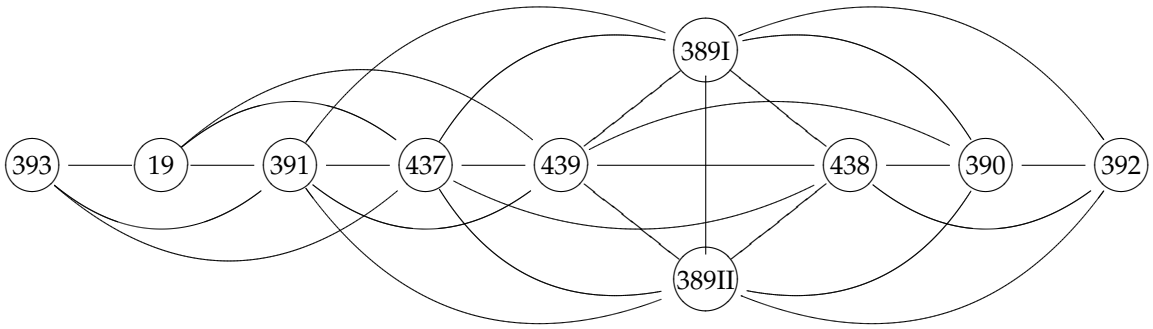
GRAFISKE MODELLER

ET MAT4-PROJEKT I GRAFISKE MODELLERS ANVENDELSE I

BESTEMMELSE AF HAPLOTYPEFREKVENS

UDARBEJDET AF MIKKEL MEYER ANDERSEN

4. JUNI 2009



VEJLEDER: POUL SVANTE ERIKSEN

GRUPPE: G4-115A

AALBORG UNIVERSITET

INSTITUT FOR MATEMATISKE FAG

AALBORG UNIVERSITET

Institut for Matematiske Fag
Fredrik Bajers Vej 7G
9220 Aalborg Øst
Telefon 99 40 88 04
Fax 98 15 81 29
http://www.math.aau.dk

Titel: Avanceret teori og metode

Hovedområde: Sandsynlighedsregning og statistik

PE-kursus: Grafiske modeller

Projektperiode: MAT4, forårssemestret 2009

Projektgruppe: G4-115a

Projektmedlem: Mikkel Meyer Andersen

Vejleder: Poul Svante Eriksen

Oplagstal: 4
Sidetal: 58
Bilagsantal og –art: Ingen
Afsluttet: 4. juni 2009.

Synopsis:

I dette projekt arbejdes der med Y-STR-data og modellering af locis afhængighed. Denne afhængighedsstruktur i Y-STR er ukendt, hvilket gør opgaven vanskelig. I et forsøg på at finde denne afhængighed, kigges der bl.a. på automatisk strukturlæring vha. PC-algoritmen samt informationskriterierne BIC og AIC. Det viser sig dog at være problematisk med automatisk strukturlæring pga. tynde datasæt. Derfor konstrueres også en grafisk model ud fra locis fysiske placering i håb om, at det måske afspejler afhængigheden. For at vurdere kvaliteten af de grafiske modeller, indføres Brier-score og estimation af den uobserverede sandsynlighedsmasse. Det viser sig, at ingen af de fundne modeller er brugbare i praksis.

For at forsøge at lave tættere datasæt, så den automatiske strukturlæring måske bliver bedre, undersøges om visse alleler kan kollapses for at minimere antal mulige haplotyper. Test for uafhængighed i forbindelse med kollapsning foretages både med Fishers eksakte test og Kruskall og Goodmans γ. Det viser sig, at der kan kollapses voldsomt – faktisk kan mulige haplotyper reduceres med mere end 90% ved brug af Fishers eksakte test.

I forbindelse med kollapsning undersøges også estimation af p-værdier vha. importance sampling af $2 \times k$-kontingenstabeller, hvilket netop er dem, der bruges i forbindelse med kollapsning. Udover modellering af locis afhængighed, udregnes også afstande mellem populationerne ved hjælp af Φ_{ST}-værdier. Ved hjælp af klusteranalyse kan der konstrueres et dendogram, der tydeligt viser, hvad man kunne forvente: der er klar sammenhæng mellem geografisk afstand mellem populationer og afstande mellem populationernes haplotyper.

Forord

Dette projekt er udarbejdet i forbindelse med MAT4-semestret i forårssemesteret 2009 af Mikkel Meyer Andersen, studerende ved Institut for Matematiske Fag på Aalborg Universitet. Projektets tema er "Avanceret teori og metode", og hovedområdet er "Sandsynlighedsregning og statistik" med "Grafiske modeller" som PE-kursus, hvor bl.a. [Edwards, 2000] er blevet gennemgået.

I studieordningen står der:

> *Studerende, for hvem MAT4-semestret ikke optræder som det afsluttende semester i deres bacheloruddannelse, kan erstatte den sædvanlige rapporteringsform med en kortere oversigt over læste emner eller anvendte metoder. Eksempelvis en oversigt over de centrale begreber og resultater på synopsis- eller artikelform.*

Da undertegnede har opnået bachelorgraden, er den sædvanlige rapporteringsform erstattet med en oversigt over læste emner og anvendte metoder. Der er således i projektperioden blevet anvendt væsentligt mere tid på at læse artikler og bøger samt implementere metoder og algoritmer end at nedskrive teori og finpudse projektet (der er blevet produceret ca. 2200 liniers R-kode og ca. 5300 liniers C#-kode).

Kildekoden til programmer og scripts konstrueret i dette projekt kan findes på http://people.math.aau.dk/~mikl/mat4/projekt.

Der skal lyde en stor tak til vejlereden Poul Svante Eriksen, der har bidraget med megen konstruktiv kritik. Derudover skal Torben Tvedebrink (ph.d.-studerende ved Institut for Matematiske Fag, Aalborg Universitet) også have en stor tak for altid at have tid til kontruktive samtaler omkring retsgenetik såvel som matematik i almindelighed.

Tilsvarende skal Retsmedicinsk Institut på Københavns Univertitet have stor tak for at levere data til dette projekt.

Bemærk at der i projektet benyttes engelske separatorer. Med andre ord benyttes komma som tusindeseparator og punktum som decimalseparator. Tilsvarende er aritmetiske gennemsnit noteret som $\overline{x}$, summation over et indeks noteres $x_{\cdot j} = \sum_i x_{ij}$ og vektorer noteres som $\boldsymbol{x}$.

Mikkel Meyer Andersen

Indhold

Kapitel 1

Introduktion

I traditionel STR på de autosomale kromosomer, er der uafhængighed mellem de forskellige loci. Denne uafhængighed er eksempelvis nyttig i udregning af den bevismæssige vægt. I Y-STR er der ikke uafhængighed mellem de forskellige loci. Det gør, at udregningen af den bevismæssige vægt er mere kompliceret, men omvendt vil afhængigheden kunne udnyttes til eksempelvis at estimere en given Y-STR-haplotypefrekvens under en grafisk model, som afspejler afhængigheden mellem loci. I resten af rapporten vil begrebet haplotype henvise til Y-STR-haplotype. Projektet vil fokusere på metoder vedrørende locis afhængighed (primært ved hjælp af grafiske modeller): hvordan man identificerer afhængighedsstrukturen mm. Udregning af den bevismæssige vægt vil ikke blive omtalt yderligere i dette projekt.

1.1 Beskrivelse af data

I dette projekt, har der været anvendt tre datasæt:

- `dane` fra [Hallenberg et al., 2004] bestående af de 10 loci DYS19, DYS389-I, DYS389-II, DYS390, DYS391, DYS392, DYS393, DYS437, DYS438 og DYS439 (locus DYS385a/b er blevet sorteret bort grundet entydighedsproblematikken[1])
- `somali` fra [Hallenberg et al., 2005] bestående af samme 10 loci som `dane`
- `euystr04` fra http://www.yhrd.org i 2004 bestående af samme loci som `dane` pånær DYS437, DYS438 og DYS439

Datasættene `dane` og `somali` kan findes på http://people.math.aau.dk/~mikl/mat4/projekt, mens `euystr04` kan fås ved at sende en e-mail til mikl@math.aau.dk.

Følgende R-script (der anvender `hclust`-funktionen til clustering på en Manhattan-afstandsmatrix) giver figur 1.1, der viser, at haplotyperne i `dane` og `somali` hovedsageligt er ret forskellige:

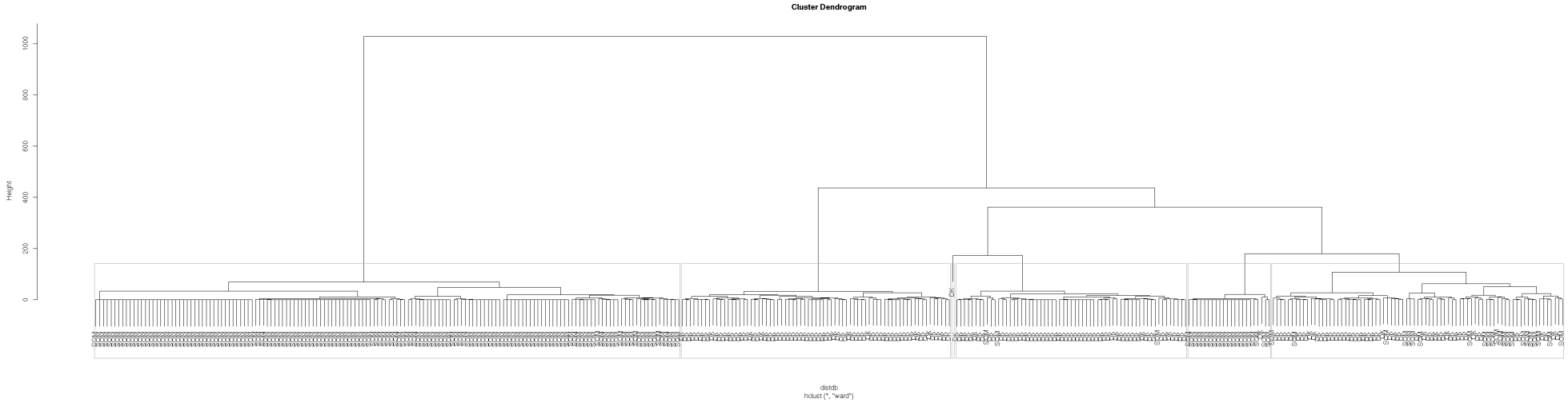
```
db <- rbind(danes, somali)
db <- as.data.frame(db)
db$pop <- c(rep("DK", nrow(danes)), rep("SOM", nrow(somali)))
distdb <- dist(db[, -ncol(db)], method="manhattan")
fit <- hclust(distdb, method="ward")
plot(fit, labels=db$pop)
groups <- cutree(fit, k=6)
rect.hclust(fit, k=6, border="darkgrey")
```

[1] Allelerne for DYS385a/b er på formen X-Y, hvor man ikke kan skelne om DYS385a=X og DYS385b=Y eller DYS385a=Y og DYS385b=X.

1.2 Berørte emner

I dette projekt er der blevet arbejdet med følgende:

- Udregning af afstande mellem populationerne på baggrund af metoden beskrevet i [Excoffier et al., 1992] udtrykt vha. Φ_{ST}-værdier på samme måde som gjort i [Roewera et al., 2000] (behandles i kapitel 2)
- Udvikling af værktøjet `SL` til strukturlæring (vha. PC-algoritmen, BIC og AIC) og udregning af sandsynligheder i grafiske modeller, som enten er fundet gennem strukturlæring eller specificeret manuelt (behandles i kapitel 3)
- Opstilling af grafiske modeller på baggrund af locis fysiske placering (behandles i afsnit 3.2)
- Udnyttelse af `SL` på de tre datasæt samt sammenligning af de resulterende grafiske modeller og modeller afspejlende locis fysiske placering ved hjælp af dækket sandsynlighedsmasse og Brier score (behandles i afsnit 3.3)
- Kollapsning af forskellige alleler for at lave tættere datasæt ved hjælp af uafhængighedstests med Fishers eksakte test samt Kruskall og Goodmans γ og efterfølgende sammenligning med ukollapsede datasæt (behandles i kapitel 4)
- Implementering af importance samplingen fra [Mehta et al., 1988] samt sammenligning med traditionel Monto Carlo-sampling i forbindelse med estimation af p-værdier for $2 \times k$-kontingenstabeller (behandles i kapitel 5)



Figur 1.1: Dendogram der viser, at haplotyperne i `dane` og `somali` hovedsageligt er ret forskellige.

Kapitel 2

AMOVA

Analyse af molekylær varians (*analysis of molecular variance*) er behandlet i [Excoffier et al., 1992]. De kommer bl.a. frem til følgende kvadratafvigelsessummer (*sum of squared deviations*) – med samme notation som i kilden (WP = *within population*; AP/WG = *among population, within group*; AG = *among group*):

$$\begin{aligned}
\mathrm{SSD}_{\mathrm{WP}} &= \sum_{g=1}^{G} \sum_{i=1}^{I_g} \frac{\sum_{j=1}^{N_{ig}} \sum_{k=1}^{N_{ig}} \delta_{jk}^2}{2N_{ig}}, \\
\mathrm{SSD}_{\mathrm{AP/WG}} &= \sum_{g=1}^{G} \left(\frac{\sum_{i=1}^{I_g} \sum_{j=1}^{N_{ig}} \sum_{i'=1}^{I_g} \sum_{k=1}^{N_{i'g}} \delta_{jk}^2}{\sum_{i=1}^{I_g} 2N_{ig}} - \sum_{i=1}^{I_g} \frac{\sum_{j=1}^{N_{ig}} \sum_{k=1}^{N_{ig}} \delta_{jk}^2}{2N_{ig}} \right), \\
\mathrm{SSD}_{\mathrm{AG}} &= \frac{\sum_{j=1}^{N} \sum_{k=1}^{N} \delta_{jk}^2}{2N} - \sum_{g=1}^{G} \frac{\sum_{i=1}^{I_g} \sum_{j=1}^{N_{ig}} \sum_{i'=1}^{I_g} \sum_{k=1}^{N_{i'g}} \delta_{jk}^2}{\sum_{i=1}^{I_g} 2N_{ig}},
\end{aligned}$$

hvor δ_{jk} er en Euklidisk afstandsmetrik mellem haplotype h_j og h_k, I_g er antal populationer i den g'te gruppe og N_{ig} er antal individer i den i'te population i den g'te gruppe (hierarkiet er individ $\subseteq$ population $\subseteq$ gruppe).

Rent beregningsmæssigt er det dog ikke den mest optimale måde. I tilfældet, hvor afstanden mellem to haplotyper blot afspejler om allellerne på de forskellige loci er ens eller ej, kan man i stedet lave kvadratafvigelsessummer mere beregningseffektive. Denne simple metrik har dog den ulempe, at den ikke inkorporerer single-step mutationer. For at lave metrikken Euklidisk, laves kvadratafstandsmålet som dummy-kodning, hvor et loci er en vektor med en længde svarende til antallet af mulige alleller og komponenterne har værdien $2^{-\frac{1}{2}}$ på pladsen svarende til det aktuelle allel og 0 på de andre pladser.

For eksemplets skyld antages haplotypen at bestå af locus DYS437, der kan antage allellerne 14, 15 og 16, samt locus DYS19, der kan antage allellerne 22, 23, 24, 25. En haplotype $h_1 = (14, 24)$ kan så dummy-kodes som vektoren $(\underbrace{2^{-\frac{1}{2}}, 0, 0}_{\text{DYS437}}, \underbrace{0, 0, 2^{-\frac{1}{2}}, 0}_{\text{DYS19}})$, og tilsvarende kan $h_2 = (16, 22)$ dummy-kodes som $(\underbrace{0, 0, 2^{-\frac{1}{2}}}_{\text{DYS437}}, \underbrace{2^{-\frac{1}{2}}, 0, 0, 0}_{\text{DYS19}})$. Dermed bliver

$$\begin{aligned}
\|h_1 - h_2\|^2 &= \left(2^{-\frac{1}{2}} - 0\right)^2 + (0-0)^2 + \left(0 - 2^{-\frac{1}{2}}\right)^2 + \\
&\quad \left(0 - 2^{-\frac{1}{2}}\right)^2 + (0-0)^2 + \left(2^{-\frac{1}{2}} - 0\right)^2 + (0-0)^2 + \\
&= \left(2^{-\frac{1}{2}}\right)^2 + \left(-2^{-\frac{1}{2}}\right)^2 + \left(-2^{-\frac{1}{2}}\right)^2 + \left(2^{-\frac{1}{2}}\right)^2 \\
&= 2,
\end{aligned}$$

hvilket netop er antallet af positioner, hvor haplotyperne er forskellige.

