathcal{I} = \{i_1, i_2, \ldots, i_q\}$ such that

$$P\left(\bigcap_{j \notin \mathcal{I}} L_j = a_j \;\middle|\; \bigcap_{i \in \mathcal{I}} L_i = a_i\right) \approx \prod_{j \notin \mathcal{I}} P\left(L_j = a_j \;\middle|\; \bigcap_{i \in \mathcal{I}} L_i = a_i\right) \quad (3.3)$$

is a good approximation.

The set $\mathcal{I}$ is called an ancestral set, because it can be interpreted as a set of alleles that is common with one's ancestors.

Note that the method is not consistent.

3.4.1 How to Choose the Ancestral Set

A straightforward way to find the ancestral set is to use a greedy algorithm that, conditional on already chosen ancestors (initially the empty set), finds at which loci the probability for the allele in question is the greatest.

Formally, the algorithm can be expressed like shown in Algorithm 1 where the stop condition in the while-loop is left unspecified for now, but will get attention in section 3.4.2 and section 3.4.3.

Algorithm 1 A greedy approach to find ancestors. The stop condition is left unspecified for now.

Require: Haplotype $h = (a_1, a_2, \ldots, a_r)$ and table P
  $\mathcal{I} \leftarrow \varnothing$
  $p \leftarrow 1$
  while stop condition is false **do**
    $\mathcal{J} \leftarrow \mathcal{I}^C = \{1, 2, \ldots, r\} \setminus \mathcal{I}$
    $i_{max} \leftarrow \arg\max_{j \in \mathcal{J}} \{P\left(L_j = a_j \middle| \cap_{i \in \mathcal{I}} L_i = a_i\right)\}$
    $p \leftarrow p \cdot P\left(L_{i_{max}} = a_{i_{max}} \middle| \cap_{i \in \mathcal{I}} L_i = a_i\right)$
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{i_{max}\}$
    check if stop condition should be set to true
  end while
  $p \leftarrow p \cdot \prod_{j \in \mathcal{I}^C} P\left(L_j = a_j \,\middle|\, \bigcap_{i \in \mathcal{I}} L_i = a_i\right)$

3.4.2 Fixed Size of the Ancestral Set

One way to stop is when a predefined number of ancestors, i.e. the number of elements in the ancestral set, is reached. In a case with $r = 10$, i.e. a haplotype consisting of 10 loci, this number could maybe be 5. Which number to take must of course be checked afterwards, e.g. by thinking of the different parameters as resulting in different models which can be compared like done later in chapter 4.

The implementation of this method can be found in the supplementary material, refer to section 1.5.

For the purpose of illustration, an example of the method will be given in the following.

Example 3.1 (Example of choosing ancestral set of fixed size). This example is based on an execution of the implementation found in the supplementary material which is described in section 1.5, hence it is easy to output some extra information, e.g. how many observations are left at each step.

This example uses the dataset provided by ***dane*** with the loci

$$\begin{array}{lll} L_1 = \texttt{DYS19}, & L_2 = \texttt{DYS389I}, & L_3 = \texttt{DYS389II}, \\ L_4 = \texttt{DYS390}, & L_5 = \texttt{DYS391}, & L_6 = \texttt{DYS392}, \\ L_7 = \texttt{DYS393}, & L_8 = \texttt{DYS437}, & L_9 = \texttt{DYS438}, \quad L_{10} = \texttt{DYS439}. \end{array}$$

First assume that we want to estimate the frequency of the haplotype

$$\begin{aligned} h = (&\texttt{DYS19} = 13, \texttt{DYS389I} = 13, \texttt{DYS389II} = 29, \\ &\texttt{DYS390} = 22, \texttt{DYS391} = 10, \texttt{DYS392} = 6, \\ &\texttt{DYS393} = 13, \texttt{DYS437} = 14, \texttt{DYS438} = 11, \texttt{DYS439} = 12) \end{aligned}$$

using the described method with 5 as the fixed size of the ancestral set.

First the unconditional marginal probabilities are found based on the original 185 observations, namely

$$\begin{aligned} P(\texttt{DYS19} = 13) &= 0.032, \\ P(\texttt{DYS389I} = 13) &= 0.481, \\ P(\texttt{DYS389II} = 29) &= 0.308, \\ P(\texttt{DYS390} = 22) &= 0.254, \\ P(\texttt{DYS391} = 10) &= 0.632, \\ P(\texttt{DYS392} = 6) &= 0.011, \\ P(\texttt{DYS393} = 13) &= 0.795, \\ P(\texttt{DYS437} = 14) &= 0.259, \\ P(\texttt{DYS438} = 11) &= 0.178, \\ P(\texttt{DYS439} = 12) &= 0.324. \end{aligned}$$

Now $i_1 = 7$, DYS393, is chosen as an ancestor because $P(\texttt{DYS393} = 13) = 0.795$ has the highest probability. This means that $\mathcal{I} = \{7\}$.

Next step is to calculate the probabilities of the remaining loci conditional on $\texttt{DYS393} = 13$, now based on 147 observations, i.e.

$$\begin{aligned} P(\texttt{DYS19} = 13 \mid \texttt{DYS393} = 13) &= 0.041, \\ P(\texttt{DYS389I} = 13 \mid \texttt{DYS393} = 13) &= 0.517, \\ P(\texttt{DYS389II} = 29 \mid \texttt{DYS393} = 13) &= 0.327, \\ P(\texttt{DYS390} = 22 \mid \texttt{DYS393} = 13) &= 0.218, \\ P(\texttt{DYS391} = 10 \mid \texttt{DYS393} = 13) &= 0.592, \\ P(\texttt{DYS392} = 6 \mid \texttt{DYS393} = 13) &= 0.014, \\ P(\texttt{DYS437} = 14 \mid \texttt{DYS393} = 13) &= 0.265, \\ P(\texttt{DYS438} = 11 \mid \texttt{DYS393} = 13) &= 0.218, \\ P(\texttt{DYS439} = 12 \mid \texttt{DYS393} = 13) &= 0.367. \end{aligned}$$

Now $i_2 = 5$, DYS391, is chosen as an ancestor because $P(\texttt{DYS391} = 10 \mid \texttt{DYS393} = 13) = 0.592$ has the highest probability. Now $\mathcal{I} = \{7, 5\}$.

Next step is to calculate the probabilities of the remaining loci conditional on DYS393 = 13 and DYS391 = 10, now based on 87 observations, i.e.

$$\begin{aligned}
P(\texttt{DYS19} = 13 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.057,\\
P(\texttt{DYS389I} = 13 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.414,\\
P(\texttt{DYS389II} = 29 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.299,\\
P(\texttt{DYS390} = 22 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.368,\\
P(\texttt{DYS392} = 6 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.023,\\
P(\texttt{DYS437} = 14 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.264,\\
P(\texttt{DYS438} = 11 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.230,\\
P(\texttt{DYS439} = 12 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) &= 0.322.
\end{aligned}$$

Now $i_3 = 2$, DYS389I, is chosen as an ancestor because $P(\texttt{DYS389I} = 13 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) = 0.414$ has the highest probability, yielding $\mathcal{I} = \{7, 5, 2\}$.

Next step is to calculate the probabilities of the remaining loci conditional on DYS393 = 13, DYS391 = 10, and DYS389I = 13, now based on 36 observations, i.e.

$$\begin{aligned}
P(\texttt{DYS19} = 13 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) &= 0.111,\\
P(\texttt{DYS389II} = 29 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) &= 0.556,\\
P(\texttt{DYS390} = 22 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) &= 0.194,\\
P(\texttt{DYS392} = 6 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) &= 0.056,\\
P(\texttt{DYS437} = 14 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) &= 0.417,\\
P(\texttt{DYS438} = 11 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) &= 0.389,\\
P(\texttt{DYS439} = 12 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) &= 0.444.
\end{aligned}$$

Now $i_4 = 3$, DYS389II, is chosen as an ancestor because $P(\texttt{DYS389II} = 29 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) = 0.556$ has the highest probability, yielding $\mathcal{I} = \{7, 5, 2, 3\}$.

Next step is to calculate the probabilities of the remaining loci conditional on DYS393 = 13, DYS391 = 10, DYS389I = 13, and DYS389II = 29, now based

on 20 observations, i.e.

$$\begin{aligned} P(\texttt{DYS19} = 13 \mid \mathcal{I}) &= 0.050, \\ P(\texttt{DYS390} = 22 \mid \mathcal{I}) &= 0.350, \\ P(\texttt{DYS392} = 6 \mid \mathcal{I}) &= 0.100, \\ P(\texttt{DYS437} = 14 \mid \mathcal{I}) &= 0.050, \\ P(\texttt{DYS438} = 11 \mid \mathcal{I}) &= 0.100, \\ P(\texttt{DYS439} = 12 \mid \mathcal{I}) &= 0.550. \end{aligned}$$

Now $i_5 = 10$, `DYS439`, is chosen as an ancestor because $P(\texttt{DYS439} = 12 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13, \texttt{DYS389II} = 29) = 0.550$ has the highest probability, yielding $\mathcal{I} = \{7, 5, 2, 3, 10\}$.

Now 5 ancestors have been found, hence the algorithm stops searching for any more ancestors. Define the event

$$C = \{\texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13, \texttt{DYS389II} = 29, \texttt{DYS439} = 12\}.$$

If we believe that (3.3) is a good approximation, then by calculating

$$\begin{aligned} P(\texttt{DYS19} = 13 \mid C) &= 0.091, \\ P(\texttt{DYS390} = 22 \mid C) &= 0.091, \\ P(\texttt{DYS392} = 6 \mid C) &= 0.182, \\ P(\texttt{DYS437} = 14 \mid C) &= 0.091, \\ P(\texttt{DYS438} = 11 \mid C) &= 0.091, \end{aligned}$$

based on the 11 observations that have the ancestral set, we get that

$$
\begin{aligned}
P(h) = P(&\texttt{DYS19} = 13, \texttt{DYS390} = 22, \texttt{DYS392} = 6, \texttt{DYS437} = 14, \texttt{DYS438} = 11 \mid C) \\
&P(\texttt{DYS393} = 13) \\
&P(\texttt{DYS391} = 10 \mid \texttt{DYS393} = 13) \\
&P(\texttt{DYS389I} = 13 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) \\
&P(\texttt{DYS389II} = 29 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) \\
&P(\texttt{DYS439} = 12 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13, \texttt{DYS389II} = 29) \\
\approx\ &P(\texttt{DYS19} = 13 \mid C) \\
&P(\texttt{DYS390} = 22 \mid C) \\
&P(\texttt{DYS392} = 6 \mid C) \\
&P(\texttt{DYS437} = 14 \mid C) \\
&P(\texttt{DYS438} = 11 \mid C) \\
&P(\texttt{DYS393} = 13) \\
&P(\texttt{DYS391} = 10 \mid \texttt{DYS393} = 13) \\
&P(\texttt{DYS389I} = 13 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10) \\
&P(\texttt{DYS389II} = 29 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13) \\
&P(\texttt{DYS439} = 12 \mid \texttt{DYS393} = 13, \texttt{DYS391} = 10, \texttt{DYS389I} = 13, \texttt{DYS389II} = 29) \\
= \ &0.091 \cdot 0.091 \cdot 0.182 \cdot 0.091 \cdot 0.091 \\
&\cdot 0.795 \\
&\cdot 0.592 \\
&\cdot 0.414 \\
&\cdot 0.556 \\
&\cdot 0.550 \\
= \ &7.384 \cdot 10^{-7}
\end{aligned}
$$

This is merely an example, and it is not clear whether 5 would be a clever fixed size for the ancestral set, neither if the idea of finding an ancestral set in this way gives good results. □

3.4.3 Threshold of the Proportion of Haplotypes Left Given the Ancestral Set

Instead of stopping at a fixed size of the ancestral set, another approach could be to stop when then ancestral set is such that less than a given proportion

of the observations satisfies the constraints set by the ancestral set. This can also be interpreted as to only continue as long as the ancestor is common.

A slightly modified version is to have a look-ahead to determine how many observations would be left in the next iteration. This is however not further pursued.

3.4.4 Mutation

Now because $\mathcal{L}_i$ are ordered sets, we can implement a single-step mutation model. When the probability of a haplotype has to be found, we find that as described earlier. When this is done, we create a single-step mutation, such that the allele on one locus is mutated. Then the probability is found for this mutated haplotype. This is done for all possible mutations corresponding to calculating the probabilities for all haplotypes with distance 1 (using the L^1/Manhattan norm). Then the haplotype is assigned the probability of the (possible mutated) haplotype with the highest probability.

3.5 Classification

The process of linking a response variable Y with several explanatory variables, $X_1, X_2, \ldots, X_p$ is called regression, when the response is continuous, and classification, when the response is discrete. Classification is also referred to as supervised learning (whereas clustering – which will be used in section 3.7 – and methods such as principal component analysis and factor analysis introduced in section 2.1 and section 2.2, respectively, are referred to as unsupervised learning). These terms are used because often classification is a two-step-process: first a training set with known categories (known response variable) is used to train the model, and then the second step is to classify the category (the response variable) for some additional data where only the explanatory variables are known. Some classification techniques also provide a probability model such that probabilities for each possible classification category can be found; the observation is often classified as belonging to the category with the greatest probability, but misclassification costs are also sometimes taken into account.

Normally regression and classification is used when having one response variable, Y, and several explanatory variables, $X_1, X_2, \ldots, X_p$.

Thinking of the loci as the variables, we do not have this clear-cut setup. This does not mean that we cannot use some of the available classification and regression techniques, but we have to create a slightly different setup.

Because we are dealing with discrete values (alleles), from now on we focus on classification and disregard regression as such, but it is worth mentioning that some of the techniques to perform classification also has a regression counterpart, e.g. the techniques presented later on in this section can also be used as regression models if changed slightly.

Let $L_1, L_2, \ldots, L_r$ be the r different loci available in the haplotype. Now since none of the loci is naturally neither a response nor an explanatory variable, we can fix L_i, say, as the response variable and then regard the rest, $L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_r$ as explanatory variables. How to make the choice of which variable to regard as a response will be addressed later. We can then perform classification using the rest as explanatory variables. We can continue like this and choose a new response variable among the remaining L_i's and use the rest as explanatory variables, but in this step, L_i cannot be used because it has already been classified.

It is a very important conceptual difference to use classification in this way than in the traditional way!

This procedure can be formalised as follows. Choose a variable L_{i_1}, say, to use as a response according to some criteria. Then the model (using the formula notation of `R`)

$$L_{i_1} \sim \sum_{k \notin \{i_1\}} L_k$$

is fitted. The next step is now to find a new response, and the rest – except for the one already used in the first step – as explanatory variables. This is repeated $r-1$ times. In total the algorithm fits the models given by (using the formula notation of `R`)

$$\begin{aligned}
L_{i_1} &\sim \sum_{k \notin \{i_1\}} L_k \\
L_{i_2} &\sim \sum_{k \notin \{i_1, i_2\}} L_k \\
&\vdots \\
L_{i_{r-2}} &\sim \sum_{k \notin \{i_1, i_2, \ldots, i_{r-2}\}} L_k \\
L_{i_{r-1}} &\sim \sum_{k \notin \{i_1, i_2, \ldots, i_{r-1}\}} L_k = L_{i_r}
\end{aligned}$$

After these $r-1$ steps, the distribution for L_{i_r} is simply the marginal distribution based on the observations.

Another approach is to find the models by searching forward, i.e. to find

$$L_{i_{r-1}} \sim L_{i_r}$$

first amongst the $r(r-1)$ possible choices of (i_{r-1}, i_r), and then to find

$$L_{i_{r-2}} \sim L_{i_{r-1}} + L_{i_r}$$

afterwards. This approach is however not pursued any further.

One of the main issues is how to chose the i_j's. A vital prior issue is how to make the actual classification because the issue of choosing i_j's depends on this. The next sections will focus on these subjects.

When using this classification approach to estimate haplotype frequencies, we are not only interested in the result of the classification, i.e. which category the data belongs to. In order to estimate a haplotype frequency by calculating probabilities, we need to get a classification distribution $p_1, p_2, \ldots, p_m$ such that the probability of the data belonging to category i is p_i assuming m different categories. To estimate a probability of a haplotype we simply take the product of the classification probabilities under the models for the actual values of the chosen response variables.

3.5.1 Classification Trees

This section is based on [Venables and Ripley, 1997, chapter 14].

A classification tree is a tree used to classify an observation based on the explanatory variables. (Similarly, a regression tree is when then response variable is continuous.) It is constructed in such a way that it only uses one of the explanatory variables at a time and splits into children nodes depending on the value of the explanatory variable.

First example 3.2 will show how classification trees work in practice, and then the theory will be described afterwards.

Example 3.2 (Example of a classification tree). Using the data in [Hallenberg et al., 2004], a classification tree for the model (using the formula notation of `R`)

$$\texttt{DYS390} \sim .$$

where `.` is the rest of the loci, i.e. `DYS19`, `DYS389I`, `DYS389II`, `DYS391`, `DYS392`, `DYS393`, `DYS437`, `DYS438`, and `DYS439`, can be found by the following `R`-code (the exact `R`-code used is in the file `classification-trees-example.R` which can be found in the supplementary material, section 1.5):

```
1 library(rpart)
2 rpart(DYS390 ~ .)
```

provided that `DYS390` is a factor (or else a regression tree will be created). The output of the `R`-code is:

```
1) root 185 131 23 (0.0054 0.22 0.29 0.29 0.16 0.038)
  2) DYS437>=15.5 62  25 22 (0.016 0.6 0.32 0.065 0 0) *
  3) DYS437< 15.5 123  73 24 (0 0.024 0.28 0.41 0.24 0.057)
    6) DYS392>=11.5 82  43 24 (0 0.037 0.37 0.48 0.085 0.037)
     12) DYS437< 14.5 16   5 23 (0 0.12 0.69 0.12 0.062 0) *
     13) DYS437>=14.5 66  29 24 (0 0.015 0.29 0.56 0.091 0.045) *
    7) DYS392< 11.5 41  19 25 (0 0 0.098 0.27 0.54 0.098)
     14) DYS438< 10.5 11   5 24 (0 0 0.36 0.55 0.091 0) *
     15) DYS438>=10.5 30   9 25 (0 0 0 0.17 0.7 0.13) *
```

The numbers in parentheses are the probability of `DYS390` taking the values $21, 22, \ldots, 26$, respectively (those are the possible values of `DYS390` in the data). Notice that no configuration of the explanatory variables results in the classification `DYS390` $= 26$, but it does of course not mean that $P(\texttt{DYS390} = 26 \mid \text{the rest}) = 0$ for all configurations of the explanatory variables.

More details can be found by getting the summary of a `rpart`-object, that is

```
1 library(rpart)
2 fit <- rpart(DYS390 ~ .)
3 summary(fit)
```

gives a verbose output. □

Theory

In general classification trees can by any type of trees, but in practice it is easier to allow only binary trees because then comparing splits is avoided, and multiway splits can be made by several binary splits. Taking a closer look at example 3.2 one can also see that only binary splits are made. This is because the `rpart`-package by [Therneau and Atkinson, 2009] in `R` is based on [Breiman et al., 1984] which only allows binary splits.

