Bayesian Networks in Forensic DNA Analysis

Master's Thesis - J. J. van Wamelen

Supervisor - prof. dr. R. D. Gill

Universiteit Leiden

Preface

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth"

Sherlock Holmes , The Sign of the Four

This thesis is written by Jasper van Wamelen in order to obtain a M.Sc in applied mathematics from the university of Leiden. My work has been supervised by Richard Gill and concerns an application of probabilistic networks in the forensic analysis of mixed DNA traces. Motivation for this subject comes from the scorpion case (a Dutch murder case) in which Richard acted as an expert witness on behalf of the prosecution.

While this is a thesis on applied mathematics I've tried to keep it accessible for a broad audience. This has led to a clearly segmented thesis which is divided into 4 chapters that can be read independent of each other without too much trouble. The segmentation also means that little specific knowledge is required from the reader to understand the big picture this thesis presents. Readers who want a more deep understanding of both the mathematical and biological part will require some prior knowledge in both fields. For the mathematical part, basic probability theory theory and statistics at a Masters level should be sufficient. For biology, some knowledge about DNA and DNA profiling is beneficiary.

The final part of the thesis deals with the scorpion case and involves computations on actual data obtained from DNA traces found at the crime scene. As the case has appeared before the appeal court during the writing of this thesis, and the suspect was acquitted, parts of chapter four are missing in the online publication. A complete version of the thesis is available for academic purposes and can be obtained by sending me an e-mail at j.wamelen@gmail.com.

Contents

Introduction

Ever since Alex Jefferys introduced DNA profiling in 1984 [2] we can no longer imagine a world without DNA evidence. Forensic investigators have replaced detectives in tv series and in almost all criminal court cases DNA evidence is present. Advances in forensic science have made it possible to profile even very small traces of DNA. With the increased complexity induced by low quality traces a qualitative analysis (a match yes or no) might sometimes be inadequate. A more quantitative method of analysis which would produce a likelihood ratio between match or no match could be of help in these situations. This thesis will give an overview on the issues and difficulties involved in producing a probabilistic model for the analysis of DNA mixtures.

The thesis consists of four independent chapters. The first chapter is the most mathematical of the four and it gives an introduction to graphical models. These graphical models are network structures for random variables, on which we can perform exact inference using the junction tree algorithm.

The second chapter presents a quantitative model to produce likelihood ratios between different scenarios of origin of a DNA mixture as it was suggested by Cowell et al. in [18]. In the third chapter we will turn our attention to the implementation of this model using matlab.

In the final chapter we will apply the model from chapter 2 through the implementation from chapter 3 on data from the scorpion case. In this case one of the DNA mixtures found at the crime scene contains a trace that appears to be neither from the suspect or the victim. With the quantitative model we will look for the most probable explanation for this trace.

Chapter 1

Graphical Models

This chapter will be an introduction to the theory of graphical models. Graphical models are network structures that represent (conditional) independencies of a joint probability distribution. The idea will be to link random random variables to nodes in a graph and represent dependencies between them as the edges between the corresponding nodes. This construction was first introduced by Judea Pearl in [16]. The main goal will be to use conditional independencies between variables to find factorizations of the joint density over the graph structure.

The factorizations of a distribution according to a graph encoding the set of r.v. $\mathcal{X}$ will be useful in finding the marginal distribution $P(X)$ of a subset of random variables $X \subseteq \mathcal{X}$. We know that we can find $P(X) = \sum_{\{Y=y\}} P(X,Y)$ but as the sample space of Y increases this becomes computationally expensive as we have to compute the sum for every instance $X = x$ of X. Now if we know that the density factorizes $P(X,Y) = f(X)g(Y)$ we can calculate the sum once and use $p(X) = f(X)\sum_{\{Y=y\}} g(Y)$ to calculate the marginals.

The models we are interested in (for applications in forensic DNA analysis) can be represented by a directed acyclic graph. The graphical models over these DAGs are called Bayesian networks, as we will see later, calculations will be performed on a factorization of the sets of variables corresponding to nodes, and edges of a so-called junction tree.

The chapter will start with some preliminaries on conditional probability and graph theory. After this introduction we are ready to formally define Bayesian networks which will form the basis of the model we will use for DNA mixture analysis in the second part of the thesis. After the directed graphs we will turn our attention to the undirected graphical models, and present some results on the relation between them and their directed counterparts. It turns out that we can make a directed graph undirected without assuming (conditional) independencies that were not present in the directed graph.

We are interested in the connection between directed and undirected models because of a property that holds for the undirected ones. From an undirected graphical model we can construct a junction tree, which is a tree structure that has subsets of variables encoded by the undirected model as its nodes. The junction tree algorithm which we will present in the final section of the chapter, gives a factorization of the joint density over the junction tree which has marginal densities over the subsets of the junction tree as its factors. This is a very convenient property as it makes inference queries a lot easier.

1.1 Preliminaries

Notation

This chapter will introduce a lot of mathematical concepts, most of which are presented in the traditional framework of definitions and theorems. However at points where we need (a lot of) terminology (for instance when talking about graphs) we choose to list important concepts in *italic* instead of presenting a list of definitions.

When we are talking about random variables or sets of random variables we will use capital letters X, Y, W, Z. Actual events $\{X = x\}$ will be denoted with small letters a, b, c (so, $a = \{X = x\}$)to prevent confusion about the two. Further we assume all random variables to have finite state space unless stated otherwise. We will denote the state space of a random variable X with Ω_X.

Conditional Probability and Indepence

A conditional probability is the probability on some event a given we have observed the occurrence of some some event b. These probabilities fit the real world situation where we observe something and want to make an inference about something related but unobserved. We denote the conditional probability of a given b by $P(a|b)$.

Definition 1.1. Given a probability space (Ω, F, P) then for every a,$b \in F$ with $P(b) > 0$, the conditional probability of a given b is defined by

$$P(a|b) = \frac{P(a \cap b)}{P(b)} \tag{1.1.1}$$

From the definition of conditional probability we can derive the result better know as Bayes' theorem.

Lemma 1.2 (Bayes' theorem)**.** *Given a probability space (Ω, F, P) we get for all a,$b \in F$ with $P(b) > 0$*

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

The proof of this lemma is just a double application of (1.1.1). We can extend the concept of conditional probability to induced distributions over random variables. The object $P(X|Y)$ with X and Y r.v. will represent a set of conditional probability distribution. This means that for every instance y of Y, $P(X|Y)$ gives a probability distribution over the possible values of X.

Definition 1.3. Let two random variables X, Y on the same probability space (Ω, F, P). We can consider, for any possible value y of Y , the conditional distribution of X given $Y = y$, denoted by $D(X|Y = y)$ This is dened if $p(y) > 0$, in which case we may call y a possible value of Y. Thus, if A denotes a set of possible values for X , then $D(X|Y = y)$ attaches to A the conditional probability value $P(X \in A|Y = y)$.

In similar fashion we can state Bayes theorem in terms of conditional probability distributions.

$$P(X|Y) = \frac{P(Y|X)P(Y)}{P(X)} \tag{1.1.2}$$

Bayes' theorem is a useful tool for inferring the posterior probability of a hypothesis based on evidence and a prior belief in the probability of different hypotheses. Often we want to form a hypotheses about the world based on observable variables. If we view Bayes' theorem in this setting we get the following; Suppose we have some hypotheses H and evidence ϵ. Bayes (1.1.1) tells us that the distribution over H given ϵ is;

$$P(H|\epsilon) = \frac{P(\epsilon|H)P(H)}{P(\epsilon)}$$

To illustrate the meaning of this equation $P(H|\epsilon)$ is usually called the posterior probability. $P(H)$ is called the prior probability, and $P(\epsilon|H)$ is called the likelihood (of the evidence). The $P(\epsilon)$ is just a normalizing constant that can be computed from the requirement that $\sum_h P(H = h|\epsilon) = 1$. So we write;

$$P(H|\epsilon) \propto P(\epsilon|H)P(H) \tag{1.1.3}$$

In real world applications, for example in forensics, we expect $P(h|\epsilon)$ to be different from $P(h)$. In other words, learning about ϵ (evidence) changes the probability over h (our hypothesis). If this is not the case, meaning that $P(h|\epsilon) = p(h)$ we call the events h and ϵ independent.

Definition 1.4. Given a probability space (Ω, F, P) and two events a,$b \in F$, with $P(a), P(b) > 0$. We call a and b independent if $P(a|b) = P(a)$. We denote this by $(a \perp b)$.

Independence is a strong and useful property but in real life, we don't often encounter truly independent events. A more common property is conditional independence, where two events are independent given we've observed a third event.

Definition 1.5. Given a probability space (Ω, F, P) and three events a,b, $c \in F$, with $P(a), P(b), P(c) > 0$. We call a and c independent given c if $P(a \mid b \cap z) = P(a|c)$. We denote this $a \perp\!\!\!\perp b|c$

Definitions 1.4 and 1.5 only deal with the independence of events while this is useful, we would also like to say something about independence relations between random variables.

Definition 1.6. Let X,Y,Z be r.v. on the same probability space (Ω, F, P). We will call X conditionally independent of Y given Z under P if

$$X = x \;\perp\!\!\!\perp\; Y = y \mid Z = z \qquad \forall\, x \in \Omega_X, \;\; \forall y \in \Omega_Y, \;\; \forall z \in \Omega_Z$$

Using the properties of conditional independence between variables will be a key factor in bringing down the computational costs of calculating the joint distribution. Before we can continue we list some basic properties of conditional independent random variables.

Lemma 1.7. *Let X,Y,W,Z be r.v. on the same probability space the following properties will hold:*

- ***Symmetry,*** *$X \perp\!\!\!\perp Y|Z$ then $Y \perp\!\!\!\perp X|Z$.*

- ***Decomposition,*** *$X \perp\!\!\!\perp (Y,W)|Z$ then $X \perp\!\!\!\perp Y|Z$.*
- ***Weak union,*** *$X \perp\!\!\!\perp (Y,W)|Z$ then $X \perp\!\!\!\perp Y|(Z,W)$.*
- ***Contraction,*** *$X \perp\!\!\!\perp W|(Z,Y)$ and $X \perp\!\!\!\perp Y|Z$ then $X \perp\!\!\!\perp (W,Y)|Z$.*

There is a fifth property, but it does not hold in general. If we have a positive and continuous distribution, the following property holds as well.

Theorem 1.8. *Let X,Y,W,Z be r.v. on the same probability space If the joint density of all variables with repect to the product measure is positive en continuous we have:*

- ***Intersection***

$$X \perp\!\!\!\perp Y|(W,Z) \quad \textit{and} \quad X \perp\!\!\!\perp W|(Y,Z) \quad \textit{then} \quad X \perp\!\!\!\perp (Y,W)|Z \tag{1.1.4}$$

Proof. Suppose that $X \perp\!\!\!\perp Y|Z$ and $X \perp\!\!\!\perp Z|Y$ and that the variables have continuous density $f(a,b,c) > 0$. Then for sutable strictly positive functions g,h,k,l we have for all (a,b,c),

$$f(a,b,c) = k(a,c)l(b,c) = g(a,b)h(b,c)$$

Since the density is continuous we have for all c,

$$g(a,b) = \frac{k(a,c)l(b,c)}{h(b,c)}$$

Now if we fix $c = c_0$ for example we get $g(a,b) = m(a)n(b)$ with $m(a) = k(a,c_0)$ and $n(b) = l(b,c_0)/h(b,c_0)$. Thus,

$$f(a,b,c) = g(a,b)h(b,c) = m(a)n(b)h(b,c) \tag{1.1.5}$$

As (1.1.3) holds for all a,b,c we can conclude that $(X \perp\!\!\!\perp (Y,Z))$. $\square$

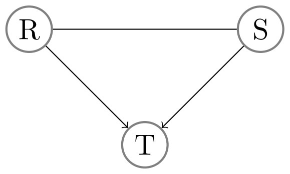
Graph Terminology

This section states some basic graph theory which we will need to define the graphical part of the graphical models in the next section. We will only list some basic terminology, for a more elaborate overview on graph theory I refere to the book *Modern graph theory* by Bollobas [?].

Definition 1.9. A graph is a pair $\mathcal{G} = (V, E)$ with V a finite set of nodes and $E \subseteq V \times V$ the set of edges.

Edges can either be directed or undirected. If for some $\mathcal{G}$ we have $v_1, v_2 \in V$ and both $(v_1, v_2) \in E$ and $(v_2, v_1) \in E$ we say that there is a undirected edge between v_1 and v_2 in $\mathcal{G}$. If there is only one edge between v_1 and v_2, say $(v_2, v_1) \in E$ we say there is a directed edge from v_2 to v_1 in $\mathcal{G}$. When we draw graphs, an undirected edge is represented by a solid line, and a directed edge by an arrow that indicates its direction.

We say the sequence $(v_1, .., v_n)$ with $v_i \in V$ is a *path* of length n in $\mathcal{G} = (V, E)$ if $\forall\ i \in \{1, .., n-1\}$ we have $(v_i, v_{i+1}) \in E$. A path that starts and ends in the same node is called a *cycle*. If a graph has no cycles, we'll call the graph *acyclic*. An example,



In this graph, we have $\mathcal{G} = (V, E)$ with $V = \{R, S, T\}$ and $E = \{(r, s), (s, r), (r, t), (s, t)\}$. The arrows also denote relations between the nodes. We call R, S *parents* of T and consequently T a *child* of R and S. We will denote the set of parents of a node x with $Pa(x)$ and the set of children with $Ch(x)$. If there is a path from a node x to a node y will call y a *descendent* (denoted $Dc(x)$)of x and x an *ancestor* ($An(y)$) of y.

If a graph contains only directed edges it is called a *directed graph.* In the same sense a graph with only undirected edges is an *undirected graph.* Intermediate variants (some edges directed, some edges undirected) are called *partially directed.* For graphical models we will focus on two types of graphs. These are *DAGs* directed acyclic graphs and undirected graphs.

Definition 1.10. Any graph $\mathcal{G} = (V, E)$ has an undirected version $\hat{\mathcal{G}} = (V\hat{E})$ where $\hat{E} = E\ \cup\ \{(v_i, v_j),\text{ if } (v_j, v_i) \in E\}$.

If two nodes are *adjacent* in $\mathcal{G}$ if they are connected in $\hat{\mathcal{G}}$. A graph in which every pair of nodes is adjacent is called *complete.* If there is a path between all nodes of $\mathcal{G}$ in $\hat{\mathcal{G}}$ we call the graph *connected.* Another important concept that we will need in undirected graphs is separation.

Definition 1.11. A set of nodes $S \in V$ is called a separator of two set $X, Y \in V$ in a graph $\mathcal{G} = (V, E)$ if every path from a node in X to a node in Y in the undirected version $\hat{\mathcal{G}}$ goes through S.

When it comes to calculations we will use another important type of graph. A *tree* is a connected graph $\mathcal{G}$ such that $\hat{\mathcal{G}}$ has no cycles. The last structure we will need is a way to view subsets of graphs.

Definition 1.12. On any graph $\mathcal{G} = (V, E)$ we can define a subgraph $\mathcal{G}_A = (A \subseteq V, E_A)$ induced by a subset A of the nodes of $\mathcal{G}$, with $E_A = \{(v_i, v_j) \in E, i, j \in A\}$.

A subgraph that is complete is called a *clique*. As we will see later, an undirected tree of cliques will play a key role in the inference on the graph. This is all we need for now, more graph theory will introduced once we start looking at junction trees and the junction tree algorithm. We are now ready to define Bayesian networks, which will be directed acyclic graphs.

1.2 Directed graphical models

1.2.1 Bayesian network definition and local Markov equivalance

One of the goals of this chapter is to represent a joint distribution P over some set of random variables $\mathcal{X} = \{X_1, ..., X_n\}$ in an efficient way. This usually requires a vast number of probabilities to be specified. For example if we take n binary-valued r.v.'s this would lead to $2^n - 1$ probabilities that need to be given. We shall see that independence relations in the distribution can be used to represent distributions more compactly.

Suppose we have a collection $\mathcal{X} = (X_v)_{v \in V}$ of random variables with finite state space Ω_{X_v} and some probability density P over them with respect to some product measure. We will use a directed acyclic Graph $\mathcal{G}$ with node set V to specify the way the joint density factorizes; in the following definition.

Definition 1.13. A Bayesian network is a pair $(\mathcal{G}, P)$ where, $\mathcal{G} = (V, E)$ is a DAG whose nodes V index some set of r.v. with finite state spaces, $\mathcal{X} = (X_v)_{v \in V}$ and P a distribution over $\mathcal{X}$ such that the edge set of $\mathcal{G}$ specifies the factorization of the density $p(x)$ of P w.r.t. some product measure:

$$p(\mathcal{X}) = \prod_{v \in V} p(X_v | Pa(X_v)). \tag{1.2.1}$$

Here we define $p(X_v|\emptyset) = p(X_v)$, and with $Pa(X_i)$ we mean the set of parental nodes of node associated X_i. We call (1.2.1) a factorization of P according $\mathcal{G}$.

The factorization of joint density has a connection to the conditional independence relations. To see that this is the case, we will need to define a way to look at the conditional independence in the setting of directed acyclic graphs.

Definition 1.14. We say that P respects the local Markov property w.r.t. $\mathcal{G}$ if we have a distribution P over $\mathcal{X} = (X_v)_{v \in V}$ a set of r.v. indexed by the node set of a DAG $\mathcal{G} = (V, E)$ such that for all nodes it holds that

$$X_v \perp\!\!\!\perp X_{Non(X_v)} | X_{Pa(X_v)}. \tag{1.2.2}$$

Here $Non(X_i)$ is the set of non-decendent nodes in $\mathcal{G}$.

Our reasoning is that both properties above are in fact equivalent and it turns out this is indeed the case, as we will prove in the next theorem.

Theorem 1.15. *P respects the local Markov property w.r.t. $\mathcal{G}$ if and only if P factorizes according to $\mathcal{G}$.*

$\Rightarrow$. Assume that P respects the local Markov property w.r.t. $\mathcal{G}$. By the chain rule we have $P(\mathcal{X}) = \prod_{i=1}^{n} P(X_i|X_1, ..., X_{i-1})$. Note that we can order the variables according to $\mathcal{G}$ without any loss of generality. From this it follows $Pa(X_i) \subseteq \{X_1, ..., X_{i-1}\}$. Thus $\{X_1, ..., X_{i-1}\} = Pa(X_i) \cup Z_i$ with $Z_i \subseteq Non(X_i)$. Since $\mathcal{G}$ is an I-Map we have $X_i \perp\!\!\!\perp Z_i | Pa(X_i)$ so;

$$P(X_i|X_1, .., X_{i-1}) = P(X_i|(Z_i, Pa(X_i))) = P(X_i|Pa(X_i))$$

$\square$

$\Leftarrow$. Assume P factorizes over $\mathcal{G}$. Let $Dc(X_i)$ denote the descendants of X_i and again denote its non-decenends by $Non(X_i)$. Note that $\{X_1, ..., X_n\} = \{X_i\} \cup Pa(X_i) \cup De(X_i) \cup Non(X_i)$ Thus we have;

$$P(X_i|Pa(X_i), Non(X_i)) = \frac{P(X_i, Pa(X_i), Non(X_i))}{\sum_{X_i = x} P(X_i, Pa(X_i), Non(X_i))} \tag{1.2.3}$$

We will work out the right hand side of (1.2.3) in parts. First we compute the nominator summing out all possible values of the descendants of X_i

$$\begin{aligned}
P(X_i, Pa(X_i), Non(X_i)) &= \sum_{Dc(X_i)} P(X_i, Pa(X_i), Non(X_i), De(X_i)) \\
&= \sum_{Dc(X_i)} P(\mathcal{X}) \\
&= \sum_{Dc(X_i)} \prod_{X\in\mathcal{X}} P(X|Pa(X)) \\
&= \prod_{X\in(\mathcal{X}\cap De(X_i))} P(X|Pa(X)) \\
&= P(X_i|Pa(X_i)) \prod_{X\in(Non(X_i)\cup Pa(X_i))} P(X|Pa(X))
\end{aligned}$$

Next we look at the denominator of (2.6). Summing out over all possible X_i we get the desired result immediately

$$\sum_{X_i=x} P(X_i, Pa(X_i), Non(X_i)) = \prod_{X\in(Non(X_i)\cup Pa(X_i))} p(X|Pa(X))$$

Putting (1.2.3) back together we get the desired result:

$$P(X_i, Pa(X_i), Non(X_i)) = P(X_i|Pa(X_i)) \Rightarrow X_i \perp\!\!\!\perp Non(X_i)|Pa(X_i)$$

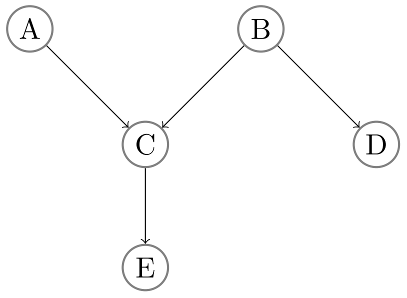
□

The fact that a distribution P over $\mathcal{X}$ factorizes according to some graph $\mathcal{G}$ over the same variables implies a set of conditional independicies that always hold and allows us to compute joint probabilities a lot faster. This is exactly the goal we set out to achieve. We illustrate what this means in a short example.

Example 1.16. Suppose we want to calculate the joint probability distribution P over a set of variables $\mathcal{X} = \{A, B, C, D, E\}$. Without any other knowledge this will be an application of the chain rule which results in the following expression;

$$\begin{aligned}
P(\mathcal{X}) &= \prod_{i=1}^{n} P(X_i|X_1, .., X_{i-1}) \\
&= P(A)P(B|A)P(C|A, B)P(E|A, B, C)P(D|A, B, C, E)
\end{aligned}$$

As we have argued in the sections above, graph structures can help us when calculating joint distributions. Assume we know that the following graph captures all conditional independencies induced by P.



An application of theorem 1.15 now tells us that $P(\mathcal{X})$ will factor over $\mathcal{G}$ which allows us to respresent the joint density as;

$$\begin{aligned} P(\mathcal{X}) &= \prod_{i=1}^{n} P(X_i|Pa(X_i)) \\ &= P(A)P(B)P(C|A,B)P(D|B)P(E|C) \end{aligned}$$

We have seen that a graph structure $\mathcal{G}$ encodes a certain set of conditional independence assumptions and that if a distribution P factorizes over $\mathcal{G}$ it satisfies the local Markov property. This is useful since we can use this factorization to speed up calculations. There is however a shortcoming in the process we used above. When we change something in the graph we need to calculate the joint density all over again.

By transforming the directed graph to an undirected equivalent we can perform inference on the junction-tree of the undirected model. This allows us to compile the density once and only adapt parts when introducing evidence. How this works will become apparent later. First we need a way to transform our directed graph into an undirected version that respects the same independencies as the original. Before we can find such a graph we will have a better look at all dependencies a Bayesian network captures.

1.2.2 Independence relations in Bayesian networks

We want to understand what other independence relations $X \perp\!\!\!\perp Y|Z$ hold (besides the local Markov relations) for all distributions P such that $(\mathcal{G}, P)$ is a Bayesian network.

To get a better understanding of these relations we will look at the converse. When is it possible for X to influence Y given Z for some distribution factorizing over a graph $\mathcal{G}$? We first define the set of all conditional independence relations that hold under some distribution P over a set of r.v. $\mathcal{X}$

Definition 1.17. Let P be a distribution over $\mathcal{X}$, we denote the set of all conditional independencies induced by P by

$$\mathcal{I}_P = \{(X, Y, Z) : X \perp\!\!\!\perp Y | Z \textit{ under } P,\ X, Y, Z \subseteq \mathcal{X}\} \tag{1.2.4}$$

The next thing of interest would be a similar expression that captures all independencies induced by a graph. Before we can specify such a set of independencies we look at all possible ways random variables can influence each other in a directed graph.

Direct connection

The most obvious way for X to influence Y in a graph is if they are directly connected through an edge $X \to Y$. If this is the case we can construct a distribution where X and Y are correlated regardless any other variables. We must however be careful with such a construction. Simply taking al variables equal ($X = Y = Z$) won't work as then obviously $X \perp\!\!\!\perp Y|Z$. Thus we choose Y depending on X and X independent of all other variables. Now it is clear that Y is indeed dependent of X given any Z.