Lad $\boldsymbol{x}_{jig}$ betegne den dummy-kodede haplotype for det j'te individ i den i'te population i den g'te gruppe og lad S være antal loci i haplotype. Dermed kan kvadratafvigelsessummer i stedet laves således, at

$$\begin{aligned}
\mathrm{SSD}_{\mathrm{WP}} &= \sum_{g=1}^{G}\sum_{i=1}^{I_g}\sum_{j=1}^{N_{ig}} \left\|\boldsymbol{x}_{jig} - \overline{\boldsymbol{x}}_{\cdot ig}\right\|^2 \\
&= \sum_{g=1}^{G}\sum_{i=1}^{I_g}\sum_{j=1}^{N_{ig}} \left\|\boldsymbol{x}_{jig}\right\|^2 - \sum_{g=1}^{G}\sum_{i=1}^{I_g}\sum_{j=1}^{N_{ig}} \left\|\overline{\boldsymbol{x}}_{\cdot ig}\right\|^2 \\
&= N - \frac{1}{2}\sum_{s=1}^{S}\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\| \frac{\boldsymbol{n}_{ig}^s}{N_{ig}} \right\|^2,
\end{aligned}$$

hvor man udnytter at dummy-kodningen resulterer i, at $\left\|\boldsymbol{x}_{jig}\right\| = 1$. Vektoren $\boldsymbol{n}_{ig}^s$ består af allel-antaller for locus s, og den har dermed antallet af forskellige alleller som dimension.

Tilsvarende kan gøres for

$$\begin{aligned}
\mathrm{SSD}_{\mathrm{AP/WG}} &= \sum_{g=1}^{G}\sum_{i=1}^{I_g}\sum_{j=1}^{N_{ig}} \left\|\overline{\boldsymbol{x}}_{\cdot ig} - \overline{\boldsymbol{x}}_{\cdot\cdot g}\right\|^2 \\
&= \sum_{g=1}^{G}\sum_{i=1}^{I_g}\sum_{j=1}^{N_{ig}} \left\|\overline{\boldsymbol{x}}_{\cdot ig}\right\|^2 - \sum_{g=1}^{G}\sum_{i=1}^{I_g}\sum_{j=1}^{N_{ig}} \left\|\overline{\boldsymbol{x}}_{\cdot\cdot g}\right\|^2 \\
&= \sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\|\overline{\boldsymbol{x}}_{\cdot ig}\right\|^2 - \sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\|\overline{\boldsymbol{x}}_{\cdot\cdot g}\right\|^2 \\
&= \frac{1}{2}\sum_{s=1}^{S}\left(\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\| \frac{\boldsymbol{n}_{ig}^s}{N_{ig}} \right\|^2 - \sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\| \frac{\boldsymbol{n}_{g}^s}{\sum_{i=1}^{I_g} N_{ig}} \right\|^2 \right)
\end{aligned}$$

og

$$\begin{aligned}
\mathrm{SSD}_{\mathrm{AG}} &= \sum_{g=1}^{G}\sum_{i=1}^{I_g}\sum_{j=1}^{N_{ig}} \left\|\overline{\boldsymbol{x}}_{\cdot\cdot g} - \overline{\boldsymbol{x}}_{\cdot\cdot\cdot}\right\|^2 \\
&= \frac{1}{2}\sum_{s=1}^{S}\left(\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\| \frac{\boldsymbol{n}_{g}^s}{\sum_{i=1}^{I_g} N_{ig}} \right\|^2 - \sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\| \frac{\boldsymbol{n}^s}{\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig}} \right\|^2 \right) \\
&= \frac{1}{2}\sum_{s=1}^{S}\left(\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} \left\| \frac{\boldsymbol{n}_{g}^s}{\sum_{i=1}^{I_g} N_{ig}} \right\|^2 - N \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 \right)
\end{aligned}$$

Ved at benytte hjælpestørrelserne[1]

$$n = \frac{\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} - \sum_{g=1}^{G}\left(\frac{\sum_{i=1}^{I_g} N_{ig}^2}{\sum_{i=1}^{I_g} N_{ig}}\right)}{\sum_{g=1}^{G} I_g - G}$$

$$n' = \frac{\sum_{g=1}^{G}\left(\frac{\sum_{i=1}^{I_g} N_{ig}^2}{\sum_{i=1}^{I_g} N_{ig}}\right) - \frac{\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig}^2}{\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig}}}{G-1} \quad \text{for } G > 1$$

$$n'' = \frac{\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig} - \frac{\sum_{g=1}^{G}\left(\sum_{i=1}^{I_g} N_{ig}\right)^2}{\sum_{g=1}^{G}\sum_{i=1}^{I_g} N_{ig}}}{G-1} \quad \text{for } G > 1$$

kan man udregne

$$\sigma_c^2 = \frac{\mathrm{SSD_{WP}}}{N - \sum_g I_g}$$

$$\sigma_c^2 + n\sigma_b^2 = \frac{\mathrm{SSD_{AP/WG}}}{\sum_g I_g - G} \Rightarrow \sigma_b^2 = \frac{\mathrm{SSD_{AP/WG}}}{n\left(\sum_g I_g - G\right)} - \frac{\sigma_c^2}{n}$$

$$\sigma_c^2 + n'\sigma_b^2 + n''\sigma_a^2 = \frac{\mathrm{SSD_{AG}}}{G-1} \Rightarrow \sigma_a^2 = \frac{\mathrm{SSD_{AG}}}{n''(G-1)} - \frac{n'\sigma_b^2 + \sigma_c^2}{n''} \quad \text{for } G > 1.$$

Den beskrevne metode resulterer udelukkende i én Φ_{ST}-værdi, så for at analysere variansen mellem populationer, kan man udregne parvise Φ_{ST}-værdier ved at konstruere én gruppe med de to populationer som [Roewera et al., 2000] eksempelvis gør. Dermed kan udregningerne reduceres til:

$$\begin{aligned}
\mathrm{SSD_{WP}} &= N - \frac{1}{2}\sum_{s=1}^{S}\sum_{i=1}^{2} N_i \left\| \frac{\boldsymbol{n}_i^s}{N_i} \right\|^2 \\
&= N - \frac{1}{2}\sum_{s=1}^{S}\left(N_1 \left\| \frac{\boldsymbol{n}_1^s}{N_1} \right\|^2 + N_2 \left\| \frac{\boldsymbol{n}_2^s}{N_2} \right\|^2 \right), \\
\mathrm{SSD_{AP/WG}} &= \frac{1}{2}\sum_{s=1}^{S}\sum_{i=1}^{2} N_i \left(\left\| \frac{\boldsymbol{n}_i^s}{N_i} \right\|^2 - \left\| \frac{\boldsymbol{n}^s}{\sum_{i=1}^{2} N_i} \right\|^2 \right) \\
&= \frac{1}{2}\sum_{s=1}^{S}\left(N_1 \left(\left\| \frac{\boldsymbol{n}_1^s}{N_1} \right\|^2 - \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 \right) + N_2 \left(\left\| \frac{\boldsymbol{n}_2^s}{N_2} \right\|^2 - \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 \right) \right), \\
\mathrm{SSD_{AG}} &= \frac{1}{2}\sum_{s=1}^{S}\left(\sum_{i=1}^{2} N_i \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 - N \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 \right) = \frac{1}{2}\sum_{s=1}^{S}\left(N_1 \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 + N_2 \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 - N \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 \right) \\
&= \frac{1}{2}\sum_{s=1}^{S}\left((N_1 + N_2) \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 - N \left\| \frac{\boldsymbol{n}^s}{N} \right\|^2 \right) \\
&= 0 \\
n &= \sum_{i=1}^{2} N_i - \left(\frac{\sum_{i=1}^{2} N_i^2}{\sum_{i=1}^{2} N_i} \right) = N - \frac{N_1^2 + N_2^2}{N} = N - \frac{(N_1 + N_2)^2 - 2N_1N_2}{N} = N - \frac{N^2 - 2N_1N_2}{N} \\
&= \frac{2N_1N_2}{N} \\
\sigma_c^2 &= \frac{\mathrm{SSD_{WP}}}{N-2} \\
\sigma_b^2 &= \frac{\mathrm{SSD_{AP/WG}} - \sigma_c^2}{n}
\end{aligned}$$

[1] I [Excoffier et al., 1992] trækkes G ikke fra i nævneren i udtrykket for n, men det er en trykfejl, hvilket manualen til programmet "Arlequin" også afspejler.

Ved at implementere dette i R, kan man vha. klusteranalysen foretaget af R-funktionen hclust få dendogrammet i figur 2.1. R-koden for plottet og AMOVA-analysen kan findes i AMOVA-*.R-filerne, der kan downloades fra http://people.math.aau.dk/~mikl/mat4/ projekt. Afstanden mellem dane og somali er udregnet på baggrund af alle 10 loci, mens afstande mellem dane/somali og populationer fra euystr04-datasættet kun sker på baggrund af de 7 loci i euystr04-datasættet (det vil sige, at locus DYS437, DYS438 og DYS439 i dane/somali bliver ignoreret i udregninger med euystr04-datasættet).

Figur 2.1: Dendogram ud fra parvise AMOVA-afstande.

Kapitel 3

Grafiske modeller

I dette kapitel arbejdes der hovedsageligt med grafiske modeller. Først kommer et afsnit om strukturlæring, dernæst kommer et afsnit omkring modelkontrol af grafiske modeller, så kommer et afsnit omkring haplotypemodellering ud fra fysisk placering og til sidst kommer et afsnit med resultater af strukturlæring og sammenligninger mellem de forskellige grafiske modeller.

3.1 SL – Structure Learner

For at få et bedre indblik i strukturinferens, er to algoritmer til strukturinferens blevet implementeret i et program, der er blevet døbt SL (for *structure learner*). Implementeringen er foretaget i programmeringssproget C#, der enten kræver .NET- eller Mono-frameworket installeret. I SL er det kun muligt at håndtere DAGs (dvs. orienterede grafer uden kredse). Dels er der blevet implementeret en scoringsbaseret grådig algoritme, hvor man kan vælge mellem enten BIC eller AIC som informationskriterium, og dels er den restriktionsbaserede PC-algoritme blevet implementeret. Den udvidede NPC-algoritme er ikke blevet implementeret.

3.1.1 Scoringsbaseret grådig algoritme

Scoringsbaseret læring foregår typisk vha. dekomposable informationskriterier på formen

$$-2 \log Q + kp,$$

hvor det gælder om at minimere scoren. Q er likelihood af data under modellen og p er kompleksiteten af modellen. Tilfældet $k = 2$ kaldes AIC (for *Akaike Information Criteria*) og $k = \log N$, hvor N er antal observationer, kaldes BIC (for *Bayesian Information Criteria*).

Hvis man transformerer scoringsfunktionen vha. $x \mapsto -\frac{x}{2}$ ligesom [Jensen and Nielsen, 2007, s. 243] (hvor det så gælder om at maksimere scoren), så kan BIC for en DAG ifølge [Jensen and Nielsen, 2007, s. 243] udtrykkes som

$$\text{BIC} = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \left(\frac{N_{ijk}}{N_{ij\cdot}} \right) - \frac{\log N_{\cdots}}{2} \sum_{i=1}^{n} q_i (r_i - 1),$$

hvor

- n er antallet af variable (knuder i grafen),
- x_i er den i'te variabel,

- r_i er antallet af tilstande i x_i,
- $q_i = \prod_{x_j \in \text{pa}(x_i)} r_j$ er antallet af konfigurationer over forældrene (med den sædvanlige konvention, at det tomme produkt er 1) og
- N_{ijk} er antallet af tilfælde i databasen med x_i i sin k'te tilstand og $\text{pa}(x_i)$ i sin j'te tilstand.

Den grådige version, der er implementeret i SL er blevet implementeret efter beskrivelsen i [Jensen and Nielsen, 2007, algoritme 7.2, s. 246]. Fordi den er grådig, finder den kun et lokalt maksimum. I SL er det gjort muligt at starte fra tilfældige startstrukturer for at teste stabiliteten af de fundne maksima. Der er også blevet forsket i at finde et globalt maksimum uden at skulle gennemsøge alle DAGs. Et eksempel på det er [Singh and Moore, 2005], der begrænser mængden af mulige DAGs på følgende måde (citat):

> We start with a simpler question than full structure discovery. Every acyclic digraph has at least one leaf. Can we identify a leaf of the best network, $x = \text{Leaf}(V)$? Since x cannot be the parent of any other variable, the score of the best network on V consists of two parts: (i) the score of the best network on $V - x$ and (ii) the score of $\text{Leaf}(V)$ given the best set of parents from $V - x$, i.e., the set of parents that maximizes the node score of x. This argument requires a decomposable scoring metric. Recursively, we can ask which node is a leaf in the best network on $V - x$. This process induces a (reverse) order on the variables. It remains to be shown how to choose $\text{Leaf}(V)$.

De finder $\text{Leaf}(V)$ vha. dynamisk programmering. På denne måde undgår de at lave en udtømmende søgning, der kører i supereksponentiel tid, men reducerer søgningen til "kun" at køre i eksponentiel tid. De skriver ydermere, at deres algoritme er praktisk anvendelig for færre end 26 knuder.

3.1.2 PC-algoritmen

PC-algoritmen er blevet implementeret efter beskrivelsen i [Jensen and Nielsen, 2007, afsnit 7.1, s. 230]. Implementationen er forholdsvis langsom, da den er implementeret i et fuldt objektorienteret sprog uden ret mange optimeringer. Det er dog værre, at den er ubrugelig, da den er implementeret præcis som foreskrevet i [Jensen and Nielsen, 2007], og ved grundig gennemlæsning af denne opdager man, at beskrivelsen er problematisk. Dels er der et mindre problem i forbindelse med introduktion af v-strukturer, og dels er der et alvorligt problem i forbindelse med den efterfølgende proces med at orientere kanterne. Forfatterne af [Jensen and Nielsen, 2007] er blevet gjort opmærksom på problemerne.

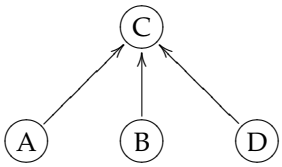
Problem med introduktion af v-strukturer

I [Jensen and Nielsen, 2007, "Rule 1", s. 231] er der en vis uklarhed over, hvornår nye v-strukturer introduceres; de skriver:

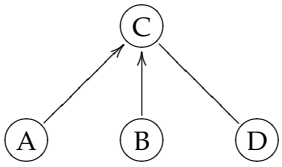
> *If you have three nodes A, B, C, such that $A - C$ and $B - C$, but not $A - B$, then introduce the v-structure $A \rightarrow C \leftarrow B$ if there exists an $\mathcal{X}$ (possible empty) such that $I(A, B, \mathcal{X})$ and $C \notin \mathcal{X}$.*

Notationen $I(A, B, \mathcal{X})$ betyder, at A er d-separeret fra B givet $\mathcal{X}$. Når alle sådanne v-strukturer er indført, skal man efterfølgende undgå at indføre nye v-strukturer. Umiddelbart gør den

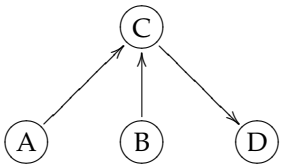
nævnte regel genskabelsen af en struktur som eksempelvis



umulig. Først ville man introducere v-strukturen $A \rightarrow C \leftarrow B$:



Herefter skal nye v-strukturer undgås, hvilket ville resultere i:

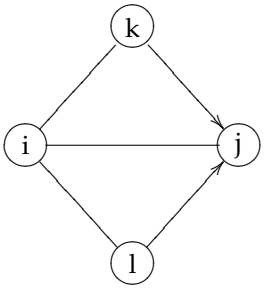


I stedet bør reglens ordlyd være:

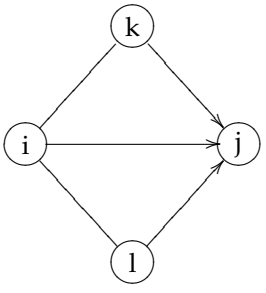
> *If you have three nodes A, B, C, such that $A - C$ and $B - C$ (or wlog. $A \rightarrow C$ and $B - C$), but not $A - B$, then introduce the v-structure $A \rightarrow C \leftarrow B$ if there exists an $\mathcal{X}$ (possible empty) such that $I(A, B, \mathcal{X})$ and $C \notin \mathcal{X}$.*

Problem med orientering af resterende kanter

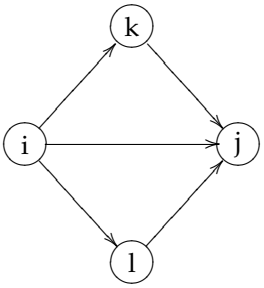
Hvis man sammenligner [Jensen and Nielsen, 2007, Rule 2-4, afsnit 7.1] med [Kalisch and Buhlmann, 2007, Algorithm 2, s. 619] ser man, at Rule 2 i [Jensen and Nielsen, 2007] (undgå nye v-strukturer) er ækvivalent med R1 i [Kalisch and Buhlmann, 2007]. Tilsvarende er Rule 3 i [Jensen and Nielsen, 2007] (undgå cykler) ækvivalent med R2 i [Kalisch and Buhlmann, 2007]. Herefter er Rule 4 i [Jensen and Nielsen, 2007], at man skal orientere kanterne tilfældigt. Dog er der to regler mere i [Kalisch and Buhlmann, 2007]: R3 siger, at hvis man har situationen



hvor k og l ikke må være naboer, så skal $i - j$ orienteres til $i \rightarrow j$, således at



Med udgangspunkt i denne regel, vil [Jensen and Nielsen, 2007] kun kunne genskabe følgende DAG ved hjælp af tilfældighed:



Grunden til det er, at $i - j$ kan risikere at blive orienteret $i \leftarrow j$ i stedet for $i \rightarrow j$, da det er tilfældighedsreglen, der vil blive anvendt.