Let $i = 1, 2, \ldots, I$ denote the nodes in the tree and $k = 1, 2, \ldots, K$ the possible values of the response variable. Then at each node we have a probability distribution $\{p_{ik}\}_{k=1}^{K}$ and observed counts $\{n_{ik}\}_{k=1}^{K}$. Each node has deviance given by

$$D_i = -2 \sum_{k=1}^{K} n_{ik} \log(p_{ik})$$

and the total deviance in the tree is simply

$$D = \sum_{i=1}^{I} D_i.$$

If we are to split a node s into nodes t and u, this changes the probability distribution at s causing a reduction of deviance giving by

$$\begin{aligned}
D_s - (D_t + D_u) &= -2\sum_{k=1}^{K} n_{sk} \log(p_{sk}) - \left(-2\sum_{k=1}^{K} n_{tk} \log(p_{tk}) - 2\sum_{k=1}^{K} n_{uk} \log(p_{uk}) \right) \\
&= 2\sum_{k=1}^{K} \left(n_{tk} \log(p_{tk}) + n_{uk} \log(p_{uk}) - n_{sk} \log(p_{sk}) \right) \\
&= 2\sum_{k=1}^{K} \left(n_{tk} \log(p_{tk}) + n_{uk} \log(p_{uk}) - (n_{tk} + n_{uk}) \log(p_{sk}) \right) \\
&= 2\sum_{k=1}^{K} \left(n_{tk} \log(p_{tk}) + n_{uk} \log(p_{uk}) - n_{tk} \log(p_{sk}) - n_{uk} \log(p_{sk}) \right) \\
&= 2\sum_{k=1}^{K} \left(n_{tk} \log\left(\frac{p_{tk}}{p_{sk}}\right) + n_{uk} \log\left(\frac{p_{uk}}{p_{sk}}\right) \right).
\end{aligned}$$

The probabilities are estimated as usual by

$$\begin{aligned}\hat{p}_{tk} &= \frac{n_{tk}}{n_{t+}},\\ \hat{p}_{uk} &= \frac{n_{uk}}{n_{u+}},\\ \hat{p}_{sk} &= \frac{n_{sk}}{n_{s+}} = \frac{n_{tk}+n_{uk}}{n_{s+}} = \frac{\hat{p}_{tk}n_{t+}+\hat{p}_{uk}n_{u+}}{n_{s+}}.\end{aligned}$$

Then the reduction in deviance is estimated to

$$2\sum_{k=1}^{K}\left(n_{tk}\log\left(\frac{n_{tk}n_{s+}}{n_{sk}n_{t+}}\right)+n_{uk}\log\left(\frac{n_{uk}n_{s+}}{n_{sk}n_{u+}}\right)\right)$$

which can easily be rewritten to use expressions of the form $n\log n$.

According to [Venables and Ripley, 1997, p. 416] and [Therneau and Atkinson, 2009] several other methods use a one-step look-ahead. So to choose the next split, the algorithm calculates the reduction in deviance of all allowed splits of all leaves and chooses the split causing the greatest reduction in deviance and stop if none. This is a greedy approach that only finds a local maximum and not the global one (if such exists). Instead of just looking at the deviance, following approaches as AIC and BIC, a punishment

$$c\log(\text{number of leaves in model})$$

is actually added according to how complex a model is. The c-parameter can be adjusted through the `rpart`-function.

One way to stop splitting is when the reduction in deviance is below some threshold. According to [Breiman et al., 1984] this is however seldom a successfully approach, and no great stopping rules were ever found. Instead, the splitting is continued until all the leaves are either pure (all the cases in the leaf is of the same class) or contains a (specified) small number of cases, and then the tree is pruned afterwards. Refer to [Breiman et al., 1984] for details on how to prune.

Modelling

The choice of the i_j's (see to the introduction of this chapter for details) is the main issue for now. One straightforward choice would be to use the model with the minimal sum of all the deviances in the model. A punishment could also be introduced. Another approach would be to use the model with the minimal entropy $\sum_i p_{ik}\log(p_{ik})$, but this is not pursued any further because according to [Venables and Ripley, 1997, p. 418] this is equivalent to using the total deviance of the tree; they only differ by a constant so in comparison between models it does not change anything.

3.5.2 Ordered Logistic Regression

First logistic regression will be described based on [Azzalini, 1996, p. 41, p. 231], and afterwards the ordered counterpart will be described based on [Agresti, 2002] and [Norušis, 2000].

Logistic Regression

Recall that the probability of an event occurring as opposed to it not occurring if often modelled by logistic regression when the occurrence of the event depends on some other explanatory variables.

First let

$$U_i \sim \text{Bin}\,(m_i, p_i) \quad \text{for} \quad i = 1, 2, \ldots, n$$

where m_i is known and p_i is unknown for all i.

Logistic regression can be stated in (at least) two ways. First a straightforward approach will be introduced, and afterward a formulation within the GLM-framework (generalised linear models) will be made. GLM is implemented in `R` via the `glm`-function.

The straightforward approach is to model the logit probability (also called log-odds) as a linear function of the explanatory variables $x_{1,i}, x_{2,i}, \ldots, x_{k,i}$ such that

$$\text{logit}\,(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i}.$$

This reduces to a linear regression with $\text{logit}\,(p_i)$ as the response instead of just p_i.

Solving for p_i means that

$$\begin{aligned}\hat{p} &= \frac{1}{1 + \exp\left(-\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k\right)\right)} \\ &= \frac{\exp\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k\right)}{\exp\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k\right) + 1}\end{aligned} \tag{3.4}$$

is an estimated probability based on the explanatory variables $x_1, x_2, \ldots, x_k$.

For the GLM-approach, consider

$$Y_i = \frac{U_i}{m_i}$$

as the response, such that the probability mass function is

$$\begin{aligned}
f_{Y_i}(y_i) &= f_{U_i}(u_i) \\
&= \binom{m_i}{u_i} p_i^{u_i} (1-p_i)^{m_i-u_i} \\
&= \binom{m_i}{m_i y_i} p_i^{m_i y_i} (1-p_i)^{m_i - m_i y_i} \\
&= \exp\left(\log\left(\binom{m_i}{m_i y_i} p_i^{m_i y_i} (1-p_i)^{m_i - m_i y_i}\right)\right) \\
&= \exp\left(\log\binom{m_i}{m_i y_i} + m_i y_i \log(p_i) + m_i (1-y_i)\log(1-p_i)\right) \\
&= \exp\left(m_i (y_i \log(p_i) + (1-y_i)\log(1-p_i)) + \log\binom{m_i}{m_i y_i}\right)
\end{aligned}$$

for $y_i = 0, \frac{1}{m_i}, \ldots, 1$ equivalent to $u_i = 0, 1, \ldots, m_i$.

Now introduce

$$\begin{aligned}
\mu_i &= \mathbf{E}[Y_i] = p_i, \\
\theta_i &= \operatorname{logit} \mu_i = \log\left(\frac{\mu_i}{1-\mu_i}\right), \\
b(\theta) &= \log(1+\exp\theta), \\
\psi_i &= 1, \\
w_i &= m_i, \\
c(y_i, \psi_i) &= \log\binom{m_i}{m_i y_i}
\end{aligned}$$

then

$$\begin{aligned}
f_{Y_i}(y_i) &= \exp\left(m_i\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right) + \log\binom{m_i}{m_i y_i}\right) \\
&= \exp\left(m_i\left(y_i \log(p_i) + \log(1-p_i) - y_i \log(1-p_i)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(m_i\left(y_i \log(\mu_i) + \log(1-\mu_i) - y_i \log(1-\mu_i)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(m_i\left(y_i\left(\log(\mu_i) - \log(1-\mu_i)\right) + \log(1-\mu_i)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(m_i\left(y_i \log\left(\frac{\mu_i}{1-\mu_i}\right) + \log(1-\mu_i)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(m_i\left(y_i \log\left(\frac{\mu_i}{1-\mu_i}\right) - \log\left(\frac{1}{1-\mu_i}\right)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(m_i\left(y_i \log\left(\frac{\mu_i}{1-\mu_i}\right) - \log\left(\frac{1-\mu_i+\mu_i}{1-\mu_i}\right)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(m_i\left(y_i \log\left(\frac{\mu_i}{1-\mu_i}\right) - \log\left(1+\frac{\mu_i}{1-\mu_i}\right)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(m_i\left(y_i\theta_i - \log(1+\exp\theta_i)\right) + c(y_i,\psi_i)\right) \\
&= \exp\left(\frac{w_i}{\psi_i}\left(y_i\theta_i - b(\theta_i)\right) + c(y_i,\psi_i)\right)
\end{aligned}$$

which according to [Azzalini, 1996, p. 226] means that it is indeed a GLM, for which very neat theory exist, e.g [Azzalini, 1996, chap. 6].

Notice that the inverse of $\theta_i = \operatorname{logit}\mu_i$ is given by $(1+\exp(-\theta_i))^{-1}$, just like in (3.4).

Because $\theta_i = \operatorname{logit}\mu_i$, the canonical link is the logit-function, which is why logistic regression is also referred to as the logit model.

It has to be noted, that instead of the logit-link, other link functions $l(p)$ give rise to a regression model for probabilities, namely

$$l(p_i) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i},$$

where table 3.3 shows some different link functions (logit is included for completeness).

It is also possible to make a multinomial regression model if several outcomes are possible. This is a generalisation of the logistic regression in the same way that the multinomial distribution generalises the binomial distribution. However, this will not be investigated further because it is not going to be used.

Name	Function	Note
Logit	$l(p) = \text{logit}(p)$	
Probit	$l(p) = \Phi^{-1}(p)$	Φ^{-1} is the inverse cumulative distribution function for the standard Gaussian distribution, i.e. with mean zero and variance 1
Complementary log-log	$l(p) = \log(-\log(1-p))$	
Negative log-log	$l(p) = -\log(-\log(p))$	
Cauchit	$l(p) = \tan(\pi(p-0.5))$	The link is the inverse cumulative distribution function for the Cauchy distribution with location parameter equal to 0 and scale parameter equal to 1

Table 3.3: Different link functions to perform logistic regression. The table is similar to the one in [Norušis, 2000, p. 84].

Odds Ratio Logistic Regression

A lot of data have an ordered response instead of just a nominal one. Generally, if using logistic (or multinomial) regression on data with ordinal responses, a lot of information are not used.

What to do instead is to make ordinal regression. One way to do this is to use what [Agresti, 2002] refers to as a cumulative link model. Now this model will be described based on [Agresti, 2002] and [Norušis, 2000].

The starting point is to define probabilities differently. In the logistic regression we modelled the probability of the event happening given the explanatory variables. In this ordinal regression, we instead model the probability that some event or the events ordered before it has happened, i.e. the cumulative probability. This means that instead of model $P(Y_i = 1 \mid \boldsymbol{x}_i)$ like in the logistic regression case, we model $P(Y_i \leq j \mid \boldsymbol{x}_i)$ for Y having levels $1, 2, \ldots, J$ given k for explanatory variables $\boldsymbol{x}_i = (x_{1,i}, x_{2,i}, \ldots, x_{k,i})$.

In other words, we model the odds given by

$$\begin{aligned}\Theta_1 &= \frac{P(Y \leq 1)}{1 - P(Y \leq 1)} = \frac{P(Y \leq 1)}{P(Y > 1)} \\ \Theta_2 &= \frac{P(Y \leq 2)}{1 - P(Y \leq 2)} = \frac{P(Y \leq 2)}{P(Y > 2)} \\ &\vdots \\ \Theta_{J-1} &= \frac{P(Y \leq J-1)}{1 - P(Y \leq J-1)} = \frac{P(Y \leq J-1)}{P(Y > J-1)}\end{aligned}$$

omitting the explanatory variables for easier reading.

This leads to the model

$$\operatorname{logit} P(Y \leq j \mid \boldsymbol{x}_i) = \zeta_{i,j} - \eta_i$$

where $\eta_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} = x_{1,i}\beta_1 + \cdots + x_{k,i}\beta_k$ is the linear predictor and $\zeta_{i,j}$ is the threshold value (corresponding to the intercept). Notice that the linear predictor η_i does not depend on the level being modelled.

Notice that this model gives the same results if adjacent levels are collapsed, of course as long as the collapsed levels are not the ones involved in the current odds modelling.

Again, different link functions than logit can be chosen.

In `R`, ordinal regression is implemented in the function `polr` from the `MASS`-package. It supports all the different links given in table 3.3 except the negative log-log. The function is based on the cumulative link model just described, and refers to [Agresti, 2002] in the help file.

The links in table 3.3 are often used in different applications according to [Norušis, 2000, p. 84]. They mention that

- Logit is usually used for evenly distributed categories
- Probit is usually used in analyses with explicit normally distributed latent variable
- Complementary log-log is usually used when higher categories is more probable
- Negative log-log is usually used when lower categories is more probable
- Cauchit is usually used on outcomes with many extreme values

Modelling

Now, if we go back to our haplotype setup with notation as introduced in section 1.4, we can use this ordinal regression as a classification model.

Let X be some event, e.g. other loci take certain allelic values. Using ordinal regressions, we can calculate

$$P(L_i = a_i \mid X) = P(L_i \leq a_i \mid X) - P(L_i \leq a_i - 1 \mid X)$$

where $P(L_i \leq a_i \mid X) = P(L_i = a_i \mid X) = 0$ for $a_i \notin \mathcal{L}_i$. The ordinal regression is used to model $P(L_i \leq a_i \mid X)$ and $P(L_i \leq a_i - 1 \mid X)$.

3.5.3 Support Vector Machines

This section is based on [Vapnik, 1998], [Cristianini and Shawe-Taylor, 2000], and [Moguerza and Muñoz, 2006].

SVMs (support vector machines) will not be exhaustively covered. This is a huge area that could easily serve as a subject for a thesis on its own. Instead some history, interesting aspects of SVMs, and the theory they are build upon will be described.

SVMs is a development in statistical machine learning theory built both upon basic, yet important, concepts like Rosenblatt's perceptron, refer to [Vapnik, 1998, p. 375] and [Cristianini and Shawe-Taylor, 2000, section 2.1.1], and upon complicated and powerful theories, e.g. within optimization theory and functional analysis. The first small steps of SVMs was according to [Vapnik, 1998, p. 711] taken in 1963 when a method corresponding to constructing an optimal hyperplane in the separable case – which is described in [Vapnik, 1998, section 10.1] – was published (by Vapnik *et al.*). The optimal hyperplane is the hyperplane that has the greatest distance to all the categories. The vectors closest to the optimal hyperplane are called the support vectors. To obtain the theory of SVMs we have today, 30 years almost passed. To quote:

> *To construct a hyperplane in high-dimensional feature space, we use a general representation of the inner product in Hilbert spaces. According to Mercer's theorem, an inner product in Hilbert spaces has an equivalent representation in kernel form. This fact was established by Mercer in 1909 [...]. Since then the Mercer theorem, the related theory of positive definite functions, and the theory of reproducing kernels Hilbert spaces have become important topics of research [...]. In particular, this theorem was used to prove the*

> *equivalence between the method of potential functions and Rosenblatt's perceptron [...].*
>
> *Therefore by the mid-1960s, two main elements of the SV machine (the expansion of the optimal hyperplane on support vectors and the constructing hyperplane in feature space using Mercer kernels) were known. It needed only one step to combine these two elements. This step, however, was done almost 30 years later in an article by Boser, Guyon, and Vapnik (1992).*
>
> *After combining the SV expansion with kernel representation of the inner product, the main idea of the SV machine was realized: One could construct linear indicator functions in high-dimensional space that had a low capacity. However, one could construct these hyperplanes (or corresponding kernel representation in the input space) only for the separable case.*
>
> *The extension of the SV techniques for the non-separable cases was obtained in an article by Cores and Vapnik (1995).*
>
> [Vapnik, 1998, p. 713]

SVMs have been shown to perform better than – or at least as good as – domain specific methods in a lot of different applications, refer e.g. to [Cristianini and Shawe-Taylor, 2000, p. 7] and [Vapnik, 1998, p. 567].

Before introducing what support vector machines (called SVMs) actually are, a short introduction to learning machines is made.

Notation

Let $X \subseteq \mathbb{R}^n$ be the set where observations are from, and Y be the set of categories. The aim is to find a classification function (also called classifier) with domain X and co-domain Y such that observations can be classified using this function.

If $Y = \{1, 2, \ldots, m\}$ the classification is called m-class classification with binary classification for $m = 2$ as a special case which sometimes use $Y = \{-1, 1\}$ (referred to as the negative and positive class, respectively). If $Y \subseteq \mathbb{R}$ the classification is called regression, but – as mentioned earlier – this case will not be treated further, it is just mentioned that this is also possible with SVMs.

Normally a classifier is build based on a l-case training set

$$S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_l, y_l)\} \subseteq (X \times Y)^l,$$

i.e. where the correct categories for the l observations are provided.

When the classifier has been built, it can be used to classify cases $\boldsymbol{x}$.

Learning Machines

Usually one restricts the set of classifiers to choose from. The classifiers left in this restricted set are also referred to as hypotheses. Learning machines using hypotheses forming linear combinations of the input variables are called linear learning machines. This type of learning machines has the advantage that developing an elaborate theory is actually possible.

As a simple example, first consider a simple binary classifier based on $f : X \subseteq \mathbb{R}^n \to R$ such that $\boldsymbol{x} \in X$ is classified to the positive class if $f(\boldsymbol{x}) \geq 0$ and to the negative class otherwise.

If f is a linear function, then f for $\boldsymbol{x} \in \mathbb{R}^n$ can be written as

$$f(\boldsymbol{x}) = \langle \boldsymbol{w} \cdot \boldsymbol{x} \rangle + b = \sum_{i=1}^{n} w_i x_i + b \tag{3.5}$$

where $\boldsymbol{w} \in \mathbb{R}^n$, the weight vector, and $b \in \mathbb{R}$, the bias, are the parameters that control the classification. These parameters must be learned from a training set. One way to do that is with The Perceptron Algorithm, which can be found in [Cristianini and Shawe-Taylor, 2000, Table 2.1, p. 12].

An important interpretation of this kind of linear classification is that $X \subseteq \mathbb{R}^n$ is split into two by the hyperplane defined by $\langle \boldsymbol{w} \cdot \boldsymbol{x} \rangle + b = 0$.

If we instead have m-classes, then a weight vector $\boldsymbol{w}_j$ and a bias b_j are associated to each class for $j = 1, 2, \ldots, m$.

Feature Spaces and Dual Forms

First the term linear separable is introduced. Training data are said to be linear separable if there exists a hyperplane that correctly classifies the training data. The definition can be extended to a linear δ-separable dataset, where the distance from the hyperplane to the closest vector is less than δ.

Not all data are linear separable in their original form. Consider a short example from [Andersen, 2009b]. The data in figure 3.9 is not linear separable. It is not possible to create a hyperplane that separates the data from the different classes.

If we transform the data from $\mathbb{R}^2$ to $\mathbb{R}^3$ with the transformation

$$\varphi(x_1, x_2) = \left(x_1, x_2, x_1^2 + x_2^2\right)$$

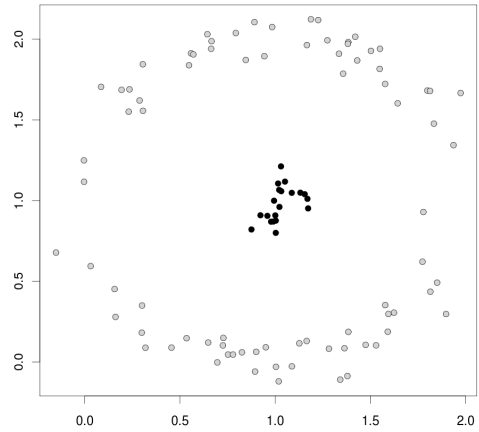

Figure 3.9: Lineary non-separable data. The classes are marked with different colors.

we get figure 3.10. As we see it is now possible to separate the data with a hyperplane.

