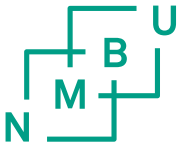

U
B
M
N

Forord

Denne masteroppgaven er utarbeidet ved avdeling for medisinsk genetikk, OUS og ved Institutt for kjemi, bioteknologi og matvitenskap ved Norges miljø- og biovitenskapelige universitet (NMBU) i perioden fra august 2013 til mai 2014.

Jeg har alltid likt utfordringer og gleden av å strekke meg etter nye ambisiøse mål. Det er akkurat det denne oppgaven har vært - en stor utfordring. Det har blitt mange våkenetter der jeg har grublet over problemstillingen. Det har vært mye frustrasjon, men også glede forbundet med prosjektet. Prosessen har vært utrolig lærerik og jeg sitter igjen med et stort faglig utbytte.

Jeg vil først takke veilederen min på medisinsk genetikk, Magnus Dehli Vigeland, for at jeg fikk ta del i prosjektet hans. Det har vært et spennende og lærerikt år. Jeg vil spesielt takke for tålmodighet, god veiledning og nødvendige kommentarer underveis i prosessen. Jeg vil også takke veilederen min på NMBU, Thore Egeland, for gode kommentarer i skriveprosessen. Deres hjelp har vært uvurderlig.

En stor takk fortjener også min kjære Marius, som har oppmuntret og holdt ut med meg når stormene har stått på som verst.

Oslo, 10.05.2014
NMBU 2014, Kristina Stormo Gjøtterud

Sammendrag

Eksomsekvensering, en storskala-sekvenseringsmetode, gir informasjon om proteinkodende områder i det humane genomet. Kartlegging av sykdommer forårsaket av en mutasjon i et gen, monogene sykdommer, har særlig dratt nytte av denne teknologien.

For recessive, monogene sykdommer, må et individ ha to kopier av en sykdomsgivende genvariant for å bli rammet. Økt forekomst av sykdom er kjent i inngiftefamilier. Dette skyldes at barn av beslektede foreldre har større sannsynlighet for å arve to kopier av en recessiv genvariant enn barn av ubeslektede foreldre. Dette er fordi foreldrenes genmateriale stammer fra samme forfar. En metode kalt autozygositetskartlegging blir derfor benyttet til å finne lange homozygote områder hos individer i inngiftefamilier siden sykdomslocus vil ligge her.

Tradisjonelt sett har autozygote områder blitt identifisert ved bruk av genetiske markører som er jevnt fordelt utover genomet. Det er derfor forventet at autozygositetskartlegging vil være vanskelig ved eksomsekvensering, da man mister informasjon knyttet til de ikke-proteinkodende områdene. En statistisk metode, en skjult markovmodell, er derfor benyttet for å kartlegge autozygote områder ved anvendelse av eksomdata.

Den skjulte markovmodellen beskrevet i denne oppgaven baserer seg på artikkelen til Leutenegger et. al (2003) og anvender to algoritmer for å kartlegge autozygote områder – forward-backward og Viterbi. Resultater fra disse algoritmene er sammenlignet med resultater fra Plink (tillegg A.1). Dette fordi Plink er et velegnet program for å finne autozygote områder.

Data, mest mulig realistisk til data fra eksomsekvensering er blitt simulert. Resultatene har vist at den skjulte markovmodellen detekterer autozygote områder ved anvendelse av eksomdata samtidig som den er mer velegnet enn Plink. Arbeidet har derfor vist at autozygositetskartlegging er mulig ved eksomsekvensering.

Summary

Exome sequencing, a high-throughput sequencing method, provides information about protein-coding regions in the human genome. Mapping of diseases caused by a mutation in a single gene, monogenic diseases, has especially benefitted from this technology.

For recessive, monogenic diseases, an affected individual must have two copies of the disease-causing gene variant. Increased occurrence of such diseases are known in consanguineous families. This is due to the fact that children of parents that are related, hence the parental genome has originated from the same ancestor, are more likely to inherit two copies of the recessive gene variant compared to children of unrelated parents. A method called autozygositymapping is therefore used to map homozygous regions of individuals of consanguineous families as the locus of the disease will be found here.

Autozygous regions are traditionally detected by the use of genetic markers that are evenly dispersed throughout the genome. It is therefore expected that autozygositymapping will be harder with exome sequencing due to information from protein coding regions are lost. Instead, a statistical method is used, the Hidden Markov Modell, where exome data is used to map autozygous regions. The Hidden Markov Model used in this thesis is based on the article of Leutenegger et al. (2003), and uses two algorithms to map autozygous regions – Forward-Backward and Viterbi. The results from these algorithms are compared with results from Plink (appendix A.1) as it is a well suited program to detect autozygous regions.

Data, as realistic as possible to data from exome sequencing, has been simulated. The results show that The Hidden Markov Model detects autozygous regions using exome data, while it performs better than Plink, hence the study therefore show that autozygositymapping is possible with exome sequencing.

Forkortelser

bp:	Basepar; fysiske posisjoner i genomet.
cM:	Centimorgan; genetisk distanse mellom to markører.
DNA:	Deoxyribonuleic acid; Arvematerialet – en dobbelttrådet helix bestående av deoxyribose, fosfat og basene A, T, C, G.
FN:	Falsk negativ; godtar H_0 når H_1 er sann – type-II feil.
FNR:	Falsk negativ rate; hyppigheten av type-II feil (FN).
FP:	Falsk positiv; godtar H_1 når H_0 er sann – type-I feil.
FPR:	Falsk positiv rate; hyppigheten av type-I feil (FP).
HGP:	Human Genome Project; internasjonalt forskningsprosjekt for å kartlegge det humane genomet.
HMM:	Skjult markovmodell; statistisk modell for å finne en tilstandssekvens bakenforliggende en observasjonssekvens.
HTS:	Storskala-sekvensering. En sekvenseringsteknikk som kan håndtere store datamengder billig og raskt.
IBS:	Identical-By-State; to alleler er identiske.
IBD:	Identical-By-Descent; to alleler er identiske og stammer fra felles forfar.
Kb:	Kilobaser; fysisk distanse mellom to markører (1 Kb = 1000 bp).
M:	Morgans; genetisk distanse mellom to markører (1M = 100 cM).
MAF:	Minor Allele Frequency; frekvensen til den sjeldne genvarianten.
Mb:	Megabase; fysisk distanse mellom to markører (1 Mb = 1000Kb).
NMBU:	Norges miljø- og biovitenskapelige universitet
OMIM:	Online Mendelian Inheritance in Man; samler informasjon om sykdomsrelaterte gener og fenotyper i det humane genomet.
OUS:	Oslo Universitetssykehus Ullevål
ROH:	Runs Of Homozygosity; et område som bare består av homozygote markører.
SN:	Sann negative; godtar H_0 når H_0 er sann.
SNP:	Enkeltbasepolymorfisme; vanlig genvariant med MAF > 1 % i aktuell befolkning.
SNR:	Sann negativ rate; hyppigheten av SN.
SNV:	Enkeltbasevariant; sjelden genvariant med MAF < 1 % i aktuell befolkning.
SP:	Sann positiv; godtar H_1 når H_1 er sann.
SPR:	Sann positiv rate; hyppigheten av SP.
VCF:	Variant Call Format; filformat for å lagre informasjon om sekvensvariasjon ved sekvensering.

Innholdsfortegnelse

1. Innledning

Fra cirka 1980 var Sanger-sekvensering det eneste verktøyet som egnet seg til DNA sekvensering. I 2005 møtte denne metoden konkurranse da storskala-sekvensering (HTS) kom på markedet.

Storskala-sekvensering har gjort det «lettere» å kartlegge genetiske årsaker til sykdom i det humane genomet, da teknikken er raskere og billigere enn tidligere Sanger-sekvensering. Sykdommer forårsaket av en mutasjon i et gen, monogene sykdommer, har særlig dratt nytte av denne teknologien. I slike tilfeller blir den kodende delen av genomet sekvensert (eksomsekvensering) i jakten på en kausal genvariant.

I denne oppgaven er fokuset rettet mot kartlegging av genetiske årsaker som er knyttet til monogene sykdommer med autosomalt recessivt nedarvingsmønster. Det vil si, sykdommen er ikke kjønnsbundet og kommer kun til uttrykk dersom et individ arver den kausale sykdomsvarianten fra begge foreldrene.

Økt forekomst av sykdom er kjent i inngiftefamilier. Dette skyldes at barn av beslektede foreldre har større sannsynlighet for å arve to kopier av en recessiv genvariant enn barn av ubeslektede foreldre. Dette er fordi foreldrenes genmateriale stammer fra samme forfar. Autozygositetskartlegging (homozygot + IBD, jamfør forkortelser) har derfor vært et viktig verktøy i jakten på recessive sykdommer hos individer av beslektede foreldre.

1.1 Formål og oppbygging

En skjult markovmodell, er benyttet i denne oppgaven for å kartlegge autozygote områder ved data fra eksomsekvensering.

Et mål er å gjennomføre autozygositetskartlegging direkte ved eksomsekvensering. Det finnes allerede gode detekteringsprogrammer for å kartlegge autozygote områder, der Plink er et av disse. Et annet mål vil derfor være å undersøke om den skjulte markovmodellen detekterer autozygote områder bedre enn Plink. Dersom dette er tilfelle, vil et siste mål være å implementere den skjulte markovmodellen inn i analyseprogrammet Filtus (tillegg A.2).

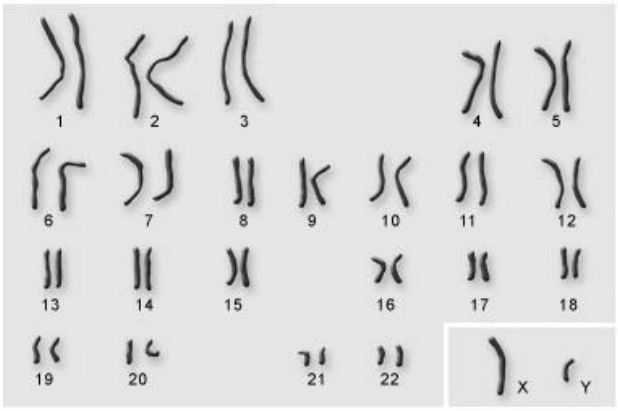
I kapittel 2 forklares teorien bak problemstillingen. Under «materialer og metoder» beskrives de simulerte- og realistiske dataene som er benyttet, den skjult markovmodell (implementert

kode i tillegg B.1-B.3) og parametervalg for Plink. Videre sammenlignes det hvor godt de ulike metodene detekterer sanne autozygote områder.

2. Bakgrunn

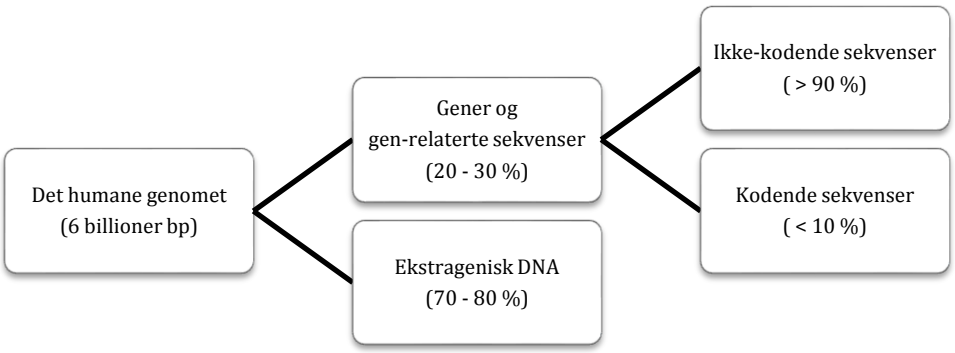
2.1 Human genetikk

Det humane genomet består av omtrent 6×10^9 basepar (bp) fordelt på 46 kromosomer. De 46 kromosomene utgjør 23 kromosompar (figur 2.1-1), hvorav 22 par av autosomer og to kjønnskromosomer – X og Y. Et kromosompar består av to homologe kromosomer, ett fra far og ett fra mor. Hvert kromosom utgjør et dobbeltrådet DNA-molekyl. Lengdene på kromosomene varierer fra omlag 5×10^7 bp til $2{,}63 \times 10^8$ bp.



Figur 2.1-1: Organisering av humane kromosompar *(EDinformatics)*

Det er estimert at ca. 20-30 % av det humane genomet er gener eller gen-relaterte sekvenser, hvorav < 10 % er protein kodende områder (Fletcher et al. 2007) (figur 2.1-2). Et gen består av en eller flere eksoner skilt av introner. Eksonene utgjør de proteinkodende områdene av et gen, mens intronene er de ikke-proteinkodende områdene.



Figur 2.1-2: Organisering av det humane genomet.

The Human Genome Project (HGP) publiserte i 2004 en nesten fullstendig genetisk sekvens, hvor det ble anslått at det humane genomet inneholder omlag 21 000 funksjonelle gener. Dette utgjør cirka 1 % av det totale genomet, hvorav det resterende er av ukjent karakter (Lander 2011)

6 mai 2014 var det ifølge OMIM (tabell 2.1-1) kartlagt 14 569 gener med kjent sekvens, og klassifisert 4 106 sykdommer med kjent molekylær bakgrunn (kjent gen- struktur/funksjon). Denne tabellen blir oppdatert fortløpende ettersom gener blir kartlagt og klassifisert.

Tabell 2.1-1: Antall oppføringer i OMIM *per 06.05.14. Baseres på genetisk molekylær bakgrunn og fenotypisk informasjon (OMIM).*

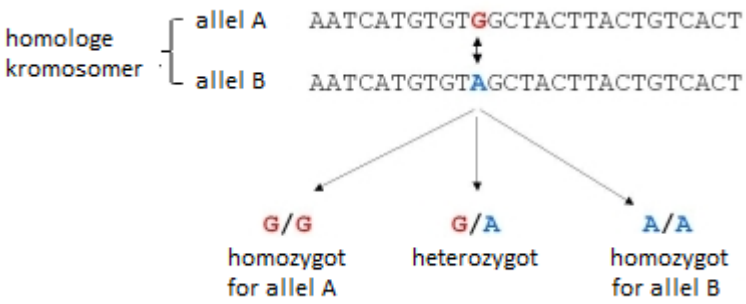
	Autosomal	X-bundet	Y-bundet	Mitokondriell	Totalt
* Gener med kjent beskrivelse	13 812	674	48	35	14 569
+ Gener og fenotype med kombinert kjent beskrivelse	99	2	0	2	103
# Fenotypisk beskrivelse, med kjent molekylær bakgrunn	3 789	285	4	28	4 106
% Mendelsk fenotype eller locus, med ukjent molekylær bakgrunn	1 568	134	5	0	1 707
Annet - hovedsakelig fenotyper med antatt mendelsk bakgrunn	1 739	115	2	0	1 856
Totalt	21 007	1 210	59	65	22 341

2.1.1 Genetisk variasjon

Basert på HGP er omtrent 99,9 % av det humane genomet likt for alle individer. De resterende 0,1 % varierer fra individ til individ (Jorde & Wooding 2004).

En SNV er en genetisk enkeltbasevariant (figur 2.1-3) ved et gitt locus som ikke er karakterisert, da varianten forekommer sjeldent i en aktuell befolkning. Frekvensen til det sjeldne allelet

betegnes MAF (Minor Allele Frequency). Dersom en SNV har MAF > 1 % betegnes enkeltbasevarianten som SNP, da den er vanlig i en aktuell befolkning. Det finnes cirka 1 SNP per 300 base i det humane genomet (Fletcher et al. 2007). Det utgjør cirka 90 % av sekvensvariasjonen, hvor de resterende 10 % er SNV'er.



***Figur 2.1-3: En SNV/SNP med to alleler G og A**. Referanseallelet (G) opptrer hyppig i en gitt befolkning mens det alternative allelet (A) opptrer sjeldnere/sjeldent. To homologe kromosomer kan dermed ha en av tre genotyper: GG, GA eller AA. (TraitGenetics)*

En genetisk markør er per definisjon et gitt locus med kjent polymorfisk egenskap (mer enn to alleler i befolkningen). SNP'er er derfor velegnet som genetiske markører og benyttes ofte ved genetisk kartlegging.

Distanser mellom markører

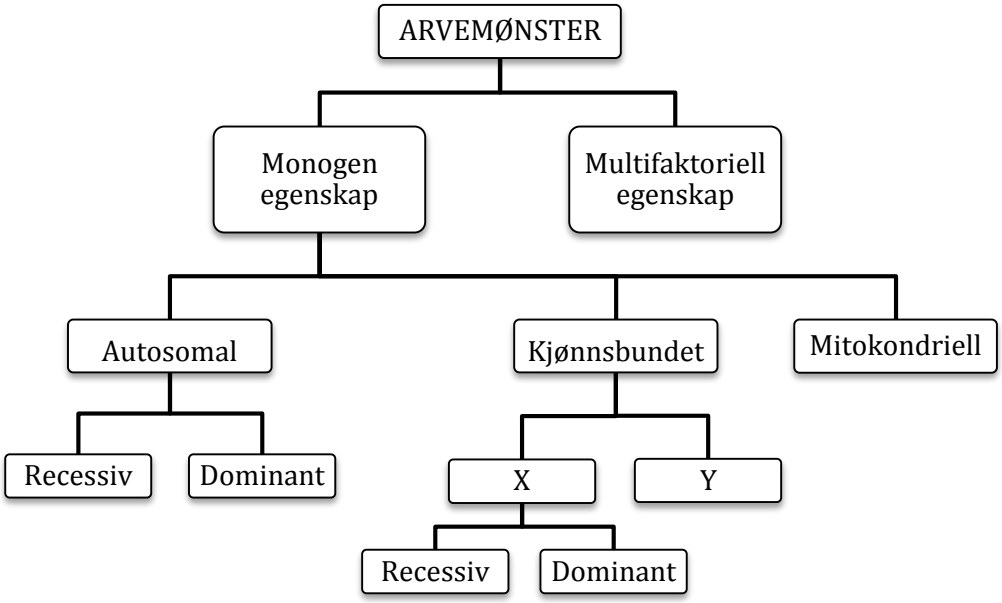
Eksomsekvensering (kapittel 2.1.4) oppgir fysisk, molekylær markørposisjon (bp). Fysisk distanse mellom to markører vil derfor være oppgitt i basepar, hvor 1 Kb = 1000 bp og 1Mb = 1000 Kb. Fysiske distanser egner seg imidlertid ikke til genetisk kartlegging, da ulike kartleggingsmetoder er basert på statistiske modeller som legger til grunn rekombinasjon (Ziegler & König 2010).