Det skal dog nævnes, at R4 i [Kalisch and Buhlmann, 2007] også har minimum én trykfejl; reglen lyder nemlig som følger:

> **R4** Orient $i - j$ into $i \rightarrow j$ whenever there are two chains $i - k \rightarrow l$ and $k \rightarrow l \rightarrow j$ such that k and l are nonadjacent.

Muligvis burde reglen i stedet ende med "such that k and j are nonadjacent" eller " such that i and l are nonadjacent".

Disse nævnte problemer gør, at PC-algoritmen i `SL` er ubrugelig. I stedet kan programmet Hugin eksempelvis anvendes.

3.1.3 Udregning af sandsynligheder

Ud over strukturlæring kan `SL` udregne sandsynligheder i DAGs. Man kan vælge både at udregne sandsynligheder i de lærte strukturer eller i en gyldig struktur, man selv specificerer (dvs. en uden kredse). Man kan vælge enten at specificere de sandsynligheder, man ønsker at få udregnet; at få udregnet dem, der er repræsenteret i datasættet eller sandsynlighederne for alle mulige konfigurationer. Da sidstnævnte mulighed kan give anledning til et enormt antal udregninger, skriver `SL` hvor mange, det drejer sig om.

3.1.4 Kom hurtigt i gang med `SL`

Dataformatet `SL` forstår består af en header, der beskriver kolonnerne, efterfulgt af data. Mulige kolonner er:

- ! angiver at kolonnen skal ignoreres
- # angiver at kolonnen er et antal (og dermed bliver rækken talt med det antal gange, der står i #-kolonnen) – kun én kolonne må være af typen #
- @ angiver at kolonnen er en faktor eller gruppering om man vil; alle rækker med samme værdi i @-kolonnen samles i én strukturlæring og sandsynlighedsudregning uden alle de andre – kun én kolonne må være af typen @
- De andre kolonner (evt. indkapslet i anførselstegn) fungerer som variabelnavne/knuder/tilstande
- Kolonner separeres af mellemrum, tab og semikolon

Et eksempel på en lovlig data-fil er følgende:

```
@ X1 X2 # !
"Type_1" 1 1 6 0.6
"Type_1" 1 0 2 0.2
"Type_1" 0 0 2 0.2
"Type_2" 1 1 10 0.5
"Type_2" 0 1 5 0.25
"Type_2" 0 0 5 0.25
```

Man kan også selv specificere DAGs, som man blot ønsker at udregne sandsynligheder i. En DAG specificeres ved at angive en tilstands forældres separeret af komma. Følgende er et eksempel, hvor `DYS19` har `DYS391` som forælder og `DYS439` har både `DYS389-I` og `DYS389-II` som forældre:

```
DYS19: DYS391
DYS389-I: DYS438, DYS389-II
DYS389-II: DYS438
DYS390: DYS392
DYS391: DYS437
DYS392:
DYS393: DYS19
DYS437: DYS439
DYS438: DYS390
DYS439: DYS389-I, DYS389-II
```

3.2 Modellering ud fra fysisk placering af loci

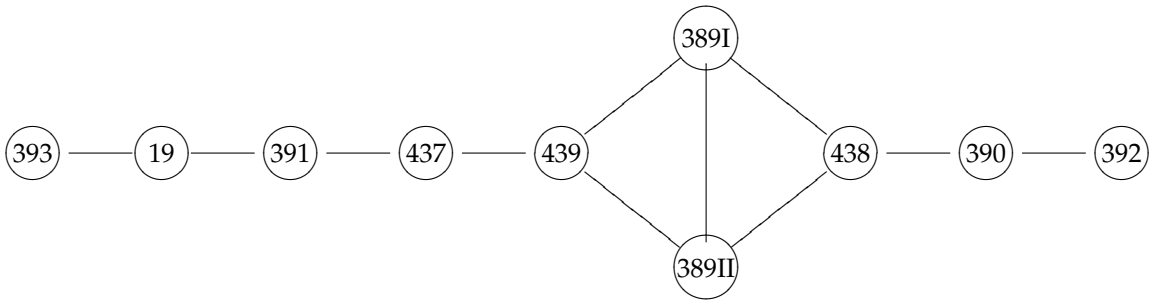
I stedet for at foretage automatisk fitning af modellerne, kan man også bruge a priori-viden til at konstruere modellerne. Umiddelbart ved vi ikke, hvordan de forskellige loci har indflydelse på hinanden; det vides blot, at de ikke er uafhængige. En mulighed kunne være, at deres indbyrdes fysiske placering på dna'et afspejlede deres indbyrdes afhængighedsforhold. I [Hanson and Ballantyne, 2005] kan de forskellige locis fysiske placering findes.

Der er lavet tre forskellige modeller (navnepræfikset "DYS" er ikke medtaget af overskuelighedsårsager):

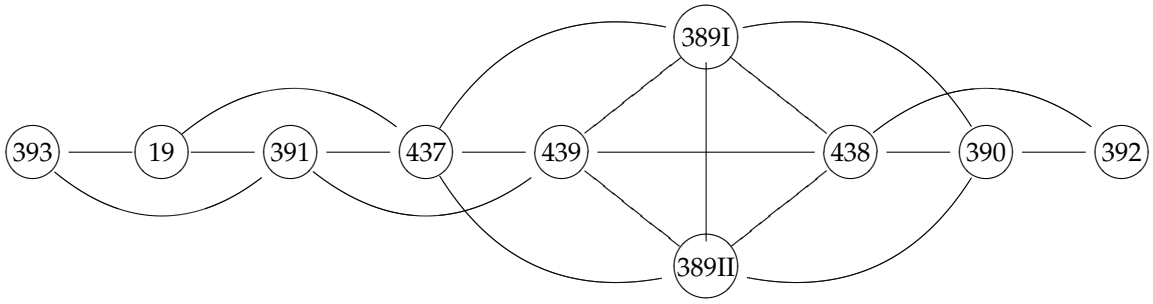
- 1. ordens model: figur 3.1. Her er det antaget, at de enkelte loci kun er afhængige af sine direkte naboer.
- 2. ordens model: figur 3.2. Her er det antaget, at de enkelte loci kun er afhængige af sine direkte naboer samt naboernes direkte naboer.
- 3. ordens model: figur 3.3. Her er det antaget, at de enkelte loci kun er afhængige af sine direkte naboer, deres direkte naboer samt naboernes naboers direkte naboer.

Her er modellerne lavet som uorienterede grafiske modeller, men de kunne lige så godt være lavet som DAGs, da det ville give de samme Markov-egenskaber idet alle knuders forældre er indicente pr. konstruktion.

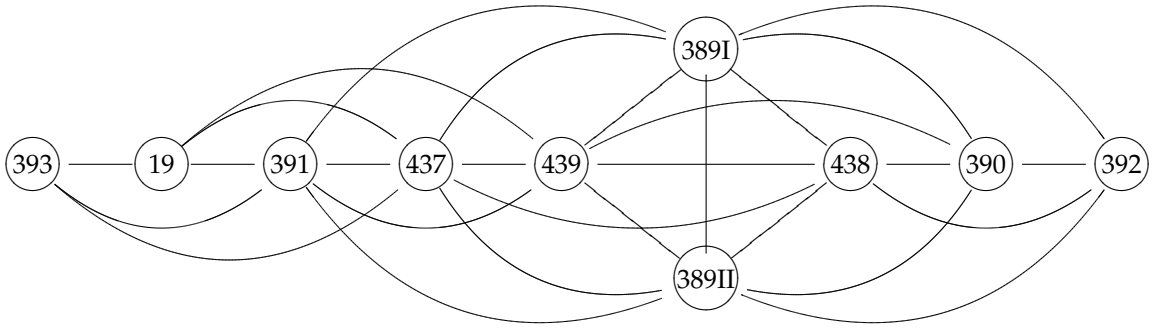
Ved at lave DAGs (se DAG for 1. ordens modellen i figur 3.4 lavet vha. `SL`) i stedet for uorienterede grafiske modeller, kan `SL` anvendes til at udregne dækket sandsynlighedsmasse. Markovegenskaberne i de uorienterede og orienterede grafiske modeller er ens, da den moraliserede graf for den orienterede graf har de samme kanter som den orienterede graf (populært sagt er alle forældrene gift).



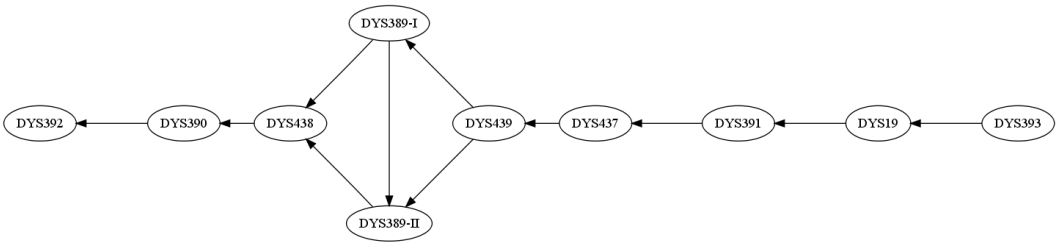
Figur 3.1: 1. ordens model efter locis fysiske placering på dna'et



Figur 3.2: 2. ordens model efter locis fysiske placering på dna'et



Figur 3.3: 3. ordens model efter locis fysiske placering på dna'et



Figur 3.4: 1. ordens modellen som DAG tegnet vha. SL.

Datasæt	Singletons	Observationer	Uobserveret masse
`dane`	112	185	0.6021505
`somali`	56	201	0.2772277

Tabel 3.1: Sandsynlighedsmasse for uobserverede haplotyper.

3.3 Resultater

I dette afsnit introduceres først to måder at vurdere, hvor godt en grafisk model modellerer loci afhængighed. Dels kan man sige noget om, hvor meget sandsynlighedsmasse uobserverede haplotyper ca. udgør, og dels kan ved hjælp af Brier score vurdere, hvor godt en grafisk model modellerer data. Det observerede data skal gerne være sandsynligt under en given model.

I dette afsnit tages der kun udgangspunkt i `dane` og `somali`, fordi `euystr04` indeholder mange forskellige populationer, og AMOVA-analysen i kapitel 2 viste, at man ikke kan tillade sig at betragte hele datasættet som én population. Samtidig viser analysen også, at `dane` og `somali` repræsenterer to vidt forskellige populationer.

3.3.1 Estimat for sandsynlighedsmassen af uobserverede haplotyper

Jf. [Robbins, 1968] kan sandsynlighedsmassen af uobserverede haplotyper tilnærmelsesvis estimeres ved

$$\frac{\text{antal singletons}}{\text{antal observationer} + 1}.$$

I tabel 3.1 ses sandsynlighedsmassen af uobserverede haplotyper for `dane` og `somali`. Datasættet `euystr04` er ikke medtaget grundet resultatet af AMOVA-analysen i kapitel 2, der viste, at man ikke kan tillade sig at betragte hele datasættet `euystr04` som én population. Tallene er fundet med følgende kode:

```
# ds is the data set in question
length(which(as.matrix(table(apply(ds, 1, paste, collapse=",")))==1))
```

Når man skal udregne den dækkede sandsynlighedsmasse under modellen, summerer man sandsynlighederne for de forskellige observerede haplotyper under den givne model.

3.3.2 Brier score

For at kontrollere, hvor god den grafiske model er, kan man sammenligne den relative frekvens med sandsynligheden udregnet under modellen. Til det formål kan man udregne den såkaldte *Brier score*. Brier scoren bruges som et nødvendigt krav: den model man har, skal som minimum modellere det observerede data godt, men det er ikke nødvendigvis et tilstrækkeligt krav til at sikre en god model.

Lad n være antal observationer i alt, k antal forskellige haplotyper, x_i antal observationer af haplotypen i og p_i sandsynligheden for haplotype i under den grafiske model. Antag at

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

hvormed

$$\begin{aligned} \mathbf{E}\,[x_i] &= np_i \\ \mathbf{Var}\,[x_i] &= np_i(1 - p_i) \end{aligned}$$

Betragt nu den stokastiske variabel

$$y_i = \left(\frac{x_i}{n} - p_i\right)^2.$$

Fra Den Centrale Grænseværdisætning fås, at

$$\sum_{i=1}^{k} y_i \xrightarrow{\sim} \mathrm{N}(\cdot, \cdot),$$

hvor middelværdien og variansen kan findes. Først bemærkes, at

$$\begin{aligned}
\mathbf{E}[y_i] &= \mathbf{E}\left[\left(\frac{x_i}{n} - p_i\right)^2\right] = \mathbf{E}\left[\left(\frac{x_i}{n} - \frac{\mathbf{E}[x_i]}{n}\right)^2\right] = \mathbf{Var}\left[\frac{x_i}{n}\right] = \frac{1}{n^2} n p_i (1 - p_i) \\
&= \frac{1}{n} p_i (1 - p_i) \\
\mathbf{Var}[y_i] &= \mathbf{E}\left[y_i^2\right] - (\mathbf{E}[y_i])^2 = \mathbf{E}\left[\left(\frac{x_i}{n} - p_i\right)^4\right] - (\mathbf{E}[y_i])^2,
\end{aligned}$$

hvor

$$\begin{aligned}
\mathbf{E}\left[\left(\frac{x_i}{n} - p_i\right)^4\right] &= \mathbf{E}\left[\left(\frac{x_i}{n}\right)^4 - 4\left(\frac{x_i}{n}\right)^3 p_i + 6\left(\frac{x_i}{n}\right)^2 p_i^2 - 4\frac{x_i}{n} p_i^3 + p_i^4\right] \\
&= \frac{1}{n^4}\mathbf{E}\left[x_i^4\right] - \frac{4p_i}{n^3}\mathbf{E}\left[x_i^3\right] + \frac{6p_i^2}{n^2}\mathbf{E}\left[x_i^2\right] - \frac{4p_i^3}{n}\mathbf{E}[x_i] + p_i^4.
\end{aligned}$$

Den momentgenerende funktion for x_i er givet ved

$$M_{x_i}(t) = \mathbf{E}[\exp(t x_i)] = \cdots = (p_i \exp(t) + (1 - p_i))^n,$$

og ydermere gælder, at

$$\mathbf{E}\left[x_i^d\right] = M_{x_i}^{(d)}(0),$$

altså den momentgenerende funktions d'te afledede. Dermed fås, at

$$\mathbf{Var}[y_i] = \left(\frac{1}{n^4} M_{x_i}^{(4)}(0) - \frac{4p_i}{n^3} M_{x_i}^{(3)}(0) + \frac{6p_i^2}{n^2} M_{x_i}''(0) - \frac{4p_i^3}{n} n p_i + p_i^4\right) - \left(\frac{1}{n} p_i (1 - p_i)\right)^2.$$

Dermed kan der standardiseres, således at der grundet uafhængighed gælder, at

$$b := \frac{\sum_{i=1}^{k} y_i - \sum_{i=1}^{k} \mathbf{E}[y_i]}{\sqrt{\sum_{i=1}^{k} \mathbf{Var}[y_i]}} \sim N(0, 1),$$

hvormed $b > 1.96$ er kritiske med et signifikansniveau på 2.5% (1.96 er 97.5%-fraktilen for standardnormalfordelingen).

Brier scoren skal dog bruges varsomt. I visse henseender kan den være kritisk; hvordan skal man eksempelvis fortolke, hvis b er signifikant negativ hvormed modellen passer "for godt" på data?