Indirect connection

Suppose that X and Y are not directly connected and the only trails between them in the graph go through Z (otherwise $X \perp\!\!\!\perp Y|Z$ is obvious). There are four different trails in which this occurs.

1. **Causal effect** This is an extension of the direct connection. Suppose there exists a trail from $X \to Z \to Y$, thus X influences Y through Z. If we suppose Z is know than clearly $X \perp\!\!\!\perp Y|Z$.
2. **Indirect evidential effect** The reverse of the Causal effect. Here we have a chain $Y \to Z \to X$ but the argument remains the same. If we are given Z, X and Y are independent.
3. **Common cause** In this case the trail looks like $X \leftarrow Z \to Y$ and it is again clear that given Z the others are independent

4. **Common effect** In the previous cases we saw that X can only influence Y if we do not have any knowledge about Z. In this trail $X \to Z \leftarrow Y$, influence can only flow from X to Y if Z or one of its descendants is known.

It will turn out that the common effect or *v -shape* will play a big part in determining whether $X \perp\!\!\!\perp Y|Z$. We need two more definitions to make this precise,

Definition 1.18 (Active trail)**.** If X can influence Y through Z we call the trail $X \rightleftharpoons Z \rightleftharpoons Y$ active.

Definition 1.19. Let $\mathcal{G} = (V, E)$ be a DAG and $\tau = (V_1 \rightleftharpoons .... \rightleftharpoons V_n)$ a trail in $\mathcal{G}$, and $Z \subseteq V$ a set of nodes. The trail τ is active given Z if

- Whenever a v-structure occurs $V_{i-1} \to V_i \leftarrow V_{i+1}$, either V_i or one of its descendants is in Z
- No other node of $Z \in \tau$

This definition of active trails will give rise to an important concept introduced by Judea Pearl in [16] called d-separation (d for directed). This concept will indeed specify all independence relations that hold for all distributions associated with some Bayesian network.

Definition 1.20 (d-separation)**.** Let $X, Y, Z \subseteq V$ be three sets of nodes in a DAG $\mathcal{G} = (V, E)$. We call X and Y d-separated given Z, notation $d - Sep_{\mathcal{G}}(X; Y|Z)$, if there is no active trail between any node in X and any node in Y given Z. We denote the set of all independencies that correspond with d-sepertion in $\mathcal{G}$ with $\mathcal{I}_{\mathcal{G}}$.

$$\mathcal{I}_{\mathcal{G}} = \{(X, Y, Z) \ : \ d - Sep_{\mathcal{G}}(X; Y|Z)\} \tag{1.2.5}$$

The aim of this section was to detect other independencies that hold for all distributions that factor according to $\mathcal{G}$. Hence we want to make sure that indeed if X and Y are d-separated by some set Z they are infect conditionally independent given Z for every distribution P that factorizes over $\mathcal{G}$.

Lemma 1.21. *If $(\mathcal{G}, P)$ is a Bayesian network we have that $\mathcal{I}_{\mathcal{G}} \subseteq \mathcal{I}_P$.*

The next thing we could try to prove is that $\mathcal{I}_{\mathcal{G}}$ captures al independencies, in other words $\mathcal{I}_P = \mathcal{I}_{\mathcal{G}}$. Unfortunately this is not true in general as we can construct a counterexample.

Example 1.22. Suppose we have a distribution P over $\mathcal{X}$ in which all r.v. $X \in \mathcal{X}$ are independent. Now $(\mathcal{G}, P)$ is a bayesian network for any graph $\mathcal{G}$ whose nodes index $\mathcal{X}$. This means we can construct a graph $\mathcal{G}'$ such that disjoint subsets X and Y are not d-separated by some set Z in $\mathcal{G}'$ ergo $(X, Y, Z) \notin \mathcal{I}_{\mathcal{G}'}$. On the other hand, since all nodes are independent, $(X, Y, Z) \in \mathcal{I}_P$ thus $\mathcal{I}_p \neq \mathcal{I}_{\mathcal{G}}$.

A slightly less strong assertion will hold.

Theorem 1.23. *Suppose we have a DAG $\mathcal{G}$ over $\mathcal{X}$ and $X, Y, Z \subseteq \mathcal{X}$ where X and Y are not d-separated given Z. Than there exists a distribution P that factors over $\mathcal{G}$ in which X and Y are dependent given Z.*

Proof. As X and Y are not d-separated, there must exist at least one trail $\tau = (E_1, ..., E_n)$ with $E_1 \in X$ and $E_n \in Y$ that is active. The goal is to think up a distribution that makes X and Y correlated. First we define all CPDs on the trail such that E_i and E_{i+1} are correlated. In the case of a "v-structure" $E_i \rightarrow E_{i+1} \leftarrow E_{i+2}$ define the CPDs on X_{i+1} and its descendents in such a way that the correlation between E_i and E_{i+2} is activated. Next we pick all other variables not on τ, or downstream at some v-structure to be independent of all other variables (uniformly distributed for example). This ensures that there is no other path that cancels out the influence of X on Y through τ $\square$

With the previous theorem we can now conclude that $\mathcal{I}_{\mathcal{G}}$ is the set of maximal independencies we can derive from $\mathcal{G}$. For any independence assertion that is not induced by d-separation in $\mathcal{G}$ we can always find a distribution P that will factorize over $\mathcal{G}$. In general though, we can expect that $\mathcal{I}_{\mathcal{G}}$ captures all independencies as the next theorem states.

Theorem 1.24. *If $(\mathcal{G}, P)$ is a Bayesian network we have that $\mathcal{I}_P = \mathcal{I}_{\mathcal{G}}$ a.s., that is for all distributions P factorizing over the graph $\mathcal{G}$ except for a set of measure zero in the space of CPTs.*

Theorem 1.24 tells us that d-separation almost always will characterize all the conditional independencies present. This allows us to abstract the graphical details of $\mathcal{G}$ and see it as just a specification of independence properties. A implication of this fact is that there is some sort of equivalence between different the graphs on which Bayesian networks are built.

Definition 1.25. We will call two graphs $\mathcal{G}_1$, $\mathcal{G}_2$ over $\mathcal{X}$ I-equivalent if $\mathcal{I}_{\mathcal{G}_1} = \mathcal{I}_{\mathcal{G}_2}$.

It won't be a surprise that the I-equivalance property has everything to do with the v-stuctures in the graphs underlying the Bayesian networks. This gives us an easy way to tell if two graphs are indeed I-equivalent.

Definition 1.26. We say that two graphs $\mathcal{G}_1, \mathcal{G}_2$ have the same skeleton if $\hat{\mathcal{G}}_1 = \hat{\mathcal{G}}_2$. Where $\hat{\mathcal{G}}_1$ and $\hat{\mathcal{G}}_2$ are the undirected versions of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively

The fact that two graphs have the same skeleton won't imply that they are I-equivalent but the converse will hold. If two graphs don't have the same skeleton we can construct (via a contruction similar to the proof of theorem 1.23) an independence relation that only holds in one of them and thus they won't be I-equivalent. The next definition will give everything we need to relate I-equivalance to the structure of Bayesian network graphs.

Definition 1.27. Let $\mathcal{G} = (V, E)$ be a DAG. We call a v-structure $X \to Z \leftarrow Y$, between the nodes $X, Y, Z \in V$ immoral if there is no direct edge between X and Y in $\mathcal{G}$.

Theorem 1.28. *If $(\mathcal{G}_1, P)$ and $(\mathcal{G}_2, P)$ are both Bayesian networks over $\mathcal{X}$. Then $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton and immoralities if and only if they are I-equivalent.*

Now if we have a Bayesian network $(\mathcal{G}, P)$ with no immoralities in $\mathcal{G}$, we could view the pair $(\hat{\mathcal{G}}, P)$ where $\hat{\mathcal{G}}$ is the undirected version of $\mathcal{G}$. The question arises if P also has a factorization w.r.t. this new undirected object. To answer this question we first need some basic theory about undirected graphical models.

1.3 Undirected models

1.3.1 Markov random field definition and Markov equivalencies

In the previous section we looked at directed graphs as the underlying structure of a graphical model and we investigated the conditional independencies these graphs encode. In this section we define the undirected equivalent of the Bayesian network, the Markov random field. This structure will be very similar to the directed case with only a few changes.

With the elimination of directions on the edges in the network, we will need to define the way in which variables interact with each-other as conditional probabilities won't longer suffice. We begin with defining a Markov random field, the undirected equivalent of definition 1.13.

Definition 1.29. Let $\mathcal{G} = (V, E)$ be an undirected graph, let $\mathcal{X}$ be a collection of r.v. indexed by the nodes V of $\mathcal{G}$ and let P be a probability distribution over $\mathcal{X}$. We call the pair $(\mathcal{G}, P)$ a Markov random field if it satisfies one of the following Markov properties:

- **Pairwise Markov property**; Any two non-adjacent variables are conditionally independent given all other variables:

$$X \perp\!\!\!\perp Y |\ \mathcal{X} \backslash \{X, Y\} \quad if \quad (X, Y) \notin E \tag{1.3.1}$$

- **Local Markov property**; A variable is conditionally independent of all other variables given its neighbors:

$$X \perp\!\!\!\perp \mathcal{X} \backslash \{Ne(X) \cup X\} | Ne(X) \tag{1.3.2}$$

- **Global Markov property**; Any two subsets of variables are conditionally independent given a separating subset:

$$X \perp\!\!\!\perp Y | Z \tag{1.3.3}$$

  where every path from any node in X to any node in Y passes through Z.

The Markov propertys listed in the definition above are in fact equivalent if the intersection property (1.1.2) from theorem 1.8 holds. Before we prove this, we prove the following relation that holds in general.

Lemma 1.30. *For any undirected graph $\mathcal{G}$ and distribution function on $\mathcal{X}$ it holds that*

$$(1.3.3) \implies (1.3.2) \implies (1.3.1)$$

(1.3.3) $\implies$ (1.3.2). Trivial, $Ne(X)$ separates X from $\mathcal{X} \backslash \{Ne(X) \cup X\}$. □

(1.3.2) $\implies$ (1.3.1). Assume we have X and Y are non adjacent than we have;

$$Y \in \mathcal{X} \backslash \{Ne(X) \cup X\}$$

Since (1.3.2) holds we get

$$X \perp\!\!\!\perp \mathcal{X} \backslash \{Ne(X) \cup X\} | Ne(X)$$

Now since $Ne(X) \cup (\mathcal{X}\backslash Ne(X)\backslash X\backslash Y) = \mathcal{X}\backslash\{X, Y\}$ we get by the weak union property

$$X \perp\!\!\!\perp \mathcal{X}\backslash Ne(X)|\mathcal{X}\backslash\{X, Y\}$$

And thus, by the decomposition property ($Y \in \mathcal{X}\backslash Ne(X)$) we get $X \perp\!\!\!\perp Y|\mathcal{X}\backslash\{X, Y\}$ □

Theorem 1.31 (Pearl and Paz)**.** *Suppose we have a probability distribution on $\mathcal{X}$ such that 1.8 holds for disjoint subsets $X, Y, W, Z \subset \mathcal{X}$. All Markov properties are then equivalent,*

$$(1.3.3) \Longleftrightarrow (1.3.2) \Longleftrightarrow (1.3.1)$$

Proof. Note that we only need to prove that the pairwise Markov property (1.3.1) implies the global Markov property (1.3.3). Then lemma 1.30 will validate all other implications. This proof goes by backwords induction on the number of vertices in the separating set Z. For details I'll refer the reader to the proof given in Lauritzen [14] (theorem 3.7 of chapter 3). □

The global Markov property is very important as it gives a general criteria for when two groups of variables X and Y are conditionally independent given a third group of variables Z, as we have seen in the Bayesian network setting, conditional independence is directly related to factorization. The same is true for the Markov properties. Before we can present a theorem that proves this we need to define what we mean by factorization in the undirected case.

Definition 1.32. Let P be a probability distribution on $\mathcal{X}$. We say P factorizes according to an undirected graph $\mathcal{G}$ if there exist non-negative functions $\phi_B(X)$ for all complete subsets of $B \subseteq V$ that only depend on X_B, and a product measure ν on $\mathcal{X}$ such that the density p of P with respect to ν factorizes over the sets B,

$$p(x) = \prod_B \phi_B(x) \tag{1.3.4}$$

These factors ϕ_B are not unique since there is some arbitrariness in the choice of the product measure ν, also we can take groups together, of split them up in many ways. To eliminate this we can assume that the only sets B we get are the maximal cliques of a graph $\mathcal{G}$.

Lemma 1.33. *If a distribution P factorizes according to an undirected graph $\mathcal{G}$, we can without loss of generality assume it factorizes according to the maximal cliques of $\mathcal{G}$.*

Proof. Any complete subset $B \subseteq V$ of $\mathcal{G}$ is contained in at least one maximal complete subset (we only have a finite number of nodes) $\mathcal{C}$, so for each factor ϕ_B contributing to the density of p we assign ϕ_B to any one particular $\mathcal{C}$. Next we denote $B_\mathcal{C}$ all sets B assigned to the clique $\mathcal{C}$. (define $\phi_{B_\mathcal{C}} = 1$ if $B_\mathcal{C} = \emptyset$) then with (1.3.4) we get;

$$p(x) = \prod_B \phi_B(x) = \prod_\mathcal{C}(\prod_{B_\mathcal{C}} \phi_B) = \prod_\mathcal{C} \phi_\mathcal{C} \tag{1.3.5}$$

□

With this definition we can state the undirected equivalent of the factorization theorem 1.15, by Hammersley and Clifford.

Theorem 1.34. *A probability distribution P with positive and continuous density p with respect to some product measure ν satisfies the pairwise Markov property (2.9) with respect to an undirected graph $\mathcal{G}$ if and only if it factorizes according to $\mathcal{G}$.*

Proof. This proof goes beyond the scope of this thesis. The implication from the factorization to the pairwise Markov property is relatively easy, but the other way around relies on some algebraic lemmas. Interested readers can find a proof in the book of Lauritzen [14] (theorem 3.9). □

1.3.2 Independence relations in Markov random fields

Just as in the directed case we might be interested in the maximal set of conditional independencies that an undirected graph induces for all distributions that factorize according to it. As we have argued above, from al three Markov properties the global Markov property is the strongest (as it implies the other two even if the intersection property does not hold). A natural way to define the set of conditional independencies would be the following.

Definition 1.35. For an undirected graph we define the set of conditional independencies it induces with

$$\mathcal{I}_{\hat{\mathcal{G}}} = \{(X, Y, Z) : Z \textit{ separates } X \textit{ and } Y \textit{ in } \hat{\mathcal{G}}\} \tag{1.3.6}$$

Obviously we will have that $\mathcal{I}_{\hat{\mathcal{G}}} \neq \mathcal{I}_P$ for some P that factorize (for example the distributions in which al variables are independent) but under the assumption of a positive and continuous density it is clear (via a similar construction as in theorem 1.23) that there is no bigger set of independencies that always hold.

1.4 The connection between directed and undirected models

Now we have some theory on undirected graphical models, let us go back to the Bayesian networks. We remember that in working with directed models we came across a property that provided an equivalence class between different graphs that depended on the underlying undirected structure and the immorilaizations in the graph. It turns out that if we eliminate the immoralities in the directed graph and forget about direction on the edges we get an undirected version that respects the independencies in the directed graph.

Definition 1.36. We will denote the moral graph obtained from a DAG $\mathcal{G} = (V, E)$ as $\mathcal{M}_\mathcal{G} = (V, \hat{E})$ with

$$\hat{E} = (E \ \cup \ \{(v_i, v_j) \ : \ (v_j, v_i) \in E\} \cup \{(v_i, v_j) \ : \ (v_i, v_k) \ \& \ (v_j, v_k) \in E\})$$

There are now two basic propertys of $\mathcal{M}_\mathcal{G}$ that we are interested in. Firstly, does every distribution that factorized over $\mathcal{G}$ also has a factorization in the undirected sense over $\mathcal{M}_\mathcal{G}$. Secondly, are there any independence relations induced by $\mathcal{M}_\mathcal{G}$ that do not hold in the original directed graph.

Theorem 1.37. *If $(\mathcal{G}, P)$ is a Bayesian network than $(\mathcal{M}_\mathcal{G}, P)$ is a Markov random field.*

Proof. We prove by construction that P has a factorization in the undirected sence witch respect to $\mathcal{M}_\mathcal{G}$. We start with the Bayesian network $(\mathcal{G}, P)$ which has by definition the property that P factorizes according to $\mathcal{G}$. If we moralize $\mathcal{G}$ we get the undirected graph in which we have that any set $\{v \cup Pa(v)\}$, $v \in V$ is now a complete set and hence in at least one clique in the moral graph. Denote all nodes in a clique $\mathcal{C}$ with $V(\mathcal{C})$. Now we can construct the following factorization:

$$\begin{aligned} p(\mathcal{X}) &= \prod_{v \in V} p(X_v | Pa(X_v)) \\ &= \prod_{\mathcal{C}} \left(\prod_{v \in V(\mathcal{C})} p(X_v | Pa(X_v)) \right) \\ &= \prod_{\mathcal{C}} \phi_\mathcal{C} \end{aligned}$$

□

Theorem 1.38. *$\mathcal{M}_{\mathcal{G}}$ does not induce any independence relations on the distributions that factorize over it that do not hold in $\mathcal{G}$, thus we have $\mathcal{I}_{\mathcal{M}_{\mathcal{G}}} \subseteq \mathcal{I}_{\mathcal{G}}$.*

Proof. Suppose that Z does not D-separate X from Y, then $(X, Y, Z) \notin \mathcal{I}_{\mathcal{G}}$. Then there is an active trail from X to Y in $\mathcal{G}$, this means that whenever there is a v-structure $\tau_{i-1} \to \tau_i \leftarrow \tau_{i+1}$ either τ_i or one of its descendants is a node of Z. Each of these head-to-head meetings gives rise to a marriage between τ_{i-1} and τ_{i+1} in the moral graph. This gives an undirected path between X and Y not separated by Z which means that $(X, Y, Z) \notin \mathcal{I}_{\mathcal{M}_{\mathcal{G}}}$. □

With the proofs of theorem 1.37 and 1.38 we can now conclude that calculations we want to perform on a Bayesian network can also be performed on the undirected model over the moralized graph underlying the Bayesian network. With this fact we can now turn our attention to a clever way to perform calculations on Markov random fields.

1.5 Junction trees

Definition of a junction tree and marginal factorization theorem

To perform calculations on undirected graphical models we will make use of junction trees. Junction trees have the convenient property that distributions that factorize over them do so according to the marginals over the set of variables that form the nodes of the junction tree. This section will start with the definition of a junction (or clique) tree. After this we will state a necessary condition for existence of such a tree and argue that we can transform any (un-) directed network to obey this condition.

Definition 1.39. Let $\mathcal{C}$ be a collection of subsets of a finite set V and $\mathcal{T}$ a tree with $\mathcal{C}$ as its set of nodes. Then $\mathcal{T}$ is a junction tree if any intersection $\mathcal{C}_1 \cap \mathcal{C}_2$ of a pair of sets $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{C}$ is contained on the unique path between C_1 and C_2 in $\mathcal{T}$. We characterize the set of edges of $\mathcal{T}$ with a set of separators S where $S_{ij} = \mathcal{C}_i \cap \mathcal{C}_j$ whenever $\mathcal{C}_i$ and $\mathcal{C}_j$ are neighbors in the tree.

In practice we want to choose the nodes of the junction tree $\mathcal{T}$ equal to the cliques in the undirected network $\hat{\mathcal{G}}$, in this case we call $\mathcal{T}$ a clique tree. It is however not clear if every undirected graph has a clique tree, a necessary condition for this comes from the next definition.

Definition 1.40. A graph $\mathcal{G}$ is said to be chordal if all cycles of length 4 and more have chords, which are edges connecting non-neighbouring nodes in the cycle.

Theorem 1.41. *The following properties are equivalent for an undirected graph* $\hat{\mathcal{G}} = (V, E)$

- *There exists a junction tree for* $\hat{\mathcal{G}}$
- $\hat{\mathcal{G}}$ *is chordal*
- $\hat{\mathcal{G}}$ *is decomposable*

Before we explain the last equivalence class we must note that theorem 1.41 only tells us something about the existence of a junction tree. This does not tell us anything about how to find it, nor that it is unique. To define when a graph is decomposable we first need to define a decomposition.

Definition 1.42. A decomposition of a graph $\mathcal{G} = (V, E)$ is a triple of disjoint non-empty subsets $X, Y, Z \subseteq V$ with $(X \cup Y \cup Z) = V$ and Z a complete subset separating X and Y.

Next we would like to know what this means for factorizations over a decomposition of a graph.

Theorem 1.43. *If (X,Y,Z) is a decomposition of an undirected graph* $\mathcal{G}$ *then a distribution P factorizes over* $\mathcal{G}$ *if and only if* $P_{X \cup Z}$ *and* $P_{Y \cup Z}$ *factorize over* $\mathcal{G}_{X \cup Z}$ *and* $\mathcal{G}_{Y \cup Z}$ *respectively and the joint density satisfies*

$$p(\mathcal{X}) = \frac{P_{X \cup Z} P_{Y \cup Z}}{P_Z} \tag{1.5.1}$$

By using the decomposition concept recursively we can break down a graph into subsets that are all complete. This is precisely the desired property we are looking for.

Definition 1.44. A graph $\mathcal{G}$ is decomposable if it is either complete or it has a decomposition (X,Y,Z) such that $\mathcal{G}_{X \cup Z}$ and $\mathcal{G}_{Y \cup Z}$ are decomposable.

The fact that $\hat{\mathcal{G}}$ is decomposable when it has a junction tree will play a vital role in the factorizations of distributions over junction trees. Please note that while a graph may have a decomposition, this does not guarantee that it is also decomposable. For a decomposable graph we get the following important property.

Theorem 1.45. *If $(\mathcal{G}, P)$ is a Markov random field and $\mathcal{G}$ a decomposable graph. P factorizes according to the cliques the $\mathcal{C}$ and the separators S of $\mathcal{G}$ in the following way*

$$p(x) = \frac{\prod_{C \in \mathcal{C}} p(x_C)}{\prod_{S \in S} p(x_S)} \tag{1.5.2}$$

Proof. The proof is by recursive use of theorem 1.42 and the fact that $\mathcal{G}$ is a decomposable graph. □

1.5.1 From an undirected graph to a junction tree

Now that we have seen the nice properties of a junction tree and we have theorem 1.41 that tells us when an undirected graph has a junction tree, we can take step back and think if we can construct a junction tree for the moral graph $\mathcal{M}_{\mathcal{G}}$. This is possible but to use theorem 1.41 we would need to have that our undirected graph is chordal. Generally this will not be the case, so we will have to make it chordal through a process called triangulation.

Definition 1.46. We can construct a chordal graph from any undirected graph by adding edges to the edge set. This process is called triangulation of the graph.

While the process of triangulation is pretty straight forward, it can be done in many ways. We won't go into any depth here, but the choice of triangulation can play an important part in finding a convenient junction tree. More on this can be found in chapter 9 of Koller and Friendman [8].

Definition 1.47. We will denote a chordal graph obtained from an undirected graph $\hat{\mathcal{G}}$ via triangulation with $Ch_{\hat{\mathcal{G}}}$.

If we view $Ch_{\mathcal{M}_{\mathcal{G}}}$, a chordal and moral version of the original directed graph $\mathcal{G}$ we can immediately state the following trivial lemma.

Lemma 1.48. *$\mathcal{I}_{Ch_{\mathcal{M}_{\mathcal{G}}}} \subseteq \mathcal{I}_{\mathcal{M}_{\mathcal{G}}} \subseteq \mathcal{I}_{\mathcal{G}}$ for all distributions that factor over $Ch_{\mathcal{M}_{\mathcal{G}}}$.*

Because the triangulation step only adds edges in the moral graph, it can only result in less triples of sets that separate each other, hence the first inclusion. The second inclusion was already proven in theorem 1.38 and thus we are still respecting the original independencies in $(\mathcal{G}, P)$. As the chordal graph is also decomposable via 1.41 we now know via 1.45 that there exits a factorization of P over a junction tree of $Ch_{\mathcal{M}_{\mathcal{G}}}$ that factorizes as the marginals of the cliques in the tree.