Det finnes ulike genetiske kartleggingsfunksjoner som knytter rekombinasjonsfrekvens oppimot genetisk distanse mellom markører. Haldanes kartleggingsfunksjon og χ^2-modell er to eksempler. Det er derfor vanlig å benytte genetisk distanse ved genetisk kartlegging, der enheten er gitt i Morgan (M), hvor 1 M = 100 cM. Imidlertid er genetisk distanse en tilnærming til fysisk distanse, hvor det grovt sett er antatt at 1 cM = 1 Mb.

2.1.2 Monogene egenskaper

Det er snart 150 år siden Gregor Johann Mendel publiserte sine studier av nedarvingsmønsteret hos erteplanter – en viktig byggestein for moderne genetikk. Genetiske egenskaper kun bestemt av ett gen betegnes som Mendelske- eller monogene egenskaper. Disse egenskapene er som oftest et resultat av en mutasjon i et gen. Det er derfor lettere å kartlegge monogene egenskaper kontra egenskaper påvirket av flere genvarianter og miljøfaktorer (multifaktorielle).

Monogene egenskaper følger i hovedsak tre nedarvingsmønster – autosomal, kjønnsbundet eller mitokondriell (figur 2.1-4).



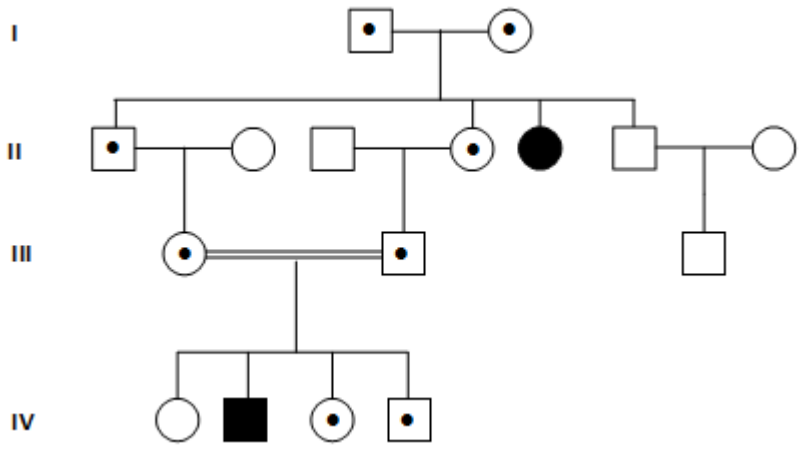
Figur 2.1-4: Genetisk nedarvingsmønstre.

Ved mitokondriell nedarving skjer genetisk overføring via mitokondrielt DNA fra mor til barn.

Som nevnt tidligere har mennesket to kjønnskromosomer, X og Y. Kvinner har XX og menn XY. Kjønnsbundet nedarving kan dermed være knyttet til X eller Y kromosomet. For X-bundet egenskaper kan nedarvingsmønsteret være recessivt eller dominant. Siden menn bare har ett Y kromosom vil Y-bundet egenskaper komme til uttrykk uavhengig av om egenskapen er recessiv eller dominat.

Dominante egenskaper trenger kun en kopi av en genvariant for å komme til uttrykk. Et individ som uttrykker en slik egenskap kan derfor være heterozygot – eller homozygot for det gitte allelet. Individer som uttrykker en recessiv egenskap må derimot ha to kopier av allelet– være homozygot.

Autosomale egenskaper er knyttet til gener med loci på et av de 22 autosomene og følger recessiv eller dominant nedarvingsmønster. Det er egenskaper knyttet til autosomal recessiv arvegang som er diskutert i denne oppgaven. For å utrykke en slik egenskap må begge foreldrene til et individ ha minst en kopi av det recessive allelet. I tilfeller hvor begge foreldrene er heterozygote for det recessive allelet vil gjennomsnittlig ¼ av barna deres være homozygote for det gitte allelet (figur 2.1-5).



Figur 2.1-5: Pedigree med autosomal recessiv arvegang*. Symbolbruk: Firkant = mann; sirkel = dame; prikk = heterozygot for det recessive allelet; svart firkant/sirkel = homozygot for det recessive allelet. Enkel strek mellom to individer = foreldre; dobbel strek mellom to individer = beslektede foreldre (Ziegler & König 2010)*

Det er verdt å merke seg at en høy andel individer i befolkningen er bærere av sjeldne recessive genvarianter. For ubeslektede foreldre vil disse egenskapene sjeldent bli uttrykt da to individer som oftest ikke er bærere av den samme genvarianten.

2.1.3 Hardy-Weinberg likevekt

Hardy-Weinberg likevekt beskriver en teoretisk sammenheng mellom allelfrekvens og genotypefrekvens fra generasjon til generasjon. Dersom allelfrekvensen er kjent i foreldregenerasjon kan genotypefrekvens hos avkom forutsies som vist i tabell 2.1-2 (Klug et al. 2010).

Tabell 2.1-2: Forventet genotypefrekvens gitt Hardy-Weinbergs likevektsfunksjon.
p = sannsynlighet for referanseallel (A); q = sannsynlighet for alternativt allel (B).

Genotype	Forventet genotypefrekvens
AA	p^2
AB	$2pq$
BB	q^2

For SNP'er med locus i eksonene er gjennomsnittlig MAF (q) estimert til å være 23.6 % (Yang et al. 2013). Det er imidlertid verdt å merke seg at dette estimatet gjelder normale SNP'er i en gitt befolkning og ikke SNV'er generelt.

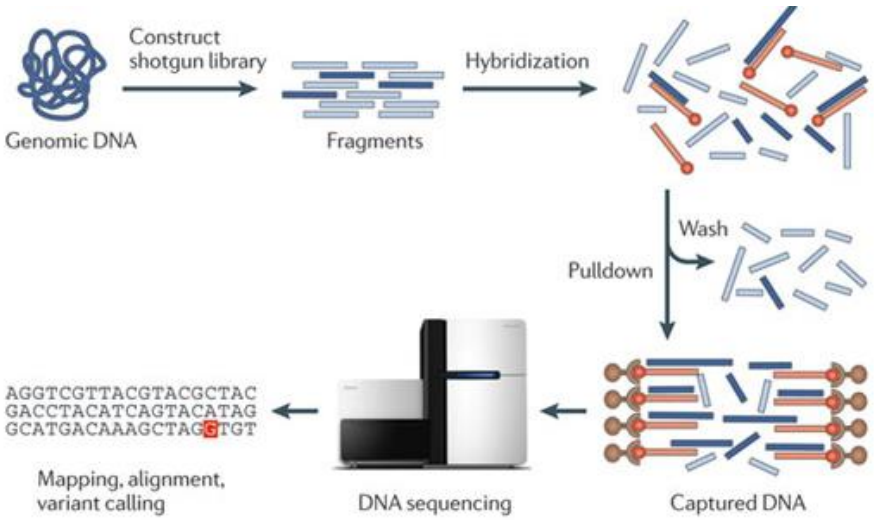
2.1.4 Eksomsekvensering

I følge Ng et al. (2010) er eksomsekvensering en god metode for å kartlegge allel- og fenotypevariasjon bakenforliggende monogene egenskaper. Dette fordi de fleste monogene egenskaper er forårsaket av en mutasjon i kodende område av et gen. Metoden er billig, forholdsvis rask og det trengs lite genmateriale. Ulempen er imidlertid at man mister informasjon knyttet til de ikke-kodende områdene av genomet, og antall SNP er ofte begrenset.

Eksomsekvensering benytter seg av storskala-sekvensering. Bamshad et al. (2011) har beskrevet de generelle stegene for denne metoden (figur 2.1-6):

- Genomisk DNA blir tilfeldig kuttet til korte fragmenter av en gitt lengde. En rekke enzym- og ligeringsreaksjoner benyttes for å amplifisere (oppkopiere) fragmentene slik at ønskelig andel genmateriale oppnås.
- Deretter tilføres spesifikke prober – oligonukleotider merket med biotin. En oligonukleotid er et kort, enkelttrådet DNA eller RNA fragment. Ved hybridisering vil fragmentene komplementære til probene komme sammen. Denne metoden kalles eksom-fanging.

- Overflødige fragmenter vaskes bort, mens fragmenter festet til oligonukleotider amplifiseres på nytt.
- En sekvensator gjennomfører storskalasekvensering av eksomet. Det finnes mange ulike sekvensatorer på markedet. De varierer noe i teknisk utførelse, men resultatet er relativt likt.
- Resultat fra eksomsekvensering (eksomdata) kan for eksempel brukes til kartlegging, sekvenssammenstilling og variasjonsdeteksjon.

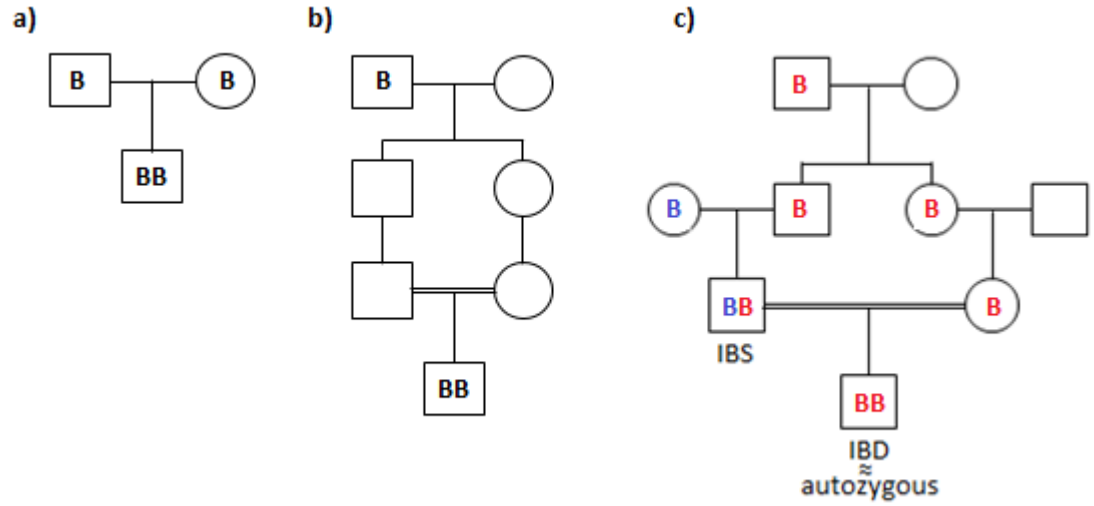


***Figur 2.1-6: Generelle steg for eksomsekvensering**. (Bamshad et al. 2011)*

2.2 Inngifte og autozygositet

To alleler ved et gitt locus er Identical-By-Descent (IBD) når de er nedarvet fra en felles forfar (figur 2.2-1). Definert slik er IBD-begrepet relatert til et gitt pedigree, og forbundet med begrepet autozygot, da de to allelene er homozygote.

Å være homozygot betyr at to alleler ved et gitt locus er identiske. Når de to identiske allelene ikke har samme opphav i pedigreet er de definert som Identical-By-State (IBS) (figur 2.2-1).



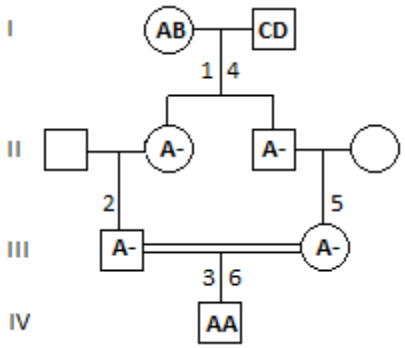
***Figur 2.2-1**: **IBS versus IBD**. a) to alleler IBS b) to alleler IBD c) begge tilstandene sett i et pedigree.*

2.2.1 Utregning av inngiftekoeffisienten

Inngiftekoeffisienten (*f*) til et individ i et gitt slektstre er sannsynligheten for at et tilfeldig autosomalt locus er autozygot. Andelen autozygositet er dermed avhengig av slektskapsforhold, eller sagt på en annen måte, antall meioser fra et individ til en forfar (McQuillan et al. 2008).

I denne oppgaven er hovedfokuset rettet mot barn av fetter og kusine. I dette tilfellet blir inngiftekoeffisienten lik 1/16, som vist i følgende utregning:

> *Oldeforeldrene til et barn av fetter og kusine har allelene A, B, C, D ved et gitt locus. Hva er sannsynligheten for at dette barnet er IBD for allel A?*



Figur 2.2-2: Slektstre der et barn av fetter og kusine er autozygot for allel A.
Firkant=mann, sirkel=dame

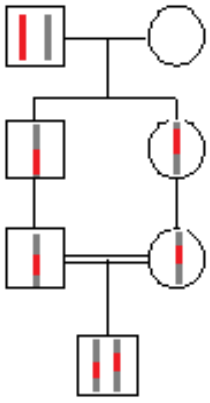
For at barnet skal være autozygot for allel A, må allelet nedarves gjennom seks meioser (n), tre på farssiden (nummerert 1, 2, 3 i figur 2.2-2) og tre på morssiden (nummerert 4, 5, 6 i figur 2.2-2). Sannsynligheten for dette er:

$$P(AA) = \left(\frac{1}{2}\right)^n = \left(\frac{1}{2}\right)^6 = 0.015625$$

Imidlertid kan barnet også være autozygot for et av de andre tre allelene. Siden dette er disjunkte hendelser, der P(AA) = P(BB) = P(CC) = P(DD), blir inngiftekoeffisienten:

$$f = 4\left(\frac{1}{2}\right)^6 = \frac{1}{16} = 0.0625$$

Under hver meiose vil det være mulighet for rekombinasjon der den genetiske sammensetningen endres (figur 2.2-3). Med fjernere slektskap viser inngiftekoeffisienten til gradvis kortere og færre autozygote områder spredt utover genomet. For eksempel er andelen autozygositet hos barn av onkel-niese 1/8 – mye høyere enn for barn av tremenninger der andelen er 1/64.



Figur 2.2-3: Rekombinasjon ved 6 meioser *fører til at IBD-området (rødt felt) gradvis blir mindre.*

2.2.2 Medisinske konsekvenser ved høy inngfitekoeffisient

Av rapport 2007:2 fra folkehelseinstituttet (Surén et al. 2007, s. 3-9) presenteres en tabell (her tabell 2.2-1) over medisinske konsekvenser for barn av beslektede og ubeslektede foreldre.

Tabellen er korrigert for andre risikoer ved hjelp av regresjonsanalyse, hvor det er kontrollert for mors alder, fødselstall, utdanningsnivå, sosioøkonomiske status og barnets fødselsår.

Tabell 2.2-1: Relativ risiko av medisinske konsekvenser for inngifte og ikke-inngifte.
1= søskenbarn eller nærmere beslektet, 2= fjernere beslektet enn søskenbarn, 3= beslektet, men ukjent slektskap. (Surén et al. 2007)

Utfall	Slektskapskategori			
	1	2	3	Ubeslektet
Dødfødsel	1,63 (1,39 – 1,91)	0,99 (0,81 – 1,21)	1,22 (1,02 – 1,46)	1,0
Spedbarnsdød	2,43 (2,11 – 2,79)	1,42 (1,19 – 1,69)	1,35 (1,12 – 1,62)	1,0
Medfødte misdannelser	2,0 (1,8 – 2,2)	1,3 (1,1 – 1,5)	1,1 (1,0 – 1,3)	1,0
Risiko for død	1,75 (1,45 – 2,11)	1,24 (1,04 – 1,49)	0,96 (0,76 – 1,22)	1,0

Beregninger basert på data fra MFR og SSB for barn født f.o.m. 1967 – 1. halvår 2005. Hele tidsperioden er inkludert, og variasjoner over tid er korrigert for ved å justere for barnets fødselsår i regresjonsanalysen. Det er også justert for mors alder, mors fødselstall og mors utdanningsnivå.

Resultatet bekrefter at medisinske konsekvenser som dødfødsel, spedbarnsdød og medfødte misdannelser forekommer hyppigere hos barn av beslektede foreldre – spesielt der slektskapet er mellom søskenbarn.

Av tabellen kommer det fram at det er 63 % høyere risiko for dødfødsel dersom beslektede søskenbarn får barn kontra ubeslektede personer. I tillegg er risikoen for spedbarnsdød 1,01 gang så høy for søskenbarn som for beslektede foreldre av fjernere slektskap. Dette er totalt 2,43 ganger høyere risiko enn for barn av ubeslektede foreldre.

Tabellen indikerer også at medfødte misdannelser øker med nærmere slektskap, der barn av søskenbarn har dobbelt så stor sjans for å arve en slik egenskap kontra barn av ubeslektede foreldre.

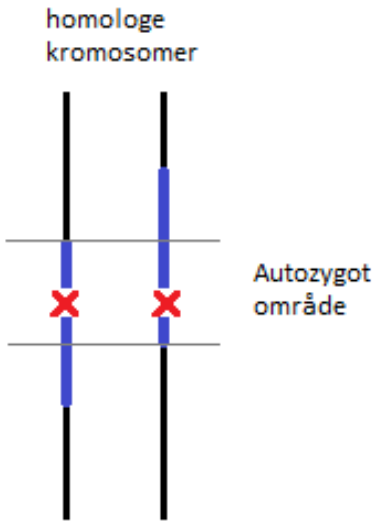
Risikoen for død gjennom hele livet grunnet genetiske konsekvenser som opptrer etter spedbarnsalderen er 51 % høyere for barn av søskenbarn kontra barn av beslektede foreldre med fjernere slektskap. I forhold til barn av ubeslektede forledre er sannsynligheten 75 % høyere.

Setesdalen har en historie om høy inngiftekoeffisient, da mange familier på midten av 1800-tallet opplevde at familiemedlemmer begynte å få ukontrollerbare rykninger i 30-40 års alderen. Tilstanden ble kalt setesdalsrykkja, før den ble kjent som Huntingtons sykdom (Lie 2008).

Et annet eksempel er Habsburg familien, en spansk kongeslekt på 15- og 1600 tallet. Dette dynastiet gikk til grunne da Charles II manglet en trone arving da han døde bare 39 år gammel. Da var han erklært impotent og led av en rekke fysiske- og mentale lidelser nedarvet gjennom generasjoner (Haraldsen 2009).

2.2.3 Autozygositetskartlegging

Autozygositetskartlegging, også kalt homozygositetskartlegging, ble introdusert av Lander og Botstein (1987). Metoden er basert på at barn av inngiftede foreldre vil ha et lengre autozygot område rundt et recessivt sykdomslocus (figur 2.2-4).



Figur 2.2-4: Autozygositetskartlegging*: Blå områder indikerer alleler nedarvet fra en felles forfar. Det autozygote området indikerer hvor den sykdomsrelaterte genvarianten (rødt kryss) kan ha sitt locus.*

Autozygositetskartlegging har blitt et viktig verktøy for å kartlegge recessive genvarianter i inngiftefamilier, da man kun trenger å undersøke relevante lange homozygote områder. Dersom

sykdommen virkelig skyldes inngifte vil sykdomslocuset ligge her. (Carr et al. 2013; Moltke et al. 2011).