En implementation af den gennemgåede Brier score kan ses i Listing 3.1, hvor udtrykket for variansen er fundet vha. Maple:

```
1 cat("En-sidet test: værdier større end 1.96 er kritiske.\n\n")
2
3 binvar <- function(p, n)
4 {
5   sum(
6   (n^4*p^4+6*n^3*p^3-6*n^3*p^4+7*n^2*p^2-18*n^2*p^3+11*n^2*p^4+n*p-7*n*p^2+12*n*p^3-6*n*p^4)/
7   n^4-4*p*(n^3*p^3+3*n^2*p^2-3*n^2*p^3+n*p-3*n*p^2+2*n*p^3)/
```

```
  n^3+6*p^2*(n^2*p^2+n*p-n*p^2)/n^2-3*p^4-p^2*(1-p)^2/n^2)
}

brier <- function(n, p, rel.freq)
{
  if (length(p) != length(rel.freq))
  {
    stop("length(p) != length(rel.freq)")
  }

  sumObs <- sum((rel.freq - p)^2)
  sumE <- sum((1/n)*p*(1-p))
  var <- binvar(p, n)

  cat("sumObs = sum((rel.freq - p)^2) =", sumObs, "\n")
  cat("sumE = sum((1/n)*p*(1-p)) =", sumE, "\n")
  cat("var = ", var, "\n")
  cat("(sumObs - sumE) / sqrt(var) = ", (sumObs - sumE) / sqrt(var), "\n")
}
```

Listing 3.1: Filen `brier.R`: Brier score

3.3.3 Korrektion

Man kan lave en korrektion, således at man i stedet for $\mathbf{E}[x_i]$ tager $\mathbf{E}[x_i|x_i > 0]$, da resten er 0. Det bemærkes, at

$$\mathbf{E}[X|X>0] = \sum_x xp(x|X>0) = \sum_x x\frac{p(x,X>0)}{p(X>0)} = \frac{\sum_x xp(x)}{p(X>0)} = \frac{\mathbf{E}[X]}{1-p(X=0)} = \frac{\mathbf{E}[X]}{1-(1-p)^n}$$

så derfor sættes

$$p_i := \frac{p_i}{1-(1-p_i)^n}.$$

Man kunne overveje at korrigere variansen på tilsvarende vis.

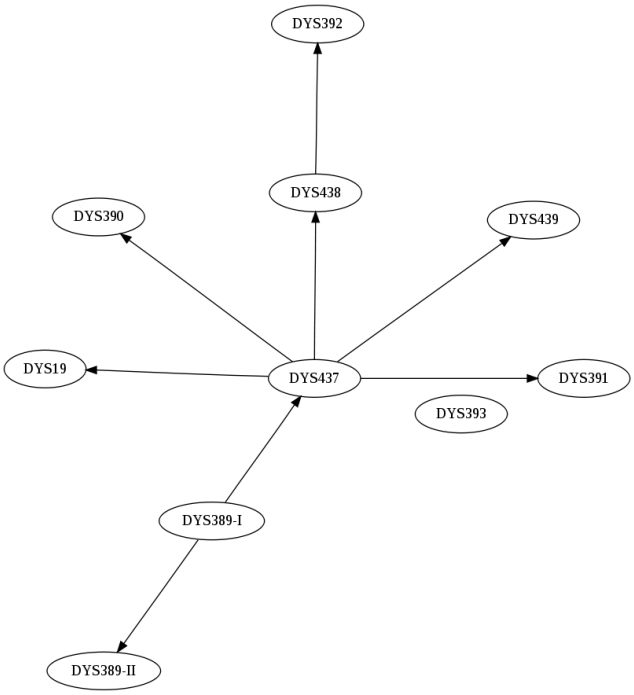
3.3.4 BIC-resultater

I figur 3.5 ses den bedste struktur, BIC har kunnet finde ud fra læring med `dane`. Den er fundet ud fra den tomme startstruktur. Der blev også forsøgt 10 tilfældige startstrukturer (hvoraf nogle gav anledning til samme score som den tomme). Listen over scores er:

```
Random08 has score = -2666,5839114582
Random01 has score = -2666,5839114582
NoEdges has score = -2666,5839114582
Random05 has score = -2666,9946075468
Random07 has score = -2679,93599738167
Random00 has score = -2725,06616107034
Random03 has score = -2742,75958985793
Random09 has score = -2754,73657430106
Random06 has score = -2762,92981626986
Random02 has score = -2792,60022950058
Random04 has score = -2796,26856204679
```

Summen af sandsynlighedsmassen for de forskellige haplotyper under modellen er 0.221265418, hvorfor det uobserverede sandsynlighedsmasse er 0.778734582.

I figur 3.7 ses den bedste struktur, BIC har kunnet finde ud fra læring med `somali`. Den er fundet ud fra en tilfældig startstruktur, der kan ses i figur 3.6. Som før blev der forsøgt med den tomme graf som startstruktur samt 10 tilfældige. Listen over scores er:



Figur 3.5: Resultatet af BIC på `dane` med den tomme graf som startstruktur.

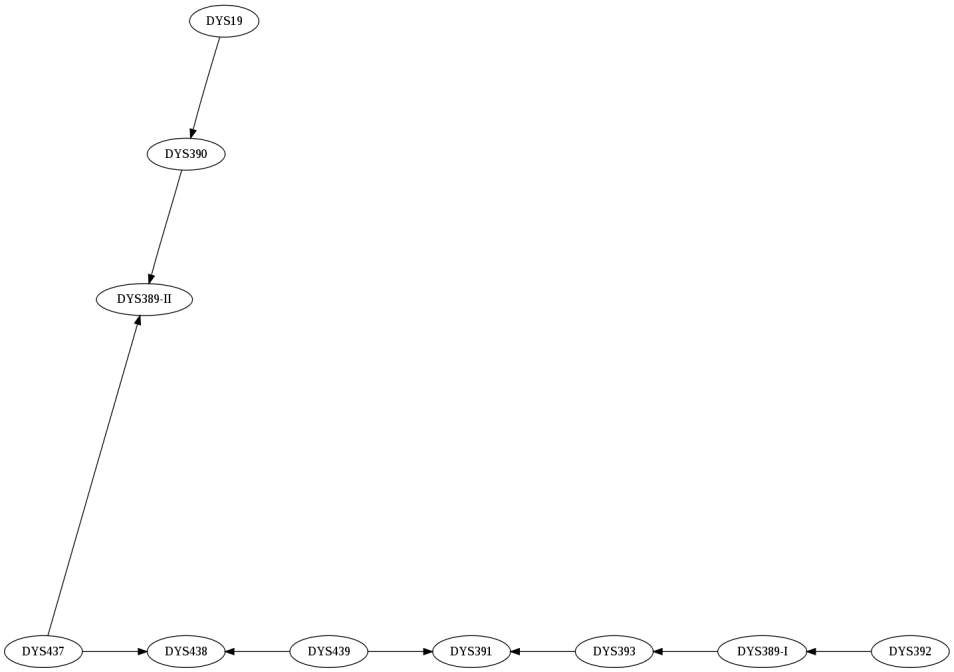
```
Random08 has score = -1848,45136159725
Random02 has score = -1850,36392877974
Random05 has score = -1857,54151356926
NoEdges has score = -1857,54151356926
Random04 has score = -1859,45408075176
Random03 has score = -1859,45408075176
Random09 has score = -1874,40120698307
Random06 has score = -1883,63094936141
Random07 has score = -1891,98694714971
Random01 has score = -1948,73756790447
Random00 has score = -1957,09674339864
```

Summen af sandsynlighedsmassen for de forskellige haplotyper under modellen er 0.57193463, hvorfor det uobserverede sandsynlighedsmasse er 0.428065368.

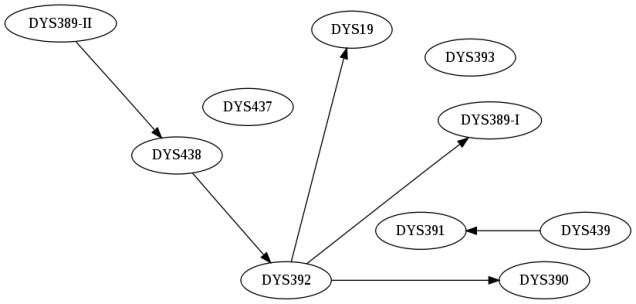
3.3.5 AIC-resultater

I figur 3.8 ses den bedste struktur, AIC har kunnet finde ud fra læring med `dane`. Den er fundet ud fra den tomme startstruktur. Der blev også forsøgt 10 tilfældige startstrukturer (hvoraf ingen gav anledning til samme score som den tomme). Listen over scores er:

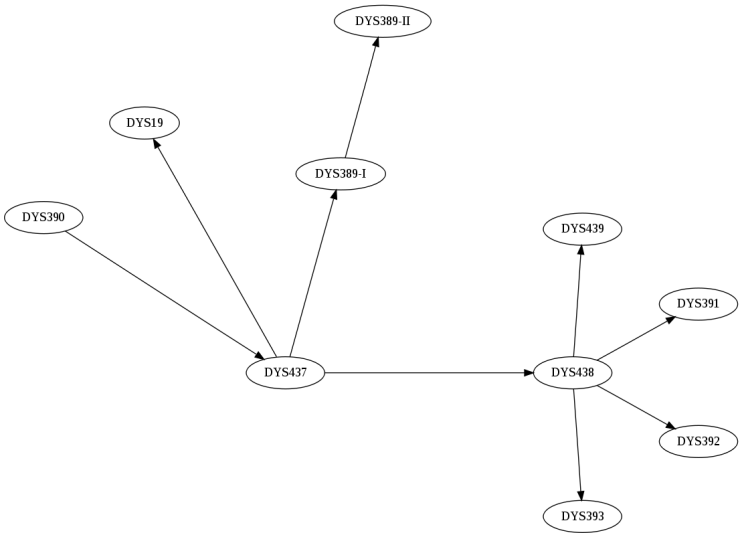
```
NoEdges has score = -2255,59675642683
Random00 has score = -2364,53504531898
Random01 has score = -2404,56730090413
Random05 has score = -2500,49165019914
Random02 has score = -2507,026382005
Random07 has score = -2561,16271340838
Random03 has score = -2586,6058436183
Random04 has score = -2601,46698010123
Random08 has score = -2613,92497316272
Random09 has score = -2654,24016580923
Random06 has score = -2682,88013978093
```



Figur 3.6: Den tilfældige startstruktur for BIC på `somali`. Resultatet kan ses i figur 3.7.



Figur 3.7: Resultatet af BIC på `somali` med startstrukturen i figur 3.6.



Figur 3.8: Resultatet af AIC på `dane` med den tomme graf som startstruktur.

Summen af sandsynlighedsmassen for de forskellige haplotyper under modellen er 0.23653702, hvorfor det uobserverede sandsynlighedsmasse er 0.763462977.

I figur 3.9 ses den bedste struktur, AIC har kunnet finde ud fra læring med `somali`. Den er fundet ud fra den tomme startstruktur. Der blev også forsøgt 10 tilfældige startstrukturer (hvoraf ingen gav anledning til samme score som den tomme). Listen over scores er:

```
NoEdges has score = -1499,82763498359
Random01 has score = -1531,94076235505
Random09 has score = -1588,83688953755
Random02 has score = -1639,41789034517
Random07 has score = -1698,01954025119
Random06 has score = -1700,24434214921
Random03 has score = -1703,72064023874
Random00 has score = -1707,45491419722
Random08 has score = -1708,9906814163
Random04 has score = -1739,89416525622
Random05 has score = -1786,93271008774
```

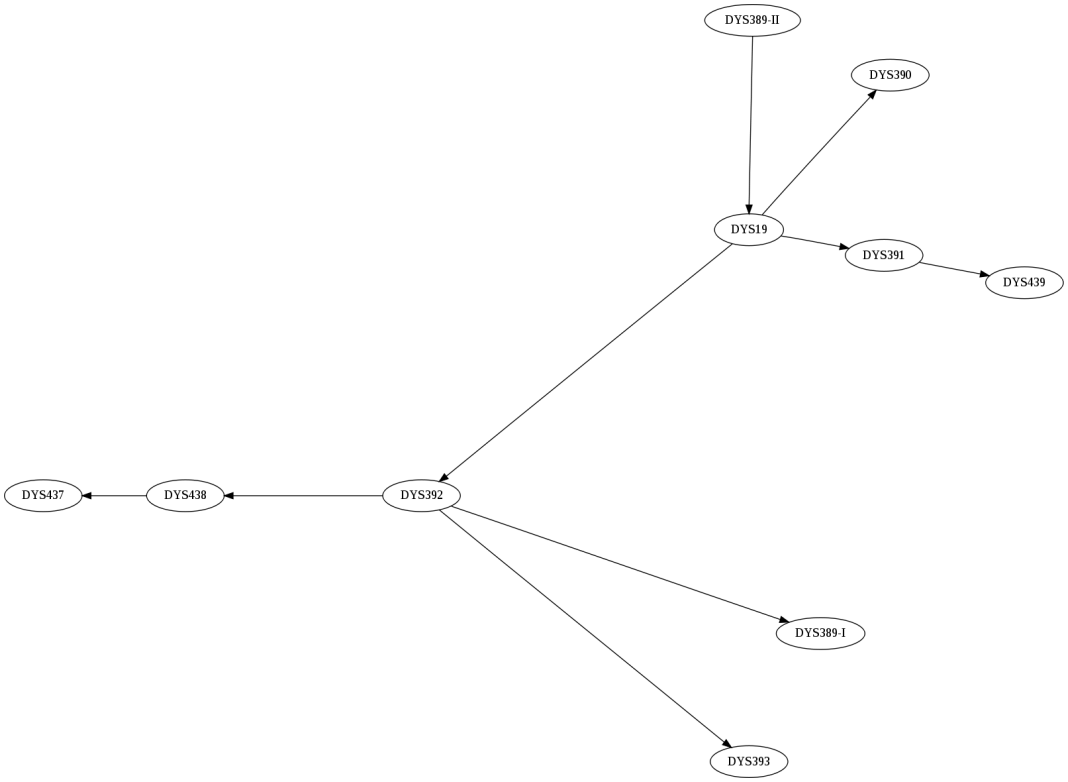
Summen af sandsynlighedsmassen for de forskellige haplotyper under modellen er 0.62915574, hvorfor det uobserverede sandsynlighedsmasse er 0.370844263.

3.3.6 NPC-resultater

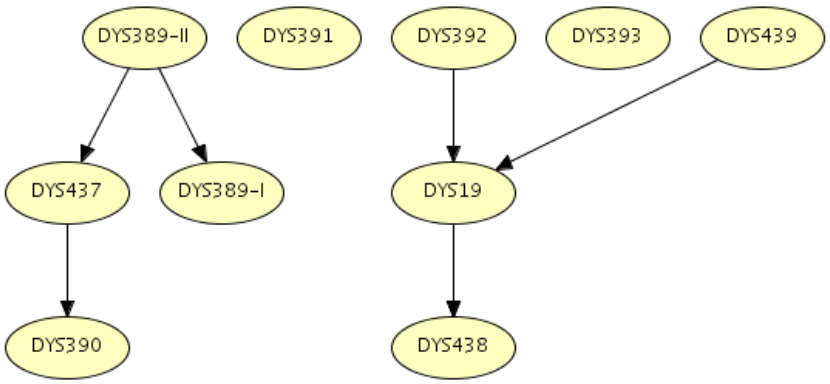
På grund af problematikken beskrevet i afsnit 3.1.2 medtages udelukkende en strukturlæring med NPC-algoritmen foretaget af programmet Hugin. For `dane` ses den grafiske model i figur 3.10 og for `somali` i figur 3.11. Da modellerne er for simple, arbejdes der ikke yderligere med dem.

3.3.7 Sammenligning af grafiske modeller

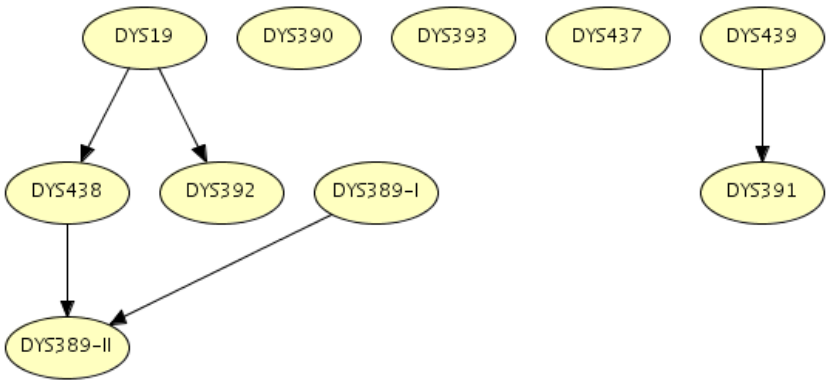
Jf. afsnit 3.3.1 kan man give et estimat for den uobserverede sandsynlighedsmasse. I tabel 3.2 ses en opsummering af, hvor meget uobserveret sandsynlighedsmasse de forskellige grafiske modeller predikterer. For en forklaring af 1., 2., og 3. ordens fysiske modeller, henvises til afsnit 3.2.