The next step is finding "this" factorization. We use quotations around "this" since we didn't discuss the uniqueness of the junction tree yet (obviously once we have a junction tree, each clique has only one marginal and thus the factorization is unique). It turns out that only in very trivial cases a junction tree is unique. This poses us with the question which junction tree to take when doing calculations. We will not go in to this subject in this thesis but interested readers can find more on this in the book by Koller and Friedman [8].

Once a junction tree is determined the quest for the marginal factorization can begin. The factorization can be found using the junction tree algorithm, which we will explain in the next section. In short it comes down to a manipulation on a factorization (for example the one we constructed in the proof of theorem 1.37) over the tree, where the marginalization is done from the outside in and back again.

Potentials and contractions

Before we can start with an algorithm that will lead to a factorization (1.5.2) we need to define the way in which we can manipulate factorizations. We start with redefining the factors we we used in definition 1.32 as potentials.

Definition 1.49. Let $\mathcal{X}$ be a set of r.v. with Ω_X the state space for $X \in \mathcal{X}$. A potential ϕ on $U = \{X_1, .., X_n\} \subseteq \mathcal{X}$ is a mapping from $\Omega_U = \Omega_{X_1} \times .. \times \Omega_{X_n} \to [0, \infty)$.

These potentials are like the conditional probabilities in the directed model, where they have the extra constraint of summing to one. We can extend these potentials to be defined on the same space in a natural way.

Definition 1.50. Let ϕ be a potential on $U \subseteq U' \subseteq V$. For $x \in \Omega_{U'}$ we define the potential $\phi'(x) = \phi(y)$ where y is the projection of x onto Ω_U. We call ϕ' the extension of ϕ to U'.

Now that potentials are all defined on the same space (namely $\Omega_{\mathcal{X}}$) we can define multiplication, addition and division point wise.

Definition 1.51. Let ϕ_1 be a potential on U_1 and let ϕ_2 be a potential on U_2. We can define multiplication, addition and division is the following way. Here ϕ_1' and ϕ_2' on the right hand side of the equations are the extensions of ϕ_1 and ϕ_2 to $U_1 \cup U_2$.

$$
\begin{aligned}
(\phi_1\phi_2)(x) &= \phi_1'(x)\phi_2'(x) \\
(\phi_1+\phi_2)(x) &= \phi_1'(x)+\phi_2'(x) \\
(\phi_1/\phi_2)(x) &= \begin{cases} 0 & if \quad \phi_2'(x)=0 \\ \phi_1'(x)/\phi_2'(x) & else \end{cases}
\end{aligned}
$$

The reason for defining the division by zero is that we can encounter such divisions when doing local computations even when the overall joint probability is well-defined. The last potential operation we will need is the marginalization of a potential.

Definition 1.52. Let $U \subseteq U' \subseteq V$ and a potential ϕ on U' we denote the margin of ϕ on U by $\sum_{U'\backslash U} \phi$ which we define for $x \in \Omega_U$ by

$$\left(\sum_{U'\backslash U} \phi\right)(x) = \sum_{z\in\{\Omega_{U'}\backslash\Omega_U\}} \phi(z.x) \tag{1.5.3}$$

Here $z.x$ is the element in Ω_U with projections x to $\Omega_{U'}$ and z to $\Omega_{U'}\backslash\Omega_U$

With the potentials defined we can state what we mean by a factorization of a density over a junction tree in terms of potentials on the cliques of the junction tree.

Definition 1.53. A charge on a junction tree $\mathcal{T} = (\mathcal{C}, S)$ is a pair of collections of potentials $\Phi = (\{\phi_C | C \in \mathcal{C}\}, \{\phi_S | S \in S\})$.

The charge on a junction tree defines a function on the underlying set of variables of the tree $\mathcal{X} = \mathcal{C} \cup S$, and we will call the Φ the representation of the function defined by the contraction.

Definition 1.54. The contraction of a charge on $\mathcal{T} = (\mathcal{C}, S)$ is a function f on $x \in \mathcal{X} = \mathcal{C} \cup S$ that is given by

$$f(x) = \frac{\prod_{C\in\mathcal{C}} \phi_C(x)}{\prod_{S\in S} \phi_S(x)} \tag{1.5.4}$$

We will call the pair $(\mathcal{T}, \Phi)$ a compiled junction tree if Φ is a charge on $\mathcal{T}$. An initial charge can be obtained from the recursive factorization of the density in similar fashion as to what we have seen in the proof of theorem 1.37.

1.5.2 Propagation of the junction tree

Once we have constructed the junction tree and compiled it using an initial charge representing the joint density of variables in the network it is time for the following step. We want to update the charge on the junction tree to a canonical charge, which is the the charge

$$\Phi_f = (\{f_C | C \in \mathcal{C}\}, \{f_S | S \in S\}) \tag{1.5.5}$$

with f_U the margin of f on the set U. This is done through a process that is called propagation of the junction tree. In this process we will manipulate the potentials in the charge over the junction tree by passing sum-flows between them. We begin by defining these flows.

Definition 1.55. A sum-flow from a clique C to a clique C' is an operation on the space of charges on a junction tree. A flow from C to C' through S' is applied to a charge $\Phi = (\{\phi_C | C \in \mathcal{C}\}, \{\phi_S | S \in S\})$ the result is a new charge $\Phi^* = (\{\phi^*_C | C \in \mathcal{C}\}, \{\phi^*_S | S \in S\})$ with

$$\phi^*_i = \begin{cases} \sum_{C \setminus S'} \phi_C : & i = S' \\ \frac{\sum_{C \setminus S'} \phi_C}{\phi_{S'}} \phi_C : & i = C' \\ \phi_i : & i \in \mathcal{C} \setminus \{C'\} \ or \ i \in S \setminus \{S'\} \end{cases} \tag{1.5.6}$$

Flows are directed and we will denote passing a flow from a clique C to a clique C' through S by $C \to C'$. As we perform these operations on a tree there is no need to specify the separator S as it is uniquely defined by C and C'.

Theorem 1.56. *Passage of a flow does not effect the contraction of a charge.*

Proof. Let f be the joint density function of interest and $\Phi = (\{\phi_C | C \in \mathcal{C}\}, \{\phi_S | S \in S\})$ the initial representation of f. If we pass a flow $C \to C'$ through S' the only changed potentials are on C' and S. The new contraction f^* is

$$\begin{aligned} f^* &= \frac{\prod_{C \in \mathcal{C}} \phi^*_C(x)}{\prod_{S \in S} \phi^*_S(x)} \\ &= \frac{\phi^*_C}{\phi^*_S} \frac{\prod_{C \in \mathcal{C} \setminus \{C'\}} \phi_C(x)}{\prod_{S \in S \setminus \{S'\}} \phi_S(x)} \\ &= \frac{\frac{\sum_{C' \setminus S'} \phi'_C}{\phi_{S'}} \phi'_C}{\sum_{C' \setminus S'} \phi'_C} \frac{\prod_{C \in \mathcal{C} \setminus \{C'\}} \phi_C(x)}{\prod_{S \in S \setminus \{S'\}} \phi_S(x)} \\ &= \frac{\sum_{C' \setminus S'} \phi_{C'}}{\sum_{C' \setminus S'} \phi_{C'}} f \end{aligned}$$

As we allow for division by zero the factors in the last equation do not necessarily cancel, we check both scenarios. Consider a point x with a projection $x_{s'}$ on $\Omega_{S'}$. For $\phi^*_{S'}(x_{s'}) = \sum_{C' \setminus S'} \phi_{C'}$ we have two possibilities. If $\phi^*_{S'}(x_{s'}) > 0$ the terms cancel and we have $f^*(x) = f(x)$. If $\phi^*_{S'}(x_{s'}) = 0$ we find $f^*(x) = 0$ by the definition in 1.51. As $\phi_{C'}$ is a non-negative function $\sum_{C' \setminus S'} \phi_{C'} = 0$ implies that $\phi_{C'}(x') = 0$ for all $x' \in \Omega_{C'}$ which projects on $x_{s'}$. This means that $\phi_{C'}(x_{c'}) = 0$ which implies that $f(x) = 0$. $\square$

The next step towards the canonical charge will be to pass the sum flows in a particular sequence over $\mathcal{T}$. This particular sequence will be defined over the subtrees $\mathcal{T}'$ of $\mathcal{T}$. A *subtree* $\mathcal{T}' = (\mathcal{C}', S')$ of a junction tree $\mathcal{T} = (\mathcal{C}, S)$ is a connected set of subsets $\mathcal{C}' \subseteq \mathcal{C}$ and the edges $S' \subseteq S$ between them. We will call a clique C a *neighbor* of a subtree $\mathcal{T}'$ if the corresponding node of $\mathcal{T}$ is not in $\mathcal{T}'$ but is connected to a vertex in $\mathcal{T}'$ by an edge in $\mathcal{T}$ and the *base* of the subtree $\mathcal{T}'$ is the set $U' = \cup_{C' \in \mathcal{C}'} C'$, all variables associated with $\mathcal{T}'$. If $\Phi = (\{\phi_C | C \in \mathcal{C}\}, \{\phi_S | S \in S\})$ is a charge on $\mathcal{T}$ then its restriction to $\mathcal{T}'$, $\Phi_{\mathcal{T}'} = (\{\phi_{C'} | C' \in \mathcal{C}'\}, \{\phi_{S'} | S' \in S'\})$ is a charge on $\mathcal{T}'$ and its potential is the contraction of $\Phi_{\mathcal{T}'}$.

We will call the sequence of flows a *schedule*, which we can characterize by an ordered list of edges in the junction tree. So an edge (C_1, C_2) will correspond to a flow $C_1 \rightarrow C_2$. Relative to a schedule we will have one special type of flow.

Definition 1.57. A flow $C_i \rightarrow C_j$ in a schedule over $\mathcal{T}$ is called active if C_i has received active flows from al its neighbors in $\mathcal{T}$ with the possible exception of C_j.

The recursion of the definition of an active flow implies that the first active flow in a schedule must originate from a leaf of the junction tree. A schedule is *full* if it contains active flows in both directions across all edges of the junction tree. We will call a schedule *active* if it only contains active flows. Combining both we get a *fully active* schedule if it is both full and active. Finally we will call a subtree $\mathcal{T}'$ *live* if at a certain stage during a schedule it has recieved active flows from all its neighbors.

Lemma 1.58. *For every junction tree $\mathcal{T}$ there exists a fully active schedule.*

Proof. This goes by induction on the number of cliques in $\mathcal{T}$. If $\mathcal{T}$ has just one clique the result is trivial. Assume it has more cliques, let C_0 be a leaf of $\mathcal{T}$ and $\mathcal{T}_0$ the subtree obtained by removing C_0 and the associated edge S_0 from $\mathcal{T}$. By induction we have a fully active schedule for $\mathcal{T}_0$. By adding a flow from C_0 over S_0 to $\mathcal{T}_0$ at the beginning of the schedule and a flow from $\mathcal{T}_0$ over S_0 to C_0 at the end of the schedule we obtain a fully active schedule for $\mathcal{T}$. □

Theorem 1.59. *If $\Phi = (\{\phi_C | C \in \mathcal{C}\}, \{\phi_S | S \in S\})$ is a representation of a function f that factorizes according to a junction tree $\mathcal{T}$. Then we can modify Φ by passing a sequence of flows according to a fully active schedule on $\mathcal{T}$ such that whenever a subtree $\mathcal{T}'$ becomes live in the schedule the potential on $\mathcal{T}'$ is the margin of f on U'.*

Proof. We start with noticing that whatever schedule of sequences we choose Φ will always be a representation of f due to theorem 1.56. The rest of the proof goes by induction. The statement is obviously true when $\mathcal{T}' = \mathcal{T}$ since then $U' = U$ and f is the marginal of U. Suppose that $\mathcal{T}' \neq \mathcal{T}$, that it is live and that is has a neighbor C^*. that was the last neighbor to have passed a flow to $\mathcal{T}'$. Now we will view the subtree $\mathcal{T}^* = (\mathcal{C}^*, S^*)$ which is $\mathcal{T}'$ with C^* and its associated edge S added. By induction we suppose the result to hold for $\mathcal{T}^*$. Now if we had $\Phi = (\{\phi_C | C \in \mathcal{C}\}, \{\phi_S | S \in S\})$ as the overall representation of f just before the last flow $C^* \to C'$ where we have $C' \in \mathcal{T}'$ we get

$$f_{U^*} = \frac{\prod_{C \in \mathcal{C}^*} \phi_C(x)}{\prod_{S \in S^*} \phi_S(x)} = \frac{\phi_{C^*}}{\phi_{S^*}} \frac{\prod_{C \in \mathcal{C}'} \phi_C(x)}{\prod_{S \in S'} \phi_S(x)}. \tag{1.5.7}$$

As we have $U' \subseteq U^*$ we can now find the margin of f on U' ($f_{U'}$) by marginalizing 1.5.6 over $U^* \backslash U' = C^* \backslash S^*$:

$$f_{U'} = \sum_{U^* \backslash U'} f_{U^*} = \sum_{U^* \backslash U'} \frac{\phi_{C^*}}{\phi_{S^*}} \frac{\prod_{C \in \mathcal{C}'} \phi_C(x)}{\prod_{S \in S'} \phi_S(x)} = \frac{\sum_{C^* \backslash S^*} \phi_{C^*}}{\phi_{S^*}} \frac{\prod_{C \in \mathcal{C}'} \phi_C(x)}{\prod_{S \in S'} \phi_S(x)} \tag{1.5.8}$$

On the other hand let's have a look at what happens to the potential on $\mathcal{T}'$ after the flow from C^* to C' is passed. From 1.55 we know that only effect of the flow on the charge of $\mathcal{T}'$ is that $\phi_{C'}$ will be updated and we find;

$$f_{U'} = \left(\frac{\sum_{C^* \backslash S^*} \phi_{C^*}}{\phi_{S^*}} \right) \phi_{C'} \frac{\prod_{C \in \{\mathcal{C}' \backslash C'\}} \phi_C(x)}{\prod_{S \in S'} \phi_S(x)} = \frac{\sum_{C^* \backslash S^*} \phi_{C^*}}{\phi_{S^*}} \frac{\prod_{C \in \mathcal{C}'} \phi_C(x)}{\prod_{S \in S'} \phi_S(x)} \tag{1.5.9}$$

□

With the previous theorem we can now deduce that whenever a clique C is live it's potential is f_C. This means that a fully active schedule over a junction tree $\mathcal{T}$ will result in a charge in which the potentials over al cliques C_i are marginals f_{C_i}. One thing remain to be shown; any time an active flow has passed an edge in $\mathcal{T}$ in both directions the potential on the separator associated with the edge should be f_S.

Lemma 1.60. *Let S be a separator of two neighboring cliques C_1 and C_2 in $\mathcal{T}$. After passage of a fully active schedule the potential ϕ_S is the margin of f on S.*

Proof. We have that the potential on S after passage of the active flows between C_1 and C_2 in both directions is

$$\sum_{C_1 \setminus S} \phi_{C_1} = \phi_S = \sum_{C_2 \setminus S} \phi_{C_2}$$

Now by theorem 1.59 the potentials ϕ_{C_1} and ϕ_{C_2} are marginals of f over their corresponding cliques and thus we must conclude that ϕ_S is the margin of f on S. □

With this we have finally proven the result we were looking for

Theorem 1.61. *If f factorizes according to a junction tree $\mathcal{T}$ then, after passage of a fully active schedule on $\mathcal{T}$ with some initial charge Φ representing f the resulting charge Φ_f is the canonical charge which represents the marginal factorization (1.5.6) of f on $\mathcal{T}$.*

$$f = \frac{\prod_{C \in \mathcal{C}} \phi_C}{\prod_{S \in \mathcal{S}} \phi_S}$$

Proof. This is a direct result of theorem 1.58, 1.59, lemma 1.60 and (1.5.4) □

1.5.3 Introduction of evidence

With the proof of theorem 1.61 we have obtained a prior joint distribution f over a set of random variables $\mathcal{X}$ that factorizes according to the prior probability distribution in each of the cliques in the junction tree associated with the DAG $\mathcal{G}$ that captured the conditional independence relations between the variables in $\mathcal{X}$. At this point we would like to introduce new information to network and obtain the posterior probabilities given the information. The introduction of information will be done in the form of *likelihood evidence* on the individual nodes for the graph $\mathcal{G}$.

For a graph $\mathcal{G}$ over a set of r.v. $\mathcal{X} = \{X_1, .., X_n\}$ with finite state space we define the set of r.v. $\mathcal{E}_V = \{\hat{X}_1, .., \hat{X}_n\}$ with possible infinite state space such that each $\hat{X}_i$ depends only on X_i through the known probability distribution $p(\hat{x}_i|x_i)$. Likelihood evidence ϵ will be an observation on the state of a variable in $\mathcal{E}_V$ which tells us the likelihood of its corresponding variables in $\mathcal{X}$ through $p(\epsilon|x_i)$. This construction of the set $\mathcal{E}_V$ also allows us to enter observations on nodes in the graph $\mathcal{G}$ by setting $\Omega_{\hat{X}_i} = \Omega_{X_i}$ and $p(\hat{x}_i|x_i) = 1$

if $\hat{x}_i = x_i$ and 0 else.

For a Bayesian network $(\mathcal{G}, P)$ over a set of r.v. $\mathcal{X} = \{X_1, .., X_n\}$ we have that the distribution of $\mathcal{X}$ factorizes according to the cliques in the junction tree associated with $\mathcal{G}$,

$$p(x) \propto \prod_{C \in \mathcal{C}} \phi_C$$

For the likelihood evidence $p(\mathcal{E}|x)$ with $\mathcal{E} = \cup \epsilon_v$ we have a similar factorization,

$$p(\mathcal{E}|x) = \prod_{C \in \mathcal{C}} l(x_c|\epsilon)$$

From (1.1.3) we can obtain the posterior of $\mathcal{X}$ given the likelihood evidence $P(\mathcal{E}|x)$.

$$\begin{aligned} P(x|\epsilon) &\propto P(x)P(\epsilon|x) \\ &\propto \left(\prod_{C \in \mathcal{C}} \phi_C(x_C)\right)\left(\prod_{C \in \mathcal{C}} l(x_C|\epsilon)\right) \\ &\propto \prod_{C \in \mathcal{C}} \phi_C(x_C) l(x_C|\epsilon) \end{aligned}$$

From this we see that the potentials ϕ^* for a charge that represents $p(x)p(\mathcal{E}|x)$ can be obtained by multiplying the likelihood evidence onto each potential ϕ in the model.

$$\phi^*_C(x_C) = \phi_C(x_C) l(x_C|\epsilon)$$

By changing the potentials we find a representation of a function that is proportional to the posterior distribution of the variables given the evidence. After entering evidence into the network we again propagate to find the marginal representation.

Chapter 2

Forensic DNA Analysis

Forensic science is the application of a board spectrum of sciences and insights to solve a legal or crime question. This can mean a handwriting expert examining a ransom note, a fire expert determining the cause of a fire or, in case of a road accident, a reconstruction of events based on skid marks and tire prints. In all cases, forensics uses expert knowledge to investigate evidence in the broadest sense of the word.

When this evidence consists of any biological samples (this can be blood, skin tissue, saliva, hair, semen etc.) analysis is usually done by a forensic DNA specialist. As the DNA of each individual is almost unique and present in all cells of the body, this process can help identifying if two biological traces match with each other. For example semen found at a rape scene could be matched to a suspect, producing conclusive proof that he had intercourse with the victim.

This second chapter of my thesis will give an introduction into the field of forensic STR (or Short Tandem Repeat) DNA analysis as it was introduced by Alex Jefferys [2]. DNA analysis using STRs deals with comparing particular regions on the DNA strand with each other as we will see later. In forensic applications of this method, the samples analyzed are usually mixtures of more than one source of DNA. Analyzing these DNA mixture models can be very difficult and the final result of this chapter will be to present a mathematical model to assist in the analysis of these mixtures.

In section 2.1 we give a short overview on how and why STR DNA analysis works, why it is helpful in forensics and the way we need to interpret results it produces. There is a lot of literature available on this subject, I'd recommend the 2005 book by Butler, *Forensic DNA Typing: Biology, Technology and Genetics of STR Markers* [4] or the book by Buckelton el al. *Forensic DNA evidence interpretation* [13] to the interested reader.

After an initial introduction to STRs we present a basic mathematical model by Robert Cowell el al. on the output of the STR DNA mixture analysis. This model will consist of two parts, a Bayesian network on top and an underlying gamma model which will be added as likelihood evidence on some of the leaf nodes of the network. The simple model makes a lot of assumptions on the underlying processes of the STR procedure, in section 2.3 we will discuss these assumptions and point out where we will need to alter the model for real life application.

In the final section we introduce solutions to the issues of section 2.3. This transforms the simple model into the model that form the basis of our analysis of the profiles in the scorpion case in the next chapter.

2.1 Preliminaries

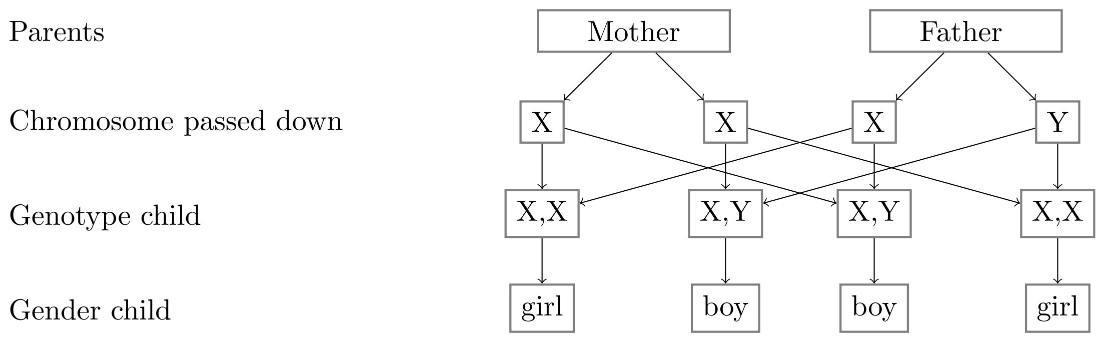
Obviously forensic DNA analysis has everything to do with DNA, as a reminder we state what we actually mean when talking about DNA. Next we introduce some terminology used in working with DNA and finally we will look at short tandem repeats and why they are useful as a forensic tool.

DNA

The basic view that we have of DNA is that of the well known double helix shaped molecule built up from four types of so called base paires G, C, A and T (or Guanine, Cytosine, Adenine and Thymine). While our human DNA actually looks like that, it is made up of not one such molecule but of 23 pairs (46 in total) of them which are called chromosomes.

A *chromosome* is a single strand of coiled DNA together with some DNA bound proteins. All but one pair of chromosomes consists of one chromosome inherited from the mother and one chromosome encoding the same genes inherited from the father. The final pair of chromosomes determines the sex of an individual, in this case the chromosomes inherited the same way (one from each parent) but they are not always the same. There are two different sex chromosomes, X and Y, a person with two X chromosomes will be a woman and a man will have an X and a Y chromosome.

Assumption 2.1. The passing down of chromosomes from parents to children is assumed to be random. This means that out of each of the 23 pairs of chromosomes parents have one is selected at random to be inherited by the child. If we take the sex chromosome as an example, we see that there are four possible combinations which each have a chance of 0.25.



The DNA on the chromosomes encodes our genes, which control the production of one or more proteins. These proteins in their turn provide all kinds of biological functions in the body. This can be on a micro scale (with functions on potassium levels within cells) or on the macro scale (growing arms and legs). Obviously, a lot of these genes are shared between all humans, in fact as much as 99,5% of our DNA is the same in every human. Still the remaining 0.5% or so does differ, but these difference occur mainly on places on the chromosome that don't code for any genes.