So even if data are not linear separable in their original form, they can sometimes be mapped to a higher dimensional space by some transformation so that the data indeed are linear separable so that the linear learning machines can be used. This is very desirable because a lot can be proved in general about linear learning machines. Mapping into a higher dimensional space is a very important concept that is used in SVM. The higher dimensional space is called the feature space. One of the questions that immediately pops up is how to choose this feature space. This question is treated a little later.

Another question that pops up is how to deal with the curse of dimensionality when one purposely introduces a higher dimensional space. Among other things, higher dimensional spaces often make the computations heavier. But it can actually be avoided making computations in the feature space.

The reason for this is that many linear learning machines can be expressed in a so-called dual form. This kind of representation is very important in the development of SVMs, because it is a form where data only appear through entries in the Gram matrix, when the machine is learned and does not depend on the dimension of the space. When classification has to be made (on a new data), the only thing needed is the inner product of this new data with the training set. So we just have to perform computations in the feature space implicitly. In the primary representation it is often necessary to e.g. sum over all dimensions of the feature space, and if a Hilbert space is used as the

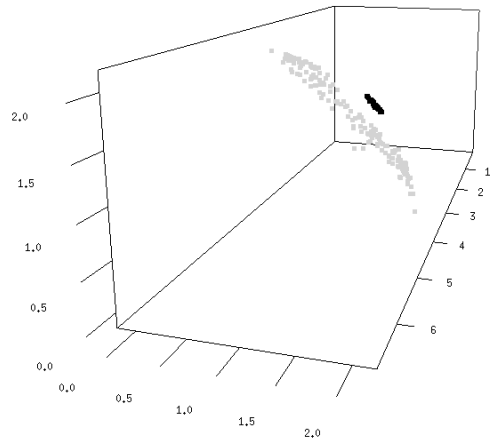

Figure 3.10: Points in figure 3.9 mapped to $\mathbb{R}^3$ in order to make the data linear separable.

feature space, it can be difficult to show convergence of the series.

As an example, the dual form of the learning machine in (3.5) gives by [Vapnik, 1998, p. 406] and [Cristianini and Shawe-Taylor, 2000, p. 18] the decision function

$$f(\boldsymbol{z}) = \left\langle \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i, \boldsymbol{z} \right\rangle + b = \sum_{i=1}^{l} \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{z} \rangle + b$$

where l denotes the number of training cases and y_i is the category of $\boldsymbol{x}_i$.

The fact that learning machines often can be expressed in dual form makes feature spaces attractive.

Kernels

Now another important concept called kernels are described. Let X denote the space the data lives in and let F be a feature space with an inner product. A kernel is a function K such that for all $\boldsymbol{x}, \boldsymbol{z}$, it is true that

$$K(\boldsymbol{x}, \boldsymbol{z}) = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{z}) \rangle$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of F. The transformation φ is often called the feature map. The key is to choose a kernel that can be evaluated efficiently such that the dimension of the feature space does not affect the computational complexity.

Kernels can be created in different ways. There is even a way to characterise if a function is a kernel. By [Cristianini and Shawe-Taylor, 2000, Proposition 3.5] we get that if $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is finite and $K(\boldsymbol{x}, \boldsymbol{z})$ is symmetric – such that $K(\boldsymbol{x}, \boldsymbol{z}) = K(\boldsymbol{z}, \boldsymbol{x})$ – then K is a kernel if and only if the matrix $(K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^{n}$ is positive semi-definite.

This gives a complete characterisation for finite input spaces. If $K(\boldsymbol{x}, \boldsymbol{z})$ is continuous and symmetric, then Mercer's theorem (1909) from functional analysis gives a complete characterisation of valid kernels. Mercer's theorem is stated in [Vapnik, 1998, p. 423] and [Cristianini and Shawe-Taylor, 2000, p. 35]. Actually Mercer's theorem ensures that $K(\boldsymbol{x}, \boldsymbol{z})$ can be expanded as a absolutely and uniformly convergent series, which by using a dual form of a linear learning machine can be rewritten as a finite sum over the test cases.

A kernel algebra is also available from [Cristianini and Shawe-Taylor, 2000, p. 42], e.g. if K_1 and K_2 are kernels, then $K_1 + K_2$ is also a kernel.

There is no such thing as a free lunch. The use of high dimensional feature spaces introduces the problem of over-fitting. The theory of avoiding this is called generalisation theory, see e.g. [Cristianini and Shawe-Taylor, 2000, chapter 4]. It describes how to make sound generalisations so that over-fitting is avoided, and it is quite complicated.

Problems

As mentioned earlier, a problem with the SVM-method is how to choose the kernel (and thereby a feature space) and kernel parameters. This is also addressed in [Moguerza and Muñoz, 2006] as an open problem in which research is being done. Some kernels are used in a lot of situations, e.g. the Gaussian kernel. But it is also possible to incorporate prior knowledge and create a kernel for a specific problem.

Final Remarks

A lot of the theory have not even been mentioned in this brief introduction to SVMs. Among the ignored areas are optimisation, theory which plays a vital role in learning in order to find the optimal hyperplanes. A lot of important inequalities used come from optimisation theory. Other ignored areas are regression and how to handle linear non-separable datasets.

3.6 Kernel Smoothing

To assign probability mass to unobserved haplotypes (in general called density estimation), one approach is to take some of the relative probability mass and smooth to the haplotype neighbours according to some distribution. One apparent way to do this, is to create a r-dimensional Gaussian distribution around each haplotype, i.e. one for each haplotype with the haplotype as the mean. Because it is a continuous density, it might give rise to problems.

Recall that we have n different haplotypes $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathcal{H}$, the observations $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{N_+} \in \mathcal{H}$ where $N_+ = \sum_{i=1}^{n} N_i$ and N_i denotes how many times the i'th haplotype has been observed. Please refer to section 1.4 for an elaborate explanation of the notation.

As a starting point for the development of the kernel smoothing model, assign to each haplotype $\boldsymbol{x}_i$ its relative frequency $\frac{N_i}{N_+}$. Now only observed haplotypes have a positive probability. In order to assign probability mass to unobserved haplotypes we want to smooth out some of the probability mass all the observed haplotypes have. There are of course no right and simple way to do this, but one approach is to put a scaled Gaussian density around each haplotype. The density has to be scaled such that it only contains the same probability mass as the relative frequency, $\frac{N_i}{N_+}$. By doing this, unobserved haplotypes also get a positive probability.

In [Seber, 1984] the Gaussian kernel

$$K\left(\boldsymbol{z}|\boldsymbol{x}_i, \lambda\right) = \left(2\pi\lambda^2\right)^{-\frac{r}{2}} \det\left(\Sigma\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\lambda^2}(\boldsymbol{x}_i - \boldsymbol{z})\Sigma^{-1}(\boldsymbol{x}_i - \boldsymbol{z})^\top\right)$$

is proposed, where λ is called a smoothing parameter. A straightforward choice for Σ is the empirical covariance matrix, say S. Now the smoothened density for any given haplotype $\boldsymbol{z} \in \mathcal{H}$ (observed as well as unobserved) is

$$g(\boldsymbol{z}) = \underbrace{\frac{1}{N_+}\sum_{j=1}^{N_+} K\left(\boldsymbol{z}|\boldsymbol{y}_j, \lambda\right)}_{\text{Summing over observations}} = \underbrace{\frac{1}{N_+}\sum_{i=1}^{n} N_i K\left(\boldsymbol{z}|\boldsymbol{x}_i, \lambda\right)}_{\text{Summing over haplotypes}},$$

i.e. we sum the contributions to the smoothened density from all the haplotypes.

Now if

$$\sum_{\boldsymbol{z}\in\mathcal{H}} K\left(\boldsymbol{z}|\boldsymbol{x}_i, \lambda\right) \approx 1 \tag{3.6}$$

for all $i = 1, 2, \ldots, n$, we have that

$$\begin{aligned}\sum_{\boldsymbol{z}\in\mathcal{H}} g(\boldsymbol{z}) &= \sum_{\boldsymbol{z}\in\mathcal{H}} \frac{1}{N_+} \sum_{i=1}^{n} N_i K\left(\boldsymbol{z}|\boldsymbol{x}_i, \lambda\right) \\ &= \frac{1}{N_+} \sum_{i=1}^{n} N_i \sum_{\boldsymbol{z}\in\mathcal{H}} K\left(\boldsymbol{z}|\boldsymbol{x}_i, \lambda\right) \\ &\approx \frac{1}{N_+} \sum_{i=1}^{n} N_i \\ &= 1.\end{aligned}$$

The assumption (3.6) is fulfilled if each density has a fairly large variance, such that the evaluations over the grid $\mathcal{H}$ gives a good approximation of the continuous density. Note that $\int_{\mathbb{R}^r} g(\boldsymbol{z})d\boldsymbol{z} = 1$. If (3.6) is not fulfilled, numerical integration could be used, but it would easily get too computational inefficient to be practically possible. Another approach is to use a discrete kernel.

In this forensic setup, we have some prior knowledge. One could ask the reasonable question: is it fair to smooth out the probability mass from haplotypes observed e.g. 9 times as much as for singletons? Instead of a given singleton, we could just as well have observed another singleton, but when a certain haplotype is observed quite often, then the relative frequency must be close to the true probability. To use this prior knowledge, one could choose to adjust the variance accordingly so that haplotypes observed a lot of times get less variance. This can be incorporated in the kernel by adjusting the variance, such that if N_i denotes the number of times haplotype number i is observed, then the kernel

$$K\left(\boldsymbol{z}|\boldsymbol{x}_i, N_i, \lambda\right) = \left(2\pi\frac{\lambda^2}{N_i}\right)^{-\frac{r}{2}} \det\left(\Sigma\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\frac{\lambda^2}{N_i}}(\boldsymbol{x}_i - \boldsymbol{z})\Sigma^{-1}(\boldsymbol{x}_i - \boldsymbol{z})^\top\right)$$

could be used.

3.7 Model-Based Clustering

Kernel smoothing was introduced in section 3.6. In order to calculate the density at a point $\boldsymbol{z}$, all the observations have to be used. In other words, the complexity of calculating a density grows with the number of observed haplotypes.

Instead of having a number of densities equal to the number of haplotypes, clustering can be used to create G groups of observations so that each group

has a density associated. In this way, in order to calculate the density at a point, only the G different densities have to be calculated instead of one at each observation. This technique is by [Fraley and Raftery, 2006] referred to as model-based clustering.

Define

$$h(\boldsymbol{z}) = \sum_{g=1}^{G} p_g f_g(\boldsymbol{z})$$

for G clusters where p_g is the a priori probability of an observation being in the g'th cluster with $\sum_{g=1}^{G} p_g = 1$ and f_g is the density for the g'th cluster. Similar to (3.6), if

$$\sum_{\boldsymbol{z} \in \mathcal{H}} f_g(\boldsymbol{z}) \approx 1,$$

for all $g = 1, 2, \ldots, G$, then

$$\sum_{\boldsymbol{z} \in \mathcal{H}} h(\boldsymbol{z}) = \sum_{\boldsymbol{z} \in \mathcal{H}} \sum_{g=1}^{G} p_g f_g(\boldsymbol{z}) = \sum_{g=1}^{G} p_g \sum_{\boldsymbol{z} \in \mathcal{H}} f_g(\boldsymbol{z}) \approx \sum_{g=1}^{G} p_g = 1.$$

Actual implementation of this method both require a way to make the clustering and a specification of the densities used. In this thesis the `R`-package `mclust` by [Fraley and Raftery, 2006] has been used. Some of the available models (different kinds of multivariate Gaussian densities) are listed in the help-file `mclustModelNames` as:

- Spherical: equal volume / unequal volume
- Diagonal: equal volume and shape / varying volume, equal shape / equal volume, varying shape / varying volume and shape
- Ellipsoidal: equal volume, shape, and orientation / equal volume and equal shape / equal shape / varying volume, shape, and orientation

In a multivariate Gaussian distribution, the covariance matrix controls the shape of the density. Using this, the theoretical background for the possible models can be explained by eigenvalue decomposition of the covariance matrix Σ_g for the g'th group. We can write

$$\Sigma_g = \lambda_g D_g A_g D_g^\top,$$

where D_g is an orthogonal matrix (such that $D_g D_g^\top = I$ making $D_g^\top = D_g^{-1}$) of the normalised eigenvectors (this does not affect the corresponding eigenvalue), A_g is a diagonal matrix with elements proportional to the eigenvalues (with proportionality factor λ_g). Because the orientation is determined by

D_g, the shape is determined by A_g, and the volume of the corresponding ellipsoid is determined by λ_g, `mclust` uses identifiers for identifying each model, where the first character concerns the volume, the second the shape, and the third the orientation. The models together with the identifiers, properties, and assumed covariance decomposition forms can be found in table 3.4.

Type	**Volume**	**Shape**	**Orientation**	**Identifier**	**Covariance**
Spherical	Equal	Equal		EII	λI
Spherical	Variable	Equal		VII	$\lambda_g I$
Diagonal	Equal	Equal		EEI	λA
Diagonal	Variable	Equal		VEI	$\lambda_g A$
Diagonal	Equal	Variable		EVI	λA_g
Diagonal	Variable	Variable		VVI	$\lambda_g A_g$
Ellipsoidal	Equal	Equal	Equal	EEE	$\lambda DAD^\top$
Ellipsoidal	Equal	Equal	Variable	EEV	$\lambda D_g A D_g^\top$
Ellipsoidal	Variable	Equal	Variable	VEV	$\lambda_g D_g A D_g^\top$
Ellipsoidal	Variable	Variable	Variable	VVV	$\lambda_g D_g A_g D_g^\top$

Table 3.4: Multivariate models available in the `R`-package `mclust`

The number of clusters to use, G, and the model (which restrictions to put on the density) is chosen via BIC by specifying a vector of possible choices. The package also has a lot of other options, please refer to [Fraley and Raftery, 2006] for details.

Compared to kernel smoothing, described in section 3.6, model based clustering has the advantage of being quicker computational-wise, but at the cost of some flexibility.

In some sense, this method identifies subpopulations – although there is no such thing as clearly distinct subpopulations; the limit is vague and overlapping occurs – in the dataset.

CHAPTER 4

Methods for Comparison of the Models

In this chapter, different ways to compare the methods to estimate haplotype frequencies will be presented.

One way is to calculate how much probability mass a model assigns to the observations, and thereby says something about how much probability mass we have not yet observed. This is motivated by [Robbins, 1968].

Another method is to compare the marginals for each locus, i.e. how the marginal distribution under the model is compared to what has been observed.

The results of the actual comparisons will be presented in chapter 5.

4.1 Observed Probability Mass

Often a reasonable assumption is that more haplotypes than the observed exist. So how much of the probability mass have we observed? And how much is still "left" to be observed? Besides being partly a philosophical question, this is an interesting statistical question where the answer provides very important information. The statistical setup is as follows and is based on the one made by [Robbins, 1968].

Let $E_1, E_2, \ldots$ be the types in a population with corresponding probabilities $p_1, p_2, \ldots$ of being sampled with $\sum_i p_i = 1$. Now assume that we take N independent observations from the population with replacement such that E_i occurs x_i times meaning that $\sum_i x_i = N$. Define

$$\varphi_i = \begin{cases} 1 & \text{if } x_i = 0, \\ 0 & \text{if } x_i > 0. \end{cases}$$

Then the unobserved probability mass is

$$U = \sum_i p_i \varphi_i.$$

The number of singletons, doubletons etc., in this context are haplotypes only observed once, twice etc., are defined to be

$$M_j = \sum_i [x_i = j]_I$$

such that M_1 is singletons, M_2 doubletons etc., and $[A]_I$ is the Iverson bracket defined as

$$[A]_I = \begin{cases} 1 & \text{if } A \text{ is true}, \\ 0 & \text{if } A \text{ is false}. \end{cases}$$

One interesting estimate of U is proposed by [Robbins, 1968], who motivates the following estimate by supposing an additional observation is made. He shows that for

$$V = \frac{M_1}{N+1} \quad \text{and} \quad W = U - V$$

then

$$\mathbf{E}[W] = 0 \quad \text{and} \quad \mathbf{Var}[W] < \frac{1}{N+1}. \tag{4.1}$$

Note that the probability of not observing E_i, because of independence, is $P(\varphi_i = 1) = (1 - p_i)^N$, hence

$$\mathbf{E}[U] = \mathbf{E}\left[\sum_i p_i \varphi_i\right] = \sum_i p_i \mathbf{E}[\varphi_i] = \sum_i p_i (1 - p_i)^N. \tag{4.2}$$

Under some regularity conditions, a limiting consistent estimate of $\mathbf{Var}[U]$ is found in [Bickel and Yahav, 1986], such that

$$\frac{\sqrt{N}(V - \mathbf{E}[U])}{\sqrt{N}\hat{\sigma}} \to N(0, 1)$$

for

$$\hat{\sigma}^2 = \frac{M_1}{N^2} - \frac{(M_1 - 2M_2)^2}{N^3}.$$

This can be used to create an $(1-\alpha)100\%$ approximate confidence interval.

Because it is a probability, it might be a bit artificial to assume normality. To acknowledge this fact, we can transform the probability via a logit-transformation and construct a symmetric interval on the logit-scale. This can be done by the δ-method.

First define

$$Y = h(V).$$

Then by the (informal) δ-method, we have that

$$\begin{aligned}\mathbf{E}[Y] &\approx h(\mathbf{E}[V]) + \frac{h''(\mathbf{E}[V])}{2}\mathbf{Var}[V] \\ \mathbf{Var}[Y] &\approx (h'(\mathbf{E}[V]))^2 \mathbf{Var}[V].\end{aligned}$$

And then an $(1-\alpha)100\%$ approximate confidence interval for $Y = h(V)$ is

$$\mathbf{E}[Y] \pm z_{\frac{\alpha}{2}} \mathbf{Var}[Y].$$

In our case, choose $h = \text{logit}$, such that

$$p = h^{-1}(x) = \text{logit}^{-1}(x) = \frac{1}{1+\exp(-x)}.$$

Then an $(1-\alpha)100\%$ approximate confidence interval for V is

$$\frac{1}{1+\exp\left(-\mathbf{E}[Y] \pm z_{\frac{\alpha}{2}} \mathbf{Var}[Y]\right)}$$

where estimates of $\mathbf{E}[Y]$ and $\mathbf{Var}[Y]$ is obtained by using V and $\hat{\sigma}^2$.

4.1.1 Verification Through Simulation

To check this estimate, several simulations have been made using a script conceptually equal to the R-script in listing 4.1, where all probabilities are equal.

```
# (Unknown) population size
Q <- 1000

# (Unknown) sampling probabilities
probs <- rep(1 / Q, Q)

# (Known) number of observations
N <- 100

# (Known) sample
observed <- sample.int(Q, N, replace = TRUE, prob = probs)
observed.unique <- unique(observed)
observed.counts <- table(observed)

# (Known) number of observed singletons
singletons <- which(observed.counts == 1)
M1 <- length(singletons)

# Robbins, 1968
estimated.unobserved.mass <- M1 / (N + 1)

# Because this is a simulation, we can calculate the true mass
true.unobserved.mass <- 1 - sum(probs[observed.unique])

deviation <- (estimated.unobserved.mass - true.unobserved.mass) /
    sqrt(true.unobserved.mass)
```

Listing 4.1: R-script to simulate the estimate given in [Robbins, 1968]

The script in listing 4.1 only makes one estimate. In order to evaluate the method, several estimates have to be simulated. The parameters are also varied in order to see how robust the estimate is.