Figur 3.9: Resultatet af AIC på `somali` med den tomme graf som startstruktur.



Figur 3.10: Resultatet af NPC-algoritmen på `dane` foretaget af programmet Hugin.



Figur 3.11: Resultatet af NPC-algoritmen på `somali` foretaget af programmet Hugin.

	dane	somali
Teoretisk estimat	0.6021505	0.2772277
BIC	0.7787346	0.4280654
AIC	0.7634630	0.3708443
1. ordens fysisk model	0.9265484	0.5669779
2. ordens fysisk model	0.7098042	0.4901236
3. ordens fysisk model	0.5019729	0.4003680

Tabel 3.2: Tabel over uobserveret sandsynlighedsmasse fundet med den bedst passende model. Det teoretisk estimatet bygger på afsnit 3.3.1, der er baseret på [Robbins, 1968].

	dane	somali	dane*	somali*
BIC	9.70329	6.94897	-4.15120	4.80894
AIC	10.58565	4.22877	-3.78828	2.39532
1. ordens fysisk model	48.70806	18.50407	-0.37752	14.62555
2. ordens fysisk model	6.17682	9.72243	-4.46116	7.17756
3. ordens fysisk model	0.00645	5.83620	-5.51097	4.04633

Tabel 3.3: Tabel over Brier-scores ved de forskellige grafiske modeller. De to kolonner længst til højre markeret med * er den korrigerede Brier-score (se afsnit 3.3.3).

I afsnit 3.3.2 blev Brier-scoren introduceret. I tabel 3.3 ses Brier-scoren for de forskellige grafiske modeller. Det ses, at den korrigerede Brier-score – specielt for `dane` – opfører sig markant anderledes end den ukorrigerede. For en forklaring af 1., 2., og 3. ordens fysiske modeller, henvises til afsnit 3.2.

I alt må det siges, at der ikke er fundet en god model.

Kapitel 4

Kollapsning

Kontingenstabellerne er meget tynde, dvs. der er mange nuller. Det giver flere problemer. Eksempelvis kan man ikke udnytte, at deviansen approksimativt er χ^2-fordelt, da det er alt for upræcist i så tynde tabeller. For at få større styrke, kan man forsøge at kollapse niveauer og på den måde få tættere kontingenstabeller.

For at kollapse to niveauer X_1 og X_2, vil man for enhver Y af de andre variable undersøge, om $P(Y|X_1) = P(Y|X_2)$ svarende til uafhængighed i en $2 \times k$-kontingenstabel. For $S+1$ variable i alt får man dermed kontingenstabellerne $2 \times c_1, 2 \times c_2, \ldots, 2 \times c_S$, der alle skal testet for uafhængighed mellem X_1 og X_2. Hvis X_1 og X_2 findes at være uafhængige i alle tabellerne, kan de kollapses.

Normalt ville man nok anvende Pearsons approksimative χ^2-test på deviansen, men pga. de tynde kontingenstabeller, er det ikke muligt. Derfor vælges i stedet eksakte test, der ikke afhænger af om kontingenstabellerne er tynde eller tætte. Man kan teste for uafhængighed i kontingenstabeller på flere måder: i det aktuelle tilfælde med $2 \times c_s$-tabeller (generelt $r \times c$-tabeller) kan man eksempelvis anvende Fishers eksakte test eller Kruskall og Goodman γ for ordinale data.

For at gennemgå udregningen af Kruskall og Goodman γ, betragtes kontingenstabellen

	Y_1	Y_2	$\cdots$	Y_s
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}

For denne $2 \times s$-kontingenstabel udregnes

$$\hat{\gamma} = \begin{cases} \frac{C-D}{C+D} & \text{for } C+D \neq 0 \\ 0 & \text{ellers} \end{cases},$$

hvor C er de konkordante par givet ved

$$\begin{aligned} C &= n_{22}n_{11} + n_{23}\left(n_{11}+n_{12}\right) + \cdots + n_{2s}\left(n_{11}+n_{12}+\cdots+n_{1(s-1)}\right) \\ &= \sum_{j=2}^{s} n_{2j} \sum_{k=1}^{j-1} n_{1k}, \end{aligned}$$

og tilsvarende er D de diskonkordante par givet ved

$$\begin{aligned} D &= n_{12}n_{21} + n_{13}\left(n_{21}+n_{22}\right) + \cdots + n_{1s}\left(n_{21}+n_{22}+\cdots+n_{2(s-1)}\right) \\ &= \sum_{j=2}^{s} n_{1j} \sum_{k=1}^{j-1} n_{2k}. \end{aligned}$$

Per definition vil $\gamma \in [-1, 1]$. Hvis der er uafhængighed, vil $\gamma = 0$. Ved positiv korrelation vil $\gamma > 0$ og for negativ korrelation vil $\gamma < 0$.

For at teste for uafhængighed, gøres således:

Algoritme 1 Test for uafhængighed vha. Kruskall og Goodmans γ

Require: t {t is the contingency table in question}
1: $N \leftarrow 1000$ {random contingency tables to generate}
2: $\alpha \leftarrow 0.05$ {significance level}
3: $e \leftarrow 0$ {number of tables with a more extreme γ-value than t}
4: **for** $i = 1$ to N **do**
5: $t' \leftarrow$ randtable(t) {generates a random contingency table with the same size and marginals as t}
6: **if** $|\gamma(t')| > |\gamma(t)|$ **then**
7: $e \leftarrow e + 1$
8: **end if**
9: **end for**
10: $p \leftarrow e/N$
11: **if** $p \geq \alpha$ **then**
12: **return true**{indepence}
13: **else**
14: **return false**{depence}
15: **end if**

Bemærk at funktionen `randtable` ikke er nærmere specificeret. Det er fordi, at den kan laves på flere forkellige måder. Eksempelvis kan Patefields algoritme anvendes til at sample tilfældige kontingenstabeller med faste marginaler (i `R` er Patefields algoritme implementeret vha. funktionen `r2dtable`).

4.1 Metode

Såvel rækkefølgen af variable, der gennemsøges, såvel som variablenes niveauer, der undersøges for kollaps, er tilfældigt. Så snart der er fundet et kollaps startes rekursivt at kollapse på det nye datasæt, hvor niveauerne er blevet kollapset. Der foretages altså ikke flere kollaps samtidigt. Som argumentation for det i tilfældet med Fishers eksakte test, betragt da kontigenstabellen

$a_{1,1}$	$\cdots$	$a_{1,n}$	R_1
$\vdots$	$\ddots$	$\vdots$	$\vdots$
$a_{m,1}$	$\cdots$	$a_{m,n}$	R_m
C_1	$\cdots$	C_n	

Under nulhypotesen om at der er uafhængighed vil celle-counts følge en generaliseret hypergeometrisk fordeling , og dermed er sandsynligheden for en kontingenstabel med de givne marginaler givet ved

$$P_{\text{cutoff}} = \frac{R_1! \cdots R_m! C_1! \cdots C_n!}{\sum_i R_i \prod_i \prod_j a_{i,j}!}.$$

I det konkrete tilfælde skal det tjekkes, om $P^{(1)}_{\text{cutoff}}$ for

$a_{1,1}$	$\cdots$	$a_{1,p}$	$a_{1,p+1}$	$\cdots$	$a_{1,n}$	R_1
$a_{2,1}$	$\cdots$	$a_{2,p}$	$a_{2,p+1}$	$\cdots$	$a_{2,n}$	R_2
C_1	$\cdots$	C_p	C_{p+1}	$\cdots$	C_n	

	Fishers eksakte test		Kruskall og Goodmans γ	
Datasæt	Niveauer kollapset	Reduktionsprocent	Niveauer kollapset	Reduktionsprocent
`dane`	12	91.96429 %	4	60.625 %
`somali`	11	93.54273 %	0	0 %

Tabel 4.1: Resultatet af den optimale kollapsning (på grundlag af 100 tilfældige rækkefølger for tests). I testet for Kruskall og Goodmans γ er der genereret 1000 tabeller – ved generering af 10000 tabeller fås samme resultater. Reduktionsprocenten er hvor mange procent antallet af mulige haplotyper er reduceret med i forhold til inden kollapsningen. I `dane` var kun ét af kollapsene det samme ved Fishers eksakte test som ved Kruskall og Goodmans γ.

er det samme som $P^{(2)}_{\text{cutoff}}$ for

$a_{1,1}$	$\cdots$	$a_{1,p} + a_{1,p+1}$	$\cdots$	$a_{1,n}$	R_1
$a_{2,1}$	$\cdots$	$a_{2,p} + a_{2,p+1}$	$\cdots$	$a_{2,n}$	R_2
C_1	$\cdots$	$C_p + C_{p+1}$	$\cdots$	C_n	

Bemærk at man, da det ikke er en rangtest, vha. ombytning af kolonnerne altid kan få de to niveauer, der skal kollapses til at hedde p og $p+1$. Dermed er

$$P^{(1)}_{\text{cutoff}} = \frac{R_1!R_2!C_1!\cdots C_n!}{(R_1+R_2)\,a_{1,1}!\cdots a_{1,n}!a_{2,1}!\cdots a_{2,n}!}$$
$$P^{(2)}_{\text{cutoff}} = \frac{R_1!R_2!C_1!\cdots (C_p+C_{p+1})!\cdots C_n!}{(R_1+R_2)\,a_{1,1}!\cdots (a_{1,p}+a_{1,p+1})!\cdots a_{1,n}!a_{2,1}!\cdots (a_{2,p}+a_{2,p+1})!\cdots a_{2,n}!}$$

Antag modsætningsvist, at $P^{(1)}_{\text{cutoff}} = P^{(2)}_{\text{cutoff}}$. I så fald skulle der gælde, at

$$\frac{C_1!\cdots C_n!}{a_{1,1}!\cdots a_{1,n}!a_{2,1}!\cdots a_{2,n}!} = \frac{C_1!\cdots (C_p+C_{p+1})!\cdots C_n!}{a_{1,1}!\cdots (a_{1,p}+a_{1,p+1})!\cdots a_{1,n}!a_{2,1}!\cdots (a_{2,p}+a_{2,p+1})!\cdots a_{2,n}!},$$

hvilket ville føre til, at

$$\frac{C_p!C_{p+1}!}{a_{1,p}!a_{1,p+1}!a_{2,p}!a_{2,p+1}!} = \frac{(C_p+C_{p+1})!}{(a_{1,p}+a_{1,p+1})!\,(a_{2,p}+a_{2,p+1})!},$$

hvilket generelt ikke gælder. Dermed kan man ikke generelt kollapse flere niveauer i samme kørsel. Dog kan man bemærke, at da kollapsning oftest giver større styrke, vil to niveauer, der inden kollaps er signifikant afhængige, som oftest også være signifikant afhængige efter kollaps, da de ikke-signifikante niveauer er blevet kollapset. Denne bemærkning bliver dog ikke benyttet i den aktuelle metode: her køres test for kollaps forfra vha. rekursion så snart et kollaps er foretaget. `R`-kildekoden til kollapsningen findes i `cluster-*.R`-filerne på http://people.math.aau.dk/~mikl/mat4/projekt.

4.2 Resultater

I tabel 4.1 ses opsummering af resultaterne for den optimale kollapsning.

Datasættet `dane` kollapses på 3 forskellige måder, når det køres 100 tests (hvor den eneste forskel er hvilken rækkefølge, kollapsene er blevet forsøgt fortaget) for kollaps med Fisher eksakte test. Hvilke niveauer der kollapses afhænger altså af den rækkefølge, de forsøges kollapset i. De tre repræsentanter for de forskellige kollaps giver 87.14286 % reduktion, 90.35714 % reduktion og 91.96429 % reduktion i antallet af mulige haplotyper. Forskellen er fx at nogle steder bliver DYS439 kollapset til 3 niveauer i stedet for 4 på grund af rækkefølgen, de er undersøgt i. Tilsvarende bliver DYS389II kollapset til enten 5 eller 6 niveauer.

Den maksimale reduktion er altså 91.96429 %, hvilket må siges at være en væsentlig reduktion. Niveauerne kollapses således (udskrift fra `R`-scriptet – allelerne er skaleret op med en faktor 10 for kun at skulle arbejde med heltal; eksempelvis findes allelet 10.2):

```
DYS19: 160 and 170 is independent so 170 has been collapsed to 160
DYS389II: 300 and 330 is independent so 330 has been collapsed to 300
DYS389II: 270 and 340 is independent so 340 has been collapsed to 270
DYS389II: 300 and 320 is independent so 320 has been collapsed to 300
DYS390: 230 and 210 is independent so 210 has been collapsed to 230
DYS391: 90 and 100 is independent so 100 has been collapsed to 90
DYS392: 120 and 160 is independent so 160 has been collapsed to 120
DYS392: 110 and 102 is independent so 102 has been collapsed to 110
DYS392: 150 and 140 is independent so 140 has been collapsed to 150
DYS393: 140 and 150 is independent so 150 has been collapsed to 140
DYS439: 100 and 140 is independent so 140 has been collapsed to 100
DYS439: 120 and 130 is independent so 130 has been collapsed to 120
```

Datasættet `somali` blev tilsvarende kollapset, hvilket resulterede i en reduktion i antallet af mulige haplotyper på 93.54273 % ved hjælp af Fishers eksakte test:

```
DYS19: 140 and 170 is independent so 170 has been collapsed to 140
DYS389I: 120 and 110 is independent so 110 has been collapsed to 120
DYS389II: 260 and 290 is independent so 290 has been collapsed to 260
DYS389II: 310 and 330 is independent so 330 has been collapsed to 310
DYS390: 220 and 230 is independent so 230 has been collapsed to 220
DYS390: 210 and 260 is independent so 260 has been collapsed to 210
DYS390: 240 and 250 is independent so 250 has been collapsed to 240
DYS393: 140 and 150 is independent so 150 has been collapsed to 140
DYS437: 160 and 150 is independent so 150 has been collapsed to 160
DYS438: 100 and 120 is independent so 120 has been collapsed to 100
DYS439: 130 and 100 is independent so 100 has been collapsed to 130
```

For Kruskall og Goodmans γ ved kørsel af 100 tests blev `somali` slet ikke reduceret, og `dane` blev ved reduceret med 60.625 % således:

```
DYS391: 120 and 110 is independent so 110 has been collapsed to 120
DYS393: 120 and 140 is independent so 140 has been collapsed to 120
DYS389II: 270 and 280 is independent so 280 has been collapsed to 270
DYS439: 120 and 130 is independent so 130 has been collapsed to 120
```

Her ses, at i DYS393 er to ikke-naboer blevet kollapset. Der kan argumenteres for, at ved anvendelse af Kruskall og Goodmans γ, må kun naboer kollapses (ellers vil der blive problemer med, hvordan ordinaliteten i data bibeholdes).

4.3 Grafiske modeller ud fra kollapsede datasæt

Hvis man laver en grafisk model til udregning af sandsynligheder i et kollapset datasæt, faktoriserer den simultane sandsynlighed ikke helt som sædvanligt. Man er nødt til at kompensere for de kollapsede niveauer ved at gange passende sandsynligheder på. Hvis eksempelvis a_1 og a_2 er to niveauer, der er blevet kollapset, så består kompensationen i, at den simultane sandsynlighed faktoriserer til

$$P(A = a_1, B = b, \ldots) = \underbrace{P(A = a_1 | A = a_1 \cup a_2)}_{\text{I oprindeligt datasæt}} \underbrace{P(A = a_1 \cup a_2, B = b, \ldots)}_{\text{I kollapset datasæt}},$$

hvor $A = a_1 \cup a_2$ er hændelsen $A \in \{a_1, a_2\}$.

	dane	somali
Teoretisk estimat	0.6021505	0.2772277
1. ordens fysisk model	0.9686405	0.2488025
2. ordens fysisk model	0.8825012	0.1364053
3. ordens fysisk model	0.8035638	0.0218503

Tabel 4.2: Tabel over uobserveret sandsynlighedsmasse for de maksimalt kollapsede datasæt, hvor kollapsningen er foretaget vha. Fishers eksakte test. Tabel over uobserveret sandsynlighedsmasse for de ukollapsede datasæt ses i tabel 3.2. Det teoretisk estimatet bygger på afsnit 3.3.1, der er baseret på [Robbins, 1968].

	dane
Teoretisk estimat	0.602151
1. ordens fysisk model	0.986097
2. ordens fysisk model	0.940390
3. ordens fysisk model	0.874453

Tabel 4.3: Tabel over uobserveret sandsynlighedsmasse for de maksimalt kollapsede datasæt, hvor kollapsningen er foretaget vha. Kruskall og Goodmans γ. Datasættet `somali` er ikke medtaget, da det ikke gav anledning til kollaps. Tabel over uobserveret sandsynlighedsmasse for de ukollapsede datasæt ses i tabel 3.2. Det teoretisk estimatet bygger på afsnit 3.3.1, der er baseret på [Robbins, 1968].