Since the bulk of our DNA across all chromosomes is actually the same, comparison of DNA samples is not done on the entire DNA string. Since if we already know that most will match there is no point in comparing every base pair. On top of that we would need a perfect DNA sample to start with, if the DNA is degraded we can simply not do a comparison. Finally sequencing an entire DNA profile is still quite expensive. Although modern techniques are advancing to overcome the last two problems the focus in DNA analysis lies on comparing specific pieces of DNA that are known to differ through the population. Before we can explain the method that we will use to compare samples, we need some terminology.

Just as our entire DNA is segmented over multiple chromosomes, these chromosomes themselves can be segmented. A specific section on of a chromosome is referred to as a *locus*. At a locus one or more strings of base pairs are known to occur, such a possible string is called an *allele*. If we can determine the DNA sequence at a locus (in a laboratory) than we call the locus a *marker*. Finally the actual string found at a marker is called the *genotype* of that marker.

Short Tandem Repeat Markers

In comparing DNA samples we now look specifically at certain markers that we know are variant across a population. One sort of these markers are called short tandem repeat markers or STRs. STRs are markers where a short sequence of base pairs or a *word* is repeated a certain number of times. These words are typically 4 base pairs long and will be repeated (depending on the specific marker) between 3 and 51 times. When working with STRs we will denote the alleles by the number of tandem repeats. In this way the genotype at a single marker becomes a pair of repeat numbers. Heres an example,

Example 2.2. Suppose we find the following DNA sequence at a known STR marker on chromosome A;

Chromosome A_1 ..CGGGTATTGATTGATTGATTGATTGATTGATTGATTGGAAAGGT.....
Chromosome A_2 ..CGGGTATTGATTGATTGATTGATTGATTGAGTTGTATGAAAGGTC...

We can clearly see a repeating pattern on both chromosomes, namely the word ATTG

Chromosome A_1 ..CGGGT $\underbrace{ATTGATTGATTGATTGATTGATTGATTGATTG}_{7\times ATTG}$GAAAGGT..
Chromosome A_2 ..CGGGT$\underbrace{ATTGATTGATTGATTGATTGATTG}_{6\times ATTG}$AGTTGTATGAAAGGT..

Here we must note that the numbering tells us nothing about which chromosome is maternal or which is paternal. So the genotype of the person on the marker viewed above would be either (6,8) or (8,6).

As there is no way to distinguish both there is no difference between the genotypes (6,8) and (8,6). We will make the convention that we denote STR genotypes ordered, thus (a, b) with $a \leq b$. (this means the genotype of the marker on chromosome A in the example above is (6,8)). A genotype with two identical repeat numbers is called a *homozygote* and a genotype with two different alleles is called a *heterozygote*.

Once we have obtained the genotype of a person at a certain marker, the question arises: what does this tell us? If we focus on the forensic application (comparing profiles from a crime scene to a reference) a match on a single STR does not tell us much as we would still expect random matches to occur. This is why DNA samples are compared using a number of STR markers. There are a lot of STR markers known, all with various properties. Out of these markers we can construct many sets of STR's, but as it turn out there are two basic standards that are used.

The first set of STR's comes from the FBI and is called CODIS (short for Combined DNA Index System) [1], the other was developed in the UK and is called SGMplus (Second Generation Multiplex plus) [3]. As the latter is the one used in the Netherlands we will focus on that one. SGMplus uses 10 STR markers and one marker indicating sex, so we can be characterize a profile by 10 pairs of repeat numbers and an indicator of either two X chromosomes or an X and a Y chromosome. The markers in SGMplus are chosen such that they each lie on a different chromosome pair, which makes them independent under assumption 2.1.1. This is what gives STR analysis its power, as we will later see.

For purposes in forensic science we would like that SGMplus genotypes are unique for all humans. This is unfortunately not true as identical twins will always have the same genotype (if no mutations occur) and there is a reasonable chance on identical profiles for children of the same parents. As the degree of the relationship decreases the chance of a random match will go down. The following table from a paper by Evett and Foreman from 2001 [7] gives what they call general match probabilities for SGMplus profiles.

Relation	Random match	Chance
Sibling	1 in 10000	1×10^{-4}
Parent/child	1 in 1 million	1×10^{-6}
Half-sibling or uncle/nephew	1 in 10 million	1×10^{-7}
First cousin	1 in 100 million	1×10^{-8}
Unrelated	1 in 1 billion	1×10^{-9}

The match probabilities for specific STR profiles are typically several orders of magnitude smaller than those given above, which were calculated from the theoretically most common SGM Plus profile.

2.1.1 DNA analysis using short tandem repeats

The process of doing a forensic DNA analysis using STR markers comes down to a few steps of which two will be discussed in the next two subsections. Obviously it all starts with taking DNA samples, a mixture sample is taken from a crime scene and reference samples are taken from individuals involved. There are strict protocols to ensure that no contamination occurs at this step. (Although a recent case in Germany proves different. Here an employee at a DNA swab manufacturer lead the police to the believe that a serial killer was on the loose because her DNA was found at every crime scene where the swabs were used. [10]).

After the collection of the sample, the amount of DNA at the STR markers is amplified using a PCR reaction. The PCR is an enzyme driven temperature dependent process in which recognizable regions of the DNA are replicated a large number of times. The process usually takes 30 or so temperature cycles, where in each cycle the number of copies is almost doubled. Thus after 30 cycles we would theoretically have 2^{30} times the amount of DNA at the target regions although in practice the PCR never achieves its optimal potential.

Once enough DNA material is obtained an electropherogram is made. To do this the amplified DNA from the PCR is injected into a gel and then an electrical current is applied. This will cause the alleles (which are all negatively charged) to move towards the positive pole. Since heavier molecules (those with higher allele numbers) will travel slower through the gel, the molecules sort themselves according to allele number. The output of this process is a graph depicting the amount of DNA of each allele in a so called *peak profile*. The height of a peak gives the amount of DNA present and its location determines the allele number. Figure 2.1 shows a typical peak profile for SGMplus.

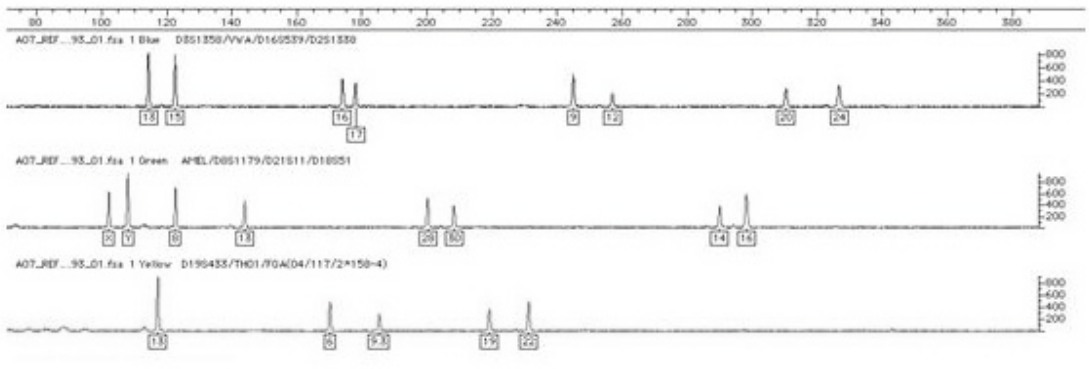


Figure 2.1: Single person SGMplus peak profile

The peak profile from figure 2.1 is made from the DNA of a single person, in it we see 21 peaks which each correspond to an allele at a certain marker but since this is a SGMplus profile, we would expect to find 22 peaks corresponding to 10 pairs of STR markers and one pair indicating sex. This 'missing' peak corresponds to a STR marker where we have the same allele on both chromosomes. In this case, the bottom left marker has a single (rather high) peak at 14 tandem repeats. This means the genotype at that marker is $(14, 14)$. The single peak is also about twice as high as the other peaks at the bottom level of the plot. With this analysis we can transform the peak profile into the following table specifying the genotype of the single contributer.

FGA	TH01	VWA	D2..	D3..	D8..	D16..	D18..	D19..	D21..	AMEL
(19,23)	(6,9.3)	(16,17)	(20,24)	(18,19)	(8,13)	(9,12)	(14,15)	(14,14)	(29,30)	(X,Y)

Table 2.1: Single person SGMplus genotype

With the genotype from the profile in figure 2.1 determined we could then try and see if it would match to the samples found at a crime scene or in a less criminal example we could perform an paternity (or maternity) test.

2.1.2 STR analysis of a DNA mixture

The analysis of the peak profile in figure 2.1 was very easy, count the peaks, figure out to what repeat number they correspond and your done. Things become more complicated when we start looking at mixtures of DNA from different persons. We will illustrate this with an example of a two person mixture at a single marker.

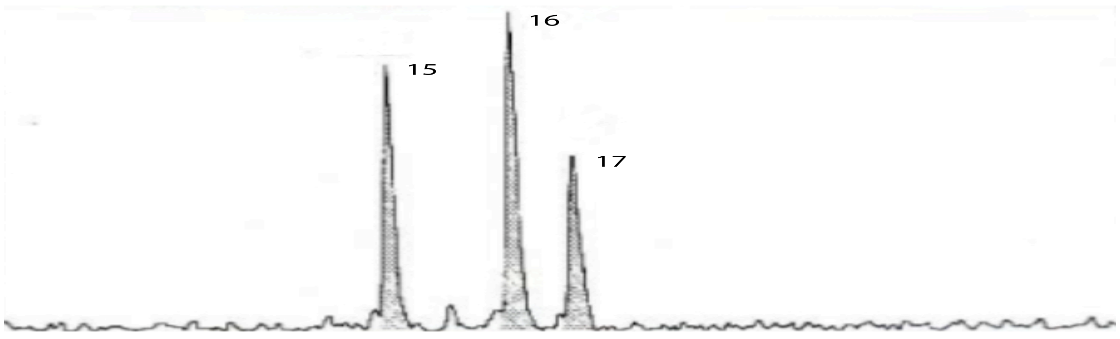


Figure 2.2: Single marker 2 person mixture

Just as in the previous peak profile we would like to produce a table containing the genotype(s) of the contributers to the profile from. If we would observe three loci at a certain marker in a 2 person mixture, we could have a lot of possible combinations for our contributers. Once DNA traces are mixed and put through the PCR and peak profile process it is impossible to determine which allele came from which DNA trace. This is what makes analyzing DNA mixture difficult.

Example 2.3. If look at the peaks in the electropherogram in figure 2.2 we observe three loci, 15, 16, and 17, at a single marker in a 2 person mixture, this leaves a lot of possible combinations for our contributers as the next table points out;

	Marker	Person 1	Person 2		Marker	Person 1	Person 2
1.	15, 16, 17	(15,15)	(16,17)	7.	15, 16, 17	(15,17)	(16,17)
2.	15, 16, 17	(15,16)	(15,17)	8.	15, 16, 17	(16,16)	(15,17)
3.	15, 16, 17	(15,16)	(16,17)	9.	15, 16, 17	(16,17)	(15,15)
4.	15, 16, 17	(15,16)	(17,17)	10.	15, 16, 17	(16,17)	(15,16)
5.	15, 16, 17	(15,17)	(15,16)	11.	15, 16, 17	(16,17)	(15,17)
6.	15, 16, 17	(15,17)	(16,16)	12.	15, 16, 17	(17,17)	(15,16)

Table 2.2: Possible contributers to a single marker 3 allele profile.

With both persons unknown we get 12 possible scenarios, non of which can be excluded. This means we will need additional evidence to determine the composition of the mixture. Additional evidence that usually is accessible in forensic cases is the genotype of one of the contributers, namely the victim. If we would know that person 1 has a $(15, 15)$ genotype we are left with only one option for person two (namely $(16, 17)$. However, If person 1 was an $(15, 16)$ we would still have 3 possibilities left for person two.

If we have a second look at figure 2.2, we notice that not all peaks in the profile are the same hight. As the peak at the 16 marker is the largest there is reason to believe that the original trace contained more DNA with a 16 marker, this could be evidence that it is the 16 marker that is shared by both contributers. The question is how we can quantify the information in the peak heights to help us determine the composition of DNA mixture models.

In the rest of this chapter we will develop a mathematical model to help answer questions like the one in the example above. With this model we want to test several hypotheses about the composition of the mixture and determine the most likely combination of contributers on the basis of the evidence available. This evidence usually consists of $\epsilon = \{$suspect genotype, victim genotype, mixture profile$\}$, and the hypotheses are for example

$$H_0 : S\&V \quad , \quad H_1 : U\&V$$

Here we denote the suspect by S, victim by V and U stands for an unknown contributer. We will calculate the weight of the evidence as the likelihood ratio

$$LR \;=\; \frac{P(\epsilon|H_0)}{P(\epsilon|H_1)} = \frac{P(H_0|\epsilon)}{P(H_1|\epsilon)} / \frac{P(H_0)}{P(H_1)}$$

To identify the genotype of each of the possibly unknown contributors to the mixture. We denote the genotypes of the contriputers by P_{gt} or if victim and/or suspect are known S_{gt} (the genotype of the suspect), V_{gt} (Genotype of the victim). Let Mix denote the peak profile of the mixture containing the peak heights of the alleles present. We would then calculate either $P(S_{gt}|V_{gt}, Mix)$ or $P(P1_{gt}, P2_{gt}|Mix)$ and find most probable combination.

2.2 A model for DNA mixtures analysis using peak heights

This paragraph will present a mathematical model on the DNA analysis using the short tandem repeats that we have seen in the previous paragraph. The aim of the model is to separate the different DNA profiles found in a mixed DNA trace. These mixed traces are in our case samples taken from a crime scene in which we would like to determine the genotypes of all contributers (the victim and one or maybe more criminals).

Several methods to do so have been suggested throughout the years. An example of one of these can be found in Evett el al. (1991) [11]. A more recent model addressing more difficult issues using baysian networks is given in Mortera el al. (2003) [12]. The common feature of all these methods is the fact they only use only a part of the information present in the peak profiles. Namely just the indication wether an allele is present or not.

As we've seen and argued in the previous paragraph the peak profile contains more information than only the fact whether an allele is present, yes or no. We would like to build a model that can use this extra information, in the form of peak height and/or areas in the profile. This will make the model rather more complex as peak areas are continuous variables in contrast to the indicator variables previously used.

A first model to use the peak areas was given in Cowell et al. (2006) [19] and uses a conditional gaussian distribution for modeling the peak area's. The model we present in this section is also by Cowell et al., and was presented in Cowell et al. (2007) [18]. This model assumes that the relative peak weights in the peak profile follow a gamma distribution dependent on the amount of DNA present before the PCR reaction. Using a gamma distribution will make the model very suitable to implement in the Bayesian networks setting.

Schematic overview of the model

The model suggested by Cowell in Cowell el al. (2007) [18] deals mixtures where DNA material is contributed by n persons. The framework of the model will consist of a Bayesian network structure as we have seen in chapter 1, this will allow use to use the junction tree algorithm to perform exact inference, avoiding MCMC procedures. At a very abstract level the network will look like this,

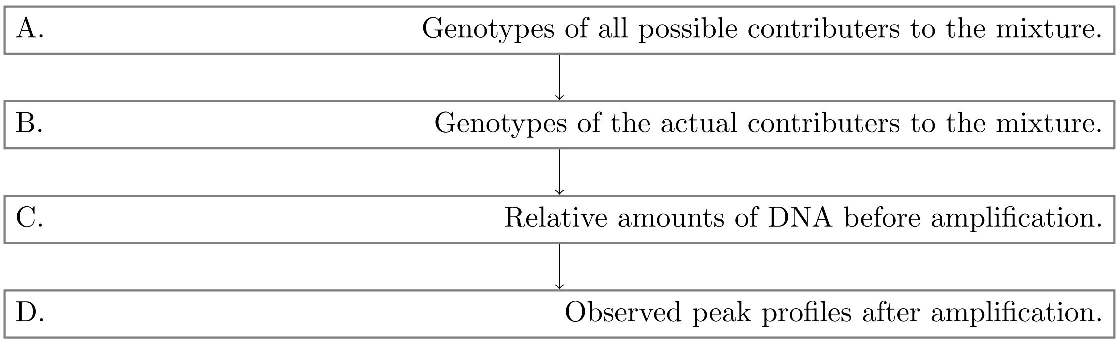

A. Genotypes of all possible contributers to the mixture.
B. Genotypes of the actual contributers to the mixture.
C. Relative amounts of DNA before amplification.
D. Observed peak profiles after amplification.

Figure 2.3: Schematic network layout for STR DNA analysis

The relations in Figure 2.3 from A to B and from B to C will be modeled in a Bayesian network and the step from C to D will consist of a conditional gamma model where the observed peaks depend on the pre-amplification amounts of DNA of every allele at each marker. How we can connect the two parts of the model will become clear after we've discussed the properties of the gamma model.

Assumptions and notation

A good model will need some reasonable assumptions. All assumptions Cowell et al. make in their paper are listed below. Whether or not these are reasonable assumptions will be discussed in the next paragraph.

Assumption 2.4. Assumptions for the simple model

1. The DNA mixture is made up of genetic material from n persons.

2. The DNA in sample analyzed is not degraded in any way so that the proportions of DNA material from different contributers are constant across all markers.

3. No PCR artifacts (we will explain these later) are present in the peak profile.

4. No Genetic anomalies occur with any of the individuals in the mixture.

5. We can compensate for repeat number scaling (we will explain this later) by using peak weights (= *peak area* $\times$ *repeat number*).

6. The peak weight of an allele in the total mixture will be equal to the sum of the peak weights on the same allele contributed by each individual separately.

As we've discussed before, we also assume that the genotypes of each person at each marker are independent (due to assumption 2.1.1). This will means that we can use the presented model on each marker independently and combine individual results afterwards. To indicate on which marker we are applying the model all variables will get a superscript m. To track variables belonging to a person we will use a subscript i. The genotype of person i at marker m becomes P_i^m. The set of all observed alleles at a marker m will be $\Omega^m = \{\alpha_1^m ... \alpha_k^m\}$. Finally, the mixing factors between different contributers will be $\theta = (\theta_1, .., \theta_n)$. The rest of the variables used will be defined along the way.

2.2.1 A Bayesian network on the possible contributers to the mixture

The Bayesian network part of the model brings together all possible contributers and all ways in which they could have contributed to produce a distribution over the possible pre-PCR amplification amounts of DNA. This network is build up in several levels, at the highest level all markers are combined together through the possible scenarios leading to the sample and the assumed mixing factor between the contributers.

As we are trying to model the STR DNA analysis from a forensic point of view, we usually have some indiction of the possible contributers to the sample found. For example, a trace found at a murder scene is likely to contain DNA material from the victim and of one or more suspects, and of course there is always a possibility that there is an unknown profile in the mix (especially if the suspect was not actually present at the all alleles). These three combined give the top part of the network, namely all possible contributers.

From all possible contributers and the number of contributers to the mixture (it is an assumption that we know this, we'll get back to this later) we can derive a set of all possible scenarios that led to the mixture. For example in a 2 person mixture from a case where we have a victim V and a suspect S this would be the set $\{V+S, V+U, S+U, U+U\}$ where U stands for an unknown person.

From all scenarios and the mixing factor between the persons contributing we can then determine the pre-PCR amounts of DNA for each allele at each marker. Here the genotypes of the contributers are included in the scenarios node. This top network will look like

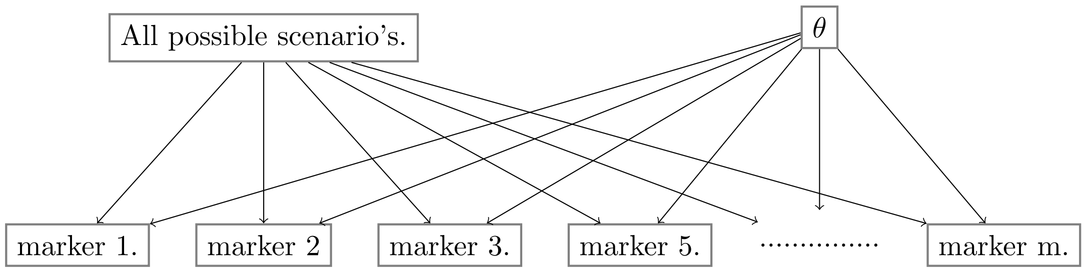


Figure 2.4: Bayesian netwerk on m markers

While this network looks small, it will pose computational issues if we'd try to determine the marginal factors simultaneously. This is due to the huge state space we have for the possible genotypes of the persons involved. We will give a small example on a sample with 4 observed alleles at each marker.

Example 2.5. Suppose we find a DNA mixture sample from 2 persons with 4 different alleles at each marker. To reconstruct all possible genotypes that are in the mixture we will look at the possible genotypes at the autosomal markers and take it to the power 10 (we can do this as markers are independent. Assume we observe alleles A, B, C and D at a certain marker. All possible genotypes that will match in this case will be in the following set

$$\{(A,A),(A,B),(A,C),(A,D),(A,X),(B,B),(B,C),(B,D),$$
$$(B,X),(C,C),(C,D),(C,X),(D,D),(D,X),(X,X)\}$$

Here X denotes the set of all other (unobserved alleles) which might have been in the mixture but disappeared in the peak profile. As we find 15 possible genotypes we can conclude that the size of the genotype state space for the autosomal markers is 15^{10}. If we multiply with the sex marker we get a figure of 30^{10} which is 5.9049×10^{14}

The only way we can do calculations is by splitting the network up into separate markers and combine the results afterwards. This brings state spaces for the genotypes back to 15 in case of 4 observed alleles if we correct (with the X) for all non observed alleles. The top of the single marker Bayesian network still has the scenarios node, and the mixture proportions which link it to all other markers but the genotypes of the persons involved are no longer included into the scenario.

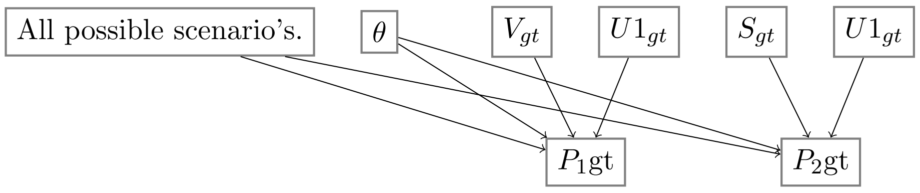


Figure 2.5: Top of the Bayesian marker network

Once all the genotypes in the mixture are determined, we can use them and the possible mixing factors to determine a pre-amplification relative allele number for each of the alleles observed. Will denote these by μ_α which are defined as

$$\mu_\alpha^m = \sum_{i=1}^{n} \theta_i n_{i\alpha}^m \tag{2.2.1}$$

Here $n_{i\alpha}^m \in \{0,1,2\}$ is the number of alleles α that person i has at marker m and θ_i the proportion of DNA contributed to the sample by person i. Figure 6 shows a network structure where we have n persons contributing;

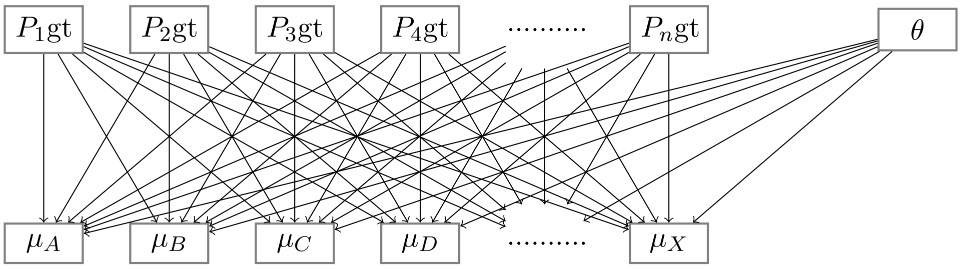


Figure 2.6: Bayesian network

The state space of the μ_α's depends on the state space for θ. In essence the θ_i's are continuous with the only restriction that $\sum_i \theta_i = 1$. In practice though we need to discretisize θ_i to be able to implement them in the Bayesian network (remember that all variables in chapter 1 had finite state space). Suggestions on how finely they should be done tend to depend on the number of persons involved. In a 2 person mixture it is usually done to 0.1.

An example the total Bayesian STR DNA network for 2 persons

If we combine figure 2.5 & 2.6 we get the total network for a marker. In figure 2.7 we give an overview of the entire network a single marker, 2 person mixture with 4 alleles observed.