Tradisjonelt er autozygositetskartlegging utført med SNP genotyping da SNP'er er lette å identifisere og jevnt fordelt utover genomet – som nevnt ca. 1 SNP per 300 base (kapitel 2.1.1). For eksomdata er situasjonen annerledes, da man mister informasjon mellom eksonene slik at markører blir ujevnt fordelt (figur 2.2-5). Dermed blir autozygositetskartlegging mer utfordrende ved bruk av eksomdata. I tillegg vil bakgrunnshomozygositet, homozygote områder som er vanlige i befolkningen, være et problem. Dette fordi områdene vil gi mer skjevhet i dataene da de ikke er autozygote områder. Dette gjelder i midlertid for autozygositetskartlegging både med data fra eksomsekvensering og med SNP-arrays.

Figur 2.2-5: Fordeling av SNP'er langs et kromosom. *De svarte strekene viser hvordan SNP'er er fordelt lang et eksom i genomdata og i eksomdata (En grå boks illustrerer et ekson).*

2.3 Skjult markovmodell

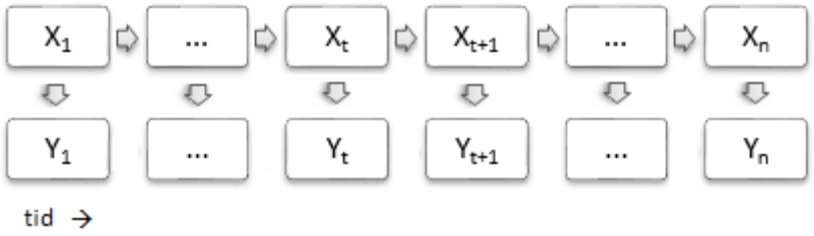
En markovmodell består av observerbare tilstander, $\{X_t\ ,\ \mathrm{t} = 1, 2, \ldots, \mathrm{n}\}$, der X-ene er stokastiske, diskrete variabler med et endelig utfallsrom $H = \{1, \ldots, N\}$. I en tidsdiskret modell er det endelige tilstandsrommet $t = \{0, 1, 2, \ldots\}$ hvor man observerer overgangen fra tilstand i til tilstand j ved tiden t, $P_{i,j} = P(X_{t+1} = j | X_t = i)$, (figur 2.3-1):

Figur 2.3-1: Markovmodell illustrert ved trellis-diagram*.Boksene indikerer de observerbare tilstandene mens pilene indikerer overgangen fra en tilstand til en annen ved tiden t.*

En stokastisk prosess har en markovegenskap dersom framtidige tilstander er betinget nåværende tilstand og uavhengig av tidligere tilstander ved et gitt tidsintervall (Ewens & Grant 2005).

En skjult markovmodell (HMM) er en generalisering av en ordinær markovmodell og dermed også mer fleksibel. Skjulte markovmodeller har lenge vært et nyttig verktøy innenfor tale- og håndskrift gjenkjenning, bildeanalyse, økonometri og bioinformatikk (Newberg 2009), da modellen sier noe om rekkefølgen av hendelser.

I en skjult markovmodell kan man ikke observere tilstander $\{X_t\,, t = 1, 2, \dots, n\}$ direkte – disse er skjult for oss. Derimot observerer man signaler $\{Y_t, t = 1{,}2, \dots, n\}$ knyttet til disse tilstandene, og på den måten predikeres de skjulte tilstandene indirekte. De observerbare signalene er på lik linje med *X*-ene stokastiske, diskrete. Dermed vil en skjult markovmodell bestå av et endelig tilstandsrom av skjulte tilstander $H = \{1, \dots, N\}$ og et endelig signalrom av observerbare signaler, $S = \{1, \dots, K\}$ (figur 2.3-2).



Figur 2.3-2: Skjult markovmodell illustrert ved trellis-diagram. *To stokastiske prosesser, der de observerbare signalene $\{Y_t\,, t = 1{,}2, \dots, n\}$ er avhengig av de skjulte tilstandene $\{X_t\,, t = 1{,}2, \dots, n\}$. Horisontale piler indikerer overgangen fra en tilstand til en annen ved tiden t, mens vertikale piler indikerer sannsynligheten for å observere et signal gitt en tilstand.*

I figur 2.3-1 og 2.3-2 indikerer horisontale piler transisjonssannsynligheter. Det vil si sannsynligheter for å gå fra en tilstand til en annen gitt markovkjeden, $p_{i,j} = P\big(X_{t+1} = H_j \big| X_t = H_i\big)$ for $1 \le i, j \le N$. Disse sannsynlighetene er representert i en transisjonsmatrise (T):

$$T = \begin{bmatrix} p_{1,1} & \cdots & p_{1,N} \\ \vdots & N \times N & \vdots \\ p_{N,1} & \cdots & p_{N,N} \end{bmatrix}$$

Radene i transisjonsmatrisa representerer nåværende tilstand (tilstanden man går fra) og kolonnene representerer neste tilstand (tilstanden man går til). Følgelig må radene i T summere til 1.

De vertikale pilene i figur 2.3-2 indikerer emisjonssannsynligheter, det vil si sannsynligheter for å observere de ulike signalene gitt tilstanden i den underliggende markovkjeden, $p_{jk} = P(Y_t = S_k | X_t = H_j)$ for $1 \leq j \leq N \ \ og \ \ 1 \leq k \leq K$. Disse sannsynlighetene er representert i en emisjonsmatrise (E):

$$E = \begin{bmatrix} p_{1,1} & \cdots & p_{1,K} \\ \vdots & N \times K & \vdots \\ p_{N,1} & \cdots & p_{N,K} \end{bmatrix}$$

Radene i emisjonsmatrisa representerer de skjulte tilstandene, mens kolonnene representerer de observerbare signalene som kan inntreffe. Derav må radene summere til 1.

Informasjon om hvordan tilstandssekvensen starter er representert i en initialvektor, $I = \pi_i$, hvor initialsannsynlighetene $\pi_i = P(X_1 = H_i)$ for $1 \leq i \ \leq N$. På lik linje med radene i transisjonsmatrisa må disse sannsynlighetene summere til 1.

For å oppsummere, er en HMM betinget en stokastisk prosess med kjent markovegenskap, hvor fem elementer må være tilstede:

1. Skjulte tilstander:

$$H = \{1, \dots, N\}$$

2. Observerbare signaler:

$$S = \{1, \dots, K\}$$

3. Transisjonssannsynligheter:

$$p_{i,j} = P(X_{t+1} = H_j | X_t = H_i) \quad for \ \ 1 \leq i, j \leq N$$

4. Emisjonssannsynligheter:

$$p_{jk} = P\left(Y_t = S_k \middle| X_t = H_j\right) \quad for\ 1 \leq j \leq N, 1 \leq k \leq K$$

5. Initialsannsynligheter:

$$\pi_i = P(X_t = H_i) \quad for\ 1 \leq i \leq N$$

Følgelig er det komplette parametersettet til en HMM: $\lambda = (T, E, I)$.

3. Materialer og metoder

3.1 Simulerte data

For å teste algoritmene og for å kunne sammenligne de med Plink, er det konstruert simulerte data basert på skriptet i tillegg B.4, og R pakkene paramlink_0.9-6 (Vigeland 2014c) og IBDsim_0.9-3 (Vigeland 2014b).

Simulert data er forsøkt gjort mest mulig realistisk, slik at deteksjon av autozygote områder vil stemme godt overens med hva som er forventet ved eksomsekvensering.

Selve simulerings funksjonen «simulation» er definert i skriptet i tillegg B.5 og er utført som beskrevet under. De to første stegene gjøres med IBDsim og tredje steget ved hjelp av skriptet i tillegg B.6:

Pedigree

- Konstruerer pedigree med slektskapsforhold mellom fetter og kusine med ett felles barn (individ 9, figur 3.1-1).

IBD mønster

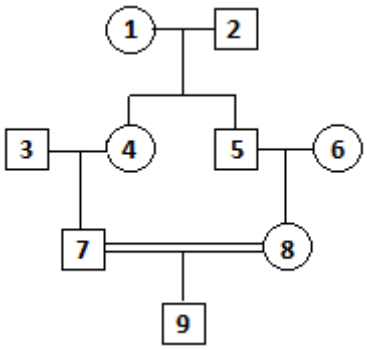
- Alle 'founderne' (individ 1, 2, 3, 6 figur 3.1-1) får unike alleler.
- Setter det genetiske kartet: deCode rekombinasjonskart (cM).
- Alle autosomale kromosomer simuleres, hvor rekombinasjon i hver meiose fra forfedre til siste ledd i pedigree blir simulert basert på det genetiske kartet. Avstanden mellom rekombinasjonshendelsene følger χ^{2}-modell, da denne modellen håndterer interferens.
- Lager en liste over alle autozygote og ikke-autozygote områder hos individ 9, basert på simulerte meioser.

Geno-typer

- Setter distanse mellom markører: basert på fysiske markørposisjoner (Mb) fra et ekte eksom.
- Setter minor allele frekvens: basert på 1000 Genomes Project.
- Setter forventet andel bakgrunnshomozygositet: ulike lengder av falske homozygote områder (Mb) settes inn tilfeldig.
- Simulerer genotyper for individ 9 betinget på IBDsim-simuleringen, MAF og ukorrekt genotyping.
- Markører homozygot for referanseallelet blir fjernet for at simulert data skal bli mest mulig likt eksomdata.
- En dataramme med 5 kolonner (MB, CM, al1, al2, trueIBD) blir returnert.

Output

- En liste med snps = genotypeinformasjon, fasit = sanne autozygote områder og map = genetisk kart



Figur 3.1-1: Fetter-kusine pedigree*. Tallene 1-9 indikerer et individs ID.*

3.2 Reelle data

VCF (Variant Call Format) er benyttet for reelle data. Dette filformatet anvendes av bioinformatikere, da det er velegnet til å lagre informasjon om sekvensvariasjon. For mer informasjon, se 1000Genomes: *http://www.1000genomes.org/.* Skriptet i tillegg B.7 henter ut informasjon om genotyper, allelfrekvenser og posisjoner fra et slikt format.

Det er benyttet eksomdata fra en pasient der en gitt sykdom er antatt monogen med autosomal recessiv arvegang. Markørene i datasettet er irreversibelt maskert for å unngå juridiske og etiske problemstillinger. Det vil si, alle variantposisjoner, navn på gener, navn på transkript samt markør ID kan ikke spores tilbake til gitt pasient.

I forbindelse med diagnostikk av denne pasienten ble det utført rutinemessig homozygositetskartlegging ved et genomvidt SNP-array med 60 000 SNP'er. Dette er relativt få, så det kan være at enkelte korte autozygote områder ikke er fanget opp. Det er allikevel valgt å bruke dette som «fasit».

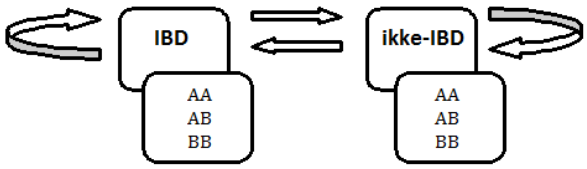
3.3 Metode: skjult markovmodell

Den skjulte markovmodellen beskrevet her baserer seg på artikkelen til Leutenegger et al. (2003) hvor IBD prosessen med tilhørende parametere langs et genom estimeres. Leutenegger et al. benytter genomdata fra individer der slektskap mellom foreldre er kjent, men der pedigree er ukjent. Den skjulte markovmodellen anvendt her, benytter derimot eksomdata fra individer der foreldrene er beslektet og hvor slektskapet og pedigree er kjent.

Et barn av beslektede foreldre, kontra barn av ubeslektede foreldre, vil ha større sannsynlighet for å arve monogene egenskaper grunnet IBD. Imidlertid vil man kun observere om markører i eksomet er homozygote eller heterozygote. Dermed kan man anta at eksomdata har egenskapen til en skjult markovmodell, slik at tilstander bakenforliggende genotyper kan estimeres (Browning & Browning 2012).

Basert på definisjonen av IBD (kapittel 2.2) vil en skjult tilstand enten være IBD eller ikke-IBD. Følgelig vil skjulte tilstander $\{X_t\,, t = 1, 2, \dots, n\}$ og observerbare genotyper $\{Y_t\,, t = 1, 2, \dots, n\}$ ha tilstandsrom $H = \{IBD,\ ikke\text{-}IBD\}$ og signalrom $S = \{AA,\ AB,\ BB\}$ (figur 3.3-1).

Tilstandsendringene $t\ \epsilon\ \{1, \dots, k\}$ skjer ved diskrete markører ved gitte kromosomposisjoner lags et kromosom.



Figur 3.3-1: Observerbare genotyper (AA, AB, BB) og bakenforliggende skjulte tilstander (IBD, ikke-IBD).

Transisjonsmatrise

Transisjonsmatrisa er som beskrevet av Leutenegger et al:

$T =$

	IBD	Ikke-IBD
IBD	$(1-qk)\cdot f + qk$	$(1-qk)\cdot(1-f)$
Ikke-IBD	$(1-qk)\cdot f$	$(1-qk)\cdot(1-f)+qk$

Her er $qk = e^{-a\cdot d_k}$ hvor d_k er den genetiske distansen i cM mellom markør $t + 1$ og t.

Transisjonssannsynlighetene bygger på to antakelser:

1. Fravær av genetisk interferens. Det vil si at overkrysning skjer uavhengig av hverandre, slik at rekombinasjonsfrekvensen er konstant langs et kromosom (Clark 1999). I motsatt fall, ved interferens, vil en overkrysning mellom to søsterkromatider minke sannsynligheten for andre overkrysninger i nærheten.
2. Det genetiske kartet, basert på rekombinasjonsfrekvensen, er kjent og uten feil.

Basert på disse antakelsene, benyttes Haldanes kartleggingsfunksjon til å finne distansen mellom to markører, der forholdet mellom rekombinasjonsfrekvens (r) og genetisk distanse (d) følger en Poissonfordeling.(Ziegler & König 2010):

$$r = \frac{1}{2}(1 - e^{-2d})$$

$$d = -\frac{1}{2}\ln(1 - 2r)$$

De resterende to parameterne, *a* og *f*, er beskrevet på følgende måte:

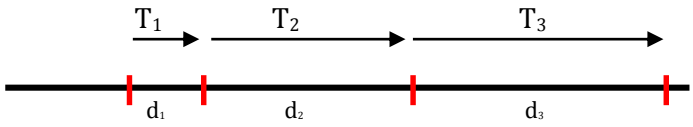
> Det er antatt at den momentane forandringen mellom tilstandene i tilstandsrommet $H = \{IBD,\ ikke\text{-}IBD\}$ har følgende ventetid: IBD er eksponensielt fordelt med α, og ikke-IBD er eksponensielt fordelt med λ.
>
> Videre er det antatt at sannsynligheten for IBD = *f*, og sannsynligheten for ikke-IBD = 1 - *f*, der *f* er inngiftekoeffisienten. Dermed kan markovprosessen forklares ved $\alpha = a(1 - f)$ og $\lambda = af$.
>
> Hovedfokuset i denne oppgaven er barn av fetter og kusine, men implementert HMM håndterer også andre slektskapsforhold.
>
> I denne oppgaven er det benyttet kjent pedigree med fetter-kusine inngifte (figur 3.1-1). Som beskrevet i kapittel 2.2.1 er inngiftekoeffisienten ved slike slektskapsforhold $\frac{1}{16} = 0.0625$. Denne sannsynligheten er basert på nedarvingsmønster gjennom seks meioser. Følgelig antas det 6 overkrysninger per Morgan. Omgjort til cM blir denne raten $6/100 = \alpha$.
>
> Dermed kan *a* estimeres slik:

$$\alpha = a(1 - f)$$

$$a = \frac{\alpha}{(1 - f)}$$

$$a = \frac{6}{100} \cdot \frac{16}{15} = 0.064$$

Til slutt, siden avstanden mellom markører varierer, er den underliggende markovkjeden inhomogen: Det vil si at hver transisjonsmatrise T_i er avhengig av avstanden d_k, illustrert i figur 3.3-2.



Figur 3.3-2: Inhomogen markovmodell. *Røde vertikale streker indikerer markører langs et kromosom (svart linje), En ny trasisjonsmatrise estimeres for hvert steg (svarte piler). Transisjonsmatrisene T_1, T_2 og T_3 er avhengig av henholdsvis distansene d_1, d_2, d_3 mellom markørene.*

Emisjonsmatrise

Som nevnt, kan man bare observere genotypen til markører langs et kromosom i det gitte signalrommet $S = \{AA,\ AB,\ BB\}$. Betinget HMM er sannsynligheten for Y_t bestemt av X_t, og en funksjon av allelfrekvensen til markøren ved kromosom posisjon t.