I stedet for at implementere understøttelse af kollapsning i `SL` (se beskrivelse i afsnit 3.1), er det i stedet implementeret fra bunden i `R` for DAGs. Både fordi datastrukturen, der registrerer hvilke alleler, der er blevet kollapset allerede var internt i `R` efter kollapsningen var foretaget (den bliver konstrueret netop til formålet at kunne udregne kompensationen); Fishers eksakte test og Patefields algoritme var implementeret i `R`, fordi at `SL` er kodet i C# og ingen af de API'er der findes til `R` kan bruges fra C#, samt at det også gav mulighed for at teste udregningerne i `SL`.

Det er implementeret på den måde, at man angiver en liste med en længde svarende til antallet af knuder, og hver indgang i listen indeholder en vektor bestående af indekser på den aktuelle knudes forældre.

Dette er brugt til at udregne dels summen af den dækkede sandsynlighedsmasse samt Brier-scoren i de maksimalt kollapsede datasæt. Når man skal udregne Brier-scoren for de kollapsede datasæt, foregår det som normalt: sandsynligheden er den kompenserede simultane sandsynlighed og den relative frekvens for en haplotype er den samme som i det ikke-kollapsede datasæt.

Ved brug af Fishers eksakte test ses den estimerede uobserverede sandsynlighedsmasse i tabel 4.2 og Brier-scores ses i tabel 4.4. Tilsvarende for Kruskall og Goodmans γ ses i henholdsvis tabel 4.3 og 4.5.

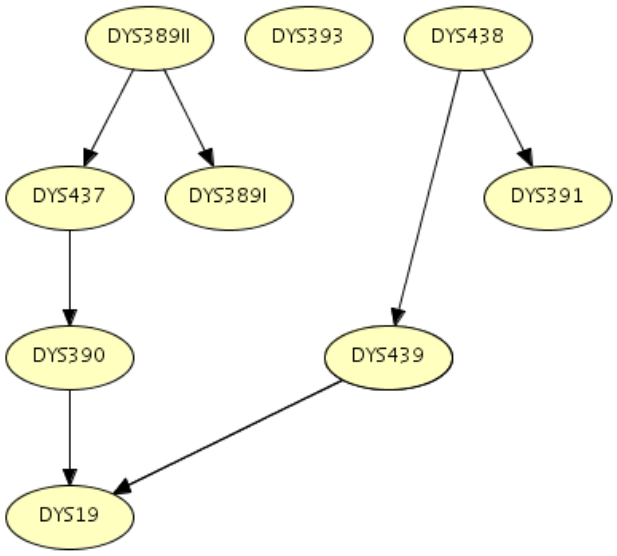
På samme måde som i resultaterne for de ukollapsede datasæt (se afsnit 3.3) ses det, at den korrigerede Brier-score opfører sig markant anderledes end den ukorrigerede på `dane`, hvorimod de ser ens ud på `somali`.

	dane	somali	dane*	somali*
1. ordens fysisk model	99.51564	28.46404	1.833620	25.52843
2. ordens fysisk model	30.30641	26.35433	1.033328	24.01145
3. ordens fysisk model	23.00919	26.50815	0.439696	24.42550

Tabel 4.4: Tabel over Brier-scores for de maksimalt kollapsede datasæt, hvor kollapsningen er foretaget vha. Fishers eksakte test. Tabel over Brier-scores for de ukollapsede datasæt ses i tabel 3.3. De to kolonner længst til højre markeret med * er den korrigerede Brier-score (se afsnit 3.3.3).

	dane	dane*
1. ordens fysisk model	223.6260	2.178214
2. ordens fysisk model	78.6837	1.311818
3. ordens fysisk model	39.4742	0.768210

Tabel 4.5: Tabel over Brier-scores for de maksimalt kollapsede datasæt, hvor kollapsningen er foretaget vha. Kruskall og Goodmans γ. Datasættet `somali` er ikke medtaget, da det ikke gav anledning til kollaps. Tabel over Brier-scores for de ukollapsede datasæt ses i tabel 3.3. Kolonnen længst til højre markeret med * er den korrigerede Brier-score (se afsnit 3.3.3).



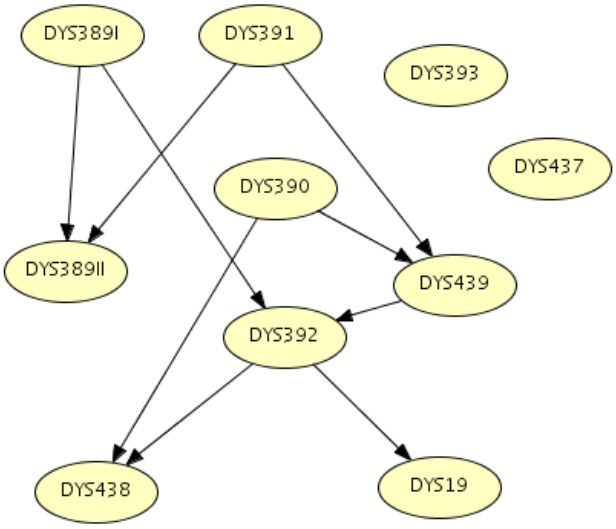
Figur 4.1: Resultatet af NPC-algoritmen på det maksimalt kollapsede `dane` foretaget af programmet Hugin.

4.3.1 Automatisk strukturlæring

En automatisk strukturlæring på de kollapsede datasæt skulle gerne resultere i en stærkere afhængighedsstruktur og dermed en graf med flere kanter. I dette afsnit vises automatisk strukturlæring med NPC vha. Hugin på datasættet kollapset med Fishers eksakte test. Funktionaliteten i `cluster*.R`-filerne vil kunne udvides på en sådan måde, at man ved hjælp af samme funktioner som bruges i udregning af dækket sandsynlighedsmasse og vektorerne til udregning af Brier-scoren ved modellerne for locis fysiske placering. Her er der dog udelukkende medtaget resultatet af NPC.

I figur 4.1 ses resultatet af NPC-algoritmen lavet ved hjælp af Hugin på `dane` (sammenlign med figur 3.10). Strukturmæssigt ligner det en form for 1. ordens model, men afhængighederne er ikke i overensstemmelse med 1. ordens modellen baseret på locis fysiske placering.

I figur 4.2 ses resultatet af NPC-algoritmen lavet ved hjælp af Hugin på `somali` (sammenlign med figur 3.11). Strukturmæssigt ligner det en form for 3. ordens model, hvis grafen moraliseres, men afhængighederne er ikke i overensstemmelse med 3. ordens modellen baseret på locis fysiske placering.



Figur 4.2: Resultatet af NPC-algoritmen på det maksimalt kollapsede `somali` foretaget af programmet Hugin.

Kapitel 5

Importance sampling

I dette kapitel kigges nærmere på metoden beskrevet i [Mehta et al., 1988], der handler om, hvordan man kan estimere eksakte p-værdier for $2 \times k$-kontingenstabeller. I første afsnit udledes formler, der ikke er udledt i artiklen, og i afsnittet herefter anvendes metoden på Fishers eksakte test, og der sammenlignes med Rs implementation i form af `fisher.test`, der anvender Patefields algoritme [Patefield, 1981]. En implementation af den beskrevne importance sampling i R kan også findes på http://people.math.aau.dk/~mikl/mat4/ projekt i `impsamp.R`-filen.

For at introducere notationen, lad $\boldsymbol{x} = (x_1, \ldots, x_k)$ være den første række i en $2 \times k$- kontingenstabel. For faste marginaler $m = \sum_{i=1}^{k} x_i$ og søjlesummerne $n_1, n_2, \ldots, n_k$ identificerer $\boldsymbol{x}$ derfor entydigt en kontingenstabel. Som hjælpestørrelse sættes $N = \sum_{i=1}^{k} n_i$ og

$$s_j = \begin{cases} \sum_{i=1}^{j} x_i & j \geq 1 \\ 0 & j = 0 \end{cases}$$

og som i tilfældet med s_j, lad der generelt gælde, at $\sum_{i=x}^{j} f(i) = 0$, når $j < x$.

Idéen i [Mehta et al., 1988] er at opbygge et netværk, hvor en vej fra starttilstanden $(0, 0)$ gennem (j, s_j) for $j = 1, 2, \ldots, k-1$ til sluttilstanden (k, m) er en kontingenstabel, og vejens sandsynlighed er kontingenstabellens unormerede sandsynlighed og vejens længde er værdien af teststørrelsen. I skridt j vil der således være mange mulige tilstande (j, s_{j1}), (j, s_{j2}), …. Se en mere detaljeret beskrivelse i [Mehta et al., 1988, afsnit 3.1]. Hvis $N_j = \sum_{i=1}^{j} n_i$, så kan s_j variere mellem $\max\left(0, N_j + m - N\right)$ og $\min\left(m, N_j\right)$.

Det er nyttigt at give mængden af mulige "børn" til en tilstand (j, s_j) betegnelsen

$$\mathbf{R}(j, s_j) = \left\{ (j+1, w) : \max\left(s_j, m - \sum_{i=j+2}^{k} n_i\right) \leq w \leq \min\left(m, s_j + n_{j+1}\right) \right\}.$$

Lad $\Gamma(j, s_j)$ betegne mængden af alle veje fra tilstanden (j, s_j) til sluttilstanden (k, m), og lad $\boldsymbol{x}_j = (x_{j+1}, \ldots, x_k)$. Inden der introduceres de essentielle funktioner, nævnes først nogle nødvendige antagelser.

Der er få generelle antagelser i [Mehta et al., 1988], bl.a. at teststørrelsen $t(\boldsymbol{x})$ skal kunne skrives på formen

$$t\left(\boldsymbol{x}\right) = \sum_{i=1}^{k} a_i(x_i)$$

og at sandsynligheden $h(\boldsymbol{x})$ for en kontingenstabel $\boldsymbol{x}$ kan skrives på formen

$$h(\boldsymbol{x}) = C^{-1} \prod_{i=1}^{k} \lambda_i(s_{i-1}, x_i)$$

for $\lambda_i : \mathbb{Z} \times \mathbb{Z} \to [0,1]$ og normaliseringskonstanten C.

Dermed kan følgende funktioner defineres:

$$
\begin{aligned}
l(j, s_j) &= \min_{\boldsymbol{x}_j \in \Gamma(j,s_j)} \left\{ \sum_{i=j+1}^{k} a_i(x_i) \right\}, \\
u(j, s_j) &= \max_{\boldsymbol{x}_j \in \Gamma(j,s_j)} \left\{ \sum_{i=j+1}^{k} a_i(x_i) \right\}, \\
q(j, s_j) &= \sum_{\boldsymbol{x}_j \in \Gamma(j,s_j)} \prod_{i=j+1}^{k} \lambda_i(s_{i-1}, x_i), \\
\mu(j, s_j) &= \mathbf{E}\left[\sum_{i=j+1}^{k} a_i(x_i) \right] \quad \text{og} \\
\sigma^2(j, s_j) &= \mathbf{Var}\left[\sum_{i=j+1}^{k} a_i(x_i) \right].
\end{aligned}
$$

Udregningen af disse vil i almindelighed være besværlig, men ved at lave baglæns induktion, kan man starte i tilstand (k, m), og derefter bevæge sig tilbage i netværket mens funktionsværdierne udregnes. For at kunne gøre det, er det naturligvis et krav at kunne omskrive funktionerne til rekursionsform. I [Mehta et al., 1988] introduceres kun rekursionsformlen for $l(j, s_j)$, så i første afsnit udledes rekursionsformlerne for de andre funktioner. Dernæst gives et konkret eksempel på, hvordan man kan anvende metoden i forbindelse med Fishers eksakte test; denne konkretisering består i at specificere a_i- og λ_i-funktionerne.

5.1 Rekursionsformler

For at udlede rekursionsformlerne bemærkes det, at der grundet konstruktionen af netværket gælder, at

$$
\sum_{\boldsymbol{x}_j \in \Gamma(j,s_j)} f_j(\boldsymbol{x}_j) = \sum_{(j+1,w) \in \mathbf{R}(j,s_j)} \sum_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} f_j(w - s_j, \boldsymbol{x}_{j+1}),
$$

hvor Σ også kan erstattes med min og max. I [Mehta et al., 1988] fås rekursionsformlen

$$
l(j, s_j) = \min_{(j+1,w) \in \mathbf{R}(j,s_j)} \left\{ a_{j+1}(w - s_j) + l(j+1, w) \right\}.
$$

Indfør hjælpestørrelsen

$$
g\left(\boldsymbol{x}_j\right) = \sum_{i=j+1}^{k} a_i(x_i).
$$

Rekursionsformlerne for $u(j, s_j)$ og $q(j, s_j)$ kan så udtrykkes analogt med den for $l(j, s_j)$,

således at

$$\begin{aligned}
u(j, s_j) &= \max_{\boldsymbol{x}_j \in \Gamma(j,s_j)} \left\{ \sum_{i=j+1}^{k} a_i(x_i) \right\} \\
&= \max_{\boldsymbol{x}_j \in \Gamma(j,s_j)} g\left(\boldsymbol{x}_j\right) \\
&= \max_{(j+1,w) \in \mathbf{R}(j,s_j)} \max_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} g(w - s_j, \boldsymbol{x}_{j+1}) \\
&= \max_{(j+1,w) \in \mathbf{R}(j,s_j)} \max_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \left(a_j(w - s_j) + \sum_{i=j+2}^{k} a_i(x_i) \right) \\
&= \max_{(j+1,w) \in \mathbf{R}(j,s_j)} \left\{ a_j(w - s_j) + \max_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \sum_{i=j+2}^{k} a_i(x_i) \right\} \\
&= \max_{(j+1,w) \in \mathbf{R}(j,s_j)} \left\{ a_{j+1}(w - s_j) + u(j+1, w) \right\}
\end{aligned}$$

og

$$\begin{aligned}
q(j, s_j) &= \sum_{\boldsymbol{x}_j \in \Gamma(j,s_j)} \prod_{i=j+1}^{k} \lambda_i(s_{i-1}, x_i) \\
&= \sum_{(j+1,w) \in \mathbf{R}(j,s_j)} \sum_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \lambda_{j+1}(w, w - s_j) \prod_{i=j+2}^{k} \lambda_i(s_{i-1}, x_i) \\
&= \sum_{(j+1,w) \in \mathbf{R}(j,s_j)} \lambda_{j+1}(w, w - s_j) \sum_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1}, x_i) \\
&= \sum_{(j+1,w) \in \mathbf{R}(j,s_j)} \lambda_{j+1}\left(w, w - s_j\right) q(j+1, w).
\end{aligned}$$

For at udlede rekursionsformlerne for resten af funktionerne, indføres først en smule notation. Jf. [Mehta et al., 1988] er sandsynligheden for $\boldsymbol{x}_j$ givet ved

$$P\left(\boldsymbol{x}_j\right) = \frac{1}{q(j, s_j)} \prod_{i=j+1}^{k} \lambda_i(s_{i-1}, x_i).$$

Dette kan nu bruges til at udlede rekursionsformlerne. Først for middelværdien, der såle-

des bliver

$$
\begin{aligned}
\mu(j,s_j) =& \mathbf{E}\left[\sum_{i=j+1}^{k} a_i(x_i)\right] = \mathbf{E}\left[g\left(\boldsymbol{x}_j\right)\right] = \sum_{\boldsymbol{x}_j \in \Gamma(j,s_j)} g\left(\boldsymbol{x}_j\right) P\left(\boldsymbol{x}_j\right) \\
=& \sum_{\boldsymbol{x}_j \in \Gamma(j,s_j)} \sum_{i=j+1}^{k} a_i(x_i) \left(\frac{1}{q(j,s_j)} \prod_{i=j+1}^{k} \lambda_i(s_{i-1},x_i)\right) \\
=& \sum_{(j+1,w)\in \mathbf{R}(j,s_j)} \sum_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \left(a_{j+1}(w-s_j) + \sum_{i=j+2}^{k} a_i(x_i)\right) \times \\
& \frac{1}{q(j,s_j)} \lambda_{j+1}(w, w-s_j) \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in \mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j) \times \\
& \sum_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \left(a_{j+1}(w-s_j) + \sum_{i=j+2}^{k} a_i(x_i)\right) \frac{1}{q(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in \mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j) \times \\
& \sum_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \sum_{i=j+2}^{k} a_i(x_i) \left(\frac{1}{q(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i)\right) + \\
& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in \mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j) a_{j+1}(w-s_j) \times \\
& \sum_{\boldsymbol{x}_{j+1} \in \Gamma(j+1,w)} \left(\frac{1}{q(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i)\right) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in \mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\mu(j+1,w) + \\
& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in \mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j) a_{j+1}(w-s_j) \frac{1}{q(j+1,w)} q(j+1,w) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in \mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\left(\mu(j+1,w) + a_{j+1}(w-s_j)\right)
\end{aligned}
$$