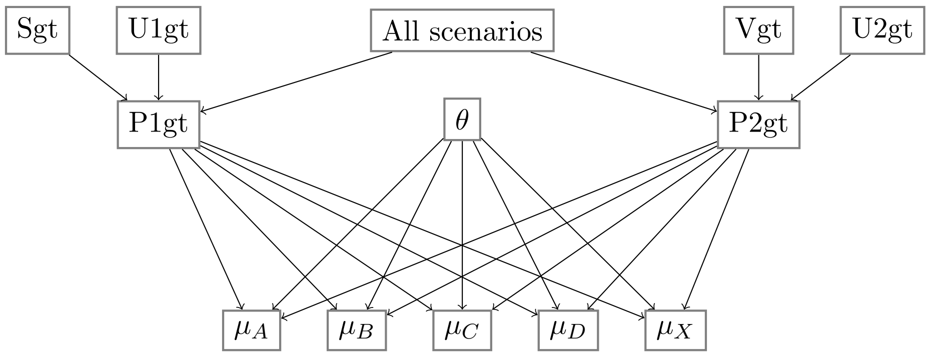


Figure 2.7: Two person 4 observed allele network for a single marker

If we round θ off to 0.1 the state spaces of the variables corresponding to the nodes in the network of figure 2.7 will be:

$$\begin{aligned}
\textit{All Scenarios} &\in \{(V+S),(V+U),(S+U),(U+U)\}\\
..gt &\in \{(A,A),(A,B),(A,C),(A,D),(A,X),(B,B),(B,C),(B,D),\\
&\quad (B,X),(C,C),(C,D),(C,X),(D,D),(D,X),(X,X)\}\\
\theta_1 &\in \{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}\\
\mu_\alpha &\in \{0,1,2\}\times\{0,1,2\}\times\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}
\end{aligned}$$

We just specify θ_1 as θ_2 is determined by the fact that both add up to one. There are two important things to point out at this moment. First, the size of the state space of μ_α depends on how fine we discretize θ. The second is that while the product state space of μ_α can become big it will always be finite. However as we've seen in the peak profiles, the post-amplification amounts of DNA at an allele are continuous. This prevents us from adding another layer of nodes beneath μ_α to represent the measured weights after the PCR process (remember that all variables in chapter 1 had discrete state space). The way to work around this is by using the measured weights as likelihood evidence on the pre amplification amounts of DNA at each allele. Before we can do this we need a model on how exactly these weights correspond to the μ_α's in the Bayesian network model. Cowell et al. [18] suggest a gamma model on a scaled version of the measured peaks which we will present in the next section.

2.2.2 A gamma model on the peak height information

Setup

We recall that the set of all possible contributers to the mixture is denoted by $i \in I = \{1,..,n\}$ and the fractions of pre-amplification DNA material contributed by person i is given as the vector $\theta = (\theta_1,..,\theta_n)$. The number of alleles α of individual i at marker m will be noted by $n^m_{i\alpha} \in \{0,1,2\}$. The area of a post-amplification peak of person i at allele α in marker m is given by $A^m_{i\alpha}$.

The areas $A^m_{i\alpha}$ are then transformed into the weights on which we will build the model $W^m_\alpha = \lambda_\alpha A^m_\alpha$, (here λ_α is the repeat number of allele α). The new *peak weights* W^m_α are now crudely corrected for repeat number scaling, by just multiplying area's with the allele numbers. Justification for this can be found in another paper by Cowell [5]. The weights are themselves sums of weights attributed by the individuals in the mixture. By the assumption that the peak weight of an allele in the total mixture will be equal to the sum of the peak weights on the same allele contributed by each individual separately, we have $W^m_\alpha = \sum_{i \in I} W^m_{i\alpha}$.

The conditional gamma model

This part of the model will deal with the conditional gamma model on the peak weights in the profile. One of the reasons a gamma distribution might be a good choice is the fact that it is closed under convolution, which is needed to respect the assumption on the distribution of the peak weights with respect to sums. More arguments for using the gamma distribution are given in [5], but the main motivation is that the gamma distribution has a very nice property that makes it suitable to be included in the Bayesian as likelihood evidence.

Recall that $W^m_{i\lambda}$ is the scaled weight of allele λ of person i in marker m and that $n^m_{i\lambda}$ is the number of alleles λ person i has at marker m. The key assumption now is that the weights contributed by each person are independently distributed according to gamma distributions depending on the number of alleles of person i and the mixing factor θ that has a shape parameter $\rho^m \gamma \theta^m_i n^m_{i\alpha}$ and an inverse scale parameter η^m.

$$W^m_{i\alpha} \backsim \Gamma(\rho^m \gamma \theta^m_i n^m_{i\alpha}, \eta^m) \tag{2.2.2}$$

Here $\Gamma(a, b)$ denotes the gamma distribution with density $f(x) = \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx}$, where we define $\Gamma(0, b)$ to be degenerate. The terms ρ^m and η^m are called amplification and respectively scaling factors. Finally, γ is some (unknown) quantitive measure of the amount of DNA pre-amplification. It is good to point out that ρ^m, η^m and γ are considered as fixed (unknown) parameters and θ^m and $n^m_{i\alpha}$ are random.

Conditionally on θ^m and $n^m_{i\alpha}$ the peak heights are assumed independent, so that any sum of gamma distributed peak heights at a given marker will again be gamma distributed (since the scale parameter is the same). We can thus write;

$$\begin{aligned} W^m_\alpha &= \sum_i W^m_{i\alpha} \backsim \Gamma\left(\rho^m \gamma \sum_i \theta^m_i n^m_{i\alpha}, \eta^m\right) \\ W^m &= \sum_\alpha W^m_\alpha \backsim \Gamma\left(\rho^m 2\gamma, \eta^m\right) \end{aligned}$$

The shape parameter in the last equation was derived from the fact that each contributer has 2 alleles at each marker. By first taking the sum over i and than over α this gives

$$\begin{aligned}\sum_\alpha \rho^m \gamma \sum_i \theta_i^m n_{i\alpha}^m &= \rho^m \gamma \sum_i \theta_i^m \sum_\alpha n_{i\alpha}^m \\ &= \rho^m \gamma \sum_i \theta_i^m 2 = 2\gamma\rho^m\end{aligned}$$

In the first term in the previous equation we can also do some rewriting. $\gamma \sum_i \theta_i^m n_{i\alpha}^m$ is the total amount of DNA of allele α pre-amplification, since there is a total of 2γ of DNA at the marker. We can write the relative proportion of allele α at marker m as

$$\begin{aligned}\mu_\alpha^m &= \frac{\gamma \sum_i \theta_i^m n_{i\alpha}^m}{2\gamma} \\ &= \frac{\sum_i \theta_i^m n_{i\alpha}^m}{2}.\end{aligned}$$

This transforms (eq...) into

$$W_\alpha^m = \sum_i W_{i\alpha}^m \backsim \Gamma\left(\rho^m \gamma \mu_\alpha^m, \eta^m\right).$$

Up to this point the exact measure of the peak heights is rather arbitrary. For reasons that will later become clear we scale them to produce *relative peak weights* per marker which sum to unity. We write

$$R_\alpha^m = \frac{W_\alpha^m}{W^m}.$$

This might seem a strange thing to do as these new weights will now be dependent on each other but as we will see the distribution of the relative peak weights has a very nice property. Since given $\mu^m = \{\mu_\alpha^m\}_\alpha$ there is independence between the W_α^m's, a known result from probability now is

$$\{R_\alpha^m\}_\alpha \backsim Dir(2\gamma\rho^m\{\mu_\alpha^m\}_\alpha).$$

Here Dir($\{\lambda_i\}_i$) is the Dirichlet distribution which has density function

$$\begin{aligned}f(x_1,...,x_k;\lambda_1,..,\lambda_K) &= \frac{1}{\beta(\lambda)} \prod_{k=1}^{k=K} x_i^{\lambda_k - 1} \\ &= \frac{\prod_{k=1}^{k=K}\Gamma(\lambda_k)}{\Gamma\left(\sum_{k=1}^{k=K}\lambda_k\right)} \prod_{k=1}^{k=K} x_i^{\lambda_k - 1}.\end{aligned}$$

The marginal distribution of each R^m_α is a beta distribution with shape parameters, $2\gamma\rho^m\mu^m_\alpha$ and $2\gamma\rho^m(1-\mu^m_\alpha)$ and has expectation and variance:

$$\begin{aligned} E(R^m_\alpha) &= \mu^m_\alpha \\ Var(R^m_\alpha) &= \frac{\mu^m_\alpha(1-\mu^m_\alpha)}{2\gamma\rho^m+1} \\ &= \sigma^m\mu^m_\alpha(1-\mu^m_\alpha). \end{aligned}$$

The final equation can be derived by rewriting the term with γ and ρ^m (which we can do as they are assumed to be fixed for each marker m) as

$$\sigma^m = \frac{1}{2\gamma\rho^m+1}.$$

The precise value of σ^m depends on several factors. Cowell suggests a value of 0.01 as this corresponds to values observed in controlled experiments. Therese Graversen has written her thesis [9] on the estimation of this parameter and for more detailed information I'll refer the reader to her results.

Thus far we have seen a model on the relative peak areas $\{R^m_\alpha\}_\alpha$ which is conditional on the relative scaled allele numbers μ^m_α, and we have a Bayesian network that deduces the μ^m_α from the possible contributers to the sample and the mixing factor θ. The next section will present a way to connect these two parts.

2.2.3 Combining the gamma model with the Bayesian network

What we would like to do, is add a final layer of nodes to our Network to include the observed relative peak weights. The problem is that these weights are continuous variables. We could choose to make them discrete (as we did with the mixing factors), but this would lead to huge state-spaces. Luckily there is an easier way of implementing the weights in the network, through the introduction of likelihood evidence on the μ^m_α.

This works because conditional on $\mu^m = \{\mu^m_\alpha\}_\alpha$ the total and relative peak weights are independent of the other nodes in the network we've presented, and because of a property of the Dirichlet distribution that we can derive on the set of relative peak weights on each marker.

We know that the likelihood of $\mu^m = \{\mu^m_\alpha\}_\alpha$ given the observed weights on a marker m is proportional to the likelihood of μ^m given the relative scaled weights

$$\begin{aligned} L(\mu^m|W^m) &= L(\mu^m|R^m, W^m) \\ &\propto L(\mu^m|R^m). \end{aligned}$$

For $L(\mu^m|R^m)$ we can derive a nice property that comes from it's Dirichlet distribution, namely that it factors over the set of observed alleles at a marker.

$$\begin{aligned} L(\mu^m|R^m) &= f(R^m|\mu^m) \\ &= \Gamma(\sum_\alpha 2\rho\gamma\mu_a)\prod_\alpha \frac{(R^m_\alpha)^{2\rho\gamma\mu_a - 1}}{\Gamma(2\rho\gamma\mu_a)} \\ &\propto \prod_\alpha \frac{(R^m_\alpha)^{\mu_a(\frac{1}{\sigma^m}-1)-1}}{\Gamma(\mu_a(\frac{1}{\sigma^m}-1))} \end{aligned}$$

This factorization property enables the introduction of the likelihood evidence in the network at each of the μ^m_α nodes. For the precise use of this evidence we refer back to chapter 1 where the theoretical basis is discussed or to chapter 3 where we use this construction in practice on the scorpion case data.

2.3 Remarks on the DNA mixture analysis model

In the previous paragraph we saw the simple STR DNA model presented by Cowell et al. in [18]. Their model made a couple of assumptions 2.1.1 that we will revisit in this paragraph. Some of the assumptions will require us to take a step back and look at some more detail to the PCR process and the possible anomalies that can occur in DNA. Once we've discussed the assumptions we will also take a better look at the gamma model and the way it deals (or can not deal) with unobserved alleles.

Anomalies in the PCR

One of the assumptions made in section 2.2 was that there were no PCR artifacts present in the peak profile. Unfortunately this is not true in general practice, as we discussed, the PCR is a very complicated temperature controlled procedure that amplifies the DNA at loci of interest. During this procedure there are several types of anomalies are known to occur. We present a short overview and refer the interested reader to chapter 5 of

Butler [4].

A ***stutter*** is a copy error in the PCR reaction where either an extra repeat is added at a marker or one repeat is mistakenly dropped. In this case we can expect to find an extra peak in the profile. Stutters are very common but are not visible in all cases as only stutters occurring during the first cycles of the PCR will be amplified enough to produce significant peaks in the profile. Therefore the so called *stutter peaks* will be smaller than the original allele peak. Most literature (chapter 3 of Butler [4]) states that the peak hight of a stutter will be less than 15% of the original peak, and in most cases they will be much smaller. The length of the repeated motive at a marker also has influence.

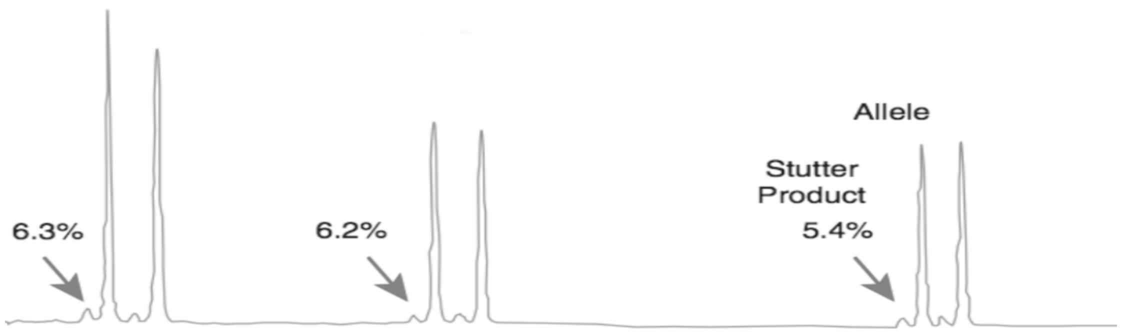


Figure 2.8: Typical stutter peaks on 3 markers.

Since the stutter peaks are small in comparison to those of the true alleles it is easy to identify stutter in a single person DNA sample but if it is known that a trace contains a mixture of DNA then stutter can become a problem. If for instance we have a 1:10 mixture, true alleles of the small contributer can be easily mistaken for a stutter of the main DNA donor and vice versa.

In contrast to the stutter phenomena where extra alleles are produced, ***drop out*** is the term for alleles that fail to be amplified during the PCR reaction. There are two ways in which a drop out occurs. The first is a mutation on the primer binding site on one of the chromosomes, this prevents the allele at that binding site from being multiplied in the PCR. The result is a so called silent or null allele. Detecting these null alleles can be difficult as there is no (easy) way to tell whether we are dealing with a homozygote- or a heterozygote genotype with a silent allele. Slight modifications to the PCR process can be helpful but they do not work in all cases. Luckily this type of drop out is very rare and most modern STR kits are designed to prevent them from happening.

The second type of drop out is a bigger challenge for forensics and it appears if a DNA trace with a low number of DNA strands (low copy number) is put through the PCR. In this case the PCR will sometimes have problems properly copying the STR sites. This effect is assumed to be stochastic and is highly dependent on the amount of DNA and the length (in base pairs) of the section of DNA that is being copied. We can use information on other peaks in the profile to tell something about the chances on drop out. A lot of high peaks are a good indicator that nothing dropped out; we must be careful if only small peaks are observed.

An other well know artifact of the PCR reaction is the ***repeat number scaling***. Alleles with a higher repeat number tend to be less amplified than alleles with a low repeat number. When analyzing a sample from a single DNA source this does not pose a problem. If we look at a DNA mixture, peak height information will become more important in identifying the genotypes in the mix. Rescaling of the peak area does compensate for the repeat number scaling effect.

The last artifact we will discuss here is called ***heterozygote imbalance***, in this case the two peaks corresponding to the alleles from a heterozygote genotype are of different height. This does not need to be a big problem in determining the genotype of a specific sample but it can cause problems when looking at mixtures of DNA, since we've argued that peak heights can help us in identifying the number of contributers to a sample.

The anomalies listed above can dramatically alter our interpretation of a peak profile. Due to stutter, we might have reason to think more persons were involved in a mixture then is actually the case. Drop out can possibly wrongly acquit someone from contributing. This means that we will need to alter the presented model to take both phenomena into account.

Genetic anomalies in the contributers

A second assumption of the model was that there are no genetic anomalies in any of the contributers to the mixture. These anomalies are possibly more rare than the ones associated with the PCR reaction, but they can have a big influence on our analysis of the peak profile. There are basically two types of genetic mutations that are known to occur.

When our chromosomes are passed down from parents on children they are not in all cases perfectly copied. Small ***mutations*** in our DNA actually occur quite often. When performing an STR analysis on for instance a paternity case we must include the small probability that a mutation has occurred at an STR marker. In crime scene analysis however, this will be less of a problem as the mutation would be in the found trace as well as in the reference sample.

A more difficult mutation to deal with that is known to take place during the embryonal phase can lead to something called ***mosaicism***. In this mutation not all cells carry the mutation, which means that an individual can have two different DNA profiles at a certain marker, depending on which cells are profiled. Moreover it could be that a reference sample will show one profile, but the DNA found at a crime scene can contain the other, thus possibly wrongly acquitting someone of a crime because the biological samples came from different part of the body of the same person.The main problem in dealing with mosaicism is that it is hard to observe and almost no relevant literature exists on the frequency of its occurrence.

An unknown number of contributers

The next assumption made in 2.2 was that the number of contributers to the mixture is known. In some cases, for instance a robbery of a rape, it might very well be that the victim can tell the number of perpetrators involved. There are however situations in which the victim can not provide any information (because he or she is dead for instance). Is it reasonable to assume we know the number of contributers in those cases?

Obviously we can make an educated guess on the number of contributers as the number of observed alleles does provide some information. This does not always lead to a perfect determination though. If we observe a marker with 4 alleles we can only infer that at least two persons must have contributed (again under the assumption that each person has at most 2 alleles). There is however no way of telling how many people there were in total. In the 4 allele case we can have, 2 heterozygotes, a heterozygote and 2 homozygotes or even 4 homozygotes, assuming that no alleles are shared between the contributers. If we include that possibility, even more combinations of contributers are possible.

Now by combining the number of alleles from all markers we would reduce the chance on some of the scenarios. For instance assume there is a marker with just two alleles, this would mean that in the case of the 4 homozygotes either 2×2 (or 1×3) of the persons had matching alleles, which is less likely than just two persons matching.

One way to overcome the process entirely would be to make the model conditional on the number of persons involved. This way we can take a lower bound on the number of persons from the largest number of observes alleles at a individual marker and run the model on all number of persons greater than the lower limit.

Actually, from a certain point the a-priori probabilities on n persons with a maximum of M observes alleles will become very small and we can possibly stop. If we assume 20 possible alleles at each marker. Then for unrelated individuals (this way we can assume alleles to be independent within markers). With some combinatorics a bound on

$$P(n > M = \max_m\{\#(\alpha^m)\})$$

can be found. This probability is small enough to give an upper limit for the number of people in the mixture in a general case. Of course things change if all suspects are related, then we have dependence between alleles and profiles sharing alleles would be more likely.

So, in conclusion: It is not a big problem if we can't be sure about the number of contributers. A lower limit can be easily found and as argued in a general case it is unlikely that there are more persons involved then the maximum number of alleles at a marker, for all remaining number of persons, we will have to calculate

Can we ignore degraded DNA?

This assumption is more important then one might think. It actually assures that each contributer to the mixture did so with two alleles on all markers which is at the basis of our model. There is not much literature to be found about the amount of degraded DNA that one would find under for instance fingernails in the general population but we can always expect there to be some DNA presumably.

Post-PCR assumptions

This concerns basically two assumptions that are stated but which sound paradoxical. Namely, the fractions of DNA pre and post amplification of a single person are the same and yet we need to scale the post amplification amounts by repeat number. Since the first suggests that every piece of DNA is amplified an equal amounts there would be no reason to correct for peak heights.

The scaling by repeat number is a generally accepted method, and thus some sort of correction on θ_i should be applied as well. The problem is that while we can observe the peak heights, θ_i are unknowns. Moreover, the scaling would depend on the genotypes of the contributers, which we are trying to figure out in the first place.

Issues with the Gamma model

As stated in the introduction to this section using the gamma model has, apart from its nice aspects also a down side. It is namely unable to model random "noise" we observe in the peak profiles. If we recall a typical profile from the biology paragraph, we have high peaks indication the alleles an some noise in the form of small peaks around them. Now if we look at the marginals of the relative peak weights which are beta distributed we find,

$$R_\alpha^m \backsim \beta(\beta_1, \beta_2)$$

$$\begin{aligned} \beta_1 &= 2\gamma\rho^m\mu_\alpha^m \\ \beta_2 &= 2\gamma\rho^m(1-\mu_\alpha^m) \end{aligned}$$

If we were to insert $\mu_\alpha^m = 0$ into this distribution we would get that $E[R_\alpha^m] = 0$ and $Var[R_\alpha^m] = 0$ meaning that with probability one, $R_\alpha^m = 0$. Ergo, any noise observed in the peak profile must have be the result of an amount of DNA in the set $\{\mu_\alpha^m\}_\alpha$.

At the moment forensic scientists use a hard cut off to eliminate this problem, meaning they discard all noise below some level and use the "cleaned" profile to do analysis. This is probably a valid method, but caution must be applied when dealing with extreme mixing factors (one large and one small contributor), where small peaks could actually indicate an allele.

2.4 An Improved model for DNA mixture analysis

This paragraph will discuss the options and ways in which we can address the issues concerning the assumptions on which the model from section 2.2 relies. There were quite some things pointed out, but in this paragraph we'll focus on the modeling of anomalies that are known to occur during PCR. The main reason to do so is that some other issues (like the questions of degenerated DNA) require more dramatic changes to the model, and other points of discussion (number of contributers etc) are critically dependent on current computing power.

2.4.1 A Bayesian network to model stutter

We recall from the previous paragraph that a stutter is a peak in the profile which originates from a faulty copy of its neighboring allele during the PCR reaction. Stutter is a difficult problem, especially in samples where the mixture ratios come in the region of 10:1 or more. In this case there will be no way to distinguish the stutter peaks from peaks corresponding to alleles of the small contributor. Luckily there is a lot known about stutter which helps in the process of modeling. The strategy will be to include a new layer of weights to the Bayesian network we presented earlier.

The new layer will allow for some of the relative pre-amplification weights to not only depend on the number of corresponding allele numbers of the contributers but also of their direct neighbor alleles. For example, in section 2.2 we took

$$\mu_\alpha^m = \sum_i n_{i\alpha}^m.$$

We will change this into a stutter version as;

$$\begin{aligned}\hat{\mu}_\alpha^m &= \sum_i n_{i\alpha}^m + \mathbf{1}_{\{st_{\alpha+1}^m\}} \sum_i n_{i(\alpha+1)}^m \\ &= \mu_\alpha^m + \mathbf{1}_{\{st_{\alpha+1}^m\}} \mu_{\alpha+1}^m.\end{aligned}$$

Here $st^m \in \{0,1\}$ is a boolean variable indicating stutter of an allele α at a specific marker m. The observant reader will have noticed at this point that this approach only models down-stutter. The reasoning behind this is that an up-stutter is about a 1000 times less likely to occur and the advantage of including it would not outweigh the increase in state space. For the same reason we will also neglect modeling double stutter (a stutter of the stutter peak). If we implement this into the network from figure 2.7 (a single marker of a network modeling a two person mixture with four consecutive observed alleles) we get the network presented in figure 2.9.