Den skjulte markovmodellen håndterer skiftet mellom IBD og ikke-IBD markører, basert på om observert markør er homozygot eller heterozygot. Dersom et større antall markører på rad er homozygote, indikerer dette et autozygot område. Et autozygot område vil derfor starte med en homozygot markør og fortsette fram til en heterozygot markør avslutter sekvensen. Dersom en heterozygot markør skulle opptre inni et lengre IBD område grunnet genotypingsfeil, vil dette påvirke korrekt deteksjon av autozygote områder. For å håndtere dette er det lagt til en feilrate for emisjonssannsynligheter der den bakenforliggende tilstanden er IBD. Emisjonsmatrisa er, som hos Leutenegger et al., basert på Hardy-Weinberg likevekt:

E =

	AA	AB	BB
IBD	$(1-\varepsilon)p + \varepsilon p^2$	ε2pq	$(1-\varepsilon)q + \varepsilon q^2$
Ikke-IBD	p^2	$2pq$	q^2

Initialvektor

Initialvektoren inneholder sannsynlighetene for hvordan tilstandssekvensen starter. I praktiske anvendelser er denne sannsynligheten ukjent, men det er kjent at inngiftekoeffisienten estimerer autozygositet i genomet. Følgende initialvektor er derfor benyttet:

I =

IBD	f
Ikke-IBD	$1\text{-}f$

3.3.1 Forward-backward algoritmen

Hensikt

Hensikten med forward-backward algoritmen er å finne sannsynligheten for at en gitt kromosomposisjon er IBD eller ikke-IBD gitt alle observasjonene, $P(X_t|Y_{1:n})$. Med andre ord, en lokal optimering der det er mulig å evaluere hver tilstand langs et kromosom.

$$P(X_t|Y_{1:n}) = \frac{P(X_t, Y_{1:n})}{P(Y_{1:n})} = \frac{P(X_t, Y_{1:t}, Y_{t+1:n})}{P(Y_{1:n})} = \frac{P(X_t, Y_{1:t}) \cdot P(Y_{t+1:n}|X_t)}{P(Y_{1:n})}$$

Algoritmen

Faktorene $P(X_t, Y_{1:t})$ og $P(Y_{t+1:n}|X_t)$ i ligning ovenfor er henholdsvis forward- (F) og backward (B) sannsynligheter der elementene i fordelingene er:

$$F_t(i) := P(X_t = i, Y_{1:t})$$

$$B_t(i) := P(X_{t+1:n} | X_t = i)$$

En videre beskrivelse av disse sannsynlighetene er gitt i henholdsvis forward- og backward algoritmen (Rabiner 1989; Ross 2007):

Forward algoritmen:

1. Initialisering

$$F_1(i) = I_i \cdot E_i(Y_1) \quad for\ 1 \leq i \leq N$$

2. Induksjon

$$F_{t+1}(j) = P(X_{t+1} = j, Y_{1:t+1}) \qquad for\ 1 \leq t \leq T-1 \\ og\ 1 \leq j \leq N$$

$$= \sum_i P(X_{t+1} = j, X_t = i, Y_{1:t+1})$$

$$= \sum_i P(X_{t+1} = j, X_t = i, Y_{1:t}, Y_{t+1})$$

$$= \sum_i P(X_{t+1} = j, Y_{t+1} | X_t = i, Y_{1:t}) \cdot P(X_t = i, Y_{1:t})$$

$$= \sum_i P(X_{t+1} = j, Y_{t+1} | X_t = i) \cdot P(X_t = i, Y_{1:t})$$

$$= \sum_i T_{i,j} \cdot E_{j,Y_{t+1}} \cdot F_t(i)$$

3. Terminering

$$P(Y|\lambda) = \sum_{i=1}^{N} F_T(i)$$

Backward algoritmen

1. Initialisering

$$B_T(i) = 1 \qquad for\ 1 \leq i \leq N$$

2. Induksjon

$$B_t(i) = P(Y_{t+1:n} | X_t = i)$$

$$= \sum_j P(Y_{t+1}, Y_{t+2:n} | X_t = i, X_{t+1} = j) \cdot P(X_{t+1} = j | X_{t=i})$$

$$= \sum_j P(Y_{t+1}, Y_{t+2:n} | X_{t+1} = j) \cdot P(X_{t+1} = j | X_{t=i})$$

$$= \sum_j P(Y_{t+1}|X_{t+1} = j) \cdot P(Y_{t+2:n}|X_{t+1} = j) \cdot T_{ij}$$

$$= \sum_j E_{jY_{t+1}} \cdot B_{t+1}(j) \cdot T_{ij}$$

A posteriori-sannsynligheter

Siste steg i forward-backward algoritmen beregner a posteriori-sannsynligheter. Det vil si den lokale optimeringen av en tilstand ved kromosomposisjon *t* normalisert over alle mulige observasjonssekvenser.

$$P(X_t = i|Y_{1:n}) = \frac{F_t(i) \cdot B_t(i)}{P(Y|\lambda)}$$

3.3.2 Viterbi-algoritmen

Hensikt

I utgangspunktet vil mange tilstandskombinasjoner generere samme sekvens av observasjoner. Viterbi-algoritmen finner den kombinasjonen av tilstander som best forklarer de observerte genotypene. Det vil si den mest sannsynlige tilstandssekvensen, $argmax_X P(X, Y|\lambda)$ for $X = X_{1:n}$, $Y = Y_{1:n}$ (Eddy 2004)

Algoritmen

Beskrivelsen av Viterbi-algoritmen er basert på artikkelen til Rabiner (1989):

To variabler, δ_t og ψ_t, holder orden på utregnede sannsynligheter slik at $argmax_X P(X, Y|\lambda)$ kan estimeres:

$\delta_t(i)$ er den høyeste sannsynligheten for en enkel sti ved kromosomposisjon *t*:

$$\delta_{t+1}(i) = \left[max_i \delta_t(i) T_{i,j}\right] \cdot E_j(Y_{t+1})$$

- $\psi_t(j)$ er en matrise som holder orden på hvilken sti som faktisk er mest sannsynlig for hver tilstand ved kromosomposisjon *t*.

1. Initialisering:

$$\delta_t(i) = I_i \cdot E_i(Y_1)\,, \qquad for\ 1 \le i \le N$$

$$\psi_1(i) = 0$$

2. Rekursjon:

$$\delta_t(j) = max_{1\le i\le N}\left[\delta_{t-1}(i) \cdot T_{i,j}\right] \cdot E_j(Y_t) \qquad for \quad 2 \le t \le T\ og \quad 1 \le j \le N$$

$$\psi_t(j) = argmax_{1\le i\le N}\left[\delta_{t-1}(i) \cdot T_{i,j}\right] \qquad for \quad 2 \le t \le T\ og \quad 1 \le j \le N$$

3. Terminering:

$$P^* = max_{1\le i\le N}[\delta_T(i)]$$

$$x_T^* = argmax_{1\le i\le N}[\delta_T(i)]$$

4. Tilbakesporing*:*

$$x_t^* = \psi_{t+1}(x_{t+1}^*) \quad for \quad t = T-1, T-2, \dots, 1$$

3.3.3 Implementering av algoritmene

Funksjon til forward-backward- og Viterbi-algoritmen er henholdsvis i tillegg B.1 og B.2. Begge funksjonene anvender skriptet i tillegg B.3. Dette skriptet inneholder de nødvendige funksjonene knyttet til den skjulte markovmodellen.

Som forklart under hensikt i kapittel 3.3.1 estimerer forward-backward algoritmen a posteriori-sannsynligheter for tilstander langs et kromosom. Dette er en lokal optimering for å finne den skjulte tilstanden i tilstandsrommet *H* som med størst sannsynlighet er tilstede ved en gitt kromosomposisjon *t* basert på en observasjonssekvens av genotyper.

Hensikten bak Viterbi-algoritmen (kapittel 3.3.2) er å finne en global optimering av skjulte tilstander langs et kromosom. Det vil si, finne den mest sannsynlige stien av skjulte tilstander i tilstandsrommet H basert på en observasjonssekvens av genotyper.

Begge algoritmene antar at den skjulte markovmodellen har et komplett parametersett (jamfør kapittel 2.3) med kjente transisjon-, emisjon- og initieringssannsynligheter.
Forward-backward og Viterbi funksjonen tar derfor de samme parametrene som input:

infile	En dataramme med fire kolonner gir informasjon om hver markør: markørposisjon (CM), referanseallel (al1), alternativt allel (al2) og frekvensen til alternativt allel (Bfreq). Antall rader er lik antall markører i datasettet.
error	Sannsynlighet for sekvenseringsfeil
a	Distansen mellom to tilstander er eksponensielt fordelt slik at a forklarer den momentane forandringen mellom IBD og ikke-IBD områder.
f	Inngiftekoeffisienten
logs	0 (standard) 1 for logtransformering. Ved logtransformering unngår man underflyt da sannsynlighetene blir additive. I tillegg vil det være lettere å skille mellom ekstremt små sannsynlighetsverdier.

Forward-backward funksjonen returnerer en $N \times 2$ matrise der kolonnene representerer markørposisjon (cM) og a posteriori-sannsynlighet for at den skjulte tilstanden ved en gitt markørposisjon er IBD. Antall rader er lik antall markører i datasettet.

Viterbi funksjonene returnerer en $N \times 2$ matrise der kolonnene representerer markørposisjon (cM) og informasjon om den skjulte tilstanden ved en gitt posisjon er IBD eller ikke-IBD. Antall rader er lik antall markører i datasettet.

3.4 Metode: Plink

Plink detekterer homozygote områder i genomet (tillegg A.1). Programmet kan dermed benyttes til å sammenligne hvor godt forward-backward- og Viterbi-algoritmen, beskrevet i kapittel 3.3, detekterer IBD områder i forhold til eksisterende programmer.

For at de to algoritmene skal bli strengest mulig bedømt, må Plink detektere flest mulige autozygote områder. Valg av parametere er derfor i henhold til dette, der fire parametersett er benyttet ved kjøring av --homozyg funksjonen i Plink (tabell 3.4-1).

***Tabell 3.4-1**: **Parametervalg ved kjøring av Plink**. Plink0 og plink1 er liberale. Plink2 er parametere tatt fra Pippucci et al. (2011). Plink3 er standard parametervalg i Plink.*

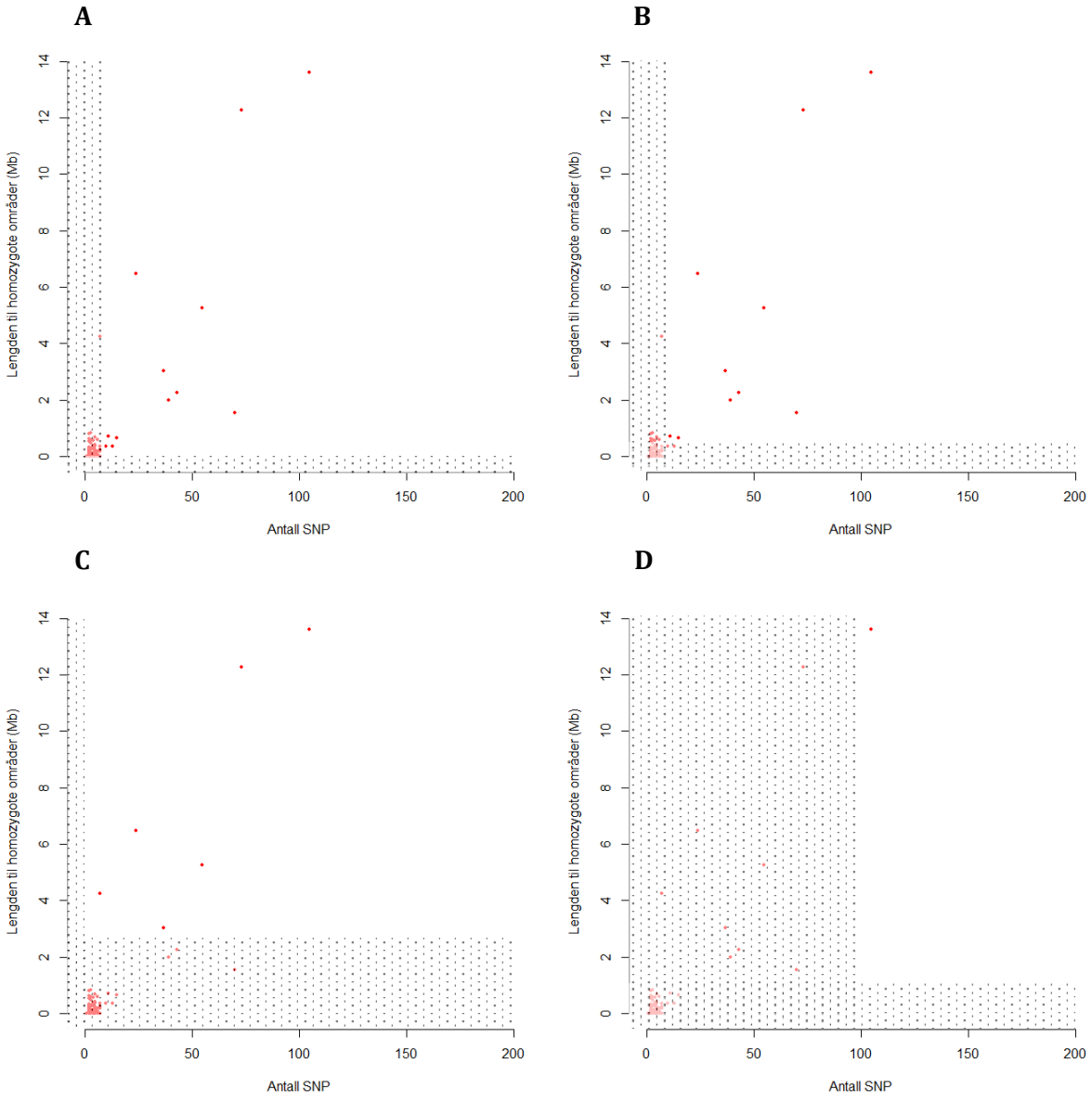
Plink parametere	plink0	plink1	plink2	plink3
--homozyg-window-kb	0	0	0	5000
--homozyg-window-snp	1	1	50	50
--homozyg-window-het	0	0	1	1
--homozyg-window-missing	0	0	5	5
--homozyg-window-threshold	1	1	0.05	0.05
--homozyg-snp	10	10	0	100
--homozyg-kb	0	500	2500	1000
--homozyg-density	10000	10000	50	50
--homozyg-gap	1000000	1000000	5000	1000

Plink0 definerer et autozygot område som minimum 10 homozygote SNP'er på rad, uten noen nedre grense for områdets lengde (figur 3.4-1 A). Den lave cut-offen er satt for å fjerne den verste støyen.

Plink1 har en noe høyere cut-off for å klassifisere et område som autozygot. Her må det være minimum 10 homozygote SNP'er på rad som spenner over et område på minst 500 kb (figur 3.4-1 B).

Plink2 er parametervalg tatt fra artikkelen ″EX-HOM (EXome HOMozygosity): A Proof of Principle″ (Pippucci et al. 2011) (figur 3.4-1 C).

Plink3 er Plinks standard parametervalg for --homozyg funksjonen. Parameterne er velegnet for deteksjon av autozygote områder med genotypedata fra SNP-arrays (figur 3.4-1 D).



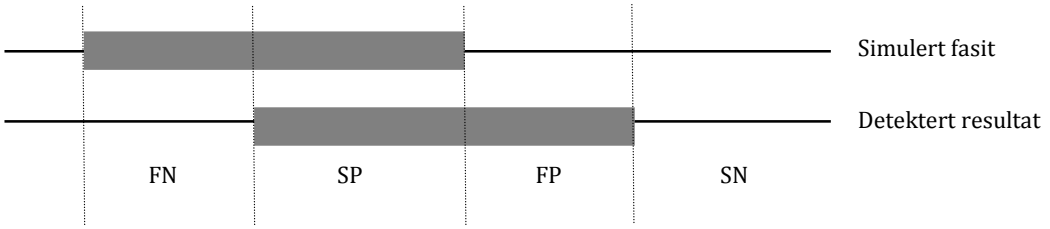
Figur 3.4-1: Hvordan parametervalg påvirker deteksjon av homozygote områder i Plink. *Homozygote områder under skraverte felt blir ikke detektert av Plink som sanne autozygote områder da de ikke oppfyller de gitte parameterkriteriene. A) plink0, B) plink1, C) plink2 og D) plink3. (Simulert data: barn av fetter og kusine).*

3.5 Validering av metoder

Det er beskrevet seks ulike metoder for å detektere autozygote områder ved eksomsekvensering. Kvaliteten til metodene ligger i hvor godt de detekterer sanne autozygote områder. Dermed er følgende hypotese antatt:

H_0: Ikke-autozygot område
H_1: Autozygot område

Validering av metodene er basert på sammenligning av simulerte autozygote områder opp mot detekterte autozygote områder langs et kromosom, som vist i figur 3.5-1.



Figur 3.5-1: Hvordan sann positiv, falsk positiv, falsk negativ og sann negativ klassifiseres *i forhold til hypotesen H_0: ikke-autozygot område VS H_1: autozygot område. Horisontale linjer = eksomsekvens; grå boks = autozygot område.*

Sensitivitet også kalt sann positiv rate (SPR) er andelen detekterte autozygote områder blant alle sanne autozygote områder. Verdien gir en indikasjon på hvor god modellen er til å klassifisere autozygositet.

$$sensitivitet = \frac{Sanne\ positive}{sanne\ positive + falske\ negative}$$

Spesifisitet også kalt sann negativ rate (SNR) er andelen detekterte ikke-autozygote områder blant alle sanne ikke-autozygote områder. I motsetning til sensitivitet, gir denne verdien en indikasjon på hvor godt modellen klassifiserer ikke-autozygote områder.

$$spesifisitet = \frac{Sanne\ negative}{falske\ positive + sanne\ negative}$$

Falsk positiv rate (FPR) indikerer hyppigheten av type-I feil, mens falsk negativ rate (FNR) indikerer hvor ofte modellen gjør feil av type-II.

$$FPR = \frac{falske\ positive}{falske\ positive + sanne\ negative} = 1 - spesifisitet$$

$$FNR = \frac{Falske\ negative}{sanne\ positive + falske\ negative} = 1 - sensitivitet$$

En optimal modell indikeres ved at SPR og SNR er tilnærmet lik 1, mens FPR og FNR er tilnærmet lik 0. Det kontrolleres for SPR, da det er ønskelig å finne en metode som detekterer flest mulig sanne autozygote områder. I tillegg kontrolleres det for FPR slik at resultatene inneholder minst mulig støy.

Validering av metodene er basert på total lengde av sanne positive, falske positive, falske negative og sanne positive områder, sammenlignet med total lengde av simulerte autozygote områder eller ikke-autozygote områder:

$$SPR = \frac{SP}{sann\ IBD} \qquad FPR = \frac{FP}{sann\ ikke{-}IBD} \qquad FNR = \frac{FN}{sann\ IBD} \qquad SNR = \frac{SN}{sann\ ikke{-}IBD}$$

Resultat fra N simulerte genom samles i en $N \times 4$ matrise, der kolonnene representerer SPR, FPR, FNR og SNR (jamfør skriptene i tillegg B.4 og B.5).

4. Resultater

4.1 Simulerte data

Det er blitt gjort 500 simuleringer på 8 ulike simulerte data, der feilrate og bakgrunnshomozygositet varierer (tabell 4.1-1). For å oppnå mest mulig realistiske simulerte data, er simuleringene gjennomført med χ^2-modell og deCode rekombinasjonskart.