The measures used to evaluate the estimate are the deviation depicted in a boxplot and sample quantiles compared to theoretical quantiles depicted in a QQ-plot.

The deviation is calculated as showed in the last line of the R-code in listing 4.1.

To calculate the sample quantiles, the expected value of the estimate is because of (4.1) equal to (4.2). The empirical variance is used.

Now let V_k denote the estimated unobserved probability masses for $k = 1, 2, \ldots, K$ where K is the number of simulations. Then by the Central

Limit Theorem, we have that

$$\frac{\hat{V}_k - \mathbf{E}[V]}{\sqrt{\mathbf{Var}[V]}}$$

are the sample quantiles.

Four different probability schemes for `probs` $= p_1, p_2, \ldots, p_Q$, the (unknown) sampling probabilities, are used (with subsequent normalization):

1. `equal`: All equal
2. `uniform`: Sampled from a uniform distribution
3. `normal`: Sampled from a normal distribution with mean 1000 and variance 10^2, and afterwards all values below 0 are changed to 10^{-5}
4. `10-90`: The first 10% are 10 times as likely as the last 90%, in this setup this corresponds to the first 100 gets unnormalized "probability" 10 and the last 900 gets unnormalized "probability" 1 – note that this assignment actually makes sense because the probabilities are normalized afterwards

Now the deviation depicted and sample quantiles are simulated as follows:

1. For each of the four probability schemes, then for `Q` $= 1000$, i.e. the (unknown) true population size, the deviation is found $10,000$ times for each `N` $= N_+$ from 100 to 1000 by 100, i.e. the values $100, 200, \ldots, 1000$.
2. For each of the four probability schemes, then for `Q` $= 10000$, i.e. the (unknown) true population size, the deviation is found $10,000$ times for each `N` from 10 to 1000 by 100, i.e. the values $10, 110, \ldots, 910$.

This setup is implemented in the `R`-file `unobserved-simulations.R` in the supplementary material (refer to section 1.5).

It turns out that all the probability schemes give very similar results – and they are all impressive. Because of this, only plots for one probability scheme, `uniform`, are included (the others are in the supplementary material and are automatically generated when executing `unobserved-simulations.R`). The boxplots can be seen in figure 4.1 and figure 4.2. The QQ-plot can be found in figure 4.3.

In order to asses the confidence intervals found both with the asymptotic variance estimate and the logit-transformed, the confidence interval coverage have been simulated.

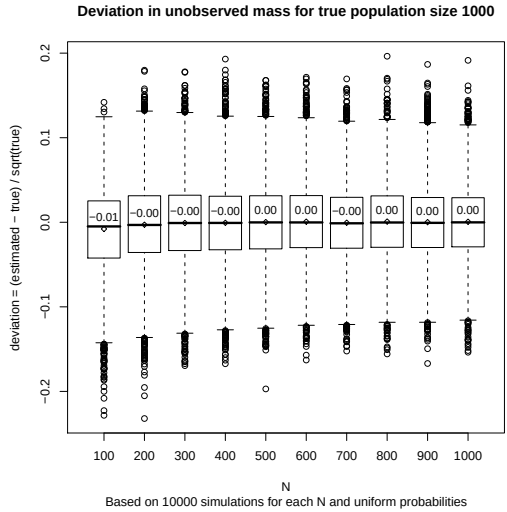

Figure 4.1: Deviation in estimate by [Robbins, 1968] for unobserved probability mass for uniformly distributed unknown probabilities.

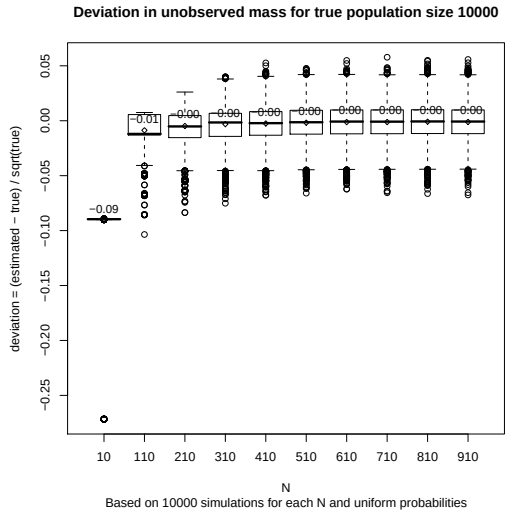

Figure 4.2: Deviation in estimate by [Robbins, 1968] for unobserved probability mass for uniformly distributed unknown probabilities.

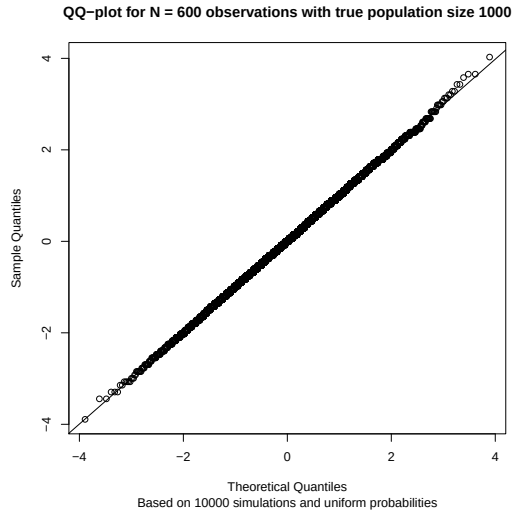

Figure 4.3: QQ-plot of estimate by [Robbins, 1968] for unobserved probability mass for uniformly distributed unknown probabilities.

Using the same setup (with the same choices for `Q`, `N`'s etc.) as calculating the deviance, then V and the corresponding confidence intervals are calculated at each simulation. The coverage is how often the confidence interval contained U.

Again, the results were quite similar for all the probability schemes. Because of this, only plots for one probability scheme, `uniform`, are included (the others are in the supplementary material and are automatically generated when executing `unobserved-simulations.R`) in figure 4.4 and figure 4.5. As seen, there is no need to use the δ-method because the estimate of the coefficient of variation $\frac{V}{\hat{\sigma}}$ is very small, hence the logit-transformation is almost linear in the confidence interval. This is also why the two confidence intervals are so similar.

It is quite amazing how well the estimate performs, and it is definitely a measure worth using to either evaluate a haplotype frequency estimation model or to fit parameters in such a model (e.g. the smoothing parameter in kernel smoothing introduced in section 3.6).

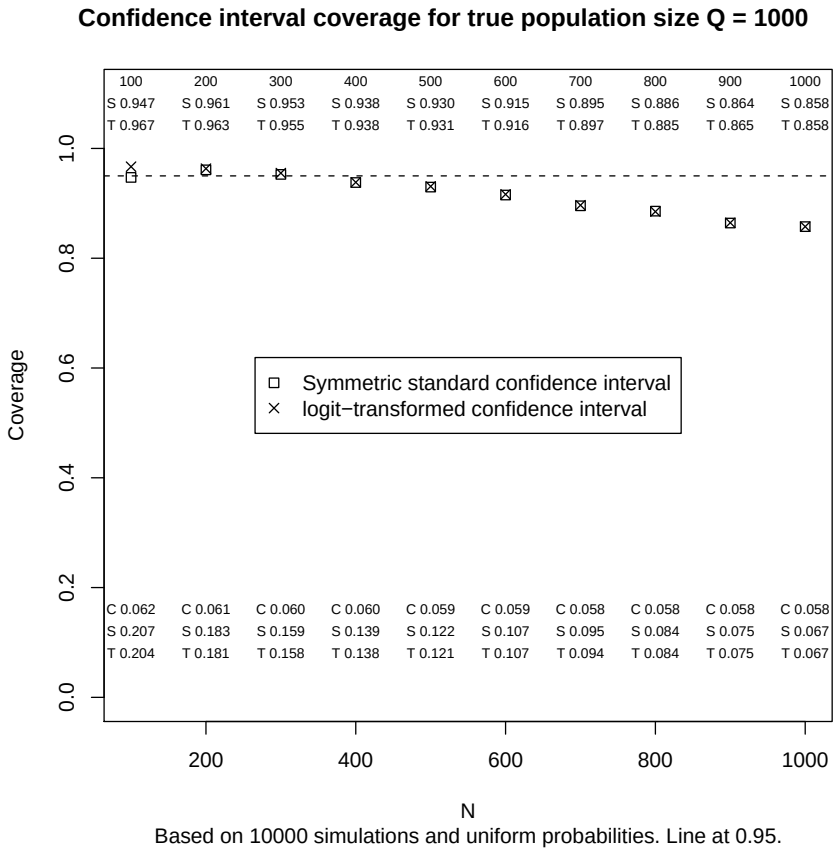

Figure 4.4: Confidence interval coverage for unobserved probability mass for uniformly distributed unknown probabilities. The number at the top is the value of `N`. The one just below with a preceding "S" is the coverage for the traditional confidence interval. The one below that, with the preceding "T" is the coverage for the logit-transformed confidence interval. The numbers in the bottom is the average width of the "S" and "T" intervals and the "C" number is an estimate of the coefficient of variation $\frac{\hat{\sigma}}{V}$

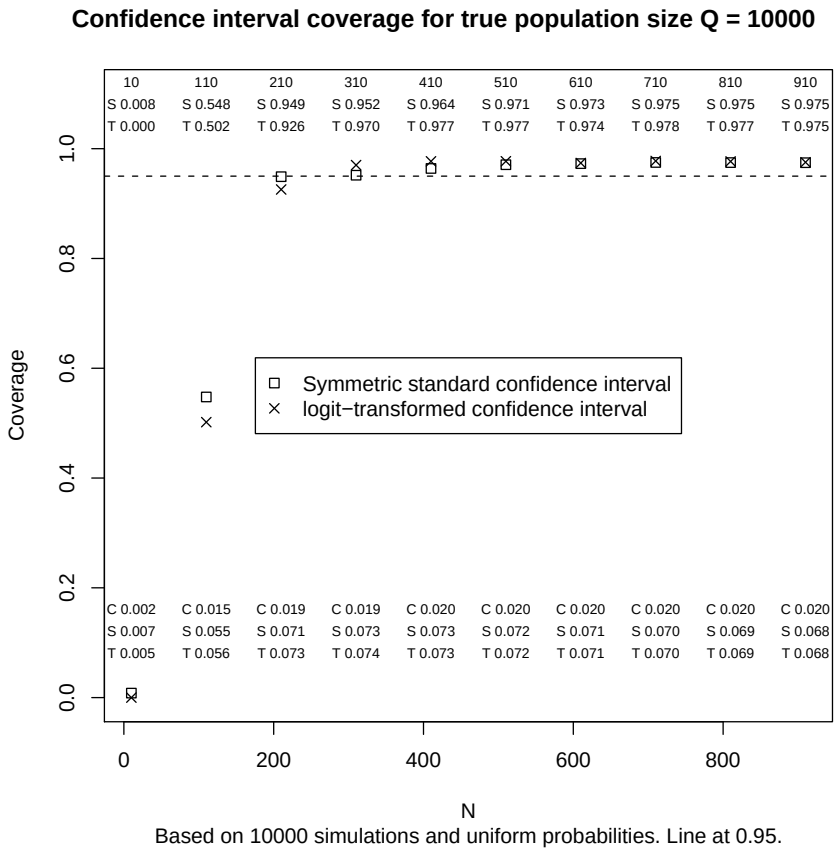

Figure 4.5: Confidence interval coverage for unobserved probability mass for uniformly distributed unknown probabilities. Please refer to the caption of figure 4.4 for further explanation of the plot.

4.2 Marginals

As stated in the introduction to this chapter, it is important that a model predicts marginals close to the observed. In this thesis the focus will only be on the single and pairwise marginals, because the marginals of higher dimension contains a lot of zeros.

A deviance measure to assess how close the predicted and observed marginals are, has been calculated.

4.2.1 Deviance

The deviance can be used to measure the difference between the table of observed counts and predicted counts under some model $\mathcal{M}_0$, say. This can be used both with one-way-tables (single locus marginal distributions), two-way-tables (pairwise loci marginal distributions) etc.

Because only the single and pairwise marginals will be checked, the focus will be on these, but the idea generalizes quite naturally.

Let $\{u\}_{ij}$ be the two-way table with the observations, $\{\tilde{p}\}_{ij}$ the table of predicted probabilities under a model $\mathcal{M}_0$, $\{\hat{p}\}_{ij}$ the relative probabilities such that $\hat{p}_{ij} = \frac{u_{ij}}{u_{++}}$.

Assume for now that

$$\sum_{i,j} \tilde{p}_{ij} = 1. \tag{4.3}$$

For the pairwise marginal tables, the deviance difference is

$$d = -2 \log \left(\frac{L\left(\{\tilde{p}\}_{ij}\right)}{L\left(\{\hat{p}\}_{ij}\right)} \right)$$

where

$$L\left(\{p\}_{ij}\right) = \prod_{i,j} p_{ij}^{u_{ij}}$$

is proportional to the likelihood with the constant $\frac{u_{++}!}{\prod_{i,j} u_{ij}}$, see e.g. [Edwards, 2000, p. 15], cancelled out in the fraction, hence unnecessary to include. With

this, the deviance difference can be rewritten as

$$\begin{aligned} d &= -2\log\left(\frac{\prod_{i,j}\tilde{p}_{ij}^{u_{ij}}}{\prod_{i,j}\hat{p}_{ij}^{u_{ij}}}\right) \\ &= -2\log\left(\prod_{i,j}\left(\frac{\tilde{p}_{ij}}{\hat{p}_{ij}}\right)^{u_{ij}}\right) \\ &= -2\sum_{i,j} u_{ij}\log\left(\frac{\tilde{p}_{ij}}{\hat{p}_{ij}}\right). \end{aligned}$$

If $\mathcal{M}_0$ is true, then a reasonable approximation is that

$$d \sim \chi^2_\nu$$

where the degrees of freedom $\nu = rc - 1$ when the deviance is based on a two-way-table with r rows and c columns, i.e. the degrees of freedom equals the number of cells minus one. If the deviances is based on a one-way-table with r cells, then $\nu = r - 1$. If our assumption in (4.3) is not fulfilled, one can use $\nu = rc$ and $\nu = r$, respectively.

When the deviance is calculated for one table, then we can check if the model $\mathcal{M}_0$ can explain the data. That is, if

$$\frac{d-\nu}{\sqrt{2\nu}} < 1.96,$$

by the central limit theorem, then we have no reason to think that $\mathcal{M}_0$ cannot explain the data – corresponding to a one-side hypothesis disregarding overfitting, i.e. if $\frac{d-\nu}{\sqrt{2\nu}} < 0$.

To compare different models $\mathcal{M}_0^1, \mathcal{M}_0^2, \ldots, \mathcal{M}_0^Q$, we can sum the deviances for the one-way-tables and compare them. Equally with the two-way tables. The one with the smallest sum of deviance can then be chosen as the best model. It would of course be preferable if it is the same model having the lowest sum of deviance for both the one- and two-way tables. This approach is in no sense standard: assuming $\mathcal{M}_0$ is true, then if the deviances for the different tables were independent, the sum of the deviances would also be χ^2-distributed with the sum of the degrees of freedom as degrees of freedom, but the deviances are in no way independent. This means that we do not know anything about how the sum of deviances is distributed, but as a naive and relative comparison between the models, the approach seems reasonable.

As mentioned earlier, the idea generalizes quite naturally for tables of higher dimension. Also, if the pairwise marginals are acceptable, then the single marginals are acceptable, too.

4.2.2 How to Find Marginals

In section 4.2.1 deviance is introduced as a measure of how well marginals under a model fits the observed marginals.

Now, one could ask how the predicted marginals under a model $\mathcal{M}_0$ are found? A very reasonable question, indeed.

We could of course base the marginals on the observed haplotypes only. But the non-trivial case, where the sum of the probabilities for the observed haplotypes under a model does not add up to one, creates some quite interesting problems. When this sum does not add up to one, the marginal tables (both single marginals, pairwise marginals etc.) does not have the same count sum (count sum is found by multiplying the cell probabilities with the number of observations, N_+, and adding all these counts together) as the observed tables.

So how should this problem be dealt with? Should we normalise the probabilities? This would solve the technical problems at hand, but the rhetorical question would of course be: Do other problems than the technical ones exist? Indeed they do. Would it not be more correct to actually get the marginals based on all possible haplotypes? This would solve the technical problems about whether to normalise or not, but it also introduces a significant practical problem of computational power because the set of all possible haplotypes is huge. Before addressing it, let us examine how marginals are actually found.

Recall the notation introduced in section 1.4, e.g. a haplotype consisting of loci $L_1, L_2, \ldots, L_r$ each taking values in $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_r$, respectively. Then the set of all possible haplotypes is $\mathcal{H} = \mathcal{L}_1 \times \mathcal{L}_2 \times \cdots \times \mathcal{L}_r$ where $\times$ denotes the Cartesian product.

Single marginals (one-way-tables) for a locus L_i is then $P(L_i = a)$ for all $a \in \mathcal{L}_i$, and pairwise marginals is $P(L_i = a, L_j = b)$ for all $a \in \mathcal{L}_i$ and $b \in \mathcal{L}_j$. The marginals can be found in different ways. The next sections will give examples of this.

Only classification models as the ones introduced in section 3.5 will be dealt with.

Exact

In this section it is shown how the marginals can be found using classification models as the ones introduced in section 3.5.

Recall that the concept of considering one locus at a time as the response

variable gave the following algorithm that fits the models given by (using the formula notation of `R`)

$$
\begin{aligned}
L_{i_1} &\sim \sum_{k \notin \{i_1\}} L_k \\
L_{i_2} &\sim \sum_{k \notin \{i_1, i_2\}} L_k \\
&\vdots \\
L_{i_{r-2}} &\sim \sum_{k \notin \{i_1, i_2, \ldots, i_{r-2}\}} L_k \\
L_{i_{r-1}} &\sim \sum_{k \notin \{i_1, i_2, \ldots, i_{r-1}\}} L_k = L_{i_r}
\end{aligned}
$$

After these $r-1$ steps, the distribution for L_{i_r} is simply the marginal distribution based on the observations.

This means, that the single marginals under a model $\mathcal{M}$ is found by exploiting that $P(L_{i_r} = a_{i_r})$ for all $a_{i_r} \in \mathcal{L}_{i_r}$ is simply the marginal distribution based on the observations.

The next step for all $a_{i_{r-1}} \in \mathcal{L}_{i_{r-1}}$ is to find

$$P\left(L_{i_{r-1}} = a_{i_{r-1}}\right) = \sum_{a_{i_r} \in \mathcal{L}_{i_r}} P\left(L_{i_{r-1}} = a_{i_{r-1}} \mid L_{i_r} = a_{i_r}\right) P\left(L_{i_r} = a_{i_r}\right)$$

where $P\left(L_{i_{r-1}} = a_{i_{r-1}} \mid L_{i_r} = a_{i_r}\right)$ is provided by our classification model and $P\left(L_{i_r} = a_{i_r}\right)$ is found in the first step.