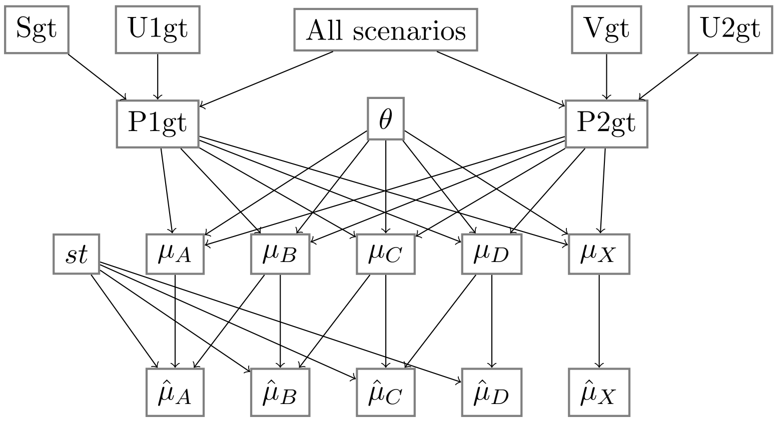


Figure 2.9: Two person 4 observed allele network for a single marker

It is important to point out that if the repeat numbers corresponding to the alleles differed by more then one repeat the bottom part of the network in figure 2.9 would look different. In this case the arrows would be rearranged to produce stutter in the X category. In real life we will have to adjust the "stutter arrows" according to the observed alleles. This means that a model incorporating stutter is always tailor-made to a specific DNA trace.

The last thing we will need to implement this stutter model are the prior probabilities for stutter on the alleles observed. For simplified reasons we will take stutter probabilities equal for all alleles on all markers (as suggested by the NFI) and we will introduce two possible levels of stutter. Just like the mixing proportion θ we would like to make st a continuous variable but again this would lead to too big state spaces.

2.4.2 An exponential model to deal with drop-out

Just like stutter a drop-out of an allele can have a big influence on the analysis of a peak profile. As we have seen in the previous paragraph, there are two types of drop out that we will need to deal with. Luckily, unlike the solution to stutter, the solution to the drop-outs can be modeled with relative ease on the general network model we've presented in the paragraph 3.2.

The changes we make are suggested by Cowell in [5] and use an exponential distribution for modeling the drop-out probabilities in the stochastic case and will add a silent allele to the possible state space of the genotypes in the top nodes of the network. We will start with a short explanation of silent alleles.

As there is no way of telling if an allele dropped out due to a mutation in the primer binding site region (anyway not without investigating those regions) we must take this into consideration when it comes to determining the genotypes that contributed to the mixture. We do this by adding a state S to the possible genotypes. This unfortunately leads to a (much) bigger state space. We can illustrate this with the new genotype state space for the network in figure 2.9:

$$\begin{aligned} ..gt \in \; & \{(A,A),(A,B),(A,C),(A,D),(A,X),(A,S),(B,B),(B,C),(B,D), \\ & (B,X),(B,S),(C,C),(C,D),(C,X),(C,S),(D,D),(D,X),(D,S),(X,X),(X,S)\} \end{aligned}$$

We can control the chance on a silent allele by changing the prior on the S state. There is unfortunately not a lot of literature known about these primer binding site mutations so putting a number on the prior is difficult, we discuss this in detail in the next chapter where we will implement the model. The reader might wonder why we don't add the state (S,S), this is again a trade off between computation time and completeness of the model. As these mutations are very rare, a double mutation (on both chromosomes) is not worth modeling.

The second drop-out phenomena will be modeled differently. It will require the network to include yet another level of nodes, this time between the persons' genotype and the relative allele numbers. Since the probability of a drop-out of a certain allele depends on the amount of DNA we'll model it in the following way, denote the allele numbers of alleles that will be amplified (thus not drop out) by $n_{i\alpha}^{amp}$. We'll assume that $P(n_{i\alpha}^{amp}|n_{i\alpha},\theta_i)$ decreases exponentially in the quantity of,

	$n_{i\alpha}$		
$n_{i\alpha}^{amp}$	0	1	2
0	1	$\exp(-\psi\theta_i)$	$\exp(-2\psi\theta_i)$
1	0	$1-\exp(-\psi\theta_i)$	$2(1-\exp(-\psi\theta_i))\exp(-\psi\theta_i)$
2	0	0	$(1-\exp(-\psi\theta_i))^2$

Table 2.3: Drop out probability table

Here we use the estimate $\psi = -\log P(\text{drop-out}= True|\theta = 1) = -\log(0.01)$, which is given in [5]. Figure 10 shows the changed part of the network for one person.

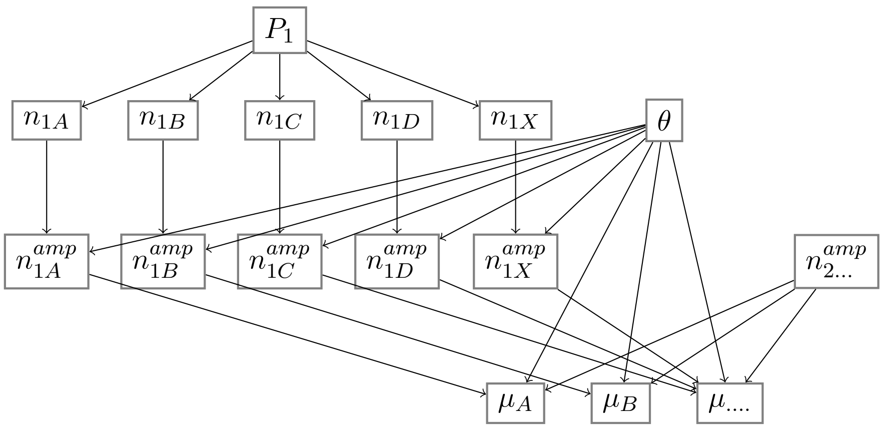


Figure 2.10: Drop out nodes included in a part of the two person example network

There is one important consequence of dealing with drop out in both ways. We can no longer use the fact that a person contributes exactly two allele s to the mixture. Where we previously used $\sum_\alpha n_{i\alpha} = 2$ to find

$$W^m \quad = \quad \sum_\alpha W_\alpha^m \backsim \Gamma\left(\rho^m 2\gamma, \eta^m\right)$$

we now get

$$\begin{aligned} W^m &= \sum_\alpha W^m_\alpha \backsim \Gamma\left(\rho^m \psi^m \gamma, \eta^m\right) \\ \psi^m &= \sum_\alpha n^{m\ amp}_{i\alpha} \end{aligned}$$

2.4.3 How to model DNA mutations and mosaicism

As discussed before, DNA mutations and mosaicism are difficult to model in general. Because there is very little known about them we must be careful in stating that mosaicisms are rare but we will assume that they are. This means that there is little reason to include the probability of a mosaicism on every allele in every person in the mixture. Producing a model that can deal with these issues will be a custom job that only need to be done for traces where there is a reason to suspect a artifact arising due to genetics. If we have such a model we could compare the likelihoods it produces with for instance a model without a genetic artifact but with more persons involved.

We'll illustrate a custom model for a case where mosaicism is suspected in figure 2.11. We could model a mosaicism as basically a large stutter on an allele α_{mos} where we also specify the direction of the mutation (up or down) but this is not what is happening in reality since the mosaicism happens only in one person.

Just as in the drop out network we'll model mosaicism on the allele numbers $n^m_{i\alpha}$ in the following way, where we have a mosaicism from allele C to D with $\lambda_C = \lambda_D - 1$ and mos denotes the levels of mosaicism (the amount of DNA transferred between loci due to the mutation).

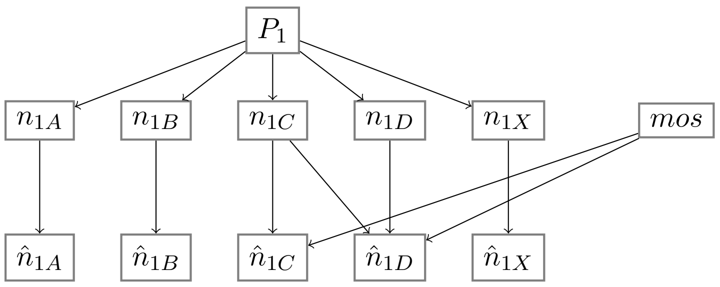


Figure 2.11: One person mosaicism model.

Now this is an easy network to implement but one fundamental thing changes. We no longer have $\hat{n}^m_{i\alpha} \in \{0, 1, 2\}$ for all alleles due to the mosaicism factor. This means that the state spaces of all nodes concerning the number of alleles of a person will be altered. This might seem to be a difficult way of modeling mosaicism however we have kept $\sum_\alpha \hat{n}_{i\alpha} = 2$ and thus nothing else (apart from a change in state spaces) changes in the model.

2.4.4 An example the expanded STR DNA network for 2 persons.

In conclusion of this chapter we will present the final model as we will be using it in the next chapter. The basic model we had in section 2.2 is expanded to include all modifications that we discussed in section 2.4. Figure 2.12 gives the Bayesian network leading up to the pre-amplification amounts of DNA μ_α. It is on those nodes that the likelihood evidence derived from measurements on the peak heights is entered.

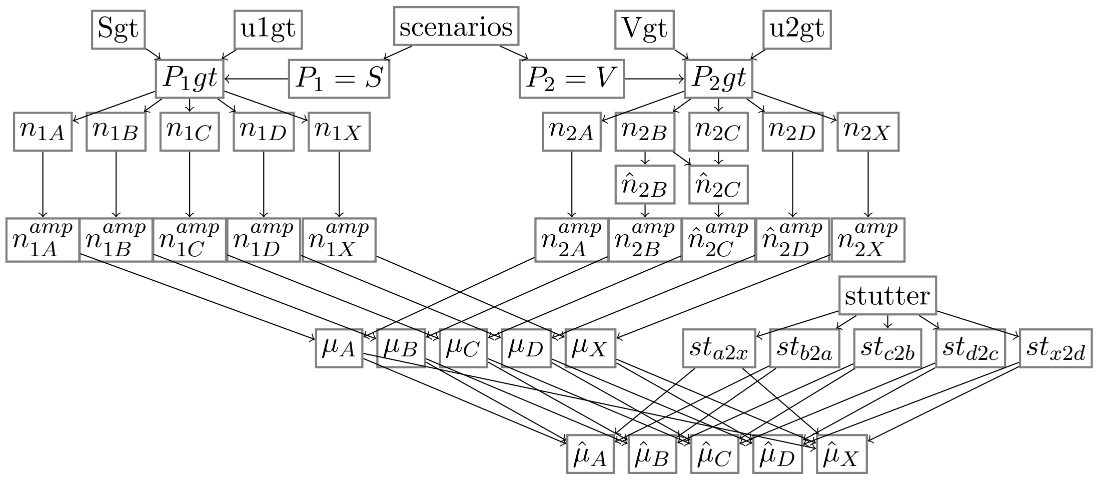

Sgt
u1gt
scenarios
Vgt
u2gt
P_1gt
$P_1 = S$
$P_2 = V$
P_2gt
n_{1A} n_{1B} n_{1C} n_{1D} n_{1X}
n_{2A} n_{2B} n_{2C} n_{2D} n_{2X}
$\hat{n}_{2B}$ $\hat{n}_{2C}$
n^{amp}_{1A} n^{amp}_{1B} n^{amp}_{1C} n^{amp}_{1D} n^{amp}_{1X}
n^{amp}_{2A} n^{amp}_{2B} $\hat{n}^{amp}_{2C}$ $\hat{n}^{amp}_{2D}$ n^{amp}_{2X}
stutter
μ_A μ_B μ_C μ_D μ_X
st_{a2x} st_{b2a} st_{c2b} st_{d2c} st_{x2d}
$\hat{\mu}_A$ $\hat{\mu}_B$ $\hat{\mu}_C$ $\hat{\mu}_D$ $\hat{\mu}_X$

Figure 2.12: 2 person model with stutter, drop-out and a mosaicism (B to C in P2)

Here we get the following state spaces, which we will take a better look at once we implement the model into the computer for use on real data in the next chapters.

$$\begin{aligned}
\textit{All Scenarios} &\in \{(V+S),(V+U),(S+U),(U+U)\}\\
..gt &\in \{(A,A),(A,B),(A,C),(A,D),(A,X),(A,S),(B,B),(B,C),(B,D)\\
&\quad (B,X),(B,S),(C,C),(C,D),(C,X),(C,S),(D,D),(D,X),(D,S)\\
&\quad (D,S),(X,X),(X,S)\}\\
\theta_1 &\in \{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}\\
n_\alpha &\in \{0,1,2\}\\
n_\alpha^{amp} &\in \{0,1,2\}\\
\mu_\alpha &\in \{0,1,2\}\times\{0,1,2\}\times\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}\\
\mu_\alpha^{amp} &\in \{0,1,2\}\times\{0,1,2\}\times\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}\times\{stutter\}
\end{aligned}$$

Chapter 3

Calculations on Bayesian networks

In this chapter we will develop a computer model that implements the extended STR model as it was presented in section 2.4. For calculations on the graph structure, the junction tree algorithm and Bayesian network structures from chapter 1 are used. As we saw in chapter 1 implementing the steps that lead up to a factorization over a junction tree associated with a Bayesian network are complicated. To overcome this issue we try to utilize existing software and adapt it to our case instead of writing difficult programs ourselves.

The implementation of the model from chapter 2 is done in Matlab with use of the Bayesian network toolbox and all code used can be found in the appendix. For all calculations I used a 2 GHZ macbook with 4gb of memory running Matlab R2009b, and version 1.0.4 of the BNT.

3.1 The Bayesian network toolkit

There is a lot of software available that can deal with Bayesian networks. Unfortunately, these programs are either free and rather limited in programming big networks (GeNie), too expensive (HUGIN) or not available for sale (MAIES, [17]). When I started working on the calculations I did so in the R language, the preferred (open source) statistics program by many known for its extensive library of add on toolkits. The R packages that can perform exact inference over network structures (for instance the junction tree algorithm) cannot handle the introduction of likelihood evidence (section 1.5.5) that we use in the peak height model. Of course writing a new package would be a good exercise, but it would take too much time.

This meant something else was needed. After some research the Bayesian network toolbox for Matlab seemed like a good platform. While Matlab can be quite expensive (a corporate license will cost upwards of 1000 US$), student and research licenses are relatively affordable. More important is the fact that the BNT can deal with the introduction of likelihood evidence which is crucial for the implementation of the model presented in chapter 2. The following sections will give a short introduction to the BNT which leads up to a short example of a total implementation of a Bayesian network in the BNT.

3.1.1 An introduction to the BNT

The Bayesian Network toolbox for Matlab was written by Kevin Murphy who is an associate professor at the University of British Columbia [15]. All documentation including the download for the entire toolbox can be found at http://code.google.com/p/bnt/. Our application of the BNT just uses a small proportion of the possibilities the toolbox has, which include multiple inference algorithms, several methods for parameter and structure learning and the possibility to include continuous (gaussian) variables in the networks used.

We will restrict ourselves to inference questions over Bayesian networks where all nodes (variables) have discrete state space, and we will use the junction tree algorithm introduces in section 1.5 to perform the inference queries. With regard to evidence we will use both hard and soft (likelihood) evidence. The next sections give a short overview of the use of the BNT. More documentation can be found online at the webpage mentioned above.

Preliminarys: download and installation

The BNT can be download from the website as a zip file. To install, unzip FullBNT.zip to a convenient folder, next open Matlab and add BNT to your path. You can go to

File→Set.Path, and then click on "Add Folder with Subfolders" and select the bnt folder. Alternatively you can set the working directory to the directory containing BNT and execute the following (which also tests if installation is successful).

```
cd ....\extraction_folder\FullBNT\FullBNT-1.0.4;
addpath(genpathKPM(pwd));
test_BNT
```

The genpathKPM function is like the builtin genpath function, but it does not add directories called Old to the path, thus preventing old versions of functions accidently shadowing new ones. The warnings occur because Matlab 7 added functions with the same names as the BNT functions. The BNT versions will shadow the built-in ones, but this should be harmless.

Data structures: creating the Bayesian network

Before we can even start to think about calculations we will need to encode the Bayesian network itself. This means not only the graph $\mathcal{G}$ but also all state spaces of the random variables $\mathcal{X} = (X_v)_{v \in V}$ it encodes and the conditional probability tables between all variables. We start by encoding the graph itself.

There is obviously more than one way to encode a graph structure. In the BNT, nodes in the graph are not labeled. This means that we just choose a numbering on the nodes in a convenient way and then specify the graph $\mathcal{G}$ through its adjacency matrix.

Definition 3.1. In a graph $\mathcal{G}$ associated with a Bayesian network over n variables we call $O_{\mathcal{X}} = (X_1, .., X_n)$ an ordering if it induces that parental nodes are always higher in the ordering then their children.

Lemma 3.2. *Each graph $\mathcal{G}$ associated with a Bayesian network has a good ordering.*

Proof. This is a proof by construction. The following algorithm provides an ordering.

- Let S denote the set of all n variables encoded by a graph $\mathcal{G}$.
- Let $O_{\mathcal{X}}$ be a vector of length n, the ordered vector of variables.

- For i=1:n, pick any node s from S without parents in $\mathcal{G}$. Set $O_{\mathcal{X}}(i) = s$ and remove s from S.

□

This construction eliminates the need to specify directions on the edges in the adjacency matrix, as they are all directed from lower to higher indices.

Lemma 3.3. *Using the ordering $O_{\mathcal{X}}$ we can specify the entire directed graph $\mathcal{G} = (V, E)$ with the binary matrix $G = O_{\mathcal{X}} \times O_{\mathcal{X}}$ with $C_{i,j} = 1$ if $(i, j) \in E$.*

Next we need to specify the state space of each variable $X_i \in O_{\mathcal{X}}$ in the graph which we will denote with Ω_X. Again the BNT does not deal with labels just, number of states is needed. If we stick to the ordering $O_{\mathcal{G}}$ we can specify the number of state spaces with a vector.

$$ns(i) = |\Omega_{O_{\mathcal{X}(i)}}| \tag{3.1.1}$$

While we specify the number of states for each node, the actual state spaces are not defined yet. It is important to point out at this point that we do not assume any ordering on the state spaces in any way; that is, they represent categorical quantities, like male and female, rather than ordinal quantities, like low, medium and high. The definition of the spaces is only needed at the point where we want to introduce evidence but in the process of defining the conditional probability tables we need to keep track of the meaning of the state spaces. We see more on this later when we apply this framework to the model on STR markers.

We know from the first chapter that a conditional probability for a variable X is a function from the product space of the state space of X with the state spaces of its parental nodes.

$$Cp(X) : \Omega_X \times \Omega_{Pa(X)} \to [0, 1]$$

In the case that X would have just one parent, we could specify $Cp(X)$ with a table of probabilities, this is why conditional probability functions are usually called conditional probability tables but as variables can have more than one parent we either need to specify the product space of its parental nodes and use this as one parent or we need some sort of multidimensional table. To Matlab both options are the same. The only thing of importance is just how Matlab stores these multidimensional arrays (and note that in Matlab, unlike C, arrays are indexed from 1). The arrays are placed in memory such that the first index toggles fastest. This means that the product space of two vectors $(x_1, .., x_n) \times (y_1, .., y_n)$ is stored as the vector $((x_1, y_1), (x_2, y_1), .., (x_n, y_1), (x_1, y_2), ..)$. In the BNT we will specify CPTs in the following way,

Definition 3.4. A CPT on a variable $X_i \in O_{\mathcal{X}}$ is the probability vector on the product space

$$\Omega_{Pa_1(X_i)} \times ... \times \Omega_{Pa_n(X_i)} \times \Omega_X$$

Here $Pa_1(X_i)$ is the parent of X_i that is highest in the ordering $O_{\mathcal{X}}$ etc..

We will illustrate just how to implement this in Matlab in a short example after the next section. Let's give a summary on the steps we need to take in BNT to specify the network.

```
n=i;  X_..=i;                   %define number of nodes and node ordering
ns=ones(n);                     %define state space size vector
ns(i)=Omega(X_i);               %define state spaces
dag=zeros(n);                   %define adjacency matrix
dag(i,j)=1;                     %edges from X_i to X_j
bnet=mk_bnet(dag,ns);           %make the network structure
bnet.CPD{Xi}=
  tabular_CPD(bnet,Xi,Cp(Xi)); %define CPT's
```

Inference: obtaining results from the Bayesian network

Having created the Bayesian network we can start to think about doing an inference query on it. There are many different algorithms for doing inference on Bayesian networks, which all make different tradeoffs between speed, complexity, generality, and accuracy. BNT offers a variety of different inference "engines". We will not discuss these in this thesis but the interested reader can find more on this in the online documentation. We will use the junction tree engine, which is the mother of all exact inference algorithms. The specific engine used can be defined as.

```
engine = jtree_inf_engine(bnet);
```

Once we have specified the inference algorithm we can start computing marginal distributions over the nodes of our interest. These marginal distributions will depend on the evidence we observe on related variables and the priors we defined on the root nodes. We can add this evidence to our network in two different ways. Hard evidence will denote observed states of a variable and can be added via

```
evidence = cell(1,N);
evidence{Observed node} = 2;   %we observed Omega_observed node (2)
```

We use a 1D cell array instead of a vector to cope with the fact that nodes can be vectors of different lengths. In addition, the value [] can be used to denote the absence of evidence, instead of having to specify the observation pattern as a separate argument (providing a list of variables on which we have evidence).

Sometimes a node is not observed, but we have some distribution over its possible values; we have likelihood evidence. This can be introduced as follows

```
soft_evidence = cell(1,N);
soft_evidence{Observed node}=[p_1,..,p_n] ;
```

where soft evidence{i} is either [] (if node i has no soft evidence) or is a vector representing the probability distribution over i's possible values. We can now add the evidence (both hard and soft) to the chosen engine.

```
[engine,loglik]=enter_evidence(engine,evidence,'soft', soft_evidence);
```

In the case of the jtree engine, enter_evidence implements a two-pass message-passing scheme. The first return argument contains the modified engine, which incorporates the evidence. The second return argument contains the log-likelihood of the evidence. We can now obtain the marginals on a variable of interest via

```
marg = marginal_nodes(engine, Node of Interest);
marg.T
```

In the case we want a joint distribution of the nodes of interest $Noi_1, .., Noi_n$ via

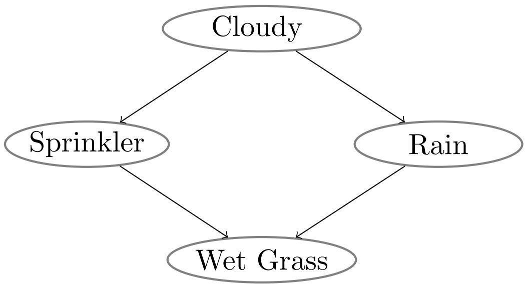
```
marg = marginal_nodes(engine, [Noi_1, Noi_2,...,Noi_n]);
marg.T
```

Computation time

Before we start with an example which will make all of the above even more clear we need to have a word about computational costs of obtaining results. As networks and the state spaces of the variables in them grow compiling a junction tree and marginalizing its cliques can become computationally heavy. Computation time is determined by the sizes of the cliques in the junction tree, thus as the network grows it becomes increasingly important to find an ordering that minimizes the sum of the clique sizes. Unfortunately finding this ordering is NP-hard. We will not to into more detail here but refer the reader to chapter 10 of Koller and Friedman [8].

3.1.2 An example using the BNT in matlab

Example 3.5. Suppose we have the following network describing the process of getting your grass wet.



Each node in this network has a state space of size 2, which corresponds to $(T, F) = ("true", "false")$ and in which we have the following conditional independency tables.