For at finde rekursionsformlen for variansen, findes først rekursionsformlen for

$$
\begin{aligned}
\nu(j,s_j) =& \mathbf{E}\left[\left(\sum_{i=j+1}^{k} a_i(x_i)\right)^2\right] = \mathbf{E}\left[\left(g\left(\boldsymbol{x}_j\right)\right)^2\right] = \sum_{\boldsymbol{x}_j\in\Gamma(j,s_j)} \left(g\left(\boldsymbol{x}_j\right)\right)^2 P\left(\boldsymbol{x}_j\right) \\
=& \sum_{\boldsymbol{x}_j\in\Gamma(j,s_j)} \left(\sum_{i=j+1}^{k} a_i(x_i)\right)^2 \left(\frac{1}{q(j,s_j)} \prod_{i=j+1}^{k} \lambda_i(s_{i-1},x_i)\right) \\
=& \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} \sum_{\boldsymbol{x}_{j+1}\in\Gamma(j+1,w)} \frac{1}{q(j,s_j)} \lambda_{j+1}(w,w-s_j) \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i)\times \\
& \left(a_{j+1}(w-s_j)+\sum_{i=j+2}^{k} a_i(x_i)\right)\left(a_{j+1}(w-s_j)+\sum_{i=j+2}^{k} a_i(x_i)\right) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\times \\
& \sum_{\boldsymbol{x}_{j+1}\in\Gamma(j+1,w)} \frac{1}{q(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i)\times \\
& \left(a_{j+1}(w-s_j)+\sum_{i=j+2}^{k} a_i(x_i)\right)\left(a_{j+1}(w-s_j)+\sum_{i=j+2}^{k} a_i(x_i)\right) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\times \\
& \sum_{\boldsymbol{x}_{j+1}\in\Gamma(j+1,w)} \left(\sum_{i=j+2}^{k} a_i(x_i)\right)^2 \left(\frac{1}{q(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i)\right) + \\
& 2\frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)a_{j+1}(w-s_j)\times \\
& \sum_{\boldsymbol{x}_{j+1}\in\Gamma(j+1,w)} \sum_{i=j+2}^{k} a_i(x_i) \left(\frac{1}{q(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i)\right) + \\
& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\left(a_{j+1}(w-s_j)\right)^2 \times \\
& \sum_{\boldsymbol{x}_{j+1}\in\Gamma(j+1,w)} \left(\frac{1}{q(j+1,w)} \prod_{i=j+2}^{k} \lambda_i(s_{i-1},x_i)\right) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\nu(j+1,w)+ \\
& 2\frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)a_{j+1}(w-s_j)\mu(j+1,w)+ \\
& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\left(a_{j+1}(w-s_j)\right)^2 \frac{1}{q(j+1,w)} q(j+1,w) \\
=& \frac{1}{q(j,s_j)} \sum_{(j+1,w)\in\mathbf{R}(j,s_j)} q(j+1,w)\lambda_{j+1}(w,w-s_j)\times \\
& \left(\nu(j+1,w)+2a_{j+1}(w-s_j)\mu(j+1,w)+\left(a_{j+1}(w-s_j)\right)^2\right)
\end{aligned}
$$

Dermed fås, at

$$\begin{aligned}\sigma^2(j,s_j) &= \mathbf{E}\left[\left(g\left(\boldsymbol{x}_j\right)\right)^2\right] - \left(\mathbf{E}\left[g\left(\boldsymbol{x}_j\right)\right]\right)^2 \\ &= \nu(j,s_j) - \left(\mu(j,s_j)\right)^2,\end{aligned}$$

hvorfor den introducerede $\nu(j,s_j) = \mathbf{E}\left[\left(\sum_{i=j+1}^{k} a_i(x_i)\right)^2\right]$ også gemmes i hver knude. Da $\left(\mu(j,s_j)\right)^2$ kan risikere at blive meget stor, kan det muligvis udnyttes, at

$$\sigma^2(j,s_j) = \left(\sqrt{\nu(j,s_j)} - \mu(j,s_j)\right)\left(\sqrt{\nu(j,s_j)} + \mu(j,s_j)\right),$$

da $\nu(j,s_j) \geq 0$, hvormed man undgår at udregne $\left(\mu(j,s_j)\right)^2$.

5.2 Baglæns induktion

For at opbygge netværket, udnyttes rekursionsformlerne vha. baglæns induktion. Først bemærkes, at

$$\begin{aligned}
\mathbf{R}(k-1,s_{k-1}) &= \{(k,m)\}, \\
l(k-1,s_{k-1}) &= \min_{\boldsymbol{x}_{k-1}\in\Gamma(k-1,s_{k-1})}\{a_k(x_k)\} = \min_{x_k\in\{m-s_{k-1}\}}\{a_k\left(x_k\right)\} = a_k\left(m-s_{k-1}\right), \\
u(k-1,s_{k-1}) &= a_k\left(m-s_{k-1}\right), \\
q(k-1,s_{k-1}) &= \lambda_k\left(s_{k-1}, m-s_{k-1}\right), \\
\mu(k-1,s_{k-1}) &= a_k\left(m-s_{k-1}\right)P\left(m-s_{k-1}\right) = \frac{a_k\left(m-s_{k-1}\right)\lambda_k(s_{k-1},m-s_{k-1})}{q(k-1,s_{k-1})} = a_k\left(m-s_{k-1}\right), \\
\nu(k-1,s_{k-1}) &= \frac{\left(a_k\left(m-s_{k-1}\right)\right)^2\lambda_k(s_{k-1},m-s_{k-1})}{q(k-1,s_{k-1})} = \left(a_k\left(m-s_{k-1}\right)\right)^2 \quad \text{og} \\
\sigma^2(k-1,s_{k-1}) &= \nu(k-1,s_{k-1}) - \left(\mu(k-1,s_{k-1})\right)^2 = 0.
\end{aligned}$$

Derefter kan netværket initialiseres som illustreret i denne algoritme:

Algoritme 2 Baglæns induktion

1: **for** $s_{k-1} = \max\left(0, N_{k-1}+m-N\right)$ to $\min\left(m, N_{k-1}\right)\}$ **do**
2: Initialisér tilstanden $(k-1,s_{k-1})$ jf. ovenstående formler
3: **end for**
4: **for** $j = k-2$ to 1 **do**
5: **for** $s_j = \max\left(0, N_j+m-N\right)$ to $\min\left(m, N_j\right)$ **do**
6: $\mathbf{R}(j,s_j) \leftarrow \left\{(j+1,w) : \max\left(s_j, m-\sum_{i=j+2}^{k} n_i\right) \leq w \leq \min\left(m, s_j+n_{j+1}\right)\right\}$
7: Udregn $l(j,s_j), u(j,s_j), q(j,s_j), \mu(j,s_j), \nu(j,s_j)$ og $\sigma^2(j,s_j)$ vha. rekursionsformlerne udledt i afsnit 5.1
8: **end for**
9: **end for**

Når først netværket er opbygget, foregår traverseringen som beskrevet i [Mehta et al., 1988]. Bemærk dog, at i [Mehta et al., 1988, (3.9)] er

$$b_{j+1} = a_{j+1}(w-s_j) + b_j,$$

altså på samme måde som definitionen af $R^*(j,s_j,b_j)$ samt [Mehta et al., 1988, (3.7)], hvilket ikke fremgår helt klart (da der summeres over w i stedet for s_{j+1}).

5.3 Fishers eksakte test

I dette afsnit beskrives, hvordan importance samplingen beskrevet i [Mehta et al., 1988] kan benyttes i forbindelse med Fishers eksakte test.

Under nulhypotesen om at der er uafhængighed, er sandsynligheden for en $2 \times k$-kontingenstabel med de givne marginaler $m, n_1, n_2, \ldots, n_k$ givet ved

$$P_{\text{cutoff}} = \frac{m!(N-m)!n_1! \cdots n_k!}{x_1! \cdots x_k!(n_1 - x_1)! \cdots (n_k - x_k)!N!}$$

Det bemærkes, at

$$\begin{aligned}\log P_{\text{cutoff}} &= \log\left(m!(N-m)!n_1! \cdots n_k!\right) - \log\left(x_1! \cdots x_k!(n_1 - x_1)! \cdots (n_k - x_k)!N!\right) \\ &= \sum_{j=2}^{m} \log j + \sum_{j=2}^{N-m} \log j + \sum_{i=1}^{k} \sum_{j=2}^{n_i} \log j - \sum_{i=1}^{k} \sum_{j=2}^{x_i} \log j - \sum_{i=1}^{k} \sum_{j=2}^{n_i - x_i} \log j - \log\left(N!\right),\end{aligned}$$

som givet de faste marginaler $m, n_1, n_2, \ldots, n_k$ kan reduceres til (da alle kontingenstabeller med disse marginaler vil have samme konstante off-set)

$$\begin{aligned}Q_{\text{cutoff}} &= -\sum_{i=1}^{k} \sum_{j=2}^{x_i} \log j - \sum_{i=1}^{k} \sum_{j=2}^{n_i - x_i} \log j \\ &= -\sum_{i=1}^{k} \left(\sum_{j=2}^{x_i} \log j + \sum_{j=2}^{n_i - x_i} \log j \right),\end{aligned}$$

hvorfor Q_{cutoff} kan skrives på formen

$$Q_{\text{cutoff}} = \sum_{i=1}^{k} a_i(x_i)$$

med

$$a_i(x_i) = -\left(\sum_{j=2}^{x_i} \log j + \sum_{j=2}^{n_i - x_i} \log j \right).$$

Da teststørrelsen er på denne form, kan der (som et alternativ til eksempelvis Patefields algoritme [Patefield, 1981]) med metoden beskrevet i [Mehta et al., 1988] laves importance sampling for at estimere eksakte signifikansniveauer.

Sæt

$$\lambda_i(s_{i-1}, x_i) = \frac{1}{x_i!(n_i - x_i)!}.$$

Da bemærkes, at

$$\lambda_i(s_{i-1}, x_i) = \exp\left(a_i(x_i)\right).$$

De specificerede teststørrelser er kritiske for små værdier, men [Mehta et al., 1988] kræver, at teststørrelsen skal være kritisk for store værdier. Derfor benyttes i stedet følgende funktioner:

$$\begin{aligned}a_i(x_i) &= \sum_{j=2}^{x_i} \log j + \sum_{j=2}^{n_i - x_i} \log j, \\ \lambda_i(s_{i-1}, x_i) &= \left(\exp\left(a_i(x_i)\right)\right)^{-1} = x_i!(n_i - x_i)!.\end{aligned}$$

5.4 Kommentarer til implementeringen

Først knyttes et par bemærkninger til [Mehta et al., 1988, (3.8)] givet ved

$$\hat{q}^*(j, s_j, b_j) = q(j, s_j)\Phi^c\left(\frac{t^* - b_j - \mu(j, s_j)}{\sqrt{\sigma^2(j, s_j)}} \right),$$

hvor Φ^c er den højre hale af standardnormalfordelingen. I det tilfælde, at $\sigma^2(j, s_j) = 0$, sættes

$$\hat{q}^*(j, s_j, b_j) = q(j, s_j),$$

da der dermed kun er én rute til sluttilstanden, så vægten sættes til 1 i stedet for $\Phi^c(\cdot)$.

Af hensyn til numerisk stabilitet, vil der blive forsøgt med følgende alternative definitioner af $\lambda_i(x_i)$ givet ved

$$\lambda_i(s_{i-1}, x_i) = \exp\left(\texttt{lchoose}(n_i, \lfloor N\hat{p}_i \rfloor) - \texttt{lchoose}(n_i, x_i)\right),$$
$$\lambda_i(s_{i-1}, x_i) = \exp\left(\texttt{lchoose}(n_i, x[i]) - \texttt{lchoose}(n_i, x_i)\right),$$

hvor `lchoose` er logaritmen til binomialkoefficienten, $x[i]$ er det observerede antal i kolonne i og $\hat{p}_i = \frac{mn_i}{N^2}$.

5.5 Resultater

Som nævnt i indledningen, kan en implementation i `R` af den beskrevne importance sampling findes på http://people.math.aau.dk/~mikl/mat4/projekt i `impsamp.R`-filen. Denne fil indeholder også logikken til automatisk at generere de LaTeX-tabeller, der findes i dette afsnit.

Da importance samplingen udelukkende er implementeret i `R`, hvor `fisher.test` (både den eksakte og Monte Carlo-simulationen) er implementeret i C, giver det ikke mening at sammenligne eksekveringstiden. Derimod kan Fishers eksakte test anvendes som p-værdi-reference og anvendes til at sammenligne præcisionen af importance sampling og Monte Carlo-simulation. Først sammenlignes forskellige valg af $\lambda_i(s_{i-1}, x_i)$-funktioner og vægtninger i $\hat{q}^*(j, s_j, b_j)$, og derefter sammenlignes den bedste kombination med Monte Carlo-simulation. For at kunne sammenligne dem, startes dog med at indføre konfidensintervaller.

5.5.1 Konfidensintervaller

Lad $p_1, p_2, \ldots, p_M$ være samplede p-værdier med $p_i = p(\boldsymbol{x})$ for tabel $\boldsymbol{x}$. Da der kun samples fra de kritiske tabeller, og samplingerne er uafhængige, vil der kunne laves følgende konstistente estimatorer:

$$\overline{p} = \frac{1}{M}\sum_{i=1}^{M} p_i \quad \text{og} \quad s^2 = \frac{1}{M-1}\sum_{i=1}^{M}(p_i - \overline{p})^2.$$

Med andre ord vil der gælde, at

$$\mathbf{E}\left[\overline{p}\right] = p^*,$$

hvor p^* er den sande p-værdi. Dermed kan der også laves et 95%-konfidensinterval for p^* givet ved

$$\left[\overline{p} - 1.96\frac{s}{\sqrt{M}} \;;\; \overline{p} + 1.96\frac{s}{\sqrt{M}}\right].$$

Tilsvarende kan laves for traditionel Monto-Carlo-sampling, hvor p-værdien findes gennem realisation af en Bernoulli-variabel. Simulationsversionen af `fisher.test`-funktionen i `R` giver kun p-værdien, men da der er tale om et binomialforsøg haves, at

$$\mathbf{Var}\left[\overline{p}\right] = \frac{p^*(1-p^*)}{M},$$

og dermed kan konfidensintervaller konstrueres på samme måde som før.

5	0	5
1	4	5
6	4	10

Tabel 5.1: Kontingenstabel med p-værdien 0.04761905.

5	0	1	6
1	4	2	7
6	4	3	13

Tabel 5.2: Kontingenstabel med p-værdien 0.03438228.

5.5.2 Kontingenstabeller

Kontingenstabellerne 5.1, 5.2, 5.3, 5.4 og 5.5 vil blive brugt til at sammenligne de forskellige metoder med.

5.5.3 Sammenligning af forskellige importance sampling-parametre

Bemærk at følgende navngivning anvendes:

$$\begin{aligned}
&\lambda_1 : \lambda_i(s_{i-1}, x_i) = \left(\exp\left(a_i(x_i)\right)\right)^{-1} = x_i!(n_i - x_i)!, \\
&\lambda_2 : \lambda_i(s_{i-1}, x_i) = \exp\left(\texttt{lchoose}(n_i, \lfloor N\hat{p}_i \rfloor) - \texttt{lchoose}(n_i, x_i)\right), \\
&\lambda_3 : \lambda_i(s_{i-1}, x_i) = \exp\left(\texttt{lchoose}(n_i, x[i]) - \texttt{lchoose}(n_i, x_i)\right).
\end{aligned}$$

Først startes med et lille trivielt tjek af kontingenstabel 5.1 i tabel 5.6. Her ses at importance sampling med det samme finder den eksakte p-værdi (da der kun er to kritiske tabeller – den anden er blot hvor rækkerne er ombyttet).

For 10, 100, 1000 og 10000 simulationer af kontingenstabel 5.2 kan middelværdier/1.96s ses i henholdsvis tabel 5.7/5.8, 5.9/5.10, 5.11/5.12 og 5.13/5.14. Generelt ser alle kombinationerne ud til at klare sig godt, når der anvendes tilstrækkeligt mange simulationer.

Det samme mønster gør sig gældende i simulationerne af kontingenstabel 5.3. For 10, 100, 1000 og 10000 simulationer kan middelværdier/1.96s ses i henholdsvis tabel 5.15/5.16, 5.17/5.18, 5.19/5.20 og 5.21/5.22.

Det viser sig, at man med λ_1 løber ind i numeriske problemer i forbindelse med kontigenstabeller med store cellecounts. Det er derfor, at λ_2 og λ_3 er blevet indført.