Then for all $a_{i_{r-2}} \in \mathcal{L}_{i_{r-2}}$ we have that

$$\begin{aligned}
P\left(L_{i_{r-2}} = a_{i_{r-2}}\right) &= \sum_{a_{i_{r-1}} \in \mathcal{L}_{i_{r-1}}} P\left(L_{i_{r-2}} = a_{i_{r-2}} \mid L_{i_{r-1}} = a_{i_{r-1}}\right) P\left(L_{i_{r-1}} = a_{i_{r-1}}\right) \\
&= \sum_{a_{i_{r-1}}} \sum_{a_{i_r}} P\left(L_{i_{r-2}} = a_{i_{r-2}} \mid L_{i_{r-1}} = a_{i_{r-1}}, L_{i_r} = a_{i_r}\right) P\left(L_{i_{r-1}} = a_{i_{r-1}}, L_{i_r} = a_{i_r}\right) \\
&= \sum_{a_{i_{r-1}}} \sum_{a_{i_r}} P\left(L_{i_{r-2}} = a_{i_{r-2}} \mid L_{i_{r-1}} = a_{i_{r-1}}, L_{i_r} = a_{i_r}\right) P\left(L_{i_{r-1}} = a_{i_{r-1}} \mid L_{i_r} = a_{i_r}\right) P\left(L_{i_r} = a_{i_r}\right)
\end{aligned}$$

where $P\left(L_{i_{r-2}} = a_{i_{r-2}} \mid L_{i_{r-1}} = a_{i_{r-1}}, L_{i_r} = a_{i_r}\right)$ and $P\left(L_{i_{r-1}} = a_{i_{r-1}} \mid L_{i_r} = a_{i_r}\right)$ is provided by the corresponding two classification models and $P\left(L_{i_r} = a_{i_r}\right)$ is found in the first step.

We continue like this all the way until the last step, where we for all $a_{i_1} \in \mathcal{L}_{i_1}$ have that

$$\begin{aligned} P\left(L_{i_1} = a_{i_1}\right) = \sum_{a_{i_2}} \cdots \sum_{a_{i_{r-1}}} \sum_{a_{i_r}} & P\left(L_{i_1} = a_{i_1} \mid L_{i_2} = a_{i_2}, \ldots, L_{i_r} = a_{i_r}\right) \times \\ & P\left(L_{i_2} = a_{i_2} \mid L_{i_3} = a_{i_3}, \ldots, L_{i_r} = a_{i_r}\right) \times \\ & P\left(L_{i_3} = a_{i_3} \mid L_{i_4} = a_{i_4}, \ldots, L_{i_r} = a_{i_r}\right) \times \\ & \vdots \\ & P\left(L_{i_{r-1}} = a_{i_{r-1}} \mid L_{i_r} = a_{i_r}\right) \times \\ & P\left(L_{i_r} = a_{i_r}\right) \end{aligned}$$

where all the probabilities are provided by the classification models.

As can be seen this is practically impossible for even small r.

Simulation

Assuming that we have a model $\mathcal{M}$, then sample a number of haplotypes, K, say. Then the marginals are the relative frequencies of these K samples.

A haplotype is sampled as follows:

1. Sample an a_{i_r} according to $P(L_{i_r} = a_{i_r})$, the marginal distribution of L_{i_r} based on the observations.
2. Sample an $a_{i_{r-1}}$ according to $P(L_{i_{r-1}} = a_{i_{r-1}} \mid L_{i_r} = a_{i_r})$
3. Sample an $a_{i_{r-2}}$ according to $P(L_{i_{r-2}} = a_{i_{r-2}} \mid L_{i_{r-1}} = a_{i_{r-1}}, L_{i_r} = a_{i_r})$
4. Continue doing this for $a_{i_{r-3}}, \ldots, a_{i_1}$

Instead of finding the marginals by the exact method, this simulation approach can be used as an approximation.

4.2.3 Normalised Marginal Approximation

Please notice that it is not possible to simulate marginals for all methods described in this thesis. The method described in section 3.2 is not build upon a statistical model and there is no straightforward way to simulate marginals under this model. What can be done instead, is to normalise the probabilities and use those instead. Whether that is a feasible approach or not, will be assessed in section 5.3.4.

For $\boldsymbol{h} = (h_1, h_2, \ldots, h_r) \in \mathcal{H}$ the single marginals are

$$P(L_i = j) = \sum_{\boldsymbol{h} \in \mathcal{H}: h_i = j} P(\boldsymbol{h}).$$

For the normalised estimated k single marginals $P(L_i = j)$, we then calculate the estimated counts

$$F_{ij} = N_+ P(L_i = j) \quad \text{for} \quad j = 1, 2, \ldots, k$$

which can be compared to the observed counts by using Pearson's χ^2-statistic given by

$$\sum_{j=1}^{k} \frac{(F_{ij} - N_i)^2}{F_{ij}} \sim \chi^2_{k-1}.$$

The validity of this approach is investigated by using it for the classification models and comparing with the simulated results. This is done later on in section 5.3.

4.2.4 Bootstrapping

Deviance was introduced in section 4.2.1. To calculate a deviance, all the observations were used. One can ask the relevant question: How do we estimate the standard deviation of the calculated deviance? One way of doing this is to use bootstrapping.

First assume that we have a test statistic based on the entire dataset. This could be the deviance.

Bootstrapping is a method to obtain several samples from only one sample as described in [Venables and Ripley, 1997, p. 186]. Assume that our sample consists of N_+ observations, then bootstrapping is done by drawing N_+ observations *with replacement* from the sample. Now the test statistic can be calculated for these N_+ observations. Repeating this a number of times yields a sample of test statistics, such that the standard deviation can be estimated.

This gives us a way of estimating how precise the calculated deviance is.

A way to validate a model is called in-bag training. First obtain a bootstrapping sample. Because bootstrapping has been used, some of the original observations have not been used – these are placed in a bag. Now the model is build based on a bootstrapping sample. The observations in the bag is now used to estimate a prediction error.

Bootstrapping requires a lot of observations, so the technique will not be used, although it might be good to keep in mind for larger datasets.

4.3 Deviance Comparing Predicted with Relative Frequencies

To evaluate how well a model performs, a Brier score can be calculated. In [Andersen, 2009a] a short introduction to the Brier score is given. It is simply a sum of squared deviations between the relative frequencies and the predicted probabilities.

As introduced in section 1.4, let N_+ be the number of observations, n the number of haplotypes, N_i the number of times the i'th haplotype has been observed, and p_i the probability of the i'th haplotype estimated under a model.

Then the Brier score is

$$\sum_{\text{all haplotypes}} \left(\frac{N_i}{N_+} - p_i \right)^2 = \sum_{\text{observed}} \left(\frac{N_i}{N_+} - p_i \right)^2 + \sum_{\text{unobserved}} p_i^2.$$

We can possibly estimate $\sum_{\text{unobserved}} p_i^2$ using doubletons as done in section 4.1. The problem is to estimate the expected value and the variance in order to standardise the score to be standard normal distributed.

Instead the deviance difference between the predicted probabilities and the observed is used. Similar to the description in section 4.2, we have

$$L(\boldsymbol{N}, \boldsymbol{p}) \propto \prod_{\text{all haplotypes}} \frac{p_i^{N_i}}{\left(\frac{N_i}{N_+}\right)^{N_i}} = \prod_{\text{all haplotypes}} \frac{p_i^{N_i}}{\left(\frac{N_i}{N_+}\right)^{N_i}} = \prod_{\text{all haplotypes}} \left(\frac{p_i}{\frac{N_i}{N_+}} \right)^{N_i}$$

and then the deviance is

$$\begin{aligned} -2 \log \left(L(\boldsymbol{N}, \boldsymbol{p}) \right) &= -2 \sum_{\text{all haplotypes}} N_i \log \left(\frac{p_i}{\frac{N_i}{N_+}} \right) \\ &= -2 \sum_{\text{observed}} N_i \log \left(\frac{p_i}{\frac{N_i}{N_+}} \right) \end{aligned} \tag{4.4}$$

following a χ^2-distribution. Note that $\log 1 = 0$ because

$$\frac{p_i^0}{\left(\frac{0}{N}\right)^0} = 1$$

such that the terms with the unobserved haplotypes vanish.

This means that we actually only have to sum over the predicted probabilities for the observed haplotypes.

4.4 Multinomial Distribution

Assume that

$$\mathcal{H} = \underbrace{\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}}_{\text{Observed}} \cup \underbrace{\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_q\}}_{\text{Unobserved}}$$

with $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \cap \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_q\} = \varnothing$. This partitioning of $\mathcal{H}$ will become clear in a moment. As described in section 1.4, $\boldsymbol{x}_i$ has been observed N_i times and $N_+ = \sum_{j=1}^{n} N_i$. Notice that $\boldsymbol{h}_j$ has been observed 0 times for all $j = 1, 2, \ldots, q$.

Now under a model $\mathcal{M}$ calculate $p_{x1} = P_{\mathcal{M}}(\boldsymbol{x}_1), \ldots, p_{xn} = P_{\mathcal{M}}(\boldsymbol{x}_n)$ and $p_{h1} = P_{\mathcal{M}}(\boldsymbol{h}_1), \ldots, p_{hq} = P_{\mathcal{M}}(\boldsymbol{h}_q)$. Let $\dot{x}_i$ be a random variable of the number of times $\boldsymbol{x}_i$ is observed, and similar $\dot{h}_j$ the number of times $\boldsymbol{h}_j$ is observed. Then assuming that

$$\left(\dot{x}_1, \ldots, \dot{x}_n, \dot{h}_1, \ldots, \dot{h}_q\right) \sim \text{Multinomial}\left(N_+, (p_{x1}, \ldots, p_{xn}, p_{h1}, \ldots, p_{hq})\right)$$

we have by [Ratnaparkhi, 2006] that the probability of the observed under the model $\mathcal{M}$ is

$$\begin{aligned} P\left(\cap_{i=1}^{n}\{\dot{x}_i = N_i\}, \cap_{j=1}^{q}\{\dot{h}_j = 0\}\right) &= \frac{N_+!}{\prod_{i=1}^{n} N_i! \prod_{j=1}^{q} 0!} \prod_{i=1}^{n} p_{xi}^{N_i} \prod_{j=1}^{q} p_{hj}^{0} \\ &= \frac{N_+!}{\prod_{i=1}^{n} N_i!} \prod_{i=1}^{n} p_{xi}^{N_i}. \end{aligned}$$

This can be used for comparing models but referring to (4.4), we can see that these two methods are equivalent for relative comparisons.

CHAPTER 5

Results for Comparison of the Models

In this chapter the models for estimating Y-STR haplotype frequencies will be evaluated using the methods introduced in chapter 4.

The main focus has been on the classification models, where the predicted marginals under the model have been found using simulation. Because of this, the results for this class of models are the ones presented first. The other methods have mainly been assessed by the unobserved probability mass and the deviance between the predicted probabilities and relative frequencies as described in section 4.3.

Before describing the actual results, a section about implementation notes will come.

5.1 Implementation

This section gives some comments on different aspects of implementing the models described earlier. It is in general highly recommended to refer to the actual implementations in the supplementary material found as described in section 1.5.

All implementations are made such that it is easy to add extra datasets be-

sides *berlin*, *dane*, and *somali* as used here. Basically, two files are needed: one of the dataset in the compact format and one in the extended format. They are of course equivalent (and the conversion is made automatically by logic in the file `include-transform-data.R`), but it is made in this way to avoid computing either format every time.

When the data exists in the correct formats (refer to the datasets `R/data/*.csv` in the supplementary material, section 1.5, for the exact format), the file `include-datasets.R` can be changed to include that particular dataset as seen in listing 5.1. The vector `dataset.names` solely specifies the datasets to be analysed, i.e. a dataset is analysed if and only if it is included in the vector `dataset.names`. This is achieved by using a loop for each analysis as exemplified in listing 5.2.

```
1 dataset.names <- c("berlin", "dane", "somali")
```

Listing 5.1: The names of the datasets to include in the analyses.

```
1 for (i in 1:length(dataset.names)) {
2   dataset.name <- dataset.names[i]
3
4   # Load dataset with logic from include-dataset-logic.R, basically:
5   dataset.compact <- get.compact.dataset(dataset.name)
6
7   # Perform analysis
8 }
```

Listing 5.2: Automatically running analyses on the specified datasets.

5.1.1 Classification Models

The classification models described in section 3.5 all have the same form, and in `R` this can be exploited by making a generic classification implementation, where the specific function, e.g. `rpart`, `ksvm`, or `polr`, is just an argument.

Such a generic framework has been implemented. Specification of a classification model is done by specifying a modelling, score, and prediction function in `include-classification-model-specification.R`. In listing 5.3 this is done for the classification trees, and it is also shown how to choose which models to actually use, such that it is possible to leave some out for some reason.

The loop in listing 5.2 for classification methods is then expanded to the one seen on listing 5.4.

```
rpart.classification.model.function <- function(formula, dataset) {
  return(rpart(formula, data=dataset))
}

generic.classification.model.score.iterator <- function(fit.list,
    score.function) {
  list.length <- length(fit.list)
  scores <- numeric(list.length)

  for (i in 1:list.length) {
    fit <- fit.list[[i]]
    scores[i] <- score.function(fit)
  }

  return(scores)
}

rpart.classification.model.score <- function(fit.list) {
  return(generic.classification.model.score.iterator(fit.list,
    function(fit) {
      # Also checks for NULL
      if (!is(fit, "rpart")) stop("fit is not a rpart object")

      leaves <- which(fit$frame$var == "<leaf>")
      devar <- fit$frame$dev[leaves]
      return(sum(devar))
    }))
}

rpart.classification.model.prediction <- function(fit, dataset) {
  if (!is(fit, "rpart")) stop("fit is not a rpart object")
  return(predict(fit, dataset))
}

classification.model.names <- c("rpart")
classification.model.functions <- c(rpart.classification.model.function)
classification.model.scores <- c(rpart.classification.model.score)
classification.model.predictions <- c(rpart.classification.model.prediction)

# This controls which models to use, similar to the dataset.names-vector
classification.model.use <- c(1)
```

Listing 5.3: Specification of classification models.

```
for (i in 1:length(dataset.names)) {
  dataset.name <- dataset.names[i]

  # Load dataset with logic from include-dataset-logic.R, basically:
  dataset.compact <- get.compact.dataset(dataset.name)

  for (j in 1:length(classification.model.use)) {
    m <- classification.model.use[j]
    classification.model.name <- classification.model.names[m]
    classification.model.function <- classification.model.functions[[m]]
    classification.model.score <- classification.model.scores[[m]]
    classification.model.prediction <- classification.model.predictions[[m]]

    # Perform analysis
  }
}
```

Listing 5.4: Automatically running classification analyses on the specified datasets.

Based on [Karatzoglou et al., 2006], the primary choice of packages for support vector machines was `kernlab`. Because of some problems (of technical nature, sometimes it could fit and other times not, which was reported to the maintainer of the package), the package `e1071` was used instead.

The tuning functionality from the `e1071`-library has been used to specify some additional models. It tunes hyperparameters by using a grid of possible parameter values, e.g. the `cp`-parameter (complexity) in `rpart`. The different classification models used can be seen in table 5.1 and for exact specification, please refer to the file `include-classification-model-specification.R`.

5.1.2 Kernel Smoothing

Kernel smoothing is described in section 3.6. Here a smoothing parameter λ is mentioned several times, but it is not mentioned how to choose an appropriate one. The way λ is found, is to find one such that the model covers the probability predicted by [Robbins, 1968] as described in section 4.1. This is done by setting two values of λ, one causing covering too much and one causing covering too little, and then use the `uniroot`-function in `R` to find a λ between the two initial values causing coverage sufficiently close to the estimated.

5.2 Properties of the Datasets

In table 5.2 the properties associated to unobserved probability mass discussed in section 4.1 are shown.

`rpart`	Classification trees as specified in listing 5.3.
`rpart-tun-dev`	As `rpart`, but with tuning each model to find the best `cp`-value among $0.0001, 0.001, 0.01$.
`svm`	Support vector machine functionality from the `e1071`-library using `C-classification`, with `radial` kernel using `cost = 10`, `gamma = 0.1`. The score is $\|\boldsymbol{a}_p - \boldsymbol{a}_o\|$, where $\|\cdot\|$ denotes a norm, $\boldsymbol{a}_p$ is the predicted alleles for locus serving as the response, and $\boldsymbol{a}_o$ is the observed alleles.
`svm-tun-perf`	As `svm`, but with tuning each model to find the best `gamma` from the values $10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}$ and the best `cost` from the values $10, 100$, and using the `performance`-value (classification error) from the tuning as score.
`svm-tun-n`	As `svm-tun-perf`, but using the norm score as `svm`.
`polr`	Ordered logistic regression using `polr` from the `MASS`-library with the logistic link.
`lrm`	Ordered logistic regression using `lrm` from the `Design`-library.

Table 5.1: The classification models described. For exact specification, please refer to the file `include-classification-model-specification.R` in the supplementary material.

5.3 Classification Models

The classification models approach was introduced in section 3.5. Refer to table 5.1 for explanation of the different models.

5.3.1 Unobserved Probability Mass

The amount of unobserved probability mass (refer to section 4.1 for details) predicted by the different classification methods can be seen in table 5.3.

5.3.2 Single Marginals

These results are based on the method described in section 4.2.

By sampling 100,000 haplotypes under each of the classification models, the marginals predicted by a model can be estimated as described in section 4.2.2.

	Estimate	CV	95% confidence interval
berlin	0.364	0.061	[0.321; 0.408]
dane	0.602	0.078	[0.510; 0.694]
somali	0.277	0.120	[0.212; 0.342]

Table 5.2: Table of estimates of unobserved probability mass. The coefficient of variation, CV, is calculated using estimates. The confidence interval is approximate and found using an asymptotic normality. Please refer to section 4.1 for further details.

	berlin	*dane*	*somali*
Estimate	0.364	0.602	0.277
CI	[0.321; 0.408]	[0.510; 0.694]	[0.212; 0.342]
`rpart`	0.478	0.71	0.42
`rpart-tun-dev`	0.451	0.71	0.42
`svm`	0.526	0.792	0.43
`svm-tun-perf`	0.633	0.873	0.578
`svm-tun-n`	0.617	0.847	0.513
`polr`	0.639	0.886	
`lrm`	0.637	0.887	

Table 5.3: Unobserved probabilities using classification methods. The missing values indicate that the method did not succeed for some reason.