$P(C = T)$	$P(C = F)$
0.5	0.5

C	$P(R = T)$	$P(R = F)$
T	0.8	0.2
F	0.2	0.8

C	$P(S = T)$	$P(S = F)$
T	0.5	0.5
F	0.9	0.1

S	R	$P(Wg = T)$	$P(Wg = F)$
T	T	1	0
F	T	0.1	0.9
T	F	0.1	0.9
F	F	0.01	0.99

The first thing we have to specify is the ordering of the variables in our network such that this ordering respects definition 3.1. In this particular network there are two such orderings $O = (C, S, R, Wg)$ and $O = (C, R, S, Wg)$. We choose the first one. As all nodes are discrete with state space of size 2 we can now code the network by the following code.

```
n=4; C=1; S=2; R=3; Wg=4;      %define number of nodes and node ordering
ns=ones(n);                    %define state space size vector
ns([C,S,R,Wg])=2;              %all state spaces have size 2
dag=zeros(n);                  %define adjacency matrix
dag(C, [S,R])=1;               %edges from C to S,R
dag(S, Wg)=1;                  %edge from S to Wg
dag(R, Wg)=1;                  %edge from R to Wg
bnet=mk_bnet(dag,ns);          %make the network structure
```

The next thing to do is to specify the conditional probability tables on the variables in the network. For all nodes without parents this means providing the prior distribution on them. Now the tables are already given above and as we discussed before, we can enter these in the BNT as vectors.

```
bnet.CPD{C} = tabular_CPD(bnet, C, [0.5 0.5]);
bnet.CPD{R} = tabular_CPD(bnet, R, [0.8 0.2 0.2 0.8]);
bnet.CPD{S} = tabular_CPD(bnet, S, [0.5 0.9 0.5 0.1]);
bnet.CPD{Wg} = tabular_CPD(bnet, Wg, [1 0.1 0.1 0.01 0 0.9 0.9 0.99]);
```

Now that we have added the CPTs to our network, we are ready to add observed or likelihood evidence and start calculations. Suppose we observed that it is raining. We can add this as hard evidence into our network.

```
evidence = cell(1,n);          %define a vector containing evidence
evidence{R}=1 ;                %ad the observation Rain = "true"
```

Next suppose we know something about the likelihood of Wet grass for instance that it is much more likely that the grass is wet no matter what:

```
soft_evidence{Wg}=[0.1,0.9] ;  %ad soft evidence on Wet grass
```

As BNT allows for several inference engines we need to specify what type we want to use. After this we add the evidence.

```
engine=jtree_inf_engine(bnet);
[engine, ll] = enter_evidence(engine, evidence,'soft',soft_evidence);
```

Finally we are ready to perform calculations. The next code prints the marginal distribution for the Cloudy variable.

```
marg = marginal_nodes(engine, C)
```

The next section will feature the implementation of a much larger network into Matlab in almost the same way.

3.2 A Matlab implementation of the STR model

Setup

The top model as it is presented in chapter 2 (section 2) is not really easy to implement into Matlab. While the BNT can do things such as object orentated Bayesian networks we will choose to do calculations on individual markers and combine the results afterwards. To keep calculation times down the networks used will be conditional on several parameters such as the mixing factor (θ) and mosaicism factors. Stutter will be included in two levels as this will be of interest to calculations on the scorpion case.

To perform the calculations we will need more than one model. The reason for this is that things as stutter and mosaicism are dependent of the alleles that we observe and those differ from marker to marker. On top of that, introducing more persons to the mixture will also require a different model. Instead of discussing all models independently we use D18 as an example (as it is the most interesting from a modeling point of view). To keep things simple we will view the two person variant with mosaicism from allele 15 to allele 16 in person two (the victim). The network used for this is the following,

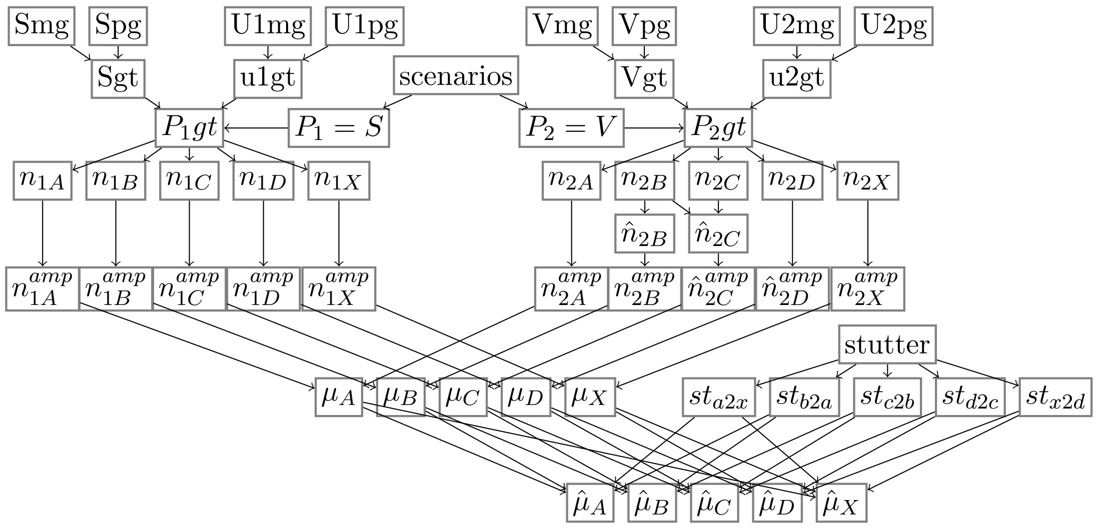


Figure 3.1: 2 person model with stutter, drop-out and a mosaicism (B to C in P2)

Structure

For the D18 marker model on two persons with mosaicism a quick count tells us we get 52 nodes and an edge set as in the graph depicted in figure 3.1. If we would have included all variables (mixing proportions etc) as nodes this number would be even higher. Although the number of nodes does play some part in computation time this is not the main reason to leave them out in this case. When it comes to parameters like the mixing factors for example we expect them to be continuous. As all our nodes are discrete it would mean we'd have to make θ discrete, leading to a large state space high up in the graph, which in turn will lead to enormous state spaces on the actual nodes where interested in, the pre-amplification amounts of DNA.

Before we can start we will need an ordering on the nodes that respects definition 3.1. As argued before such an ordering is not unique nor is it very difficult to find one. We choose the following.

```
u1mg=1;   u1pg=2;   smg=3;     spg=4;   u2mg=5;  u2pg=6;   vmg=7;
vpg=8;    sgt=9;    u1gt=10;   vgt=11;  u2gt=12; p1iss=13; p2isv=14;
p1gt=15;  p2gt=16;  target=17; n1A=18;  n1B=19;  n1C=20;   n1D=21;
n1X=22;   n2A=23;   n2B=24;    n2C=25;  n2D=26;  n2X=27;   hn1A=28;
```

```
hn1B=29;  hn1C=30;  hn1D=31;   hn1X=32; hn2A=33; hn2B=34;  hn2C=35;
hn2D=36;  hn2X=37;  muA=38;    muB=39;  muC=40;  muD=41;   muX=42;
AtX=43;   BtA=44;   CtB=45;    DtC=46;  XtD=47;  hmuA=48;  hmuB=49;
hmuC=50;  hmuD=51;  hmuX=52;
```

With an ordering on the nodes in place and an edge set defined by the graph on the previous page (for implementation please see the attached code, we can focus on their state spaces and defining the CPTs to connect them.

State spaces and CPTs

With regard to the state spaces we will work a little differently than in the example 3.5. Since state spaces are not at all obvious in our case we will discuss them on each level of the graph. At the same time we will give the CPTs on the variables. We'll work top to bottom as nodes high up influence state spaces on the nodes descending from them. There will be two type of state spaces we're using. High up we use state spaces that are as small as possible with the disadvantage that the conditional probability tables (which will be discussed in the next section) become more difficult. Once we're getting closer to the leaf nodes we'll choose larger than necessary state spaces to deal with influences from parameters on the state space and the difficult CPTs we would need otherwise.

The conditional dependency tables for all dependent nodes can also be divided into two subcategories. One category deals with what we'll call logical updates and the other deals with probabilistic updates. The states of nodes with logical updates are deterministic given the states of their parents. On the other hand the states of nodes with probabilistic updates depend on their parental nodes in some probabilistic way. We won't go into the process of defining the logical updates here, we will just state them as a matrix and refer to the attached code for precise definitions. The only probabilistic tables on nodes in our graph are involved in updating the allele number counts after drop-out. As these tables are small we'll list them below in full detail.

Maternal and paternal genes

At the highest level of the graph the maternal and paternal genes for each person are specified. Their states depend on the alleles observed. Since there are 4 observed alleles, we get 5 possible observable states (four for each allele and one state of all unobserved alleles.), combine these with a state representing an silent allele and we get 6 states. In general we'll choose the state space ordered starting with the lowest repeat number and ending with the silent allele.

$$for\ \ i \in \{1,..,8\}: \quad \begin{cases} \Omega_i = (A, B, C, D, X, S) \\ ns(i) = |\Omega_i| = 6 \\ CPT_i = prior \end{cases}$$

Genotypes

Next are the genotypes of the possible contributers (Suspect, Victim and two Unknown persons) and the actual contributers to the mixture (Person1 and Person2). Their state space obviously depends on their maternal and paternal genes but since the genotype is unordered we get just 21 states (instead of $6 \times 6 = 25$). We'll actually just use 20 since two silent alleles won't be an observable state.

$$for\ \ i \in \{9,..,12\}: \quad \begin{cases} \Omega_i = (AA, AB, AC, AD, AX, BB, BC, BD, BX, CC, CD, CX, \\ \qquad\qquad DD, DX, XX, AS, BS, CS, DS, XS) \\ ns(i) = |\Omega_i| = 20 \\ CPT_i = LU_1 \end{cases}$$

The other genotype nodes are the ones of the persons actually in the mixture. These are denoted P_1 and P_2 in the network. Their state space is the same as the other genotype nodes but the CPTs on them depend on the other genotype nodes and the targets $P1 = S$ and $P2 = V$, which we will discuss next.

$$for\ \ i \in \{15,16\}: \quad \begin{cases} \Omega_i = (AA, AB, AC, AD, AX, BB, BC, BD, BX, CC, CD, CX, \\ \qquad\qquad DD, DX, XX, AS, BS, CS, DS, XS) \\ ns(i) = |\Omega_i| = 20 \\ CPT_i = LU_2 \end{cases}$$

Target

On the same level, we have the variable of interest namely, the possible scenarios. This has in the case of a two person mixture four states. Either it is a mixture of the Suspect and the Victim, Suspect and unknown, victim and unknown or two unknowns. As we want to specify that the victim is indeed in the mixture we get two more boolean variables between the Scenarios and the persons contributing,

$$for\ \ i \in \{13, 14\} : \quad \begin{cases} \Omega_i = (true, false) \\ ns(i) = |\Omega_i| = 2 \\ CPT_i = prior \end{cases}$$

$$for\ \ i = \{15\} : \quad \begin{cases} \Omega_i = \Omega_{13} \times \Omega_{14} = ("S+V", "S+U", "V+U", "U+U") \\ ns(i) = |\Omega_i| = 4 \\ CPT_i = [1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1] \end{cases}$$

Counting variables

One level down in the graph we get the counting variables ($n_{i\alpha}$ and $n_{i\alpha}^{amp}$) which are necessary to obtain the pre-amplification amounts of DNA later on. On top of that the amplified counting variables incorporate the drop-out.

$$for\ \ i = \{18, .., 27\} : \begin{cases} \Omega_i = (0, 1, 2) \\ ns(i) = |\Omega_i| = 3 \\ CPT_i = Count \end{cases}$$

After the pre-drop-out alleles are counted the drop out will be modeled. Drop out depends on both θ, a parameter ψ and the pre-drop out alleles. The parameters are dealt with outside the network, so the CPT we are getting depend only on the $n_{i\alpha}$'s. We get the following 3×3 table with probabilities:

	$n_{i\alpha}^{amp} = 0$	$n_{i\alpha}^{amp} = 1$	$n_{i\alpha}^{amp} = 2$
$n_{i\alpha}$=0	0	0	0
$n_{i\alpha}$=1	$\exp(-\psi\theta_i)$	$1 - \exp(-\psi\theta_i)$	0
$n_{i\alpha}$=2	$\exp(-2\psi\theta_i)$	$2(1 - \exp(-\psi\theta_i)) \exp(-\psi\theta_i)$	$(1 - \exp(-\psi\theta_i))^2$

This gives the following for the pre drop out counting variables

$$for\ \ i = \{28, .., 37\} : \begin{cases} \Omega_i = (0, 1, 2) \\ ns(i) = |\Omega_i| = 3 \\ CPT_i = Drop\ out\ probability\ table \end{cases}$$

Pre amplification pre stutter amounts of DNA

Up to now the state spaces of all variables were very logical and determined by the biological equivalent of the variable in question.But after these variables the size of the state space becomes more difficult. If we look at the way we defined μ_α we see it depends on the θ_i's, or in the two person case just θ,

$$\mu_\alpha = \frac{1}{2} \sum_i \theta_i n_{i\alpha}^{amp}.$$

Now for a two person mixture with $n_{i\alpha}^{amp} \in \{0, 1, 2\}$ and θ the mixing factor of person one (and thus, $1 - \theta$ for person 2) we have

$$\mu_\alpha \in \{0, \theta/2, \theta, (1 - \theta)/2, 0.5, (1 + \theta)/2, (2 - 2\theta)/2, (2 - \theta)/2, 1\}.$$

Thus we conclude that μ_α has $3 \times 3 = 9$ states. This is true in most cases but if we take $\theta = 0.5$ the above translates to,

$$\mu_\alpha \in \{0, 0.25, 0.5, 0.25, 0.5, 0.75, 0.5, 0.75, 1\} = \{0, 0.25, 0.5, 0.75, 1\}$$

So for $\theta = 0.5$ only 5 states would do. This dependency on the state space size is something that we encounter in other variables as well, partly because the size of the states on children of the μ's depend on the state space of μ_α. On the other side, a similar thing is happening due to the stutters. We choose to give μ_α nine states and figure out what to do with them later.

$$for\ \ i = \{38, .., 42\} : \begin{cases} \Omega_i = (0, \theta/2, \theta, (1 - \theta)/2, 0.5, (1 + \theta)/2, (2 - 2\theta)/2, (2 - \theta)/2, 1) \\ ns(i) = |\Omega_i| = 9 \\ CPT_i = eye(9); \end{cases}$$

Stutter

Since the total amount of DNA does not change due to a stutter, having a stutter results in two effects that need to be reflected in the state spaces. There will be a negative effect at the allele where the stutter originates and a positive effect at the allele it stutters to. The size of these effects will be the same in both ways, but there are two possible levels of stutter, giving the stutter state space,

$$for\ \ i = \{43, .., 47\} : \begin{cases} \Omega_i = (\text{"}none\text{"},\ \text{"}low\text{"},\ \text{"}high\text{"}) \\ ns(i) = |\Omega_i| = 3 \\ CPT_i = Prior \end{cases}$$

It is good to point out again that stutter only happens from an allele with repeat number λ to an allele with repeat number $\lambda - 1$. This means that incorporating stutter in any model is always a custom job since the stutter will depend on the alleles that are observed. In the case of marker D18 all observed alleles differed only one repeat number and thus stutter could occur between any two neighboring alleles. We take X as a neighbor of both A and D but we will correct the chance of a stutter from $X\ To\ D$ to the fact that x has a possible $n -$ #observed neighbors.

Pre amplification post stutter amounts

For the state space of $\hat{\mu}_\alpha$ there are now 9 scenarios on the change of pre amplification amounts before the PCR reaction. This leads to the following table:

Stutter for α		$\alpha_{+1}2\alpha$		
		non	low	high
$\alpha 2\alpha_{-1}$	non	μ_α	$\mu_\alpha + st_1\mu_{\alpha+1}$	$\mu_\alpha + st_2\mu_{\alpha+1}$
	low	$(1 - st_1)\mu_\alpha$	$(1 - st_1)\mu_\alpha + st_1\mu_{\alpha+1}$	$(1 - st_1)\mu_\alpha + st_2\mu_{\alpha+1}$
	high	$(1 - st_2)\mu_\alpha$	$(1 - st_2)\mu_\alpha + st_1\mu_{\alpha+1}$	$(1 - st_2)\mu_\alpha + st_2\mu_{\alpha+1}$

Here st_1 is the amount of stutter in case of a low stutter, and st_2 the amount of from a high stutter. In the actual calculations we can vary these, but on suggestion of Ate Kloosterman during a discussion of the scorpion case a low stutter is 4% of the original peak height and high stutter is 14%.

The total size of the state space for a $\hat{\mu}_\alpha$ now depends on the state spaces of μ_α and $\mu_{\alpha+1}$. We recall that these depended on both θ and on the question whether there was a mosaicism. The whole thing is becoming increasingly difficult to figure out which is going to give problems at the point where we need to specify the CPT's to update these state spaces.

Here again we'll make the trade off between a larger state space and easy CPT's over a state space depending on stutter and mixing factors. This will mean that the state space for $\hat{\mu}_\alpha$ is the product space of all variables it depends on, these are in order μ_α , $\mu_{\alpha+1}$, $\alpha\ To\ \alpha_{-1}$ and $\alpha_{+1}\ To\ \alpha$, giving

$$\begin{aligned} ns(\hat{\mu}_\alpha) &= ns(\mu_\alpha) \times ns(\mu_{\alpha+1}) \times ns(\alpha 2 \alpha_{-1}) \times ns(\alpha_{+1} 2 \alpha) \\ &= 9 \times 9 \times 3 \times 3 = 729 \end{aligned}$$

We recall that Matlab layers multidimensional arrays out in memory such that the first index toggles fastest. The state space depends on product space of $\Omega_{\mu_\alpha} \times \Omega\mu_{\alpha+1} \times \Omega_{\alpha\ To\ \alpha_{-1}} \times \Omega_{\alpha_{+1}\ To\ \alpha}$. The next table lists the state space on the left and the events each state space depends on on the right.

	test							
$\Omega_{\hat{\mu}_\alpha}(1)$	μ_α	$=0$	$\mu_{\alpha+1}$	$=0$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(2)$	μ_α	$=\theta/2$	$\mu_{\alpha+1}$	$=0$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(...)$	μ_α	$=...$	$\mu_{\alpha+1}$	$=0$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(10)$	μ_α	$=0$	$\mu_{\alpha+1}$	$=\theta/2$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(11)$	μ_α	$=\theta/2$	$\mu_{\alpha+1}$	$=\theta/2$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(...)$	μ_α	$=...$	$\mu_{\alpha+1}$	$=...$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(82)$	μ_α	$=0$	$\mu_{\alpha+1}$	$=0$	$\alpha\ To\ \alpha_{-1}$	$=$ "*low*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(83)$	μ_α	$=\theta/2$	$\mu_{\alpha+1}$	$=0$	$\alpha\ To\ \alpha_{-1}$	$=$ "*low*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(...)$	μ_α	$=...$	$\mu_{\alpha+1}$	$=...$	$\alpha\ To\ \alpha_{-1}$	$=...$	$\alpha_{+1}\ To\ \alpha$	$=$ "*none*"
$\Omega_{\hat{\mu}_\alpha}(244)$	μ_α	$=0$	$\mu_{\alpha+1}$	$=0$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*low*"
$\Omega_{\hat{\mu}_\alpha}(245)$	μ_α	$=\theta/2$	$\mu_{\alpha+1}$	$=0$	$\alpha\ To\ \alpha_{-1}$	$=$ "*none*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*low*"
$\Omega_{\hat{\mu}_\alpha}(...)$	μ_α	$=...$	$\mu_{\alpha+1}$	$=...$	$\alpha\ To\ \alpha_{-1}$	$=...$	$\alpha_{+1}\ To\ \alpha$	$=...$
$\Omega_{\hat{\mu}_\alpha}(729)$	μ_α	$=1$	$\mu_{\alpha+1}$	$=1$	$\alpha\ To\ \alpha_{-1}$	$=$ "*high*"	$\alpha_{+1}\ To\ \alpha$	$=$ "*high*"

Now we need to translate this table to a vector of state spaces. In the first 9 states there is no stutter in or out and thus the state spaces will be a copy of μ_α. Actually in the next 72 states there is no in or out stutter as well so we start the state space with 9 copies of μ_α. At the point where there is an influence from $\mu_{\alpha+1}$ we no longer get 9 copies of the same vector but each of the 9 copies now is summed with some "incoming" allele weight. The resulting vector, broken up in 9 pieces will give the following table.

For j in	$\Omega_{\hat{\mu}_\alpha}(j)$		
1:81	$\Omega_{\mu_\alpha}(j \mod (9))$	$+$	0
82:162	$(1-st1)\times\Omega_{\mu_\alpha}(j \mod (9))$	$+$	0
163:243	$(1-st2)\times\Omega_{\mu_\alpha}(j \mod (9))$	$+$	0
242:324	$\Omega_{\mu_\alpha}(j \mod (9))$	$+$	$st1\times\Omega_{\mu_{\alpha+1}}(\lceil(j-243)/9\rceil)$
323:405	$(1-st1)\times\Omega_{\mu_\alpha}(j \mod (9))$	$+$	$st1\times\Omega_{\mu_{\alpha+1}}(\lceil(j-324)/9\rceil)$
404:486	$(1-st2)\times\Omega_{\mu_\alpha}(j \mod (9))$	$+$	$st1\times\Omega_{\mu_{\alpha+1}}(\lceil(j-405)/9\rceil)$
485:567	$\Omega_{\mu_\alpha}(j \mod (9))$	$+$	$st2\times\Omega_{\mu_{\alpha+1}}(\lceil(j-486)/9\rceil)$
566:648	$(1-st1)\times\Omega_{\mu_\alpha}(j \mod (9))$	$+$	$st2\times\Omega_{\mu_{\alpha+1}}(\lceil(j-567)/9\rceil)$
649:729	$(1-st2)\times\Omega_{\mu_\alpha}(j \mod (9))$	$+$	$st2\times\Omega_{\mu_{\alpha+1}}(\lceil(j-648)/9\rceil)$

Now we've defined all state spaces we can insert them into the BNT. We can state the final layer of nodes in our network.

$$for\ \ i=\{47,..,52\}:\begin{cases}\Omega_i(j)=\Omega_{\hat{\mu}_\alpha}(j) \quad j\in(1,..,729)\\ ns(i)=|\Omega_i|=729\\ CPT_i=Id_{729};\end{cases}$$

Parameters and priors

With the entire network encoded we can start to look at introducing evidence and producing results. Without introduction of evidence the priors we chose on all ancestral nodes and the conditional parameters (for example the mixing factors) will determine the final results. This is an important fact to realize!

The priors used in the code are chosen on basis of expert opinions if they are of biological nature or taken to be uniform in the case of the target nodes, because in the end we are interested in the likelihoods over them. The parameter conditioning is again done on expert opinions.

Adaptation of the model for different markers

As noted at the beginning of the model description for the SRT marker model on 4 consecutive observed alleles adaptations are needed if we observe a different set of alleles in the peak profile. Without too much depth we list the most obvious adaptations.

Non consecutive repeat numbers; In the D18 marker we observed alleles with repeat numbers $14, 15, 16$ and 17. This meant that stutter was easily dealt with. In situations where allele repeat numbers differ more then one from each other we need to rethink what this means for stutter. This might require some more information from experts on the way stutter is passed on during the PCR process.

Less observed alleles; If we observe less than 4 alleles, say 3, we can do two things. We might change our entire network accordingly, dropping the number of n, μ, etc variables from 5 to 4 (three observed alleles + the X factor). Or we could just set the likelihood evidence over one of the alleles to 0 for all states of non observed alleles. Obviously the latter is easier, but computation times will be better on the smaller model.

Concluding remarks

My thesis started with an overview on graphical models in which I've tried to present the theory from an applied point of view. Once we obtained a directed graphical model, the independencies it represented led us to an undirected equivalent, which in turn led to a junction tree respecting the independencies in the original DAG. Using the junction tree for exact inference involved an application of the junction tree algorithm. The proof of this algorithm, as presented, is an improvement over the technical and often algebraic proofs found in literature.

In the second chapter we saw an introduction to forensic DNA analysis. The model by Cowell et. al. that uses the peak areas in the peak profile to determine the contributers to a DNA mixture was presented, and we discussed some of its shortcomings. Some of the issues raised were addressed in the adapted model, but there remains a lot of work to correctly incorporate all anomalies and do more research on the effects of mosaicism.

The use of the BNT to do a computer implementation of the model from chapter 3 was very successful. While we are not (yet?) able to do calculations on entire profiles, the calculations on single markers using a computer package that does not cost a fortune is an improvement for anyone interested in doing these calculations. The code of one of the models used in chapter 4 is attached, anyone interested in the actual .m files for both models can get them by sending me an e-mail (j.wamelen@gmail.com).

Application of the matlab model on the D18 marker data in one of the DNA traces in the scorpion case produced results that indicate a likelihood in favour of the suspect being in the mixture. Personally I think that application of mathematical methods to quantify DNA mixture analysis will become a standard practice in the future, but it will require more work on the model and possibly a crash course in statistics for judges.