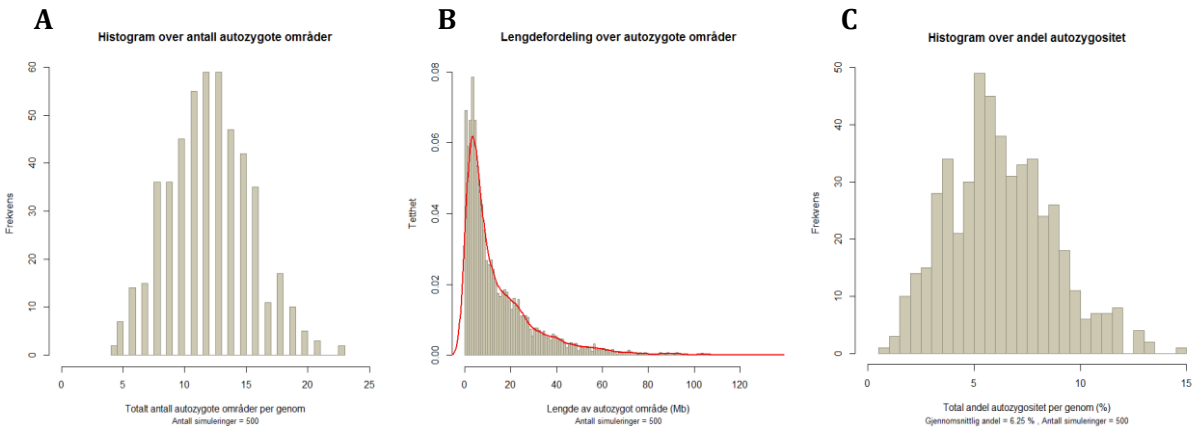
***Tabell 4.1-1: Oversikt over de 8 simulerte datasettene benyttet**. Parametervalg for feilrate og bakgrunnshomozygositet varierer.*

	Feilrate	Bakgrunnshomozygositet
Simdata1	0.0001	1Mb*22chr = 22Mb
Simdata2	0.001	1Mb*22chr = 22Mb
Simdata3	0.005	1Mb*22chr = 22Mb
Simdata4	0.01	1Mb*22chr = 22Mb
Simdata5	0.0001	5Mb*22chr = 110Mb
Simdata6	0.001	5Mb*22chr = 110Mb
Simdata7	0.005	5Mb*22chr = 110Mb
Simdata8	0.01	5Mb*22chr = 110Mb

Den høyeste feilraten brukt i simuleringene var 1%, som nok er noe høyere enn hva som er forventet i data fra eksomsekvensering. Det er allikevel valgt å ta med denne feilraten for å vise hvordan de ulike metodene påvirkes av genotypingsfeil.

4.1.1 Fordeling av sanne autozygote områder

Fasit simuleringen påvirkes ikke av feilrate og bakgrunnshomozygositet da disse parameterne kun påvirker simulerte genotyper. Fordelingen av fasitIBD, vist i figur 4.1-1, vil derfor være lik for alle simdata.



Figur 4.1-1: Sanne autozygote områder i simulerte data*. Fordeling av antall (A), lengder (B) og andel (C) sanne autozygote områder ved 500 simuleringer.*

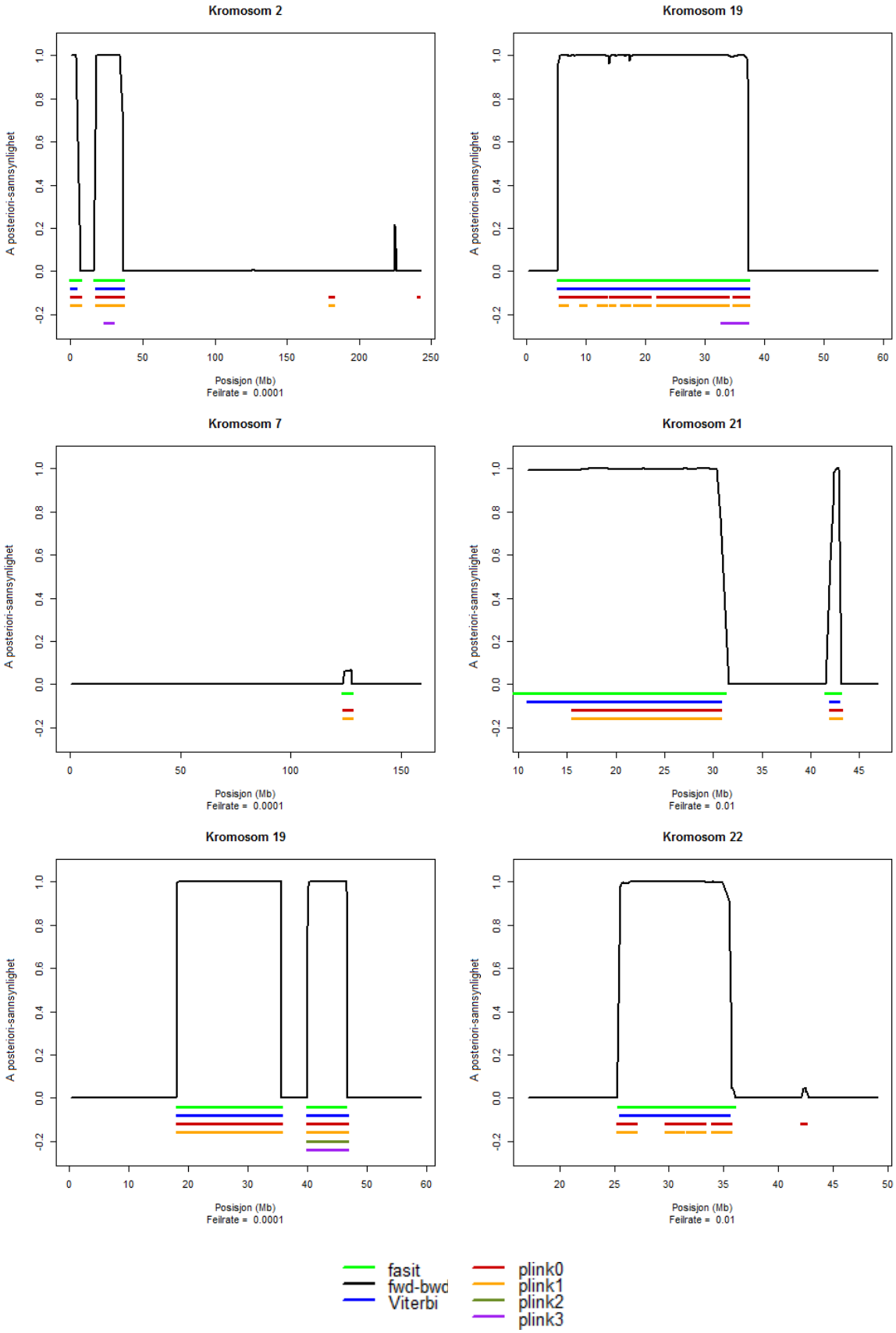
Fordelingen over antall autozygote områder per simulert genom er symmetrisk fordelt (figur 4.1-1 A) med gjennomsnitt tilnærmet lik 12 autozygote områder.

Figur 4.1-1 B viser at tetthetsfordelingen over lengdene av autozygote områder er høyreskjev. Dette er en indikasjon på at simulerte data inneholder mange korte og få lange autozygote områder.

Andel autozygositet (figur 4.1-1 C) er noenlunde normalfordelt, med gjennomsnitt lik 6,25 %. Dette samsvarer med inngiftekoeffisienten beskrevet i kapittel 2.2.1.

4.2 Sammenligning av metoder ved simulerte datasett og reelt eksom

Skriptet i tillegg B.4 benytter funksjonene i B.5 og validerer hvor godt de ulike metodene detekterer sanne autozygote områder (jamfør kapittel 3.5). Funksjonen «analyse_sim» plotter hvordan de ulike metodene detekterer autozygote områder langs et kromosom. Det er valgt å vise tre plot ved feilratene 0.0001 og 0.01 (figur 4.2-1), da de visualiserer hvordan metodene detekterer autozygositet ved lav og høy feilrate. Resultatet for hvordan metodene fanger opp autozygositet for de simulerte datasettene i tabell 4.1-1 er forøvrig vist i kapittel 4.2.1 og 4.2.2.



Figur 4.2-1: Eksempler på autozygositetskartlegging med simulerte data ved feilrate 0.0001 og 0.001.
De grønne segmentene viser de sanne autozygote områdene, mens de andre fargene viser hva som ble detektert av de ulike metodene. Den svarte kurven er a posteriori-sannsynligheter for IBD ifølge forward-backward. Merk i tillegg at y-aksen kun er knyttet til forward-backward algoritmen.

Grovt sett kan de seks ulike metodene deles inn i tre grupper, da resultatene viser at metodene i hver gruppe detekterer autozygote områder tilnærmet likt:

	Metode
Gruppe 1	forward-backward, Viterbi
Gruppe 2	plink0, plink1
Gruppe 3	plink2, plink3

Figur 4.2-2 til 4.2-4 samt figur 4.2-6 til 4.2-8 viser fordelinger over detekterte autozygote områder for de seks metodene ved ulike feilrater.

For hver feilrate er det beregnet SPR, FPR, FNR og SPR for de to algoritmene og de fire Plink metodene (figur 4.2-5 og figur 4.2-9). Terskelverdien for å klassifisere et område som IBD eller ikke-IBD ved bruk av forward-backward algoritmen er satt til 0.5.

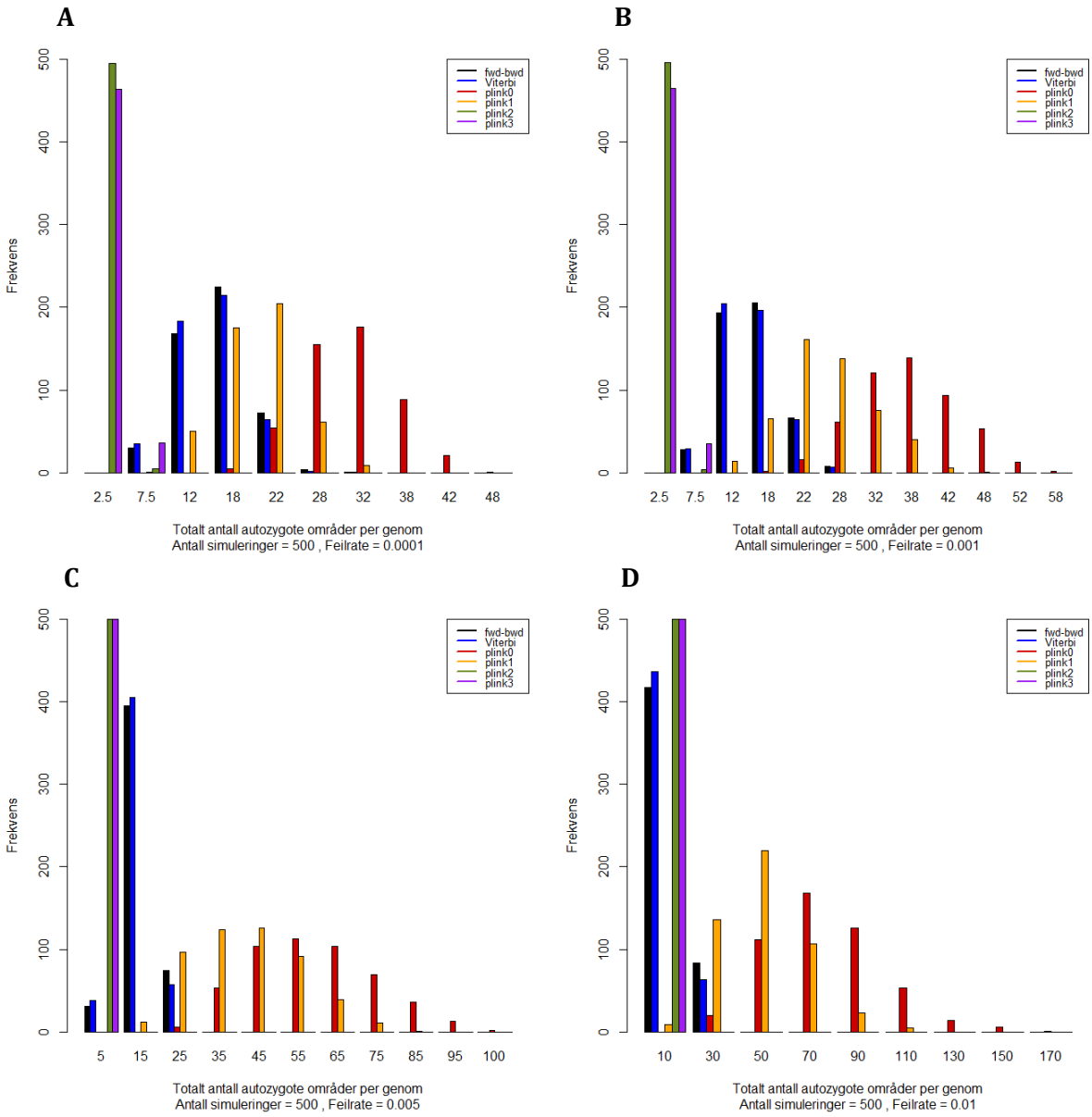
Basert på resultatene fra gruppe 3 (figur 4.2-1 til 4.2-9) er det valgt å se bort fra denne gruppa da deteksjon av autozygote områder er ekstremt dårlig.

4.2.1 Simulerte datasett med bakgrunnshomozygositet lik 22 Mb

Hvordan de ulike metodene detekterer antall autozygote områder per genom er vist i figur 4.2-2.

Detekterte autozygote segmenter ved forward-backward- og Viterbi-algoritmen er symmetrisk fordelt. Fordelingen påvirkes i liten grad av feilrate og er ikke altfor ulik antall sanne autozygote områder (jamfør kapittel 4.1.1).

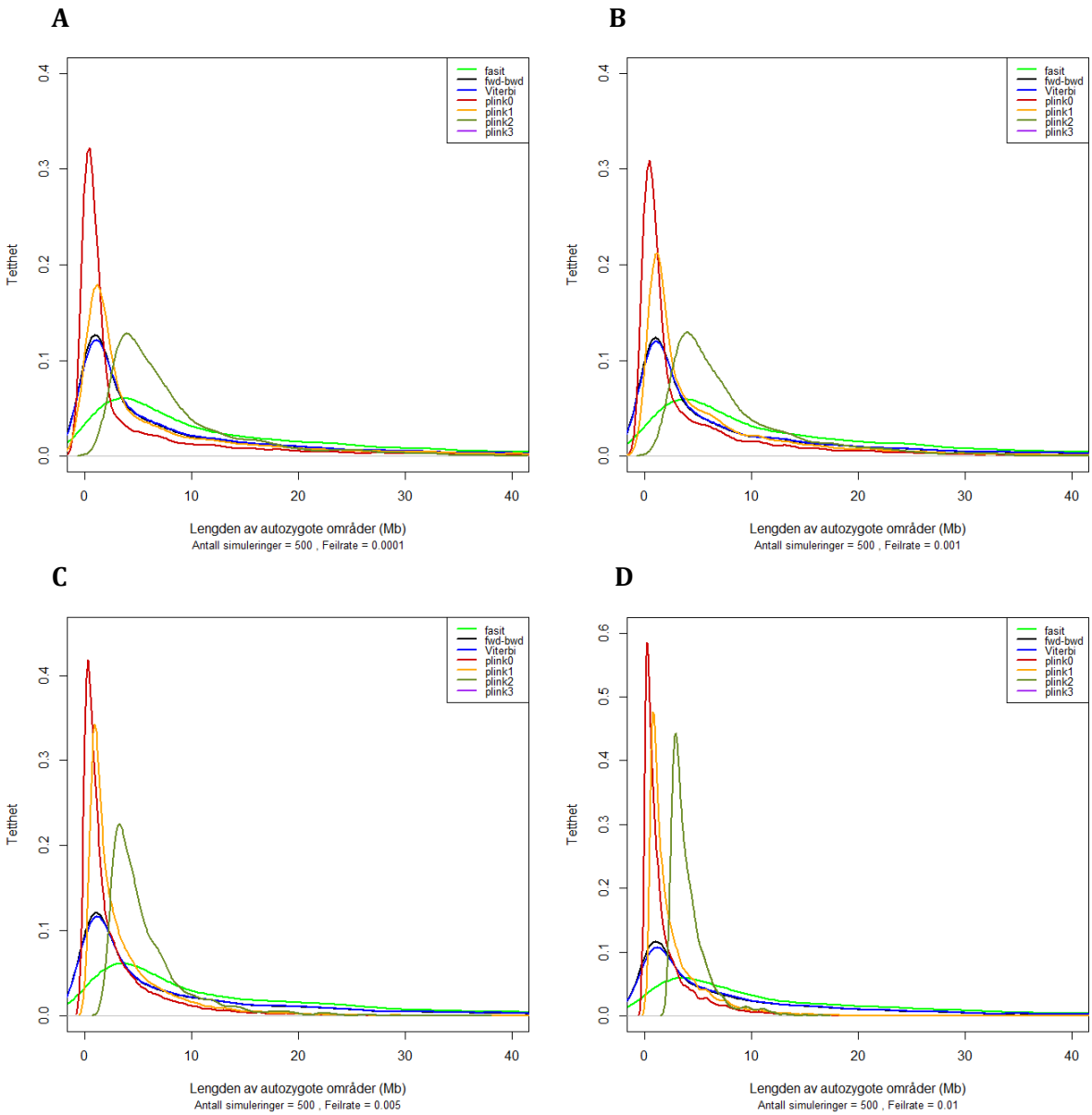
Plink0 og plink1 viser også en symmetrisk fordeling over antall autozygote områder. I forhold til gruppe 1 er gjennomsnittet forskjøvet mot høyre, der plink0 har høyere gjennomsnitt enn plink1. Med økende feilrate øker gjennomsnittet for både plink0 og plink1 slik at andelen detekterte områder blir ekstremt høyt i forhold til antall sanne områder.



Figur 4.2-2: Antall autozygote områder detektert *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 22 Mb. Tallene på x-aksen er midtpunktet i hver 'bin'.*

Figur 4.2-3 viser tetthetsfordelingen over lengdene av autozygote områder detektert av de seks ulike metodene.