The deviances of such a simulation can be seen in table 5.6 for *berlin*, and the corresponding p-values can be seen in table 5.4. The deviances for *dane* can be seen in table 5.5 and for *somali* in table 5.7. These tables and the p-values for all three datasets can be found in the supplementary material in the file `results-classification.pdf`.

	rpart	rpart-tun-dev	svm	svm-tun-perf	svm-tun-n	polr	lrm
DYS19	1.000	1.000		2.192e-05	0.000	0.934	0.545
DYS389I	0.909	0.968	0.000	0.000e+00	0.000	0.797	0.752
DYS389II	1.000	1.000	0.000	0.000e+00	0.000	1.000	1.000
DYS390			0.000				0.998
DYS391	0.984	0.999	0.000	0.000e+00	0.000	0.995	0.995
DYS392	1.000	1.000	0.000	0.000e+00	0.000	1.000	1.000
DYS393	1.000	1.000	0.000	0.000e+00	0.000	1.000	

Table 5.4: P-values for each loci for ***berlin*** dataset for observed single marginals vs. simulated single marginals for the classification method specified in each row. The missing value is for the locus where the observed marginal distribution has been used.

	rpart	rpart-tun-dev	svm	svm-tun-perf	svm-tun-n	polr	lrm
DYS19	0.346	0.298	45.324	2.842	1.719	0.930	0.276
DYS389I	0.001	0.011	429.444	324.124	265.818	0.050	0.111
DYS389II	0.003	0.016		361.196	361.559		0.052
DYS390			99.076		112.374	0.206	1.180
DYS391	0.067	0.059	343.926	368.196	442.527	0.625	0.006
DYS392	0.362	0.280	427.035	452.997		0.119	0.704
DYS393	0.053	0.050	386.076	488.271	573.695	0.131	
DYS437	0.413	0.438	18.896	5.096	9.121	0.695	1.061
DYS438	0.186	0.198	369.918	289.744	369.075	1.844	6.609
DYS439	0.017	0.059	244.164	232.435	236.350	0.024	0.028

Table 5.5: Deviances for each loci for ***dane*** dataset for observed single marginals vs. simulated single marginals for the classification method specified in each row. The missing value is for the locus where the observed marginal distribution has been used.

	rpart	rpart-tun-dev	svm	svm-tun-perf	svm-tun-n	polr	lrm
DYS19	0.004	0.034		29.124	990.758	1.306	4.031
DYS389I	0.545	0.255	125.077	2441.356	1697.606	1.016	1.206
DYS389II	0.049	0.025	1845.990	1861.456	1192.436	0.471	0.215
DYS390			800.304				0.275
DYS391	0.158	0.015	2321.402	1752.595	3779.290	0.073	0.068
DYS392	0.431	0.233	6184.992	6477.672	5626.380	0.114	0.048
DYS393	0.048	0.088	2156.867	2530.527	2280.330	0.135	

Table 5.6: Deviances for each loci for ***berlin*** dataset for observed single marginals vs. simulated single marginals for the classification method specified in each row. The missing value is for the locus where the observed marginal distribution has been used.

	rpart	rpart-tun-dev	svm	svm-tun-perf	svm-tun-n	polr	lrm
DYS19	3.522e-02	0.072		71.795			
DYS389I	1.424e-02	0.004	1265.321	1276.953	1129.937		
DYS389II			806.328		775.947		
DYS390	3.447e-03	0.014	600.665	351.185	1225.655		
DYS391	1.811e-01	0.251	1046.742	951.723	839.583		
DYS392	1.188e-02	0.033	44.416	86.231	70.017		
DYS393	6.674e-03	0.004	977.487	966.367	510.341		
DYS437	5.596e-04	0.003	1.753	13.765	22.098		
DYS438	1.417e-02	0.036	671.610	1086.726	552.016		
DYS439	4.272e-04	0.007	249.942	380.558	347.514		

Table 5.7: Deviances for each loci for *somali* dataset for observed single marginals vs. simulated single marginals for the classification method specified in each row. The missing value is for the locus where the observed marginal distribution has been used.

	berlin	*dane*	*somali*
`rpart`	1.236	1.449	0.268
`rpart-tun-dev`	0.650	1.408	0.425
`svm`	13434.632	2363.861	5664.264
`svm-tun-perf`	15092.731	2524.901	5185.304
`svm-tun-n`	15566.800	2372.237	5473.109
`polr`	3.114	4.623	
`lrm`	5.842	10.025	

Table 5.8: Sum of deviances for observed single marginals vs. simulated single marginals for the classification method specified in each row.

The sum of the deviances for all datasets can be seen in table 5.8.

5.3.3 Pairwise Marginals

	berlin	*dane*	*somali*
`rpart`	Inf	768.444	772.936
`rpart-tun-dev`	458.284	770.766	779.840
`svm`	Inf	24971.264	53797.852
`svm-tun-perf`	Inf	Inf	Inf
`svm-tun-n`	Inf	Inf	Inf
`polr`	1761.153	Inf	
`lrm`	1778.344	2126.960	

Table 5.9: Sum of deviances for observed pairwise marginals vs. simulated pairwise marginals for the classification method specified in each row.

Similar as for the single marginals, the pairwise marginals have been investigated through simulation. The sum of the deviances for all datasets can be seen in table 5.9, and in table 5.10 the number of loci rejected for each method and dataset is shown. In the latter table, the row labled "Maximum" is the number of pairs of loci, i.e.

$$(r-1)+(r-2)+\cdots+1=\sum_{i=1}^{r-1} i=\frac{r(r-1)}{2}=\frac{r!}{2!(r-2)!}=\binom{r}{2},$$

which can also be deducted easily if counting the times the `for`-loop creating the tables as shown in listing 5.5 is ran through, namely $\sum_{i=1}^{r-1}\sum_{j=i+1}^{r} 1$.

	berlin	*dane*	*somali*
Maximum	21	45	45
`rpart`	18	35	34
`rpart-tun-dev`	17	34	36
`svm`	21	45	45
`svm-tun-perf`	21	45	45
`svm-tun-n`	21	45	45
`polr`	21	45	
`lrm`	21	45	

Table 5.10: Number of loci rejected for observed pairwise marginals vs. simulated pairwise marginals for the classification method specified in each row.

```
1 for (i in 1:(r-1)) {
2   for (j in (i+1):r) {
3     # create pairwise marginals between i and j
4   }
5 }
```

Listing 5.5: An example of a `for`-loop to create pairwise marginals

5.3.4 Assessing Validity of Normalising Predicted Marginals

	berlin	*dane*	*somali*
`rpart`	Inf	Inf	Inf
`rpart-tun-dev`	Inf	Inf	Inf
`svm`	Inf	Inf	Inf
`svm-tun-perf`	Inf	Inf	Inf
`svm-tun-n`	Inf	Inf	Inf
`polr`	Inf	Inf	
`lrm`	Inf	Inf	

Table 5.11: Sum of deviances for simulated pairwise marginals vs. predicted pairwise marginals for the classification method specified in each row.

As described in section 4.2.3, it is not possible to sample haplotypes according to their predicted probabilities under all models, e.g. the surveying frequency method introduced in section 3.2. One way to get marginals under a model instead of using simulation, is to normalise the probabilities and get predicted counts of the observed haplotypes, which are then used to create the single and pairwise marginals. Whether that is a feasible approach or not, can be investigated using the classification methods, and then compare such predicted marginals with the simulated marginals. The result of this for

	berlin	*dane*	*somali*
Maximum	21	45	45
`rpart`	21	45	45
`rpart-tun-dev`	21	45	45
`svm`	21	45	45
`svm-tun-perf`	21	45	45
`svm-tun-n`	21	45	45
`polr`	21	45	
`lrm`	21	45	

Table 5.12: Number of loci rejected for simulated pairwise marginals vs. predicted pairwise marginals for the classification method specified in each row.

single marginals using *berlin* can be seen in table 5.13. The other datasets give similar results, which can be found in the supplementary material in the file `results-classification.pdf`. As it can be seen, it really gives no meaning to use such approximation to the predicted marginals, neither to compare with the observed marginals. This is supported by the results in table 5.11 and table 5.12.

	rpart	rpart-tun-dev	svm	svm-tun-perf	svm-tun-n	polr	lrm
DYS19	6146.435	3613.047	2427.304	6056.612	101496.550	11626.910	11333.339
DYS389I	6800.568	6003.647	47391.265	284175.694	213863.411	15433.152	15224.203
DYS389II	10305.569	10280.674	957939.025	1025902.574	950110.699	19670.806	19607.415
DYS390	4819.944	3644.867	183202.984	3174.681	7256.472	7926.205	7156.653
DYS391	7247.513	6387.390	792277.877	608403.021	1071823.289	8984.748	9093.375
DYS392	9781.260	10381.544	1759601.916	1723088.658	1796827.694	27489.492	28663.305
DYS393	17173.738	18181.792	570667.028	596488.035	588687.332	28473.844	26948.848

Table 5.13: Deviances for each loci for ***berlin*** dataset for simulated single marginals vs. predicted single marginals for the classification method specified in each row.

5.3.5 Deviance Comparing Predicted with Relative Frequencies

The results in table 5.14 are based on the comparison method described in section 4.3. As described in section 4.4, this comparison is equivalent to using the probability under a multinomial model for relative comparisons, so the latter has been omitted.

	berlin	*dane*	*somali*
`rpart`	1730.27	1043.21	662.39
`rpart-tun-dev`	1616.90	1043.21	662.39
`svm`	1627.41	883.21	474.45
`svm-tun-perf`	2284.33	1302.00	802.80
`svm-tun-n`	2482.69	1235.06	786.74
`polr`	2589.24	1502.96	
`lrm`	2592.33	1510.01	

Table 5.14: Comparing deviance between relative frequencies and predicted probabilities found using classification methods.

5.4 Frequency Surveying

The frequency surveying approach was introduced in section 3.2. This method is only evaluated by the unobserved probability mass as described in section 4.1 and the deviance comparing the predicted probabilities with the relative frequencies as described in section 4.3. The results can be seen in table 5.15 and table 5.16, respectively. Also refer to the model specific model control made when describing the method in section 3.2.

	berlin	*dane*	*somali*
Estimate	0.364	0.602	0.277
CI	[0.321; 0.408]	[0.510; 0.694]	[0.212; 0.342]
Full regression model		0.643	
Reduced regression model	0.479	0.703	0.335

Table 5.15: Unobserved probabilities using frequency surveying. The missing values indicate that the method did not succeed for that particular model.

	berlin	*dane*	*somali*
Full regression model		549.40	
Reduced regression model	2056.88	937.43	619.81

Table 5.16: Comparing deviance between relative frequencies and predicted probabilities found using frequency surveying. The missing values indicate that the method did not succeed for that particular model.

5.5 Ancestral Awareness

Ancestral awareness was introduced in section 3.4. The predicted unobserved probability mass (refer to section 4.1) can be seen in table 5.17 and the deviance comparing the predicted probabilities with the relative frequencies (refer to section 4.3) can be seen in table 5.18.

Stop criteria	Mutation	*berlin*	*dane*	*somali*
4 ancestors	Yes	−3.364	−0.663	−2.533
5 ancestors	Yes	−5.984	−1.565	−4.759
6 ancestors	Yes	−6.002	−2.037	−5.082
0.1 threshold	Yes	−6.073	−3.134	−6.85
0.15 threshold	Yes	−3.635	−2.66	−7.1
0.2 threshold	Yes	−2.408	−1.34	−7.721
4 ancestors	No	0.448	0.741	0.423
5 ancestors	No	0.307	0.689	0.349
6 ancestors	No	0	0.59	0.283
0.1 threshold	No	0.39	0.589	0.182
0.15 threshold	No	0.454	0.668	0.22
0.2 threshold	No	0.466	0.713	0.246

Table 5.17: Table of unobserved probability mass using ancestral awareness. ***berlin*** estimated to 0.364 (and with 95% probability within $[0.321; 0.408]$), ***dane*** estimated to 0.602 (and within 95% probability within $[0.510; 0.694]$), and ***somali*** estimated to 0.277 (and within 95% probability within $[0.212; 0.342]$). Also refer to table 5.2.

5.6 Kernel Smoothing

The kernel smoothing was introduced in section 3.6. In table 5.19 the λ causing the estimated amount of unobserved probability mass (refer to section 4.1 for details) and the deviance comparing the predicted probabilities with the relative frequencies (as described in section 4.3) is shown.

Stop criteria	Mutation	*berlin*	*dane*	*somali*
4 ancestors	Yes	-1133.158	154.804	-115
5 ancestors	Yes	-1832.135	-147.204	-331.068
6 ancestors	Yes	-1904.99	-260.113	-385.526
0.1 threshold	Yes	-1714.93	-399.85	-526.29
0.15 threshold	Yes	-1037.173	-306.212	-515.36
0.2 threshold	Yes	-566.644	-70.764	-534.509
4 ancestors	No	1681.957	1389.787	1340.967
5 ancestors	No	796.162	989.062	747.293
6 ancestors	No	0	639.172	400.191
0.1 threshold	No	1498.875	811.1	345.078
0.15 threshold	No	1965.47	1093.083	369.113
0.2 threshold	No	2151.174	1221.66	555.189

Table 5.18: Comparing deviance between relative frequencies and predicted probabilities found using ancestral awareness.

	λ	Deviance
berlin	1.12	2634.36
dane	0.86	801.19
somali	1.63	918.39

Table 5.19: Results for the kernel smoothing. The optimal λ's found is in the λ-column. The deviance-column is the deviance between relative frequencies and predicted probabilities found using the kernel smoothing.

5.7 Model Based Clustering

The model based clustering was introduced in section 3.7. The R-library `mclust` uses BIC to determine the number of clusters to use given a range of allowable model choices. Here the allowable number of clusters was specified to be between 1 and 20. The estimated amount of unobserved probability mass (refer to section 4.1 for details) can be seen in table 5.20, the deviance comparing the predicted probabilities with the relative frequencies (as described in section 4.3) in table 5.21, the number of clusters in table 5.22, and the BIC-scores in table 5.23.

For all the datasets and all the methods, several plots were made. Figure 5.1 shows the relative frequencies compared to the smoothed probabilities using the first two sample principal components. Note that the smoothed probabilities are not normalised. In figure 5.2 the clusters are shown. Similar figures for the other datasets and models can be found in the supplementary material.

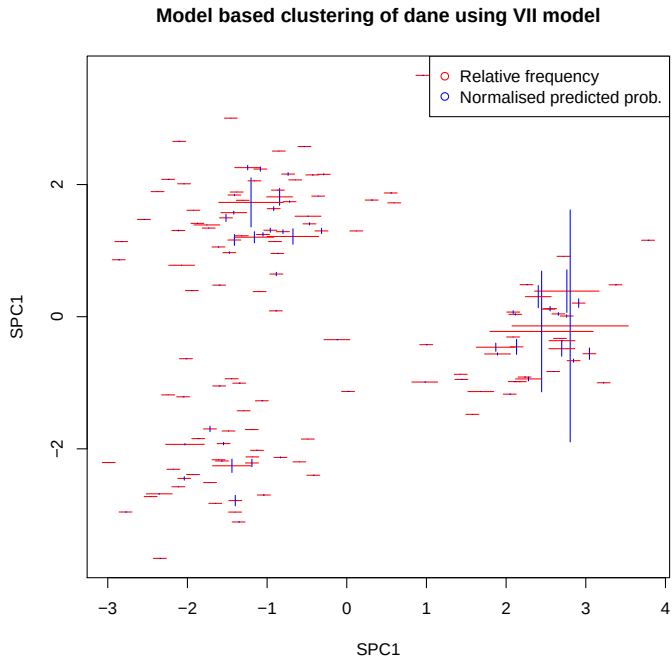

Figure 5.1: Comparing the relative frequencies compared to the (non-normalised) smoothed probabilities for ***dane*** using the VII-model.

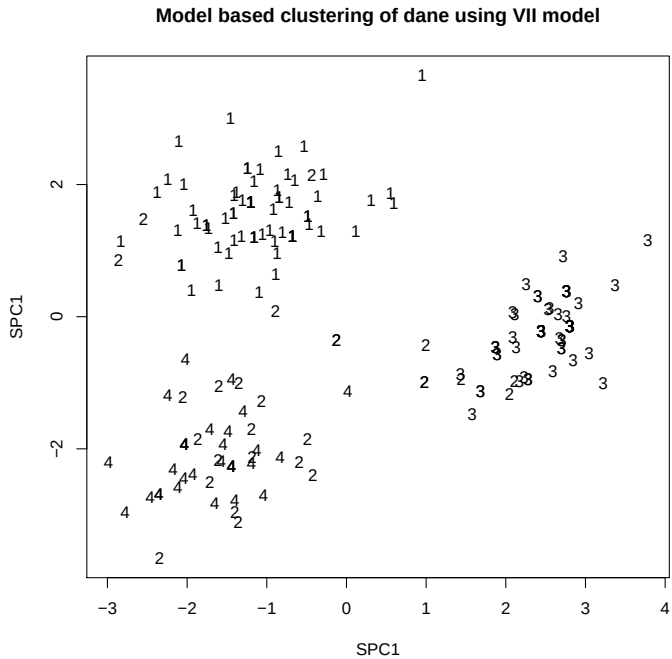

Figure 5.2: The observations for *dane* marked by the number of the cluster to which they belong using the VII-model. As seen in table 5.22 there are 4 clusters in this case.

	berlin	*dane*	*somali*
Estimate	0.364	0.602	0.277
CI	[0.321; 0.408]	[0.510; 0.694]	[0.212; 0.342]
EII	0.475	0.875	-73.752
VII	0.681	0.639	-1.27
EEI	-5.543	-5.981	-1872.639
EVI		0.593	
VEI	0.214	-4.379	
VVI		-1.255	
EEE	-3.875	-7.84	-992.039
EEV	-49.342	-88.285	-499.667
VEV	-5.515	-1.802	
VVV		-2.577	

Table 5.20: Unobserved probabilities using model based clustering. The missing values indicate that the method did not succeed for some reason.

5.8 Comparing Models

Out of the classification models, it seems like the classification trees outperform the other classification methods based on the marginals. Especially support vector machines look poor in this regard, which is probably because of a wrong kernel and wrong kernel parameters. A grid search among possible kernels and corresponding parameters could be made to get better performance.

The frequency surveying also performs well in the comparisons made for that one, but the important area of marginal deviance is not checked due to the lack of underlying model.

Both these approaches overestimate the unobserved probability mass, corresponding to underestimating the observed probability mass.

Besides the comparison methods for classification models used here, it is also possible to approximate $\mathbf{E}[K_j]$, where K_j denotes the number of haplotypes observed $j+1$ times, like done for frequency surveying in section 3.2.3 by using simulation. If N_+ denotes the number of observations in the dataset, then N_+ haplotypes can be simulated M times, and the average of the observed K_j's in the simulated data approximate $\mathbf{E}[K_j]$ for large M. A prototype of an implementation of this can be found in `classification-count-tables.R` in the supplementary material (refer to section 1.5).

The idea behind kernel smoothing is worthy of a note, although more model control has to be done for this model in order to validate it. The huge drawback for this model is the inefficiency of calculating a probability. This leads

	berlin	*dane*	*somali*
EII	2214.84	1630.88	-144.19
VII	3548.33	1505.86	1162.00
EEI	-103.37	408.50	-1397.08
EVI		1329.43	
VEI	2926.29	1069.19	
VVI		1182.89	
EEE	103.28	253.49	-1174.59
EEV	-2796.34	-649.09	-1048.61
VEV	1593.86	1128.98	
VVV		1105.19	

Table 5.21: Comparing deviance between relative frequencies and predicted probabilities found using model-based-clustering. The missing values indicate that the method did not succeed for some reason.

	berlin	*dane*	*somali*
EII	15	4	19
VII	3	4	2
EEI	16	17	19
EVI		3	
VEI	3	4	
VVI		3	
EEE	16	17	16
EEV	13	7	5
VEV	3	2	
VVV		2	

Table 5.22: Number of clusters used. The missing values indicate that the method did not succeed for some reason.

to the model based clustering where there is a huge difference between the different models, and the performance in general is poor, probably because of a bad discretization of the continuous density.