Samlet set ser kombinationen λ_2 sammen med vægtningen Φ^c ud til at klare sig bedst, specielt ved få simulationer, men der er ikke en entydig bedste kombination. Dog vælges denne kombination til at sammenligne variansen af importance sampling med traditionel Monte Carlo-sampling.

15	6	13	34
5	2	0	7
20	8	13	41

Tabel 5.3: Kontingenstabel med p-værdien 0.1246539.

10	9	10	9	10	48
0	1	0	1	0	2
10	10	10	10	10	50

Tabel 5.4: Kontingenstabel med p-værdien 1.

24	78	33	76	211
6	2	2	24	34
30	80	35	100	245

Tabel 5.5: Kontingenstabel med p-værdien $4.484936 \cdot 10^{-5}$.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.04761905	0.04761905	0.04761905
$\Phi(\cdot)$	0.04761905	0.04761905	0.04761905
1	0.04761905	0.04761905	0.04761905

Tabel 5.6: Middelværdier for kontingenstabellen 5.1 ved 1 simulation.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.02817731	0.02583006	0.03253463
$\Phi(\cdot)$	0.02784013	0.02096697	0.02566147
1	0.02348281	0.02800872	0.2528839

Tabel 5.7: Middelværdier for kontingenstabellen 5.2 ved 10 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.008984306	0.009598886	0.01145531
$\Phi(\cdot)$	0.01291608	0.01195501	0.01161321
1	0.009942635	0.01112580	0.4270073

Tabel 5.8: $1.96s$ for kontingenstabellen 5.2 ved 10 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.02338166	0.0502278	0.02546046
$\Phi(\cdot)$	0.02680138	0.02609720	0.02619836
1	0.04773012	0.02569519	0.02522574

Tabel 5.9: Middelværdier for kontingenstabellen 5.2 ved 100 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.003361452	0.04296157	0.003400258
$\Phi(\cdot)$	0.003548232	0.003587635	0.003246130
1	0.04300172	0.00338879	0.003411060

Tabel 5.10: $1.96s$ for kontingenstabellen 5.2 ved 100 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.03651139	0.0324111	0.03668231
$\Phi(\cdot)$	0.03382171	0.0354194	0.03402454
1	0.03010191	0.03032173	0.03839301

Tabel 5.11: Middelværdier for kontingenstabellen 5.2 ved 1000 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.009628825	0.007498937	0.00962975
$\Phi(\cdot)$	0.008632129	0.008625504	0.008632087
1	0.006161586	0.006154513	0.01053269

Tabel 5.12: 1.96s for kontingenstabellen 5.2 ved 1000 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.03450258	0.03532276	0.03236194
$\Phi(\cdot)$	0.03678219	0.03412725	0.03404763
1	0.03642218	0.03292664	0.03877833

Tabel 5.13: Middelværdier for kontingenstabellen 5.2 ved 10000 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.002761249	0.002858083	0.002408221
$\Phi(\cdot)$	0.003073225	0.002694391	0.002694430
1	0.003043789	0.002445481	0.003301179

Tabel 5.14: 1.96s for kontingenstabellen 5.2 ved 10000 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.2765958	0.09834367	0.2768341
$\Phi(\cdot)$	0.09389142	0.2714131	0.1024707
1	0.0877532	0.09058398	0.1032747

Tabel 5.15: Middelværdier for kontingenstabellen 5.3 ved 10 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.3384936	0.0182338	0.3385484
$\Phi(\cdot)$	0.01515197	0.3394836	0.01417765
1	0.01265649	0.01113595	0.01392115

Tabel 5.16: 1.96s for kontingenstabellen 5.3 ved 10 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.1316222	0.1238346	0.1716778
$\Phi(\cdot)$	0.1367253	0.1578523	0.1140042
1	0.09630993	0.1094525	0.1201027

Tabel 5.17: Middelværdier for kontingenstabellen 5.3 ved 100 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.04795636	0.03977603	0.06439787
$\Phi(\cdot)$	0.0484205	0.05855816	0.03422243
1	0.004416460	0.02842859	0.03988596

Tabel 5.18: 1.96s for kontingenstabellen 5.3 ved 100 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.1214172	0.1291677	0.1162850
$\Phi(\cdot)$	0.1237452	0.1233202	0.1256487
1	0.1260409	0.1283025	0.1357193

Tabel 5.19: Middelværdier for kontingenstabellen 5.3 ved 1000 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.01247074	0.01391967	0.01103404
$\Phi(\cdot)$	0.01295039	0.01279114	0.01319136
1	0.01321217	0.01391020	0.01574811

Tabel 5.20: $1.96s$ for kontingenstabellen 5.3 ved 1000 simulationer.

5.5.4 Sammenligning af importance og Monte Carlo-sampling

I sammenligningen af importance samplingen med traditionel Monte Carlo-sampling, vil λ_2 blive benyttet. Det vil sige, at

$$\lambda_i(s_{i-1}, x_i) = \exp\left(\texttt{lchoose}(n_i, \lfloor N\hat{p}_i \rfloor) - \texttt{lchoose}(n_i, x_i)\right)$$

samt den oprindelige definition af $\hat{q}^*(j, s_j, b_j)$ med vægtningen Φ^c vil blive anvendt.

En oversigt over hvilke tabeller, der viser hvilke resultater, kan findes i tabel 5.23.

5.5.5 Opsummering

I dette afsnit er importance sampling blevet sammenlignet med Monto Carlo-sampling. Der blev fundet ud af, at der i importance sampling skal simuleres langt færre tabeller end i Monto Carlo-sampling for at opnå en lille varians, hvilket stemmer overens med formålet med importance sampling. Det bemærkes dog, at importance sampling udelukkende ser ud til at være markant bedre ved små p-værdier, hvilket dog også er de mest interessante tilfælde. Et eksempel på dette er sampling af p-værdien $4.484936 \cdot 10^{-5}$ for kontingenstabel 5.5, hvor resultatet af traditionel Monto Carlo-sampling ses i tabel 5.31, og resultatet af importance sampling ses i tabel 5.32. Her er en markant forskel på præcisionen og antallet af nødvendige simulationer.

En oversigt over hvilke tabeller, der viser hvilke resultater, kan findes i tabel 5.23. Det har dog ikke været muligt at sammenligne eksekveringstider, da sammenligningen blev foretaget med `R`'s implementation af Monte Carlo-sampling (gennem `fisher.test`, der er implementeret i C), mens importance sampling blev implementeret til formålet i `R` efter [Mehta et al., 1988]. For at kunne sammenlige eksekveringstider ville importance samplingen også skulle implementeres i C. En sådan implementation vil dog muligvis stadig have numeriske problemer i forbindelse med importance sampling af store tabeller, da netværket, man opbygger, kan blive forholdsvist stort og kan indeholde mange tabeller med lille sandsynlighed. Den baglænse induktion er ekstremt smart til opbygning af netværket, men det gør det samtidigt svært at optimere og udelukke usandsynlige veje, da man først kender normaliseringskonstanten for sent i forløbet.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.1257726	0.1233305	0.1282754
$\Phi(\cdot)$	0.1259505	0.1267349	0.1230619
1	0.1233150	0.1225369	0.1253797

Tabel 5.21: Middelværdier for kontingenstabellen 5.3 ved 10000 simulationer.

	λ_1	λ_2	λ_3
$\Phi^c(\cdot)$	0.004201634	0.004114148	0.004417331
$\Phi(\cdot)$	0.004279821	0.004297719	0.004093885
1	0.004076431	0.004014054	0.004229442

Tabel 5.22: $1.96s$ for kontingenstabellen 5.3 ved 10000 simulationer.

Kontingenstabel	p-værdi	Monte Carlo-sampling	Importance sampling
5.2	0.03438228	Tabel 5.24	Tabel 5.25
5.3	0.1246539	Tabel 5.26	Tabel 5.27
5.4	1	Tabel 5.28	Tabel 5.29
5.5	$4.484936 \cdot 10^{-5}$	Tabel 5.31	Tabel 5.32

Tabel 5.23: Tabel med hvor resultater til sammenligning af importance sampling og Monte Carlo-sampling findes.

	Middelværdi	$1.96s$
100	0.04950495	0.04251629
1000	0.03896104	0.01199339
10000	0.0339966	0.00355192
1e+06	0.03434997	0.0003569679

Tabel 5.24: Resultater for Monte Carlo-sampling for kontingenstabellen 5.2. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
10	0.02096697	0.01195501
100	0.05180344	0.04293709
1000	0.03182091	0.007502714
10000	0.03351884	0.002590797

Tabel 5.25: Resultater for importance sampling for kontingenstabellen 5.2. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
100	0.1584158	0.07156557
1000	0.1258741	0.02055944
10000	0.1221878	0.00641905
1e+06	0.1244889	0.0006470714

Tabel 5.26: Resultater for Monte Carlo-sampling for kontingenstabellen 5.3. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
10	0.08767966	0.01323679
100	0.1280028	0.03959602
1000	0.1253166	0.01319751
10000	0.1227366	0.004034505

Tabel 5.27: Resultater for importance sampling for kontingenstabellen 5.3. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
100	1	0
1000	1	0
10000	1	0
1e+06	1	0

Tabel 5.28: Resultater for Monte Carlo-sampling for kontingenstabellen 5.4. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
10	0.6024984	0.0998042
100	0.7854818	0.08794402
1000	0.7780894	0.03019005
10000	0.7532751	0.009019746

Tabel 5.29: Resultater for importance sampling for kontingenstabellen 5.4. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
10	0.95	0.098
100	0.865	0.04372721
1000	0.847	0.01428841
10000	0.7532751	0.009019746

Tabel 5.30: Resultater for importance sampling for kontingenstabellen 5.4 med vægtningen 1. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
100	0.00990099	0.01940594
1000	0.000999001	0.001958042
10000	0.00029997	0.0003394141
1e+06	5.099995e-05	1.399684e-05

Tabel 5.31: Resultater for Monte Carlo-sampling for kontingenstabellen 5.5. Kolonnen til venstre angiver antal udførte simulationer.

	Middelværdi	$1.96s$
10	6.568262e-06	7.305999e-07
100	7.02142e-06	7.187673e-07
1000	7.523566e-06	3.036694e-07
10000	4.732183e-05	7.434721e-05

Tabel 5.32: Resultater for importance sampling for kontingenstabellen 5.5. Kolonnen til venstre angiver antal udførte simulationer.

Resultatet af importance sampling af kontingenstabel 5.4 kan ses i tabel 5.29. Her ser resultaterne meget underlige ud. Det skyldes muligvis vægtningen. Hvis man i stedet anvender vægtning 1, så fås resultaterne i tabel 5.30, hvilket ser bedre ud, men stadigvæk ikke optimalt.

I stedet for den anvendte sammenligningsmetode, kunne man også kigge på sekventiel sampling, hvor man i stedet vurderede, hvor mange tabeller der skal samples inden variansen kommer under en foruddefineret tærskel. Eksempelvis kunne man vælge at stoppe, hvis $\overline{p} - \frac{3s}{\sqrt{M}} > 0.05$.

Kapitel 6

Opsummering

Projektets fokus var at undersøge locis afhængighed i Y-STR-haplotyper ved hjælp af grafiske modeller samt at undersøge de teoretiske områder, der var forbundet med dette problem.

Afhængighedsstrukturen blev forsøgt modelleret ved hjælp af strukturlæringsalgoritmer som BIC/AIC som PC-algoritmen (se afsnit 3.1). Derudover blev der konstrueret 1., 2. og 3. ordens modeller ud fra locis fysiske placering på dna'et i håb om, at dette afspejlede afhængighedsstrukturen (se afsnit 3.2). Som det kan ses i afsnit 3.3, gav ingen af metoderne anledning til en endegyldig god grafisk model, der afspejler locis afhængighedsstruktur.

En af de primære årsager til det, er at datasættene er meget tynde. Ved hjælp af automatisk kollapsning (se kapitel 4) ved brug af både Fishers eksakte test og Kruskall og Goodmans γ, blev datasættene voldsomt kollapset – se tabel 4.1. Det gav dog ikke anledning til, at de grafiske modeller ud fra locis fysiske placering blev bedre.

Der blev introduceret mål for, hvor godt en model passer; dels blev sandsynlighedsmassen for uobserverede haplotyper introduceret i afsnit 3.3.1 og dels blev Brier-scoren introduceret i afsnit 3.3.2. Umiddelbart ser ingen af målene ud til at være entydige gode indikatorer for, hvor god en given model er. Ved at sammenholde tabel 3.3 og 4.4 ses, at Brier-scoren og den korrigerede opfører sig vidt forskelligt på `dane`, hvorimod de opfører sig meget ens på `somali`.

Til at sample $2 \times k$-kontingenstabeller, blev importance sampling undersøgt i kapitel 5. Der blev fundet ud af, at for små p-værdier gav importance sampling anledning til langt mindre varians end traditionel Monte Carlo-sampling.

6.1 Videre arbejde

Nogle af de ting, der kunne være interessante at kigge nærmere på er kollapsning, importance sampling og modelverifikation.

Det kunne det være interessant at undersøge om der fandtes andre måder at kollapse alleler på, der gav anledning til en stærkere afhængighedsstruktur. Det ville tilsvarende være interessant at få data med haplotyper baseret på flere loci.

I forbindelse med importance sampling ville det være interessant at kigge på nogle af de optimeringer artiklen nævner, samt at implementere det i C (da det af en eller anden grund ikke findes i `R` pt.) og teste bl.a. eksekveringstiden op mod tiden for traditionel Monte Carlo-sampling.

Udover sandsynlighedsmassen for uobserverede haplotyper og Brier-score, kunne det være interessant at undersøge, om man kunne lave mere robuste mål for, hvor god en model er.

Litteratur

David Edwards. *Introduction to Graphical Modelling*. Springer, 2. edition, 2000. ISBN 0-387-95054-0. 5

L. Excoffier, P. E. Smouse, and J. M. Quattro. Analysis of molecular variance inferred from metric distances among dna haplotypes: Application to human mitochondrial dna restriction data. *Genetics*, 131(2):479–491, June 1992. ISSN 0016-6731. URL http://view.ncbi.nlm.nih.gov/pubmed/1644282. 10, 13, 15

Charlotte Hallenberg, Karsten Nielsen, Bo Simonsen, Juan Sanchez, and Niels Morling. Y-chromosome str haplotypes in danes. December 2004. URL http://www.ncbi.nlm.nih.gov/pubmed/16226159. 9

Charlotte Hallenberg, Bo Simonsen, Juan Sanchez, and Niels Morling. Y-chromosome str haplotypes in somalis. January 2005. URL http://www.ncbi.nlm.nih.gov/pubmed/15939170. 9

Erin K K. Hanson and Jack Ballantyne. Comprehensive annotated str physical map of the human y chromosome: Forensic implications. *Leg Med (Tokyo)*, December 2005. ISSN 1344-6223. URL http://dx.doi.org/10.1016/j.legalmed.2005.10.001. 23

Finn V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, 2. edition, 2007. ISBN 978-0-387-68281-5. 19, 20, 21, 22

Markus Kalisch and Peter Buhlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007. URL http://jmlr.csail.mit.edu/papers/volume8/kalisch07a/kalisch07a.pdf. 21, 22

Cyrus R. Mehta, Nitin R. Patel, and Pralay Senchaudhuri. Importance sampling for estimating exact probabilities in permutational inference. *Journal of the American Statistical Association*, 83(404):999–1005, December 1988. URL http://www.jstor.org/stable/2290126. 10, 41, 42, 43, 46, 47, 52

W. M. Patefield. Algorithm as 159: An efficient method of generating random r x c tables with given row and column totals. *Applied Statistics (Series C)*, 30(1):91–97, 1981. URL http://www.jstor.org/stable/2346669. 41, 47

Herbert E. Robbins. Estimating the total probability of the unobserved outcomes of an experiment. *The Annals of Mathematical Statistics*, 39(1):256–257, 1968. URL http://www.jstor.org/stable/2238931. 25, 32, 37

L. Roewera, M. Kayserb, P. de Knijffc, K. Anslingerd, A. Betze, A. Cagliaf, D. Corach, S. Furedi, L. Henke, M. Hidding, H.J. Kargel, R. Lessig, M. Nagy, V.L. Pascali, W. Parson, B. Rolf, C. Schmittj, R. Sziborn, J. Teifel-Gredingo, and M. Krawczakp. A new method for the evaluation of matches in non-recombining genomes: application to y-chromosomal short tandem repeat (str) haplotypes in european males. *Forensic Science International*, 114(1):31–43, October 2000. URL http://linkinghub.elsevier.com/retrieve/pii/S0379073800002875. 10, 15

Ajit P. Singh and Andrew W. Moore. Finding optimal bayesian networks by dynamic programming (cmu-cald-05-106). 2005. URL http://www.autonlab.org/autonweb/library/papers.html. 20