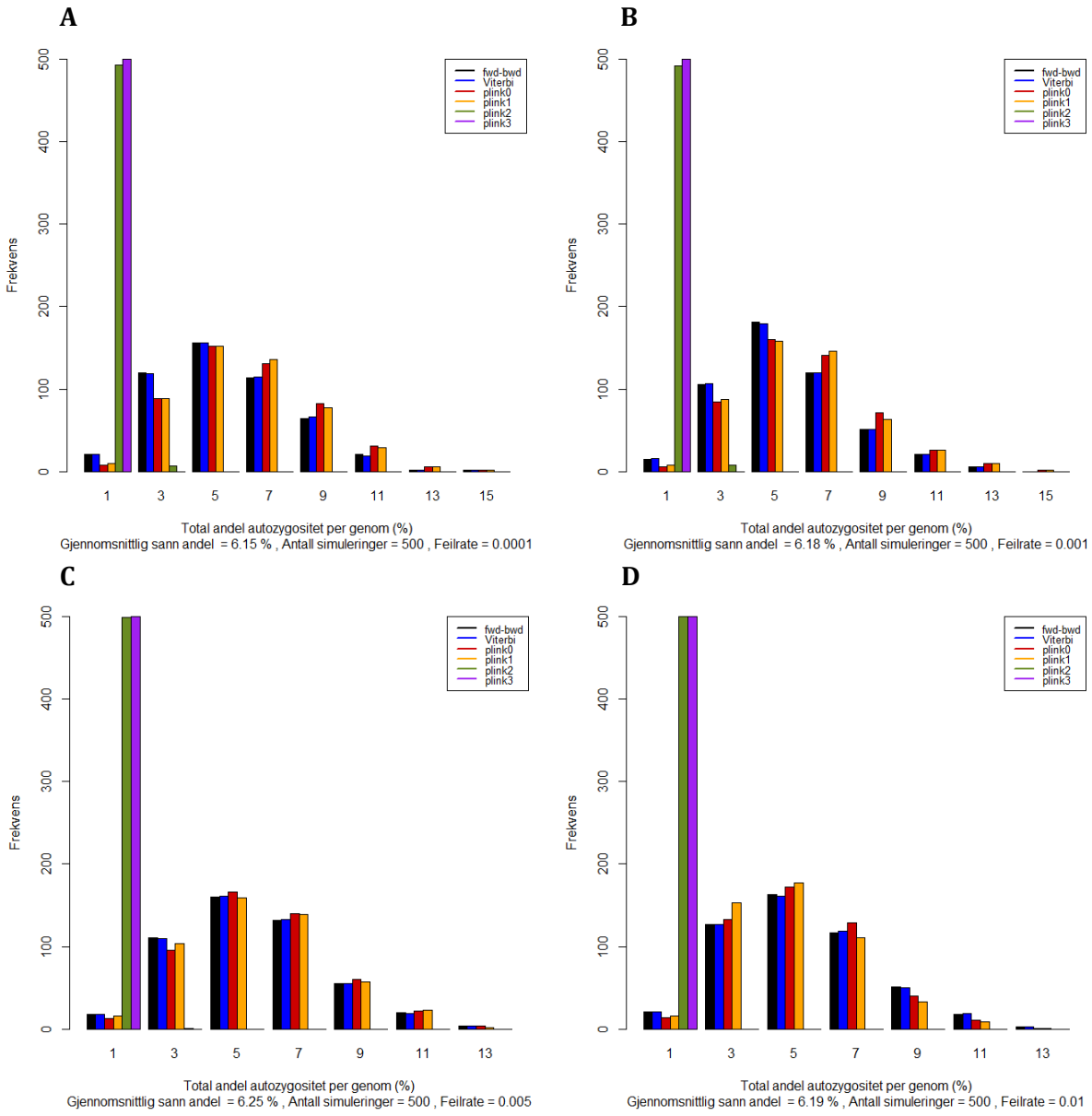
Lengdefordelingen for gruppe 1 er høyreskjev og blir ikke påvirket av feilraten. Det er verdt å merke seg at forward-backward-kurven er skjult bak Viterbi-kurven da observasjonene er tilnærmet like. For Plink metodene forskyves forventningen mot venstre og standardavviket minker med økende feilrate.



Figur 4.2-3: Lengdefordelingen av autozygote områder (Mb) detektert *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 22 Mb. Fasit = sanne autozygot lengdefordeling.*

Hvor stor andel autozygositet de ulike metodene detekterte er vist i figur 4.2-4.

Fordelingen er normalfordelt og tilnærmet lik for både forward-backward- og Viterbi-algoritmen. Plink0 og plink1 har lik fordeling som gruppe 1, men gjennomsnittet er forskjøvet mot høyre. Både gruppe 1 og gruppe 2 fungerer best ved lav feilrate da detektert autozygositet blir tilnærmet lik sann autozygositet.



Figur 4.2-4: Andel autozygositet (%) detektert *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 22 Mb. Tallene på x-aksen er midtpunktet i hver 'bin'.*

Figur 4.2-5 viser hvor godt de ulike metodene detekterer autozygote områder for henholdsvis simdata1, 2, 3 og 4. For gruppe 1 er sann positiv rate tilnærmet lik for alle feilratene. Gruppe 2 påvirkes i større grad av feilraten da økt feilrate gir lavere sann positiv rate. For begge gruppene blir medianen forskjøvet mot øvre kvartil når feilraten minker.

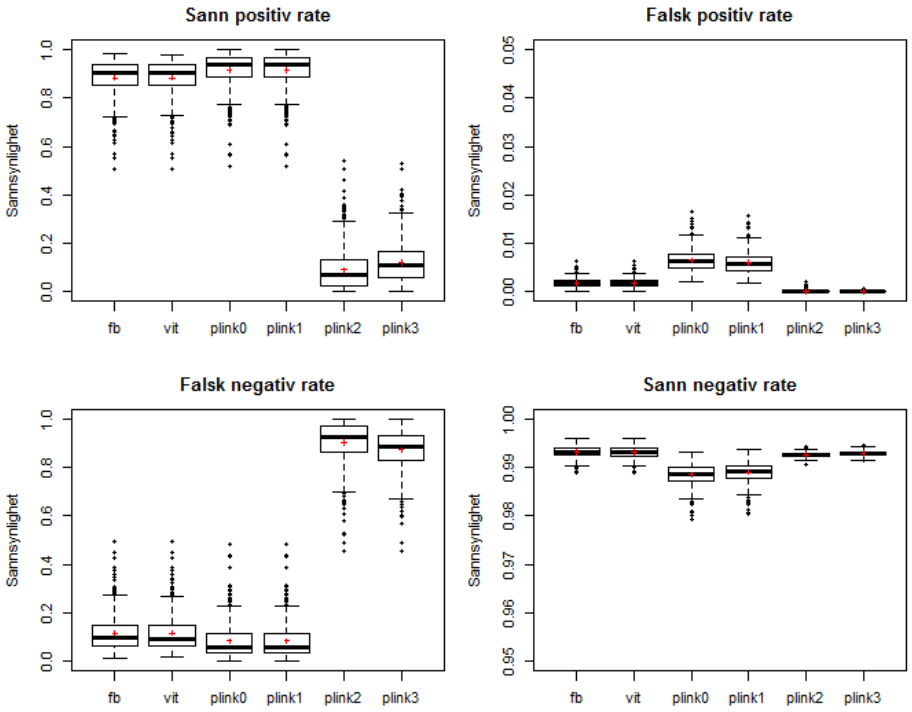
Gruppe 1 har en ekstremt lav falsk positiv rate, der kvartilbredde er tilnærmet lik median. Variasjonsbredden er noe bredere, og resultatet påvirkes i liten grad av økende feilrate. Dette

gjelder ikke for gruppe 2 der falsk positiv rate og kvartilbredde øker med økende feilrate. I tillegg forskyves medianen mot nedre kvartil.

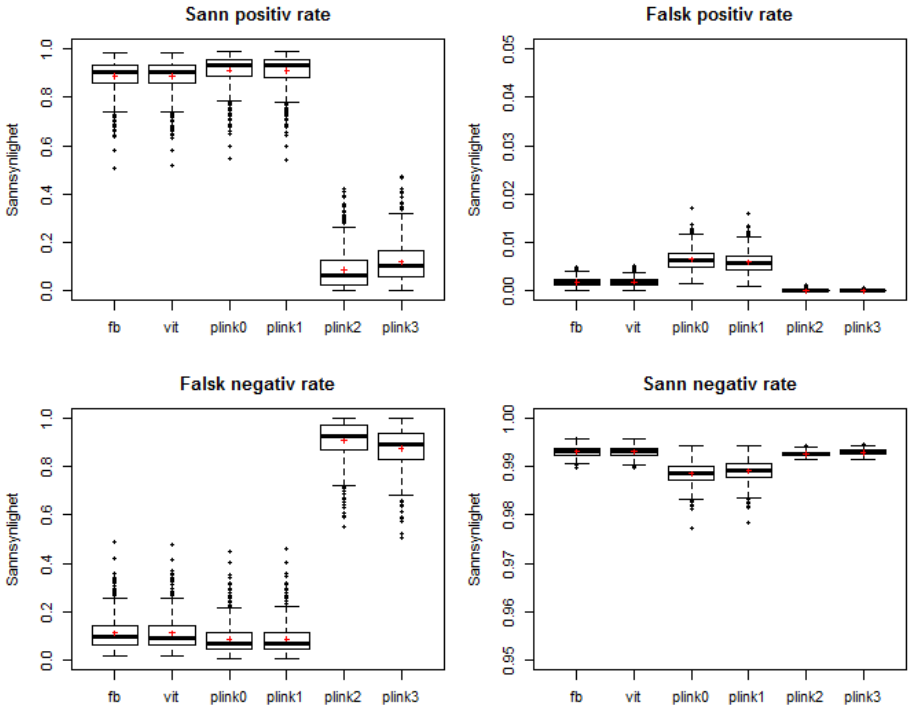
Falsk negativ rate øker med økende feilrate, hvor størst utslag er observert for gruppe 2. Resultatene viser at økt feilrate øker kvartilbredden samtidig som medianen forskyves fra nedre kvartil mot symmetrisk fordeling.

For alle gruppene er sann negativ rate tilnærmet lik 1, da den gjenspeiler falsk positiv rate. Gruppe 1 har høyest sann negativ rate ved alle feilratene, der kvartilbredde og median i liten grad påvirkes av økt feilrate. Gruppe 2 har lavest sann negativ rate med bredest variasjons- og kvartilbredde, hvor medianen er tilnærmet symmetrisk fordelt.

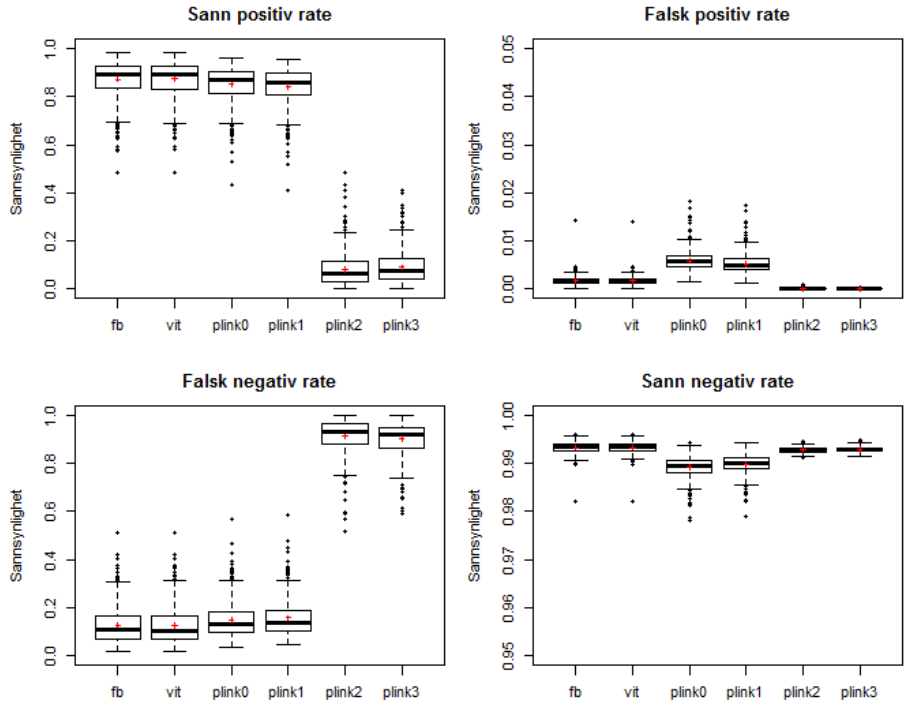
A



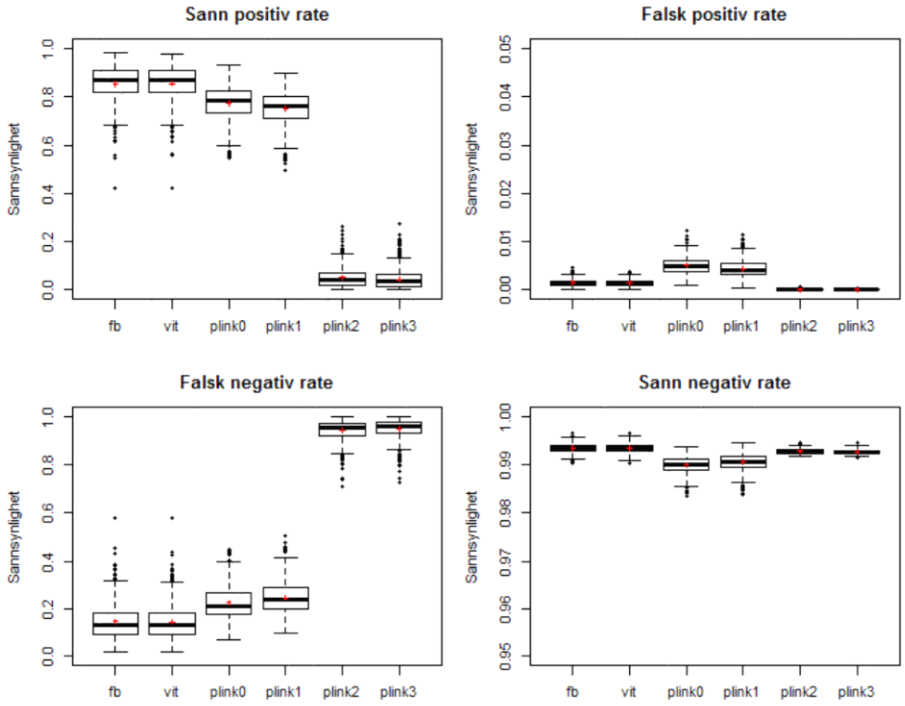
B


Sann positiv rate
Falsk positiv rate
Falsk negativ rate
Sann negativ rate
Sannsynlighet
fb
vit
plink0
plink1
plink2
plink3

C


Sann positiv rate
Falsk positiv rate
Falsk negativ rate
Sann negativ rate
Sannsynlighet
fb
vit
plink0
plink1
plink2
plink3

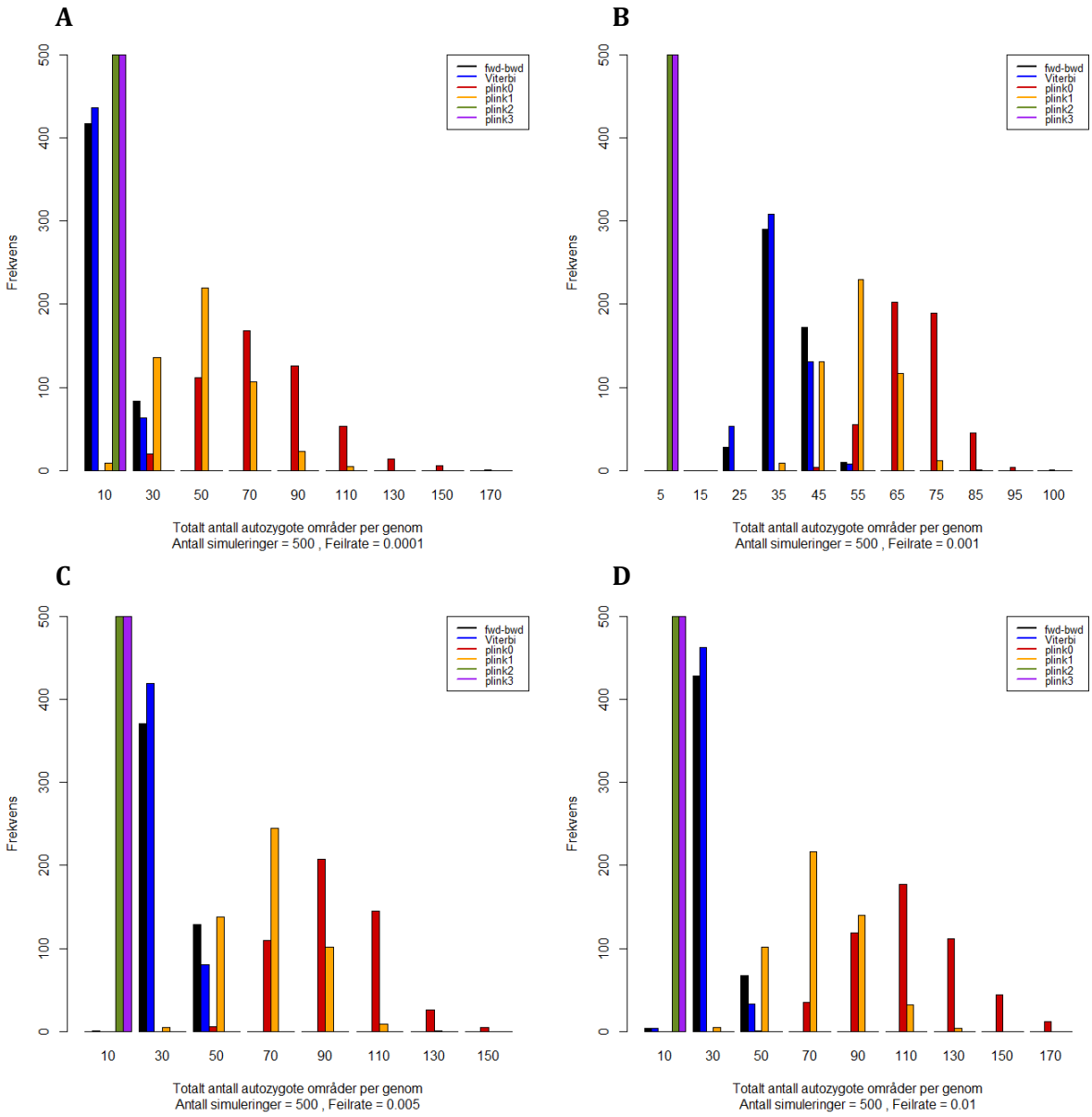
D



Figur 4.2-5: Sammenligning av metoder *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 22 Mb. Boksene i indikerer nedre og øvre kvartil, mens linjene ut i fra boksene (the whiskers) indikerer minste og største observasjon hvor uteliggere (prikker) er ekskludert. Horisontal linje indikerer median og rødt kryss indikerer gjennomsnitt. Legg merke til aksene for FPR og SNR.*