In conclusion, this chapter shows that it is really important to validate a model control in a number of different ways. Because of this, the classification models with classification trees as a worthy representative has to be emphasised because of its appealing theoretical nature.

	berlin	*dane*	*somali*
EII	-9884.18	-3608.05	-2272.23
VII	-10608.55	-3498.69	-2592.00
EEI	-7656.69	-3179.17	-1067.07
EVI		-3390.13	
VEI	-10025.38	-3109.00	
VVI		-3254.03	
EEE	-7999.42	-3259.07	-1353.19
EEV	-6577.25	-3191.75	-1792.07
VEV	-9101.20	-3513.34	
VVV		-3536.53	

Table 5.23: The BIC-score for each model. The missing values indicate that the method did not succeed for some reason.

CHAPTER 6

Evidence

In section 1.2 calculating statistical evidence was briefly discussed. Calculating statistical evidence is very important in e.g. crime cases, so good and reliable methods have to be available. The actual calculations of the likelihood ratio for Y-STR are more complex than for autosomal STR because of the loci dependency. It also means that the methods developed for the autosomal case cannot be used with Y-STR haplotypes directly, although some ideas and principles can be reused.

In this chapter, evidence for Y-STR haplotypes will be dealt with shortly in order to further motivate why it is so important to have reliable methods of estimating probabilities for Y-STR haplotypes.

6.1 Two Contributors

In section 1.2 a short example of a LR was given and the expression

$$LR = \frac{P\left(T \ominus \boldsymbol{h}_s\right)}{\sum_{(\boldsymbol{h}_1,\boldsymbol{h}_2)\equiv T} P\left(\boldsymbol{h}_1\right) P\left(\boldsymbol{h}_2\right)}$$

for one unknown contributor was deducted.

To assess the number of terms in the denominator, let r be the number of

loci, let $T = (T_1, T_2, \ldots, T_r)$ be the trace, where T_i is a set of alleles such that $|T_i| \in \{1, 2\}$. As earlier, let $\boldsymbol{h}_s$ denote the suspect's haplotype and $\boldsymbol{h}_1$ denote the one additional contributor's haplotype. Also let $\mathcal{H}_T = T_1 \times T_2 \times \cdots \times T_r$ be the Cartesian product of $T_1, T_2, \ldots, T_r$. In the non-trivial case, a $j \in \{1, 2, \ldots, r\}$ exists such that $|T_j| = 2$ with $T_j = \{a_1, a_2\}$, say. Let $T_j' = \{a_1\}$ (such that one of the alleles is removed) and

$$\mathcal{H}_T' = T_1 \times \cdots \times T_{j-1} \times T_j' \times T_{j+1} \times \cdots \times T_r$$

Now the denominator of LR can be written as

$$\begin{aligned}\sum_{(\boldsymbol{h}_1, \boldsymbol{h}_2) \equiv T} P(\boldsymbol{h}_1) P(\boldsymbol{h}_2) &= \sum_{\boldsymbol{h}_1 \in \mathcal{H}_T} P(\boldsymbol{h}_1) P(T \ominus \boldsymbol{h}_1) \\ &= 2 \sum_{\boldsymbol{h}_1 \in \mathcal{H}_T'} P(\boldsymbol{h}_1) P(T \ominus \boldsymbol{h}_1)\end{aligned}$$

by exploiting the fact that the second contributor's haplotype is uniquely determined by the first contributor's. Note that if k denotes the number of loci in the trace with only one allele, and we assume that we have the non-trivial case with $0 \leq k < r$, we have that

$$|\mathcal{H}_T| = \prod_{i=1}^{r} |T_i| = 2^{r-k}$$

such that

$$|\mathcal{H}_T'| = \frac{|\mathcal{H}_T|}{2} = 2^{r-k-1} \leq 2^{r-1}.$$

This means that for r loci, a maximum of $2 \cdot 2^{r-1} = 2^r$ haplotype frequencies have to be calculated, e.g. is $2^{10} = 1024$ and $2^{17} = 131,072$.

If a trace has two contributors with no known suspects, the two most likely contributers can be chosen to be

$$\arg\max_{\boldsymbol{h}_1 \in \mathcal{H}_T'} P(\boldsymbol{h}_1) P(T \ominus \boldsymbol{h}_1).$$

The same formulation can be used for a trace with a suspect and two additional contributors by subtracting the suspect's haplotype from the trace. An example of this strategy will now be given.

Example 6.1 (Identifying two contributors to a trace). In this example the dataset ***dane*** has been used together with the classification trees (refer to section 3.5.1) for estimating haplotype frequencies.

Say that the contributors' true (unknown) haplotypes are

$$\begin{aligned}\boldsymbol{h}_1 &= (14, 12, 29, 25, 11, 13, 13, 14, 12, 11) \quad \text{and} \\ \boldsymbol{h}_2 &= (13, 14, 32, 22, 10, 11, 13, 14, 10, 11).\end{aligned}$$

The names of the loci are not relevant for this example, so they have been omitted. Then the known trace is

$$T = (\{14,13\},\{12,14\},\{29,32\},\{25,22\},\{11,10\},\\ \{13,11\},\{13\},\{14\},\{12,10\},\{11\}).$$

Here the number of loci in the trace with only one allele is $k = 3$, so the number of haplotype probabilities to calculate is $2 \cdot 2^{10-3-1} = 2 \cdot 2^6 = 2 \cdot 64 = 128$. Iterating through these different possibilities gives a top-3, where the contributors giving the highest probability are

$$\begin{aligned} \boldsymbol{h}_1 &= (14,12,29,25,11,13,13,14,12,11) \quad \text{and} \\ \boldsymbol{h}_2 &= (13,14,32,22,10,11,13,14,10,11) \quad \text{results in} \\ P(\boldsymbol{h}_1)P(\boldsymbol{h}_2) &= 5.551 \cdot 10^{-9}, \end{aligned}$$

the contributors giving the second highest probability are

$$\begin{aligned} \boldsymbol{h}_1 &= (14,12,29,22,11,13,13,14,12,11) \quad \text{and} \\ \boldsymbol{h}_2 &= (13,14,32,25,10,11,13,14,10,11) \quad \text{results in} \\ P(\boldsymbol{h}_1)P(\boldsymbol{h}_2) &= 5.186 \cdot 10^{-11}, \end{aligned}$$

and the contributors giving the third highest probability are

$$\begin{aligned} \boldsymbol{h}_1 &= (14,14,32,22,11,11,13,14,10,11) \quad \text{and} \\ \boldsymbol{h}_2 &= (13,12,29,25,10,13,13,14,12,11) \quad \text{results in} \\ P(\boldsymbol{h}_1)P(\boldsymbol{h}_2) &= 5.359 \cdot 10^{-13}. \end{aligned}$$

As seen, the true contributors are the ones giving the highest probabilities.□

4	5	6	7	8	9	10	11
0.0375	0.0181	0.0092	0.0091	0.0049	0.0035	0.0014	0.0018
12	13	14	15	16	17	18	19
0.0015	0.0011	0.0010	0.0009	0.0003	0.0002	0.0001	0.0003
20	21	22	23	24	25	27	29
0.0001	0.0004	0.0002	0.0002	0.0001	0.0002	0.0003	0.0001
30	31	40	42	45	49	58	
0.0001	0.0001	0.0002	0.0001	0.0001	0.0001	0.0001	

Table 6.1: Number of simulations causing a rank greater than or equal to 4. The integer in the odd numbered rows is the true contributors' rank and the numbers in the even numbered rows are how many times it happened in the simulations converted to a frequency. E.g. in 3.75% of the simulations, the true contributors' haplotype probabilities were ranked 4. Notice that in general, the higher the rank, the lower the frequency.

In example 6.1 the true contributors gave the highest probability. To assess whether this is always the case, a simulation study has been made.

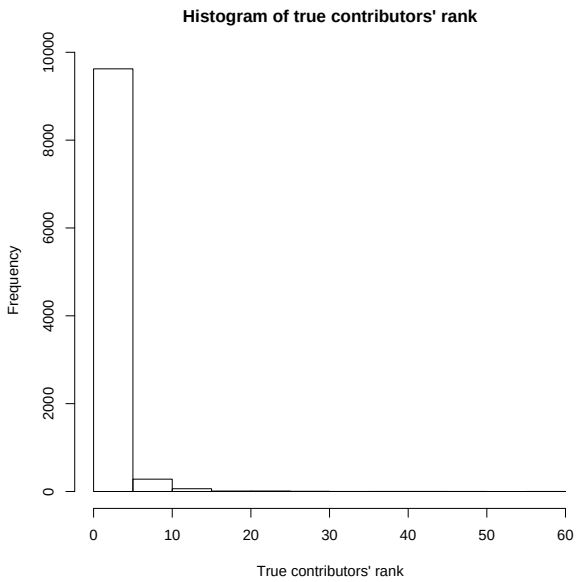

Figure 6.1: A histogram of the ranks of the true contributors haplotype probabilities in the simulation study. The plot is auto-scaled by `R`, so the reason of the wide horizontal axis is because of outliers.

Simulating $10,000$ experiments as the one in example 6.1, still with ***dane*** and classification trees, the sample mean for the rank of the true contributors' haplotype probabilities was 1.828 with a sample variance of 2.203^2. So on average the true contributors lie in top-2 based on this experiment.

A histogram of the ranks can be seen in figure 6.1. The plot is auto-scaled by `R`, so the reason of the wide horizontal axis is because of outliers. A table of number of simulations causing a rank greater than or equal to 4 can be seen in table 6.1. Notice that in general, the higher the rank, the lower the frequency. A table of sizes of $\mathcal{H}'_T$ in the cases, where the true contributors' haplotype probabilities where not in top-10, can be seen in table 6.2. Notice that the probability does not increase with the size of $\mathcal{H}'_T$.

Lastly, figure 6.2 gives expressive statistics about how many of the cases the true contributors appeared in top-x for x from 1 to 20. As seen, the probability increases quickly and is over 90% already for top-3.

The file `classification-evidence-2-contributors.R` contains the code used to perform the simulation study. Example 6.1 has also been made in this way, actually it is just the result of one of the simulations with a verbose output.

16	32	64	128	256
0.125	0.135	0.260	0.333	0.146

Table 6.2: The frequencies of the size of $\mathcal{H}'_T$ in the 96 cases out of the 10,000 simulations corresponding to a bit below 0.1% where the true contributors' haplotype probabilities were not in top-10, i.e. they had a rank strictly grater than 10. E.g. in 12.5% of the times where the true contributors' haplotype probabilities where not in top-10, the size of $\mathcal{H}'_T$ was 16. Notice that the probability does not increase with the size of $\mathcal{H}'_T$.

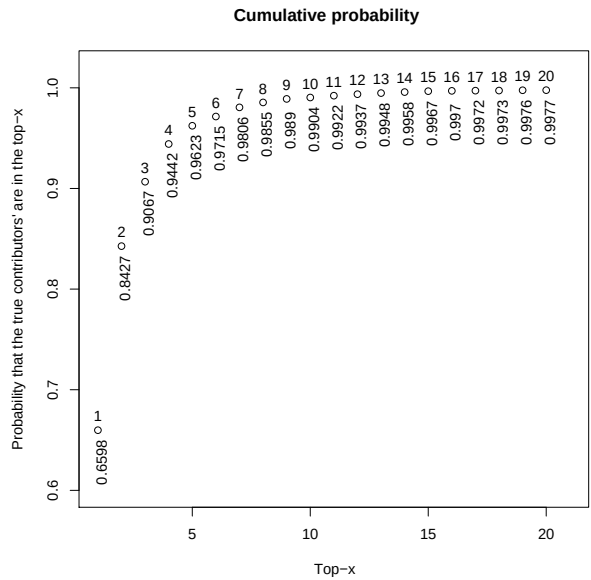

Figure 6.2: The probability that the true contributors' were in top-x for x from 1 to 20.

6.2 n Contributors

Instead of just having two contributors as described in section 6.1, a generalised form for n contributors has been described in [Wolf et al., 2005]. This approach will be discussed in this section.

To use the same notation as done in the article, let n be the number of unknown contributors and m the number of loci. Let $E_{t,i}$ be the set of alleles on the i'th locus in the trace and similar $E_{s,i}$ for the suspect and $E_{k,i}$ for the known contributors. Let $A_{n,i}$ denote the set of alleles carried by n individuals on the i'th locus.

Let

$$P_n\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right)=P\left(\bigcap_{i=1}^{m}\{W_i\subseteq A_{n,i}\subseteq V_i\}\right)$$

denote the probability that $A_{n,i}$ is included in V_i and contains $W_i\subseteq V_i$ for all $i=1,2,\ldots,m$. Then as [Wolf et al., 2005, equation (2)], we have that

$$LR=\frac{P_n\left(\bigcap_{i=1}^{m}\{E_{t,i};E_{t,i}\setminus(E_{s,i}\cup E_{k,i})\}\right)}{P_{n+1}\left(\bigcap_{i=1}^{m}\{E_{t,i};E_{t,i}\setminus E_{k,i}\}\right)}.$$

If the loci were independent, which we know they are not, we would get that

$$P_n\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right)=\prod_{i=1}^{m}P_n\left(V_i;W_i\right).$$

Now, assume without loss of generality, that $W_i=\{1,2,\ldots,k_i\}$ and $V_i=\{1,2,\ldots,s_i\}$ with $0\leq k_i\leq s_i$, such that

$$\begin{aligned}
P_n\left(\bigcap_{i=1}^{m}\{V_i;\varnothing\}\right) &= P\left(\bigcap_{i=1}^{m}\{A_{n,i}\subseteq V_i\}\right)\\
&= P\left(\left[\bigcap_{i=1}^{m}\{A_{n,i}\subseteq V_i\}\right]\cap\left[\bigcap_{i=1}^{m}\{W_i\subseteq A_{n,i}\}\right]\right)+\\
&\quad P\left(\left[\bigcap_{i=1}^{m}\{A_{n,i}\subseteq V_i\}\right]\cap\left[\bigcap_{i=1}^{m}\{W_i\subseteq A_{n,i}\}\right]^{C}\right)\\
&= P\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right)+\\
&\quad P\left(\left[\bigcap_{i=1}^{m}\{A_{n,i}\subseteq V_i\}\right]\cap\left[\bigcup_{i=1}^{m}\{W_i\not\subseteq A_{n,i}\}\right]\right)
\end{aligned}$$

where

$$
\begin{aligned}
& P\left(\left[\bigcap_{i=1}^{m}\{A_{n,i}\subseteq V_i\}\right]\cap\left[\bigcup_{i=1}^{m}\{W_i\not\subseteq A_{n,i}\}\right]\right)\\
&= P\left(\left[\bigcap_{i=1}^{m}\{A_{n,i}\subseteq V_i\}\right]\cap\left[\bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}\{\{j\}\not\subseteq A_{n,i}\}\right]\right)\\
&= P\left(\bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}\left\{\{j\}\not\subseteq A_{n,i}\cap\bigcap_{r=1}^{m}\{A_{n,r}\subseteq V_i\}\right\}\right)\\
&= P\left(\bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}\left\{\bigcap_{r=1}^{m}\{\{j\}\not\subseteq A_{n,r}\subseteq V_i\}\right\}\right)\\
&= P\left(\bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}\left\{\bigcap_{r=1}^{m}A_{n,r}\subseteq\left((V_i\setminus\{j\})\cap\bigcap_{\substack{r=1\\r\neq i}}^{m}V_r\right)\right\}\right)
\end{aligned}
$$

such that

$$
\begin{aligned}
P_n\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right) = {} & P_n\left(\bigcap_{i=1}^{m}\{V_i;\varnothing\}\right)-\\
& P\left(\bigcup_{i=1}^{m}\bigcup_{j=1}^{k_i}\left\{\bigcap_{r=1}^{m}A_{n,r}\subseteq\left((V_i\setminus\{j\})\cap\bigcap_{\substack{r=1\\r\neq i}}^{m}V_r\right)\right\}\right)
\end{aligned}
$$

which can be interpreted as the probability that all $A_{n,i}$'s are included in the respective V_i minus the probability that at least one allele from at least one W_i is lacking.

To calculate this, the inclusion-exclusion principle will be used. From a probability theoretical point of view, it states that for events $B_1, B_2, \ldots, B_n$ it is true that

$$
P\left(\bigcup_{i=1}^{n}B_i\right)=\sum_{k=1}^{n}(-1)^{k-1}\sum_{\substack{I\subseteq\{1,2,\ldots,n\}\\|I|=k}}P\left(\bigcap_{i\in I}B_i\right)
$$

giving the well known result $P(B_1\cup B_2)=P(B_1)+P(B_2)-P(B_1\cap B_2)$ for $n=2$.

Now

$$P_n\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right) = \sum_{T\subseteq\{(i,j):1\le i\le m,1\le j\le k_i\}} (-1)^{|T|}\times P_n\left(\bigcap_{i=1}^{m}\{V_i\setminus\{j:(i,j)\in T\};\varnothing\}\right)$$

where $T=\varnothing$ gives the term $P_n\left(\bigcap_{i=1}^{m}\{V_i;\varnothing\}\right)$. Two ways of calculating this are discussed by [Wolf et al., 2005], but here only the recursive method will be described. Note that $P_n\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right)=0$ if $|W_i|>n$ for at least one i. This corresponds to more alleles on at least one locus than can be explained by the assumed n contributors. If this is not the case, the recursive formula [Wolf et al., 2005, equation (5)] states that

$$P_n\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right) = \sum_{j_1\in D_1}\cdots\sum_{j_m\in D_m} f(j_1,\ldots,j_m)\times P_{n-1}\left(\bigcap_{i=1}^{m}\{V_i;W_i\setminus\{j_i\}\}\right) \tag{6.1}$$

where $P_0(\cdot)=1$, $f(j_1,\ldots,j_m)$ denotes the frequency of the Y-STR haplotype $(j_1,\ldots,j_m)$, and

$$D_i=\begin{cases}W_i, & \text{if } |W_i|=n, \text{ and}\\ V_i, & \text{if } |W_i|<n.\end{cases}$$

As seen, it is crucial to have good models to estimate all the $f(j_1,\ldots,j_m)$'s.

The structure (factorisation of the simultaneous probability mass function) of the classification models described in section 3.5 can be exploited to gain efficiency when calculating (6.1) because not all factors depend on all the loci. If the haplotypes are represented as a DAG (directed acyclic graph) where each node is an allele on a specific locus, then the complexity of such an optimisation can be found by moralising and triangulating the DAG.

The specific case, where the n males in question are assumed to be unrelated, is discussed in [Wolf et al., 2005].

Please refer to [Wolf et al., 2005] for a discussion on the computational efficiency of the two methods.

6.2.1 Approximation with Known Error Bound

In [Wolf et al., 2005] only the exact value of $P_n\left(\bigcap_{i=1}^{m}\{V_i;W_i\}\right)$ is dealt with. Because of elegant inequalities, it is actually possible to get approximations, where the error can be bounded.