Bibliography

[1] Codis. *http://www.fbi.gov/about-us/lab/codis/codis.*

[2] S.W. Thein A.J. Jeffreys, V Wilson. Hypervariable 'minisatellite' regions in human dna. 314:67–73, 1984.

[3] Applied Biosystems. Ampflstr sgm plus pcr amplification kit user manual. *http://www3.appliedbiosystems.com/sup/URLRedirect/index.htm*, 2006.

[4] John M. Butler. *Forensic DNA Typing, biology, technology and genetics of STR markers*. Graduate Texts in Mathematics. Elsevier Academic Press, 2005.

[5] R. G. Cowell. Validation of an str peak area model. *Forensic Science International: Genetics*, 3:193 – 199, 2009.

[6] Robert Cowell. Scorpion case mixes. June 2010.

[7] L.A. Foreman and I.W. Evett. Statistical analyses to support forensic interpretation for a new 10-locus str profiling system. *International Journal of Legal Medicine*, 114:131183, 2003.

[8] Daphne Koller & Nir Friedman. *Probabilistic Graphical Models, principles and techniques*. Adaptive computation and machine learning series. The MIT Press, 2009.

[9] Therese Graversen. Parameter estimation for dna mixtures. June 2010.

[10] Claudia Hilmmelreich. Germany's phantom serial killer: Dna blunder. *Time magazine*, March 2009.

[11] G. Wilcott I.W. Evett, C. Bu?ery and D. Stoney. A guide to interpreting single locus proles of dna mixtures in forensic cases. *Journal of the Forensic Science Society*, 31:41–47, 1991.

[12] A.P. Dawid J. Mortera and S.L. Lauritzen. Probabilistic expert systems for dna mixture proling. *Theoretical Population Biology*, 63:191205, 2003.

[13] S.J. Walsh J. S. Buckleton, C.M. Triggs. *Forensic DNA Evidence Interpretation.* Applications of Mathematics. CRC Press, 2001.

[14] Stephan Lauritzen. *Graphical Models.* Oxford statistical science series. Oxford University Press, 1996.

[15] Kevin Murphy. Computing science and statistics. 33, 2001.

[16] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* The Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1991.

[17] J. Mortera R. G. Cowell and S.L. Lauritzen. Maies: A tool for dna mixture analysis. *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (eds: Rina Dechter and Thomas Richardson)*, pages 90 – 97, 2006.

[18] J. Mortera R. G. Cowell and S.L. Lauritzen. A gamma model for dna mixture analyses. *Bayesian Analysis*, 2:333348, 2007.

[19] J. Mortera R. G. Cowell and S.L. Lauritzen. Identification and separation of dna mixtures using peak area information. *Forensic Science International*, pages 28–34, 2007.

Code attachement

```
%clear all
clear all
%Ad BNtoolbox
addpath(genpathKPM(pwd))

%--Made by Jasper van Wamelen------------------------------------------
%---------------------------------------------------------------------
%--Model Discription--------------------------------------------------
%---------------------------------------------------------------------
%Gamma model on a single marker with 4 observed alleles A,B,C and D from
%two persons where we have a mos from B to C in p2. Theta1 is the mix-
%factor between the persons, theta2 is the mos factor.
%---------------------------------------------------------------------

%--evidence-----------------------------------------------------------

%marginal paternal and maternal alleles
%              A      B      C      D      X     s
gen_marg=[0.1374 0.1589 0.1391 0.1258 0.4387 0.001];


%measured allele weights
%       A     B    C    D    X
%ev=[11918  9371  2595 1486  638]; %AHH352_2_1a
%ev=[11918  9371  2595 1486    0]; %AHH352_2_1b
%ev=[11918 11966     0 1486    0]; %AHH352_2_1c
```

```
%ev=[24027 21791  3973 3242 1796]; %AHH352_2_2a
%ev=[24027 21791  3973 3242    0]; %AHH352_2_2b
%ev=[24027 25764     0 3242    0]; %AHH352_2_2c

%ev=[27919 23354  4018 3785 1950]; %AHH352_2_2a
%ev=[27919 23354  4018 3785    0]; %AHH352_2_2b
%ev=[27919 27372     0 3785    0]; %AHH352_2_2c

%ev=[63864 54516 10586 8513 4384]; %AHH352_2_2_sum_a
%ev=[63864 54516 10586 8513    0]; %AHH352_2_2_sum_b
 ev=[63864 65102     0 8513    0]; %AHH352_2_2_sum_c

%factors
theta =0.10;   %mix factor
st1   =0.14;   %apriori low stutter chance
st2   =0.07;   %apriori high stutter chance
stf1  =0.16;   %low stutter factor
stf2  =0.04;   %high stutter factor
phi   =24;     %drop-out factor
sigma =0.005;   %gamma model scaling factor

%specify allele numbers
A=14; B=15; C=16; D=17;

% Genotype translation table
% 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 21
%AA AB AC AD AX BB BC BD BX CC CD CX DD DX XX 00

ev_vgt=2;         %vgt=(A,B)=(14,15)
ev_sgt=11;        %sgt=(D,D)=(17,17)
ev_p2isv=1;       %p2gt=vgt

%--Setup----------------------------------------------------------------

%define nodes
u1mg=1;   u1pg=2;   smg=3;     spg=4;   u2mg=5;  u2pg=6;   vmg=7;
```

```
vpg=8;    sgt=9;    u1gt=10;    vgt=11;  u2gt=12; p1iss=13; p2isv=14;
p1gt=15;  p2gt=16;  target=17; n1A=18;  n1B=19;  n1C=20;   n1D=21;
n1X=22;   n2A=23;   n2B=24;    n2C=25;  n2D=26;  n2X=27;   hn1A=28;
hn1B=29;  hn1C=30;  hn1D=31;   hn1X=32; hn2A=33; hn2B=34;  hn2C=35;
hn2D=36;  hn2X=37;  muA=38;    muB=39;  muC=40;  muD=41;   muX=42;
AtX=43;   BtA=44;   CtB=45;    DtC=46;  XtD=47;  hmuA=48;  hmuB=49;
hmuC=50;  hmuD=51;  hmuX=52;

%count nodes
n=52;

%make node levM1els for discrete nodes
ns=ones(1,n);
ns([u1mg,u1pg,smg,spg,u2mg,u2pg,vmg,vpg])=5;
ns([u1gt,sgt,u2gt,vgt,p1gt,p2gt])=15;
ns([p1iss,p2isv])=2;
ns([AtX,BtA,CtB,DtC,XtD])=3;
ns([target])=4;
ns([n1A,n1B,n1C,n1D,n1X,n2A,n2B,n2C,n2D,n2X])=3;
ns([hn1A,hn1B,hn1C,hn1D,hn1X,hn2A,hn2B,hn2C,hn2D,hn2X])=3;
ns([muA,muB,muD,muC,muX])=9;
ns([hmuA,hmuB,hmuD,hmuC,hmuX])=729;

%specify all that al nodes are discrete
dnodes=(1:n);

%define the adjecency matrix
dag=zeros(n);
dag(u1mg,u1gt)=1;  dag(u1pg,u1gt)=1;  dag(smg,sgt)=1;
dag(spg,sgt)=1;    dag(u2mg,u2gt)=1;  dag(u2pg,u2gt)=1;
dag(vmg,vgt)=1;    dag(vpg,vgt)=1;    dag(sgt,p1gt)=1;
dag(u1gt,p1gt)=1;  dag(u2gt,p2gt)=1;  dag(vgt,p2gt)=1;

dag(p1iss,[p1gt,target])=1;        dag(p2isv,[p2gt,target])=1;
dag(p1gt,[n1A,n1B,n1C,n1D,n1X])=1; dag(p2gt,[n2A,n2B,n2C,n2D,n2X])=1;
dag(n1A,hn1A)=1;   dag(n1B,hn1B)=1;    dag(n1C,hn1C)=1;
dag(n1D,hn1D)=1;   dag(n1X,hn1X)=1;    dag(n2A,hn2A)=1;
dag(n2B,hn2B)=1;   dag(n2C,hn2C)=1;    dag(n2D,hn2D)=1;
dag(n2X,hn2X)=1;   dag(hn1A,muA)=1;    dag(hn1B, muB)=1;
dag(hn1C,muC)=1;   dag(hn1D,muD)=1;    dag(hn1X,muX)=1;
dag(hn2A,muA)=1;   dag(hn2B,muB)=1;    dag(hn2C,muC)=1;
dag(hn2D,muD)=1;   dag(hn2X,muX)=1;    dag(muA,hmuA)=1;
dag(muB,hmuB)=1;   dag(muC,hmuC)=1;    dag(muD,hmuD)=1;
dag(muX,hmuX)=1;   dag(muA,hmuX)=1;    dag(muB,hmuA)=1;
dag(muC,hmuB)=1;   dag(muD,hmuC)=1;    dag(muX,hmuD)=1;

dag(AtX,[hmuA,hmuX])=1;   dag(BtA,[hmuB,hmuA])=1;
dag(CtB,[hmuC,hmuB])=1;   dag(DtC,[hmuD,hmuC])=1;
dag(XtD,[hmuX,hmuD])=1;

bnet=mk_bnet(dag,ns,'discrete', dnodes);   %make the BN

%--CDP's----------------------------------------------------------------

%Incoporate maternal/ paternal inheratence
bnet.CPD{u1mg}=tabular_CPD(bnet,u1mg,'CPT',gen_marg);
bnet.CPD{u1pg}=tabular_CPD(bnet,u1pg,'CPT',gen_marg);
bnet.CPD{smg}=tabular_CPD(bnet,smg,'CPT',gen_marg);
bnet.CPD{spg}=tabular_CPD(bnet,spg,'CPT',gen_marg);
bnet.CPD{u2mg}=tabular_CPD(bnet,u2mg,'CPT',gen_marg);
bnet.CPD{u2pg}=tabular_CPD(bnet,u2pg,'CPT',gen_marg);
bnet.CPD{vmg}=tabular_CPD(bnet,vmg,'CPT',gen_marg);
bnet.CPD{vpg}=tabular_CPD(bnet,vpg,'CPT',gen_marg);

%Logic update from paternal and maternal to genotype
%Logic update from paternal and maternal to genotype
s1=
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s2=
[0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s3=
[0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s4=
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s5=
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0];
s6=
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s7=
[0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s8=
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s9=
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0];
s10=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s11=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s12=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0];
s13=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
s14=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0];
s15=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0];
s16=
[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0];
s17=
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0];
s18=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0];
s19=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0];
s20=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0];
s21=
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1];

LU=[s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17
    s18 s19 s20 s21];
bnet.CPD{sgt}=tabular_CPD(bnet,sgt,'CPT',LU);
bnet.CPD{u1gt}=tabular_CPD(bnet,u1gt,'CPT',LU);
bnet.CPD{vgt}=tabular_CPD(bnet,vgt,'CPT',LU);
bnet.CPD{u2gt}=tabular_CPD(bnet,u2gt,'CPT',LU);

bnet.CPD{sgt}=tabular_CPD(bnet,sgt,'CPT',LU);
bnet.CPD{u1gt}=tabular_CPD(bnet,u1gt,'CPT',LU);
bnet.CPD{vgt}=tabular_CPD(bnet,vgt,'CPT',LU);
bnet.CPD{u2gt}=tabular_CPD(bnet,u2gt,'CPT',LU);

%Logic update to p1 and p2 genotype
tar=[1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1];
pis=[1 1]; piv=[0.5 0.5];
bnet.CPD{p1iss}=tabular_CPD(bnet,p1iss,'CPT',pis);
bnet.CPD{p2isv}=tabular_CPD(bnet,p2isv,'CPT',piv);
bnet.CPD{target}=tabular_CPD(bnet,target,'CPT',tar);

%Update to the persons in the mixture
s=eye(21); s1=s(1,:); s2=s(2,:); s3=s(3,:); s4=s(4,:); s5=s(5,:);
s6=s(6,:); s7=s(7,:); s8=s(8,:); s9=s(9,:); s10=s(10,:);
s11=s(11,:); s12=s(12,:); s13=s(13,:); s14=s(14,:); s15=s(15,:);
s16=s(16,:); s17=s(17,:); s18=s(18,:); s19=s(19,:); s20=s(20,:);
s21=s(21,:);

T1=reshape(s1(ones(1,21),:).',[1,441]);
T2=reshape(s2(ones(1,21),:).',[1,441]);
T3=reshape(s3(ones(1,21),:).',[1,441]);
T4=reshape(s4(ones(1,21),:).',[1,441]);
T5=reshape(s5(ones(1,21),:).',[1,441]);
T6=reshape(s6(ones(1,21),:).',[1,441]);
T7=reshape(s7(ones(1,21),:).',[1,441]);
T8=reshape(s8(ones(1,21),:).',[1,441]);
```

```
T9=reshape(s9(ones(1,21),:).',[1,441]);
T10=reshape(s10(ones(1,21),:).',[1,441]);
T11=reshape(s11(ones(1,21),:).',[1,441]);
T12=reshape(s12(ones(1,21),:).',[1,441]);
T13=reshape(s13(ones(1,21),:).',[1,441]);
T14=reshape(s14(ones(1,21),:).',[1,441]);
T15=reshape(s15(ones(1,21),:).',[1,441]);
T16=reshape(s16(ones(1,21),:).',[1,441]);
T17=reshape(s17(ones(1,21),:).',[1,441]);
T18=reshape(s18(ones(1,21),:).',[1,441]);
T19=reshape(s19(ones(1,21),:).',[1,441]);
T20=reshape(s20(ones(1,21),:).',[1,441]);
T21=reshape(s21(ones(1,21),:).',[1,441]);

i=ones(1,21); o=ones(1,21)*0;
 F1=[i o o o o o o o o o o o o o o o o o o o o];
 F2=[o i o o o o o o o o o o o o o o o o o o o];
 F3=[o o i o o o o o o o o o o o o o o o o o o];
 F4=[o o o i o o o o o o o o o o o o o o o o o];
 F5=[o o o o i o o o o o o o o o o o o o o o o];
 F6=[o o o o o i o o o o o o o o o o o o o o o];
 F7=[o o o o o o i o o o o o o o o o o o o o o];
 F8=[o o o o o o o i o o o o o o o o o o o o o];
 F9=[o o o o o o o o i o o o o o o o o o o o o];
F10=[o o o o o o o o o i o o o o o o o o o o o];
F11=[o o o o o o o o o o i o o o o o o o o o o];
F12=[o o o o o o o o o o o i o o o o o o o o o];
F13=[o o o o o o o o o o o o i o o o o o o o o];
F14=[o o o o o o o o o o o o o i o o o o o o o];
F15=[o o o o o o o o o o o o o o i o o o o o o];
F16=[o o o o o o o o o o o o o o o i o o o o o];
F17=[o o o o o o o o o o o o o o o o i o o o o];
F18=[o o o o o o o o o o o o o o o o o i o o o];
F19=[o o o o o o o o o o o o o o o o o o i o o];
F20=[o o o o o o o o o o o o o o o o o o o i o];
F21=[o o o o o o o o o o o o o o o o o o o o i];
```

```
LU1=[T1 F1 T2 F2 T3 F3 T4 F4 T5 F5 T6 F6 T7 F7 T8 F8];
LU2=[T9 F9 T10 F10 T11 F11 T12 F12 T13 F13 T14 F14 T15 F15 T16 F16];
LU3=[T17 F17 T18 F18 T19 F19 T20 F20 T21 F21];

bnet.CPD{p1gt}=tabular_CPD(bnet,p1gt,'CPT',[LU1 LU2 LU3]);
bnet.CPD{p2gt}=tabular_CPD(bnet,p2gt,'CPT',[LU1 LU2 LU3]);

%Count the number of alleles in each person gt.
n14_0=[0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1];
n15_0=[1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1];
n16_0=[1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 1 1 1];
n17_0=[1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 1];
 nx_0=[1 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1];

n14_1=[0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0];
n15_1=[0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0];
n16_1=[0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 0];
n17_1=[0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0];
 nx_1=[0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0];

n14_2=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
n15_2=[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
n16_2=[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0];
n17_2=[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0];
 nx_2=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0];

countA=[nA_0 nA_1 nA_2];
countB=[nB_0 nB_1 nB_2];
countC=[nC_0 nC_1 nC_2];
countD=[nD_0 nD_1 nD_2];
countX=[nX_0 nX_1 nX_2];

bnet.CPD{n1A}=tabular_CPD(bnet,n1A,'CPT',countA);
bnet.CPD{n1B}=tabular_CPD(bnet,n1B,'CPT',countB);
bnet.CPD{n1C}=tabular_CPD(bnet,n1C,'CPT',countC);
bnet.CPD{n1D}=tabular_CPD(bnet,n1D,'CPT',countD);
bnet.CPD{n1X}=tabular_CPD(bnet,n1X,'CPT',countX);
bnet.CPD{n2A}=tabular_CPD(bnet,n2A,'CPT',countA);
bnet.CPD{n2B}=tabular_CPD(bnet,n2B,'CPT',countB);
bnet.CPD{n2C}=tabular_CPD(bnet,n2C,'CPT',countC);
bnet.CPD{n2D}=tabular_CPD(bnet,n2D,'CPT',countD);
bnet.CPD{n2X}=tabular_CPD(bnet,n2X,'CPT',countX);

%drop-out factors
p=phi;   t1=theta1;   t2=1-t1;

dropP1=[1 exp(-p*t1) exp(-2*p*t1) 0 (1-exp(-p*t1))
                2*(1-exp(-p*t1))*exp(-p*t1) 0 0 (1-exp(-p*t1))^2 ];
dropP2=[1 exp(-p*t2) exp(-2*p*t2) 0 (1-exp(-p*t2))
                2*(1-exp(-p*t2))*exp(-p*t2) 0 0 (1-exp(-p*t2))^2 ];

bnet.CPD{hn1A}=tabular_CPD(bnet,hn1A,'CPT',dropP1);
bnet.CPD{hn1B}=tabular_CPD(bnet,hn1B,'CPT',dropP1);
bnet.CPD{hn1C}=tabular_CPD(bnet,hn1C,'CPT',dropP1);
bnet.CPD{hn1D}=tabular_CPD(bnet,hn1D,'CPT',dropP1);
bnet.CPD{hn1X}=tabular_CPD(bnet,hn1X,'CPT',dropP1);
bnet.CPD{hn2A}=tabular_CPD(bnet,hn2A,'CPT',dropP2);
bnet.CPD{hn2B}=tabular_CPD(bnet,hn2B,'CPT',dropP2);
bnet.CPD{hn2C}=tabular_CPD(bnet,hn2C,'CPT',dropP2);
bnet.CPD{hn2D}=tabular_CPD(bnet,hn2D,'CPT',dropP2);
bnet.CPD{hn2X}=tabular_CPD(bnet,hn2X,'CPT',dropP2);

%make mu's
mus=reshape(eye(9), [1,81]);

bnet.CPD{muA}=tabular_CPD(bnet,muA,'CPT',mus);
bnet.CPD{muB}=tabular_CPD(bnet,muB,'CPT',mus);
bnet.CPD{muC}=tabular_CPD(bnet,muC,'CPT',mus);
bnet.CPD{muD}=tabular_CPD(bnet,muD,'CPT',mus);
bnet.CPD{muX}=tabular_CPD(bnet,muX,'CPT',mus);
```

```
%stutter correction
bnet.CPD{AtX}=tabular_CPD(bnet,AtX,'CPT',[st,st2,1-st-st2]);
bnet.CPD{BtA}=tabular_CPD(bnet,BtA,'CPT',[st,st2,1-st-st2]);
bnet.CPD{CtB}=tabular_CPD(bnet,CtB,'CPT',[st,st2,1-st-st2]);
bnet.CPD{DtC}=tabular_CPD(bnet,DtC,'CPT',[st,st2,1-st-st2]);
bnet.CPD{XtD}=tabular_CPD(bnet,XtD,'CPT',[st/20,st2/20,1-st/20-st2/20]);

hmus=reshape(eye(729), [1,81*9*9*81]);
bnet.CPD{hmuA}=tabular_CPD(bnet,hmuA,'CPT',hmus);
bnet.CPD{hmuB}=tabular_CPD(bnet,hmuB,'CPT',hmus);
bnet.CPD{hmuC}=tabular_CPD(bnet,hmuC,'CPT',hmus);
bnet.CPD{hmuD}=tabular_CPD(bnet,hmuD,'CPT',hmus);
bnet.CPD{hmuX}=tabular_CPD(bnet,hmuX,'CPT',hmus);

%--Make the modified Gamma model----------------------------------------

t1=theta1;  r=[0 1 2];

r1=reshape(r(ones(1,3),:),[1,9]);
r=reshape(r(ones(1,3),:).',[1,9]);

mu=(t1*r+(1-t1)*r1)/2;
minmu=mu-stf1*mu;
min2mu=mu-stf2*mu;

for(j=1:9)
hulpmu1(j,:)=minmu+stf1*mu(j);
hulpmu2(j,:)=min2mu+stf1*mu(j);
hulpmu3(j,:)=mu+stf1*mu(j);
hulpmu4(j,:)=minmu+stf2*mu(j);
hulpmu5(j,:)=min2mu+stf2*mu(j);
hulpmu6(j,:)=mu+stf2*mu(j);
end

hmu1=reshape(hulpmu1.',[1,81]);
hmu2=reshape(hulpmu2.',[1,81]);
```

```
hmu3=reshape(hulpmu3.',[1,81]);
hmu4=reshape(hulpmu4.',[1,81]);
hmu5=reshape(hulpmu5.',[1,81]);
hmu6=reshape(hulpmu6.',[1,81]);
hmu7=reshape(minmu(ones(1,9),:).',[1,81]);
hmu8=reshape(min2mu(ones(1,9),:).',[1,81]);
hmu9=reshape(mu(ones(1,9),:).',[1,81]);

hmu=[hmu1 hmu3 hmu2 hmu4 hmu5 hmu6 hmu7 hmu8 hmu9];
hmu22=[hmu1 hmu4 hmu7 hmu2 hmu5 hmu8 hmu3 hmu6 hmu9];
hmua=hmu; hmub=hmu; hmuc=hmu; hmud=hmu; hmux=hmu22;

ev2(1)=ev(1)*A; ev2(2)=ev(2)*B; ev2(3)=ev(3)*C;
ev2(4)=ev(4)*D; ev2(5)=ev(5)*((A+B+C+D)/4);

norm=sum(ev2);  ev2=ev2/norm;   sig=(1/sigma -1);

for (i=1:729)
    sevA(i)=log(ev2(1)^(hmua(i)*sig))-log(gamma(hmua(i)*sig));
    sevB(i)=log(ev2(2)^(hmub(i)*sig))-log(gamma(hmub(i)*sig));
    sevC(i)=log(ev2(3)^(hmuc(i)*sig))-log(gamma(hmuc(i)*sig));
    sevD(i)=log(ev2(4)^(hmud(i)*sig))-log(gamma(hmud(i)*sig));
    sevX(i)=log(ev2(5)^(hmux(i)*sig))-log(gamma(hmux(i)*sig));
    if sevA(i)==-inf sevA(i)=0;
    if sevB(i)==-inf sevB(i)=0;
    if sevC(i)==-inf sevC(i)=0;
    if sevD(i)==-inf sevD(i)=0;
    if sevX(i)==-inf sevX(i)=0;
end

if(sum(sevX)==0) sevX(1)=1; end %compensate for no X
if(sum(sevC)==0) sevC(1)=1; end %compensate for no C

soft_evidence{hmuA}=[sevA];  soft_evidence{hmuB}=[sevB];
soft_evidence{hmuC}=[sevC];  soft_evidence{hmuD}=[sevD];
soft_evidence{hmuX}=[sevX];
```

```
%--Execute -----------------------------------------------------------

%introduce evM1idence
evidence = cell(1,n);  evidence{vgt}=ev_vgt ;
evidence{sgt}=ev_sgt;  evidence{p2isv}=ev_p2isv;

%jtree inference engine
engine = jtree_inf_engine(bnet);
[engine, ll] = enter_evidence(engine, evidence,'soft',soft_evidence);

%produce marginals of target node
marg = marginal_nodes(engine, target);
Target=marg.T;

%likelihood ratio v+s/v+u
LHR=marg.T(1)/marg.T(3)
```