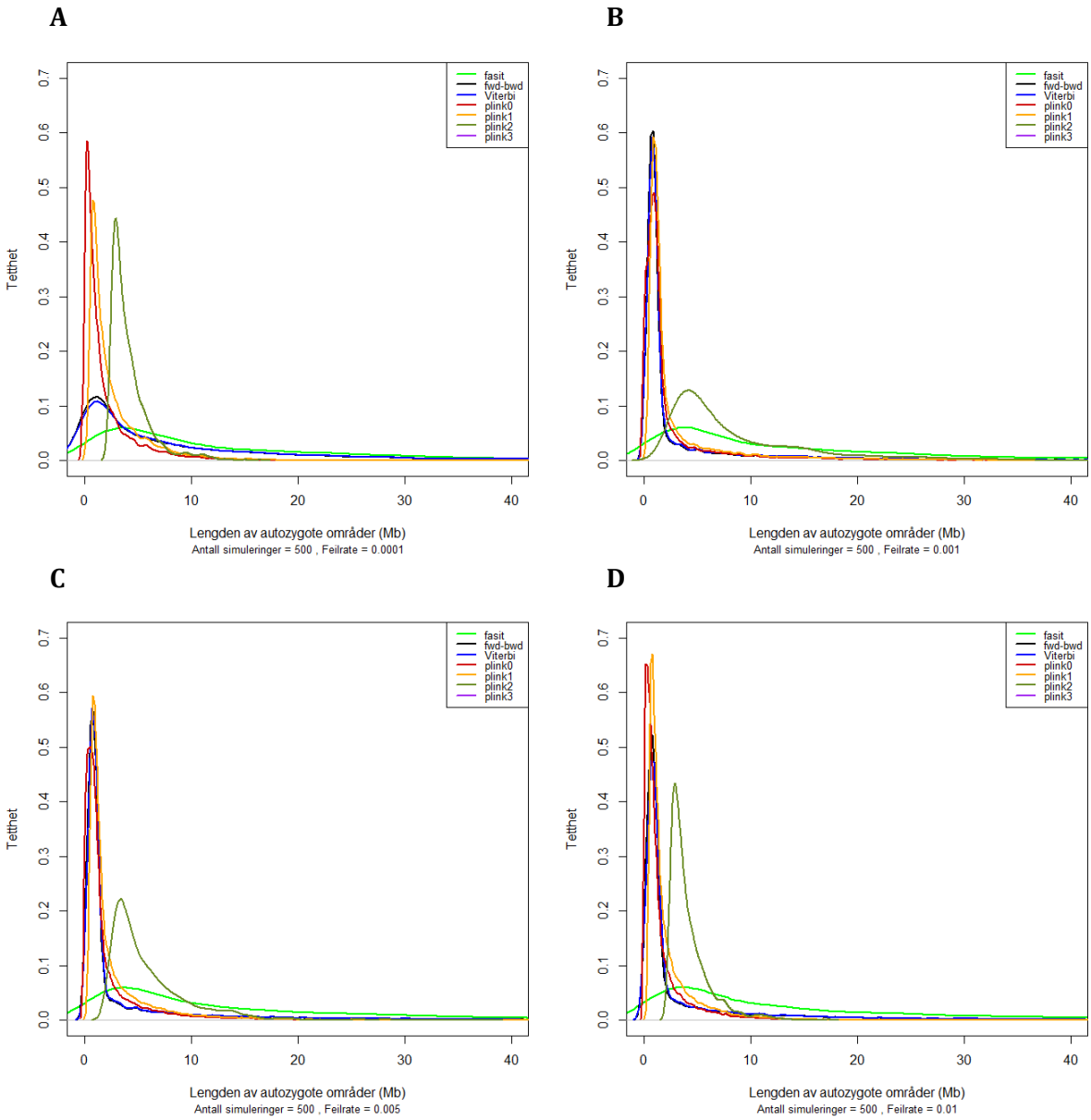
4.2.2 Simulerte datasett med bakgrunnshomozygositet lik 110 Mb

Sammenlignet med figur 4.2-2 vises det til at økt bakgrunnshomozygositet øker antall detekterte autozygote områder per simulert genom (figur 4.2-6). For gruppe 1 minker gjennomsnittet med økende feilrate, mens for Plink metodene øker gjennomsnittet med økende feilrate.



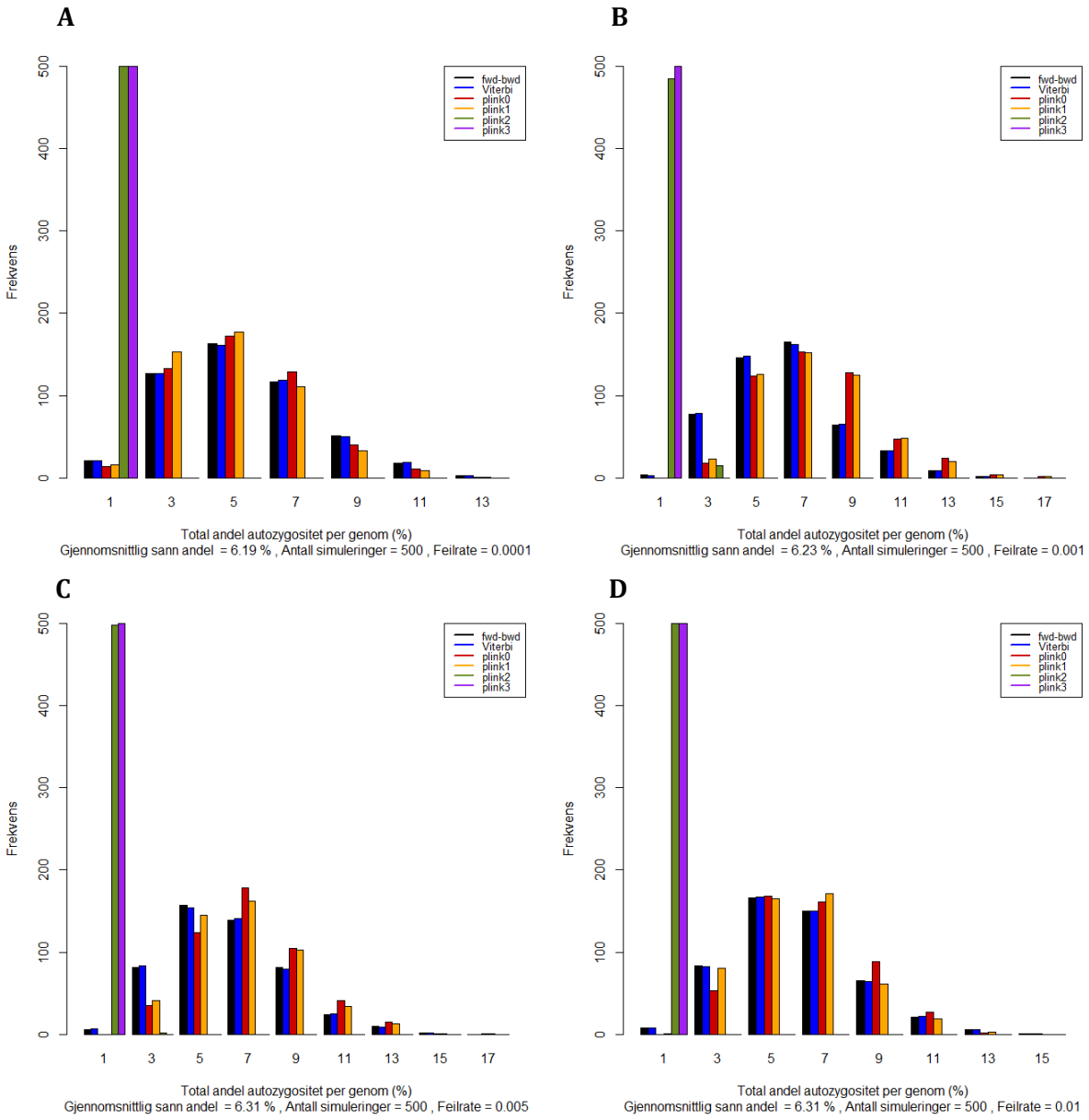
Figur 4.2-6: Antall autozygote områder detektert *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 110 Mb. Tallene på x-aksen er midtpunktet i hver 'bin'.*

Når bakgrunnshomozygositeten øker forskyves tetthetsfordelingen mot venstre (figur 4.2-7), slik at forventningen til lengdefordelingen minker (jamfør figur 4.2-3). Det er kun Plink metodene som påvirkes av minkende feilrate, ved at fordelingene blir ytterligere forskjøvet mot venstre.



Figur 4.2-7: Lengdefordelingen av autozygote områder (Mb) detektert *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 110 Mb. Fasit = sanne autozygot lengdefordeling.*

Figur 4.2-8 viser at økt bakgrunnshomozygositet også medfører økt andel detektert autozygositet. Gjennomsnittet er for alle metodene forskjøvet mot høyre (jamfør figur 4.2-4) der Plink metodene forskyves mest. Metodene fungerer best ved lav feilrate, da detektert autozygositet kommer nærmere sann autozygositet.

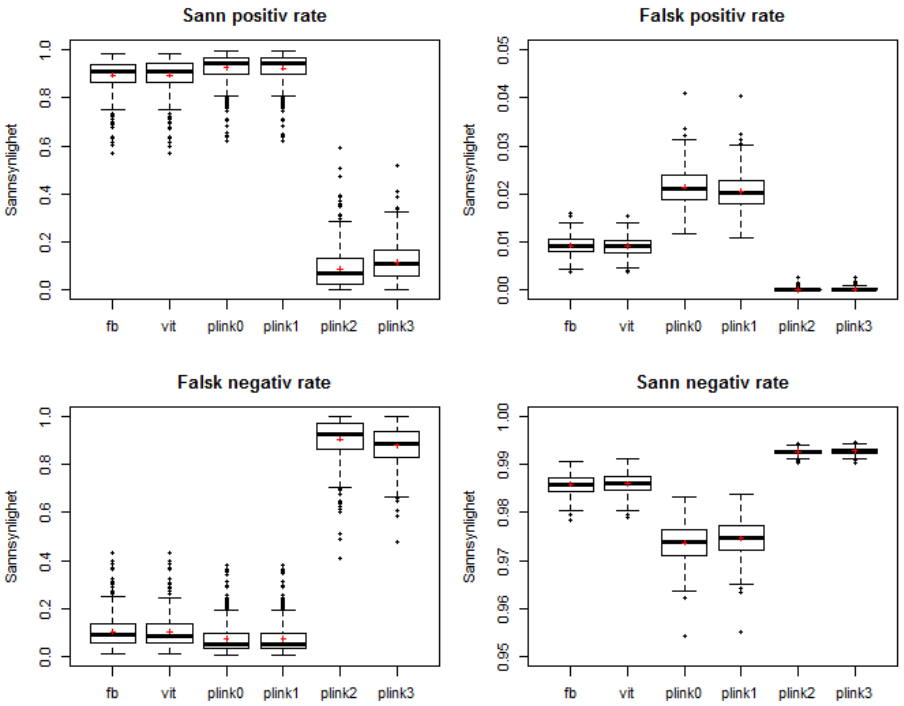


Figur 4.2-8: Andel autozygositet (%) detektert *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 110 Mb. Tallene på x-aksen er midtpunktet i hver 'bin'.*

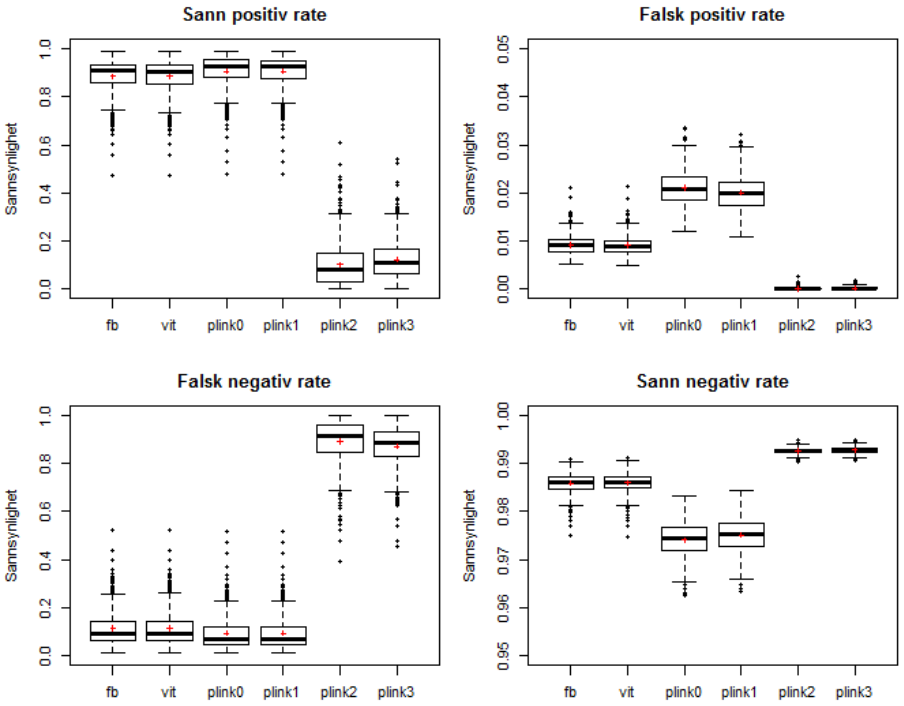
Med økt bakgrunnshomozygositet (figur 4.2-9) øker falsk positiv rate i forhold til resultatene vist i figur 4.2-5. Kvartilbredde og variasjonsbredde øker der median er symmetrisk fordelt i boksen.

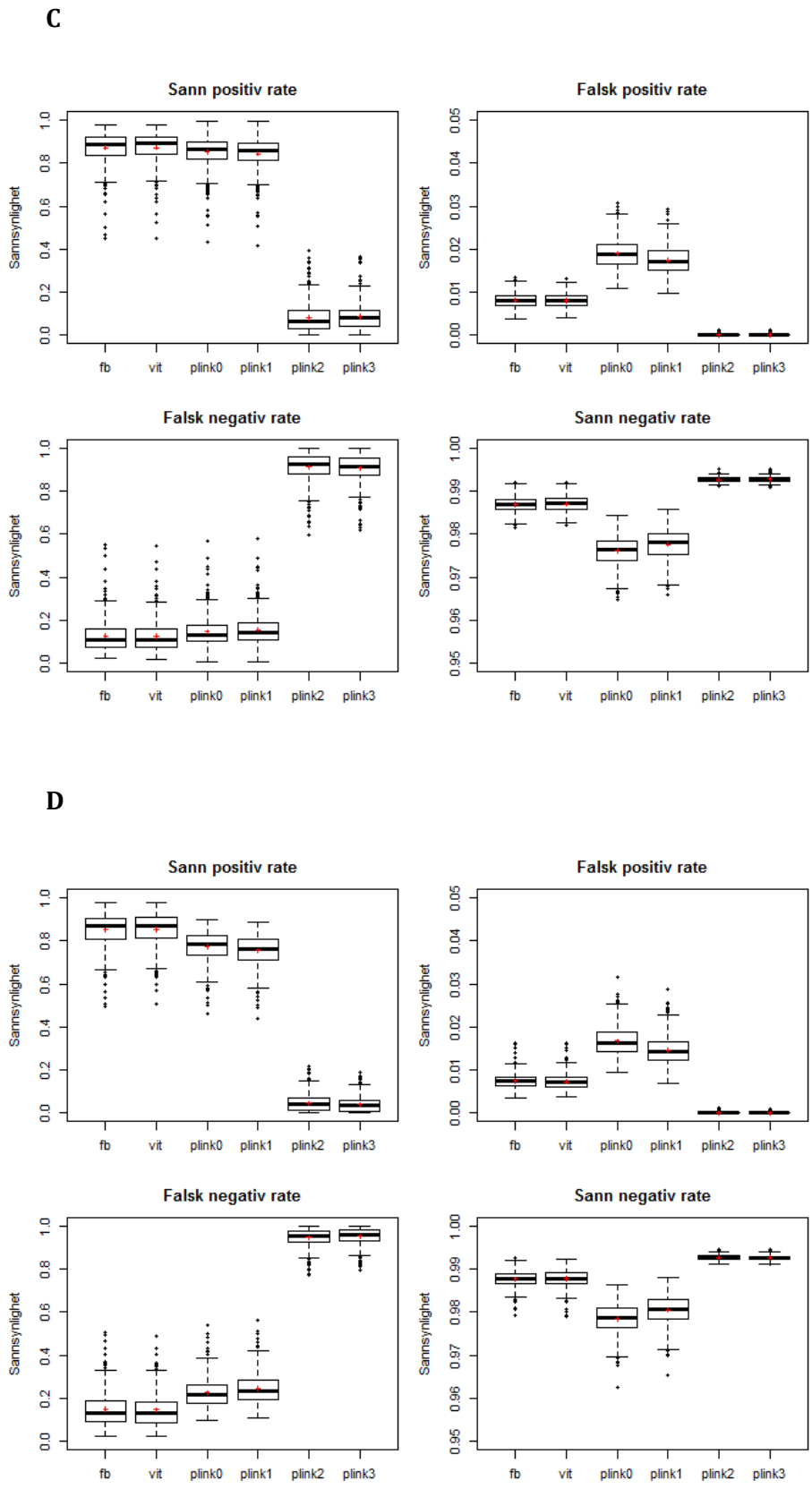
Når falsk positiv rate øker vil sann negativ rate minke. Figurene viser at gruppe 1 fungerer bedre ved økt bakgrunnshomozygositet enn gruppe 2 da det vises til smaler variasjons- og kvartilbredde for de to algoritmene. Sann positiv rate og falsk negativ rate er tilnærmet lik som ved lav bakgrunnshomozygositet.

A


Sann positiv rate
Falsk positiv rate
Falsk negativ rate
Sann negativ rate
Sannsynlighet
fb
vit
plink0
plink1
plink2
plink3

B


Sann positiv rate
Falsk positiv rate
Falsk negativ rate
Sann negativ rate
Sannsynlighet
fb
vit
plink0
plink1
plink2
plink3



Figur 4.2-9: Sammenligning av metoder *ved feilratene A) 0.0001, B) 0.001, C) 0.005 og D) 0.01 når bakgrunnshomozygositeten er 110 Mb. Boksene i indikerer nedre og øvre kvartil, mens linjene ut i fra boksene (the whiskers) indikerer minste og største observasjon hvor uteliggere (prikker) er ekskludert. Horisontal linje indikerer median og rødt kryss indikerer gjennomsnitt. Legg merke til aksene for FPR og SNR.*

4.2.3 Reelt eksom

For et reelt eksom er det antatt at et SNP-array resultat (ACGH) inneholder de sanne autozygote områdene. Disse dataene ble imidlertid skaffet rett før innlevering av oppgaven, og analysene er derfor ufullstendige.