The upper bound

$$P\left(\bigcup_{i=1}^{n} B_i\right) \leq \sum_{i=1}^{n} P\left(B_i\right).$$

is given by Boole's inequality. Alternating upper and lower bounds can be found using the more general Bonferroni inequalities. For odd $m \geq 1$ we get

$$U_m = \sum_{k=1}^{m} (-1)^{k-1} \sum_{\substack{I \subseteq \{1,2,\ldots,n\} \\ |I|=k}} P\left(\bigcap_{i \in I} B_i\right) \geq P\left(\bigcup_{i=1}^{n} B_i\right)$$

and for even $m \geq 2$ we get

$$L_m = \sum_{k=1}^{m} (-1)^{k-1} \sum_{\substack{I \subseteq \{1,2,\ldots,n\} \\ |I|=k}} P\left(\bigcap_{i \in I} B_i\right) \leq P\left(\bigcup_{i=1}^{n} B_i\right).$$

Note that

$$\begin{aligned} L_m &= U_{m-1} + (-1)^{m-1} \sum_{\substack{I \subseteq \{1,2,\ldots,n\} \\ |I|=m}} P\left(\bigcap_{i \in I} B_i\right) \\ &= U_{m-1} - \sum_{\substack{I \subseteq \{1,2,\ldots,n\} \\ |I|=m}} P\left(\bigcap_{i \in I} B_i\right) \end{aligned}$$

Let

$$T_m = \begin{cases} U_m, & \text{if } m \geq 1 \text{ is odd, and} \\ L_m, & \text{if } m \geq 2 \text{ is even.} \end{cases}$$

If ε is the maximum allowed approximation error, then algorithm 2 approximates $P\left(\bigcup_{i=1}^{n} B_i\right)$ within ε of the true value.

This might be used in order to avoid calculating the entire sum as done in [Wolf et al., 2005], although it is difficult to provide an estimate of the gain of such an approximation because it depends on how quick the sum converges.

Algorithm 2 Approximating $P\left(\bigcup_{i=1}^{n} B_i\right)$ with maximal error ε

Require: $n \geq 1$, $\varepsilon \in [0,1]$
  if $\varepsilon = 0$ **then**
    return T_n
  end if
  $s \leftarrow \sum_{i=1}^{n} P(B_i)$
  if $\varepsilon = 1$ **then**
    return $T_1 = s$
  end if
  $k \leftarrow 2$
  $\delta \leftarrow 1$
  while $\delta \geq \varepsilon$ **do**
    $\delta = \sum_{\substack{I \subseteq \{1,2,\ldots,n\} \\ |I|=k}} P\left(\bigcap_{i \in I} B_i\right)$
    $s \leftarrow s + (-1)^{k-1} \delta$
    $k \leftarrow k + 1$
  end while
  return s

6.3 The κ-model

A quite new article, [Brenner, 2010], addresses the issues of how to deal with singletons in calculating evidence. A key concept in the article is to assume that all observed singletons are equally likely; this assumption will be addressed in the end of this section.

Let α denote the number of singletons in a sample of n haplotypes, and denote

$$\kappa = \frac{\alpha}{n}.$$

Using this notation, they assume that the probability of the suspect's haplotype being a singleton if another observation is made, is equal to the proportion of non-singletons in the current database given by $1 - \kappa$.

These concepts are used to derive that

$$\frac{n}{1-\kappa} \tag{6.2}$$

is the evidential strength for matching a previously unobserved haplotype.

One key assumption made in the article, for among other things deriving (6.2), is that the widely accepted single-step mutation model is not used. To quote the article itself:

> *I take the DNA sequence of a haplotype as being no more than an arbitrary name; in this respect my model differs from that of Krawczak [8].*
>
> [Brenner, 2010, p. 2]

where Krawczak [8] refers to [Roewer et al., 2000]. This assumption violates the general accepted view. In other words, assume that our database consists only of 10 observations of the haplotype $(1, 1)$. A consequence of the assumptions in [Brenner, 2010], is that the probability of observing $(1, 10)$ in the next observation is as likely as observing $(1, 2)$.

Because of this fact, the κ-model seems inadequate because it actively disregards the single-step mutation model.

At the 7^{th} International Y Chromosome User Workshop in Berlin, Germany, April 2010, it was in general urged to try not to establish the point of view that the single-step mutation model is false. It was also suggested that a consensus paper was written in order to establish a united view of the statisticians instead of giving the impression that statisticians in this area are divided.

CHAPTER 7

Recapitulation

This chapter starts by recapping some of the aspects of estimating Y-STR haplotype frequencies and calculating evidence. Afterwards some ideas for further work is described.

In this thesis, calculating evidence as introduced by a simple example in section 1.2 and later in a general form in chapter 6 has been used as motivation as to why it is so crucial to be able to estimate Y-STR haplotype frequencies.

Several methods to estimate Y-STR haplotype frequencies have been described in chapter 4. In the beginning of this chapter, some desirable properties of such models are stated, e.g. consistency. Then the frequency surveying approach from [Roewer et al., 2000] is described and some problems with the approach are pointed out. Afterwards, three new models and a class of models are developed. The class of models is classification models where classification trees, support vector machines, and ordered logistic regression are examples of instances of this class. The three models developed besides classification models are ancestral awareness, kernel smoothing, and model based clustering.

Methods to compare the models are developed in chapter 4. One of the comparison methods are the amount of predicted unobserved probability mass, based on an estimate given in [Robbins, 1968]. This estimate is verified through a simulation study. Also, comparing single and pairwise marginals

using deviance and comparing predicted and relative frequencies using deviance are described.

The results of the comparisons are given in chapter 5. Not all the methods developed for estimating Y-STR haplotype frequencies in chapter 4 give satisfying results, but the classification models (especially using classification trees) described in section 3.5 seems like a reasonable model. Although classification models exhibit a theoretically beautiful structure, not all of them incorporate the prior knowledge we have (like the single step mutation model), which is a point of criticism.

Besides having an accurate model, in practise it is also important to have a model that can be explained to a judge by forensic geneticists in court. We claim that the form of the classification models described in section 3.5 are at least as easy to understand and explain as the frequency surveying approach described in section 3.2.

The methods are build and verified upon small datasets, and it would of course be valuable to have access to larger dataset similar to the one which `http://www.yhrd.org` is based on. Unfortunately that dataset is not publicly available due to some problems with the rights. A straightforward intermediate solution is to make the datasets already available as articles and supplementary material available, so that the work done by `http://www.yhrd.org` by gathering the data does not need to be done by every statistician with interest in this field.

7.1 Further Work

In this section some proposals for further work are presented.

7.1.1 Statistical Model Incorporating Genetic Knowledge

The surveying approach described in section 3.2 lacks an underlying statistical model, but incorporates prior knowledge. The classification trees model is build on a statistical model and perform quite well, but does not incorporate prior knowledge. Being able combine these to create a true statistical model that incorporates genetic knowledge is desirable.

One direction to go to achieve this goal, is to try to smooth contingency tables according to the single step mutation model. This might be usable in both learning a graphical model with the PC-algorithm and subsequent estimation using this model.

7.1.2 Y-STR Mixtures

In section 1.2 and section 6.1 evidence for Y-STR mixtures were dealt with in a simple case of two contributors and in general for n contributors in a qualitative way. Taking that as a starting point, there are several directions to go.

One direction is to pursue an improvement in regards to computational runtime of the generalised qualitative method described in section 6.2 based on [Wolf et al., 2005]. One approach is briefly mentioned in section 6.2.1, where the idea is to use approximations instead of exact values, and to top it all off, the maximum error committed can be controlled.

Another important direction is to use quantitative information instead of only the qualitative. If for example there is a lot more DNA material from one contributor than the other, the EPG (electropherogram) reflects this, but the qualitative data do not. This important information should be used to perform a better separation of the profiles.

7.1.3 Subpopulations

Although there is no fixed boundary between subpopulations, there is a vague partitioning of the humans such that some Y-STR haplotypes are more com-

mon in some part of the world than in the rest. An ambitious goal would be to define subpopulation effects for Y-STR and maybe even include them in models for Y-STR haplotype frequency estimation.

7.1.4 Extended Models

Problems such as mutations, drop-ins, and drop-outs were mentioned briefly in section 1.1. In a perfect model for calculating evidence and separating mixtures, these problems have to be addressed.

7.1.5 Signal Processing

Another interesting area is to try to model the signal in the EPG. This can be used to estimate parameters for drop-outs and similar phenomena.

Bibliography

Alan Agresti. *Categorical Data Analysis*. Wiley, 2. edition, 2002. 59, 62, 63

Mikkel Meyer Andersen. Graphical Models – A MAT4-project in the use of graphical models to estimate haplotype frequencies. An 8th Semester-project at Department of Mathematical Sciences, Aalborg University, Denmark (available in Danish only), 2009a. URL `http://people.math.aau.dk/~mikl/mat4/projekt/`. 20, 21, 45, 46, 91

Mikkel Meyer Andersen. Report on Stay Abroad at La Trobe University, Melbourne, Australia. A Review of The Course Content, 2009b. 21, 66

Adelchi Azzalini. *Statistical Inference – Based on the Likelihood*. Chapman & Hall, 1996. ISBN 0-412-60650-X. 59, 61

P. J. Bickel and J. A. Yahav. On estimating the Total Probability of the Unobserved Outcomes of an Experiment. *Lecture Notes-Monograph Series – Adaptive Statistical Procedures and Related Topics*, 8:332–337, 1986. URL `http://www.jstor.org/stable/4355542`. 76

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1. edition, 1984. ISBN 978-0412048418. 56, 58

Charles H. Brenner. Fundamental problem of forensic mathematics – The evidential value of a rare haplotype. *Forensic Science International: Genetics*, In Press, Corrected Proof, 2010. ISSN 1872-4973. doi: 10.1016/j.fsigen.2009.10.013. URL `http://www.sciencedirect.com/science/article/B8CX7-4Y4W3SD-1/2/b994679273cd0d16e7afa8e4e53e8907`. 6, 124, 125

John M. Butler. *Forensic DNA Typing: Biology, Technology, and Genetics of STR Markers*. Academic Press, 2005. ISBN 978-0121479527. 15

John M. Butler. *Fundamentals of Forensic DNA Typing*. Academic Press, 2009. ISBN 978-0123749994. 15

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 1. edition, 2000. ISBN 0-521-78019-5. 64, 65, 66, 68, 69

David Edwards. *Introduction to Graphical Modelling*. Springer, 2. edition, 2000. ISBN 0-387-95054-0. 84

L. Excoffier, P. E. Smouse, and J. M. Quattro. Analysis of molecular variance inferred from metric distances among dna haplotypes: Application to human mitochondrial dna restriction data. *Genetics*, 131(2):479–491, June 1992. ISSN 0016-6731. URL `http://view.ncbi.nlm.nih.gov/pubmed/1644282`. 20

C. Fraley and A. E. Raftery. MCLUST Version 3 for R: Normal Mixture Modeling and Model-Based Clustering. Technical Report 504, Department of Statistics, University of Washington, 2006. URL `http://www.stat.washington.edu/mclust/`. Revised: January 2007, November 2007, November 2009, December 2009. 72, 73

Charlotte Hallenberg, Karsten Nielsen, Bo Simonsen, Juan Sanchez, and Niels Morling. Y-chromosome STR haplotypes in Danes. December 2004. URL `http://www.ncbi.nlm.nih.gov/pubmed/16226159`. 19, 55

Charlotte Hallenberg, Bo Simonsen, Juan Sanchez, and Niels Morling. Y-chromosome STR haplotypes in Somalis. January 2005. URL `http://www.ncbi.nlm.nih.gov/pubmed/15939170`. 19

Myles Hollander. Dependence, Tests for. In *Encyclopedia of Statistical Sciences*. John Wiley & Sons, Inc., 2006. doi: 10.1002/0471667196.ess0483.pub2. 46

Finn V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, 2. edition, 2007. ISBN 978-0-387-68281-5. 45

R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 5 edition, 2001. 23

Markus Kalisch and Peter Buhlmann. Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007. URL `http://jmlr.csail.mit.edu/papers/volume8/kalisch07a/kalisch07a.pdf`. 46

Alexandros Karatzoglou, David Meyer, and Kurt Hornik. Support Vector Machines in `R`. *Journal of Statistical Software*, 15, 2006. 96

M. Krawczak. Forensic evaluation of y-str haplotype matches: a comment. *Forensic Science International*, 118(2–3):114–115, May 2001. URL http://dx.doi.org/10.1016/S0379-0738(00)00479-5. 35

Javier M. Moguerza and Alberto Muñoz. Support Vector Machines with Applications. *Statistical Science*, 21:322–336, 2006. doi: 10.1214/088342306000000493. 64, 69

Kevin Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33:331–350, 2001. URL http://people.cs.ubc.ca/~murphyk/Papers/bnt.pdf. Can be downloaded at http://code.google.com/p/bnt/. 46

Marija J. Norušis. *PASW Statistics 18 Advanced Statistical Procedures*. Prentice Hall, 2000. ISBN 978-0-321-69057-9. 59, 62, 63

Luke Prendergast and Paul Kabaila. *STA3AS Unit Text*. La Trobe University, Australia, 2009. 23

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. URL http://www.R-project.org. ISBN 3-900051-07-0. 7

M. V. Ratnaparkhi. Multinomial Distributions. In *Encyclopedia of Statistical Sciences*. John Wiley & Sons, Inc., 2006. doi: 10.1002/0471667196.ess1697.pub2. 92

Herbert E. Robbins. Estimating the Total Probability of the Unobserved Outcomes of an Experiment. *The Annals of Mathematical Statistics*, 39 (1):256–257, 1968. URL http://www.jstor.org/stable/2238931. 6, 75, 76, 78, 80, 81, 96, 127

L. Roewer, M. Kayser, P. de Knijff, K. Anslinger, A. Betz, A. Caglià, D. Corach, S. Füredi, L. Henke, M. Hidding, H.J. Kärgel, R. Lessig, M. Nagy, V.L. Pascali, W. Parson, B. Rolf, C. Schmitt, R. Szibor, J. Teifel-Greding, and M. Krawczak. A new method for the evaluation of matches in non-recombining genomes: application to Y-chromosomal short tandem repeat (STR) haplotypes in European males. *Forensic Science International*, 114(1):31–43, October 2000. URL http://linkinghub.elsevier.com/retrieve/pii/S0379073800002875. 5, 10, 18, 19, 34, 35, 36, 37, 125, 127

G. A. F. Seber. *Multivariate Observations*. Wiley, 1984. ISBN 0-471-88104-X. 70

Terry M. Therneau and Beth Atkinson. *rpart: Recursive Partitioning*, 2009. URL http://CRAN.R-project.org/package=rpart. R port by Brian Ripley. 56, 58

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998. ISBN 0-471-03003-1. 64, 65, 68, 69

Allard Veldman. Evidential strength of Y-STR haplotype matches in forensic DNA casework. Master's thesis, Mathematisch Institut, Universiteit Leiden and Nederlands Forensisch Instituut, 2007. 36

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer, 2. edition, 1997. ISBN 0-387-98214-0. 23, 24, 27, 55, 58, 90

Andreas Wolf, Amke Caliebe, Olaf Junge, and Michael Krawczak. Forensic interpretation of Y-chromosomal DNA mixtures. *Forensic Science International*, 152(2-3):209 – 213, 2005. ISSN 0379-0738. doi: DOI:10.1016/j.forsciint.2004.07.021. URL `http://www.sciencedirect.com/science/article/B6T6W-4DTKYKY-2/2/72c703e981d875eb3df776c67f5e3fea`. 6, 10, 119, 120, 122, 123, 129

Raanan Yehezkel and Boaz Lerner. Bayesian Network Structure Learning by Recursive Autonomy Identification. *Journal of Machine Learning Research*, 10:1527–1570, 2009. URL `http://jmlr.csail.mit.edu/papers/volume10/yehezkel09a/yehezkel09a.pdf`. 46

APPENDIX A

Correlation Matrices

Correlations matrices for *berlin*, *dane*, and *somali* have been calculated using different methods.

Correlations matrices using Pearson's correlation coefficient can be found in table A.1, table A.2, and table A.3.

Correlations matrices using Kendall's τ and Spearman's ρ have been omitted, because they were basically the same as with Pearson's correlation coefficient. They can be found in the supplementary material (refer to section 1.5), where the code for generating the matrices also can be found.

	DYS19	DYS389I	DYS389II	DYS390	DYS391	DYS392	DYS393
DYS19		0.022	0.000	0.000	0.028	0.000	0.003
DYS389I	0.090		0.000	0.000	0.001	0.000	0.592
DYS389II	0.246	0.624		0.000	0.174	0.001	0.004
DYS390	0.353	0.317	0.323		0.004	0.020	0.001
DYS391	-0.086	0.128	0.053	0.111		0.000	0.047
DYS392	-0.350	0.221	-0.125	-0.091	0.319		0.709
DYS393	0.116	0.021	0.114	-0.130	-0.078	-0.015	

Table A.1: Correlation matrix for ***berlin*** using Pearson's correlation coefficient. The lower triangular matrix shows the actual correlations, and the upper triangular matrix shows the two-sided p-values from testing if the correlations are equal against being different.

	DYS19	DYS389I	DYS389II	DYS390	DYS391	DYS392	DYS393	DYS437	DYS438	DYS439
DYS19		0.001	0.000	0.000	0.303	0.000	0.049	0.000	0.079	0.000
DYS389I	0.249		0.000	0.000	0.001	0.000	0.012	0.000	0.000	0.963
DYS389II	0.341	0.784		0.000	0.066	0.733	0.011	0.000	0.094	0.290
DYS390	0.376	0.430	0.504		0.000	0.017	0.252	0.000	0.000	0.511
DYS391	-0.076	0.244	0.135	0.328		0.000	0.592	0.000	0.000	0.150
DYS392	-0.283	0.283	0.025	0.176	0.486		0.842	0.003	0.000	0.000
DYS393	0.145	0.184	0.188	-0.085	-0.040	-0.015		0.017	0.104	0.202
DYS437	-0.405	-0.640	-0.650	-0.659	-0.289	-0.221	-0.176		0.000	0.076
DYS438	-0.130	0.348	0.124	0.467	0.622	0.778	-0.120	-0.394		0.000
DYS439	-0.347	-0.003	-0.078	-0.049	0.106	0.432	-0.094	0.131	0.298	

Table A.2: Correlation matrix for ***dane*** using Pearson's correlation coefficient. The lower triangular matrix shows the actual correlations, and the upper triangular matrix shows the two-sided p-values from testing if the correlations are equal against being different.

	DYS19	DYS389I	DYS389II	DYS390	DYS391	DYS392	DYS393	DYS437	DYS438	DYS439
DYS19		0.000	0.000	0.000	0.141	0.000	0.614	0.004	0.000	0.528
DYS389I	0.361		0.334	0.000	0.067	0.000	0.337	0.886	0.000	0.375
DYS389II	-0.543	0.069		0.000	0.760	0.000	0.285	0.496	0.000	0.304
DYS390	-0.621	-0.267	0.313		0.443	0.036	0.001	0.000	0.000	0.333
DYS391	-0.104	-0.129	-0.022	0.054		0.962	0.035	0.030	0.077	0.005
DYS392	0.260	0.500	-0.303	-0.148	0.003		0.479	0.052	0.000	0.124
DYS393	0.036	0.068	0.076	-0.241	-0.149	0.050		0.000	0.153	0.576
DYS437	0.204	-0.010	-0.048	-0.336	-0.153	-0.138	0.260		0.416	0.009
DYS438	-0.799	-0.464	0.623	0.469	0.125	-0.564	0.101	-0.058		0.274
DYS439	-0.045	-0.063	0.073	-0.069	0.199	-0.109	0.040	0.185	0.078	

Table A.3: Correlation matrix for *somali* using Pearson's correlation coefficient. The lower triangular matrix shows the actual correlations, and the upper triangular matrix shows the two-sided p-values from testing if the correlations are equal against being different.