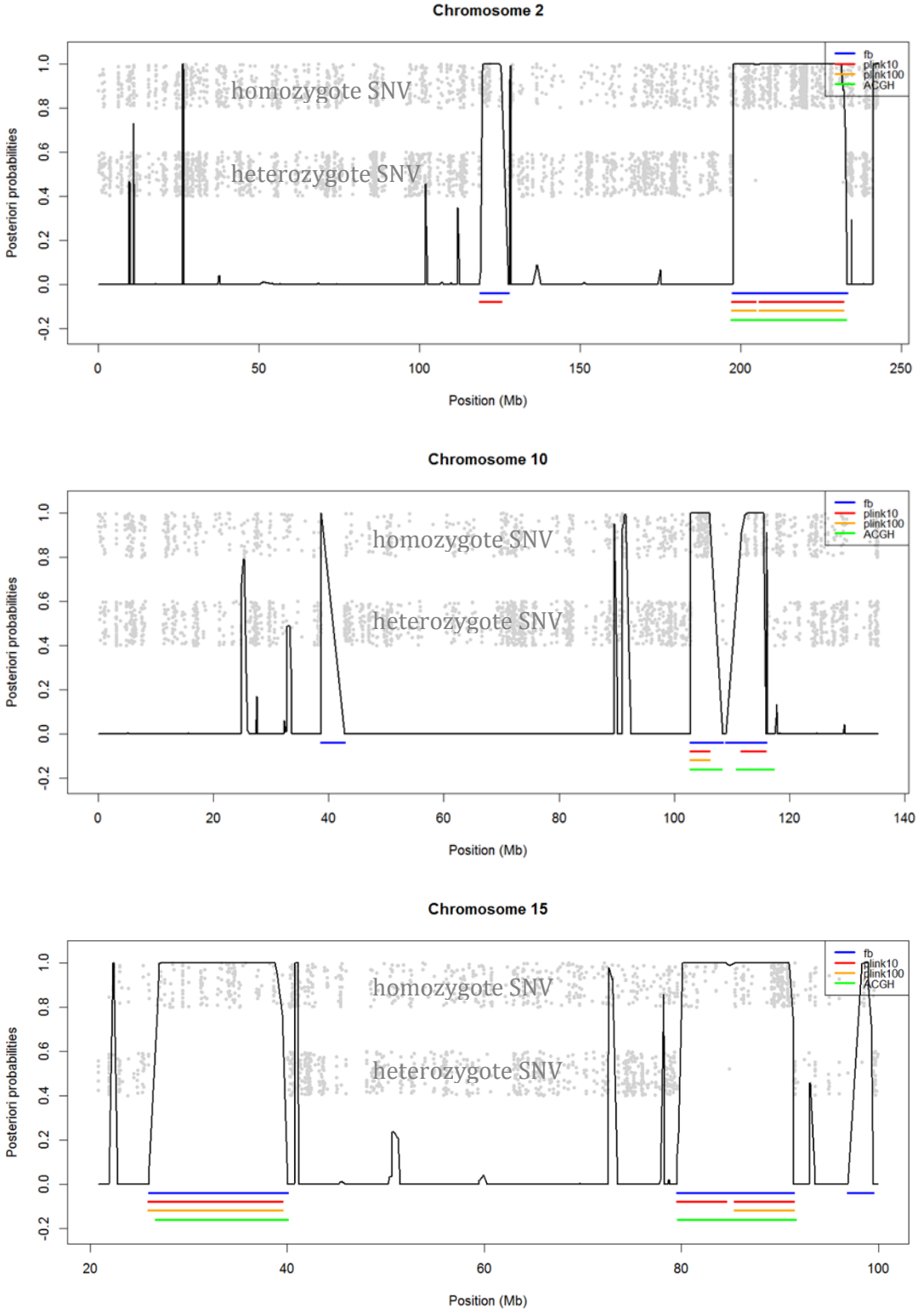
Forward-backward algoritmen, sammenlignet med plink0 og plink1 er benyttet for å detektere de autozygote områdene i eksomet. På grunn av tidspresset er ikke Viterbi-algoritmen benyttet. Dette fordi resultatene fra kapittel 4.2.1 og 4.2.2 viser til at forward-backward algoritmen er noe mer velegnet. Plink2 og plink3 er heller ikke benyttet, da de samme resultatene viser at disse metodene ikke er egnet for eksomdata.

Feilraten for sekvenseringsfeil ble satt til 0,1 %. Autozygote områder detektert av forward-backward algoritmen har en terskelverdi på 0.1 og en nedre grense på 2 Mb.

Styrken til forward-backward algoritmen, plink0 og plink1 er vist i tabell 4.2-1. Her kommer det fram at andelen sanne positive er høyest for forward-backward algoritmen. I tillegg er det vist tre plot i figur 4.2-10 for å visualisere hvordan metodene detekterer autozygote områder.

Tabell 4.2-1: Styrken til forward-backward algoritmen, plink0 og plink1 ved reelt eksom*, når feilrate = 0.001. Forward-backward algoritmen har en terskelverdi på 0.1 og en nedre grense på 2Mb.*

	forward-backward	**plink0**	**plink1**
Sanne positive	96.3 %	89.7 %	77.4 %
Falske negative	3.4 %	10.0 %	22.3 %
Falske positive	2.9 %	1.7 %	0.5 %
Sanne negative	96.6 %	97.8 %	98.8 %
Antall segmenter	29	27	14



Figur 4.2-10: Deteksjon av autozygote områder ved eksomsekvensering der feilraten =0.001.
Grønne segmenter = sanne autozygote områder fra SNP-array (ACGH). De andre fargene viser hva som ble detektert av de ulike metodene. Den svarte kurven er a posteriori-sannsynligheter for IBD ifølge forward-backward. Merk i tillegg at y-aksen kun er knyttet til forward-backward algoritmen.

5. Diskusjon

I denne oppgaven er det benyttet en skjult markovmodell for å detektere autozygote områder ved eksomsekvensering. Det er beskrevet to algoritmer – forward-backward og Viterbi. Metodene klassifiserer markører langs et kromosom som IBD eller ikke-IBD. Forward-backward algoritmen beregner a posteriori-sannsynligheter for hver markør, mens Viterbi-algoritmen leter etter den mest sannsynlige sekvensen av IBD og ikke-IBD områder. For å verifisere hvor godt de to algoritmene fungerer, er det benyttet simulerte data med genotyper fra en inngiftefamilie. I tillegg er resultatene sammenlignet med resultater fra Plink.

Autozygositetskartlegging er et nyttig verktøy for å finne gener involvert i recessive sykdommer og egenskaper, spesielt i familier med inngifte. Metoden har til nå hovedsakelig basert seg på genotypedata fra SNP-arrays med jevnt fordelte SNP'er, da dette er mer anvendelig for å kartlegge lange autozygote områder. Anvendelsen av autozygositetskartlegging ved eksomsekvensering er imidlertid forventet mer problematisk. Dette er fordi tettheten av genetisk informasjon langs kromosomene varierer (Browning & Browning 2012; Carr et al. 2013). Resultatene i denne oppgaven indikerer imidlertid at deteksjon av auozygote områder ved anvendelse av eksomdata ikke er like problematisk som først antatt. En mulig årsak kan være høy tetthet av SNV'er langs kromosomene. En indikasjon på dette er illustrert i figur 4.2-10, der det er vist hetero- og homozygote enkeltbasevarianter langs kromosomene.

Det har vært vanskeligere enn forventet å simulere realistiske eksomdata. For det første varierer MAF fra markør til markør. For det andre må det være realistiske markørposisjoner. For det tredje vil homozygote områder som er vanlig i en befolkningen være tilstede. For det fjerde vil interferens påvirke rekombinasjonsfrekvensen. Til slutt, og i følge Leutenegger et al. (2003), vil håndtering av sekvenseringsfeil være helt nødvendig.

Problemstillingene i avsnittet ovenfor er håndtert på følgende måte: MAF for hver markør er basert på 1000 Genome Project, der markørposisjonene er hentet fra et ekte eksom. For å håndtere bakgrunnshomozygositet er det simulert tilfeldige homozygote områder. Disse homozygote områdene vil ikke være sanne autozygote områder. Siden informasjon om faktiske rekombinasjonshendelser er ukjent, er χ^2-modell benyttet, da den håndterer interferens og følgelig er en god tilnærming (Browning 2003). I reelle eksomdata vil hver markør være annotert med en individuell feilrate. I mangel på slik informasjon, er det benyttet en konstant feilrate for å håndtere sekvenseringsfeil. En forbedring vil derfor være å inkorporere faktisk feilrate for hver enkelt markør.

Det har også vært vanskelig å skaffe reelle eksomdata hvor man har informasjon om kjente autozygote områder. Dette har vært ønskelig for å teste metodene opp mot reelle eksom og ikke bare simulerte data. Et slik datasett ble imidlertid anskaffet rett før innlevering. Grunnet tidspresset er analysen knyttet til det reelle eksomet ufullstendig, men allikevel en god indikasjon på at autozygositetskartlegging er mulig ved eksomsekvensering.

Den skjulte markovmodellen beskrevet i denne oppgaven antar kjent slektskap og kjent pedigree mellom beslektede foreldre. Dersom det benyttes individer der foreldre er beslektet, men ikke angitt hvordan, er det antatt at modellen ikke egner seg like godt. Dette fordi modellen antar kjent inngiftekoeffisient, en informasjon som nå er ukjent. Allikevel vil *a* parameteren si noe om forventet lengde av autozygote områder, da den er basert på forventet rekombinasjonsfrekvens. En forbedring av den skjulte markovmodellen vil derfor være om modellen også kan håndtere eksomdata av denne karakter.

Forward-backward er forventet noe bedre enn Viterbi, basert på at algoritmen finner a posteriori-sannsynligheter til hver tilstand langs et kromosom. A posteriori-sannsynlighetene påvirkes av MAF og gjør det mulig å evaluere hver tilstand langs kromosomet. Dersom MAF er lav og et locus er homozygot for det alternative allelet, trengs det få markører på rad for sterk IBD indikasjon. Ved høy MAF må det derimot være mange homozygote markører på rad da en homozygot markør er tilnærmet vanlig i befolkningen. Om forward-backward algoritmen klassifiserer et område som autozygot er bestemt av en terskelverdi. Lav terskelverdi gir flere og lengre autozygote områder, mens høy terskelverdi gir færre og kortere områder.

Plink0 og plink1 tillater ingen heterozygote markører i et autozygot område. Som illustrert i figur 4.2-10, vil derfor et langt autozygot område bli delt i to dersom en heterozygot markør, grunnet genotypingsfeil, skulle være tilstede. Den skjulte markovmodellen håndterer slike heterozygote markører, da det er lagt til en feilrate i emisjonssannsynlighetene. Det er derfor ikke overraskende at forward-backward algoritmen fungerer bedre enn Plink i disse tilfellene.

Slik modellen er beskrevet er transisjonssannsynlighetene basert på fravær av genetisk interferens. Et kjent rekombinasjonskart er benyttet slik at distansen mellom to markører kan beregnes ved hjelp av Haldanes kartleggingsfunksjon. Denne antakelsen er imidlertid en forenkling av IBD prosessen, men som Leutenegger et al. (2003) hevder, spiller dette liten rolle. En forbedring vil imidlertid være om HMM håndterer interferens, men som Browning og Browning (2012) hevder, vil dette være vanskelig da det vil redusere hastigheten til modellen.

Plink benytter en sliding-window metode for å detektere autozygote områder. Om Plink klassifiserer et område som autozygot eller ikke, er avhengig av 9 parametere (tillegg A.1). Standard parametervalg er velegnet for SNP-arrays, men som resultatene viser, fungerer de dårlig på eksomdata. Det er imidlertid vanskelig å vite hvilke parametervalg som er optimale for Plink, da små endringer kan påvirke resultatet drastisk. Å finne optimale parametervalg for Plink, anvendt mot eksomdata, er vanskelig og innehar stor grad av usikkerhet.

Ut ifra valgte parametersett er det forventet at når både *homozyg-kb* og *homozyg-snp* er lav, vil det detekteres mange homozygote områder. I tillegg vil parametervalgene *homozyg-density* og *homozyg-gap* ha mye å si for deteksjonen av autozygote områder, da de setter kriterier til distanse mellom markører. For plink0 og plink1 er disse parameterne satt høyt for å håndtere ujevn SNP-fordeling bedre enn plink2 og plink3. Resultatene viser også til at standard parametervalg fungerer dårlig på eksomdata.

Selv om Plink er et velegnet detekteringsprogram for autozygote områder er det langt fra perfekt (Howrigan et al. 2011). En årsak er at programmet ikke legger til grunn en biologisk modell. Det vil si, Plink antar jevn fordeling av markører og tar ikke høyde for rekombinasjon eller pedigree. Den skjulte markovmodellen er derimot bygget på en biologisk modell og derfor forventet å detektere autozygote områder på en mer tilfredsstillende måte.

Ved autozygositetskartlegging leter man etter kausale genvarianter i autozygote områder. Som nevnt i kapittel 3.5 kontrolleres det for SPR og FPR for å få minst mulig støy i dataene. Når det kontrolleres for FPR vil FNR bli noe høyere. Ulempen med dette er at den kausale genvarianten kan ha sitt sykdomslocus i et autozygot område som ikke blir detektert. Det er imidlertid ikke så farlig å miste noen autozygote markører, da den kausale genvarianten som oftest er lokalisert i et relativt langt autozygot område.

Figur 4.1-1 indikerer at simulerte data inneholder mange korte og få lange autozygote områder. Det er forventet at de to algoritmene skal slite med å fange opp de korte områdene (Carr et al. 2009). For plink0 og plink1 er dette tatt høyde for da cut-off verdiene er lave slik at små områder skal fanges opp. Med bakgrunn i dette, er det forventet at plink0 og plink1 både skal ha høyere sann positiv rate og falsk positiv rate enn forward-backward- og Viterbi-algoritmen. Dermed inneholder resultatene fra gruppe 2 mer støy og gjør det problematisk å skille sanne autozygote områder fra falske autozygote områder. Siden FNR = 1-SPR vil falsk negativ rate være høyere for gruppe 1 enn for gruppe 2. På samme måte vil også SNR = 1- FPR være høyere for gruppe 1.

Med økt bakgrunnshomozygositet er det forventet at metodene får høyere falsk positiv rate. Dette fordi en større andel homozygote markører i eksomdata vil være IBS. Siden falsk positiv rate øker, vil sann negativ rate minke. Sann positiv rate og falsk negativ rate er forventet tilnærmet uendret da andel sanne autozygote områder forblir det samme. Allikevel er det overraskende at økt bakrunnshomozygositet ikke påvirker resultatene mer. Hvorfor det er slik er noe usikkert, men en mulig årsak kan være ukorrekt simulering av bakgrunnshomozygositet. Det kunne derfor vært interessant å simulere data uten bakgrunnshomozygositet, for å se hvordan dette påvirker resultatene.

Den skjulte markovmodellen beskrevet i denne oppgaven er velegnet til kartlegging av autozygote områder av inngiftefamilier med kjent slektskapsforhold og slektstre. Det er imidlertid verdt å bemerke at mangel på feilhåndtering er en svakhet i koden (tillegg B.1-B.3). Til tross for viktighet av feilhåndtering, er dette arbeidet ikke påbegynt grunnet tidsbegrensninger. Dette må håndteres før algoritmen kan implementeres i Filtus (tillegg A.2), slik at koden blir sikrere, raskere og mer pålitelig.

6. Konklusjon

Resultatene viser at den skjulte markovmodellen er velegnet til å detektere autozygote områder ved anvendelse av eksomdata. Dermed er autozygositetskartlegging mulig ved eksomsekvensering.

De to algoritmene knyttet til den skjulte markovmodellen, forward-backward- og Viterbi-algoritmen, er tilnærmet like gode og påvirkes i liten grad av sekvenseringsfeil. Små forskjeller viser imidlertid at forward-backward algoritmen er noe bedre egnet enn Viterbi-algoritmen til å detektere sanne autozygote områder. Dette fordi forward-backward algoritmen beregner a posteriori-sannsynligheter for om markører langs et kromosom er autozygote eller ikke.

Resultatene viser i tillegg at Plink (tillegg A.1) fungerer overraskende bra, men at programmet er sensitiv ovenfor parametervalg. Den skjulte markovmodellen har imidlertid vist seg bedre, da den er basert på en biologisk modell og dermed håndterer ujevn fordeling av SNV'er. Neste steg vil derfor være å implementere algoritmen i Filtus (tillegg A.2). Det må imidlertid gjøres noe feilhåndtering i koden (tillegg B.1-B.3) før implementering.

7. Litteraturliste

Bamshad, M. J., Ng, S. B., Bigham, A. W., Tabor, H. K., Emond, M. J., Nickerson, D. A. & Shendure, J. (2011). Exome sequencing as a tool for Mendelian disease gene discovery. *Nature Reviews Genetics*, 12: 745-755.

Browning, S. (2003). Pedigree Data Analysis With Crossover Interference. *Genetics*, 164: 1561-1566.

Browning, S. R. & Browning, B. L. (2012). Identity by Descent Between Distant Relatives: Detection and Applications. *Annual Review of Genetics*, 46: 617-633.

Carr, I. M., Sheridan, E., Hayward, B. E., Markham, A. F. & Bonthron, D. T. (2009). IBDfinder and SNPsetter: Tools for Pedigree-Independent Identification of Autozygous Regions in Individuals with Recessive Inherited Disease. *Human Mutation*, 30 (6): 960-967.

Carr, I. M., Bhaskar, S., O'Sullivan, J., Aldahmesh, M. A., Shamseldin, H. E., Markham, A. F., Bonthron, D. T., Black, G. & Alkuraya, F. S. (2013). Autozygosity Mapping with Exome Sequence Data. *Human Mutation*, 34 (1): 50-56.

Clark, A. G. (1999). The Size Distribution of Homozygous Segments in the Human Genome. *The American Journal of Human Genetics*, 65: 1489-1492.

Eddy, S. R. (2004). What is a hidden Markov model? *Nature Biotechnology*, 22 (10): 1315-1316.

EDinformatics. *What is the human genome and how big is it?* Tilgjengelig fra: http://www.edinformatics.com/math_science/human_genome.htm (lest 24.04.14).

Ewens, W. J. & Grant, G. R. (2005). *Statistical Methods in Bioinformatics: An Introduction*. 2 utg. New York: Springer.

Fletcher, H., Hickey, I. & Winter, P. (2007). *BIOS Instant Notes in Genetics*. 3. utg. Abingdon, UK: Taylor & Francis Group.

Haraldsen, I. (2009). *Innavl til kongelig besvær*. forskning.no. Tilgjengelig fra: http://www.forskning.no/artikler/2009/april/217035 (lest 09.02.2014).

Howrigan, D. P., Simonson, M. A. & Keller, M. C. (2011). Detecting autozygosity through runs of homozygosity: A comparison of three autozygosity detection algorithms. *BMC Genomics*, 12: 460.

Jorde, L. B. & Wooding, S. P. (2004). Genetic variation, classification and 'race'. *Nature Genetics Supplement*, 36 (11): S28-S33.

Klug, W. S., Cummings, M. R., Spencer, C. A. & Palladino, M. A. (2010). *Essentials of Genetics*. 7. utg. San Francisco: Pearson Education.

Lander, E. S. & Botstein, D. (1987). Homozygosity Mapping: A Way to Map Human Recessive Traits with the DNA of Inbred Children. *Science*, 236: 1567-1570.

Lander, E. S. (2011). Initial impact of the sequencing of the human genome. *Nature*, 470: 187-197.

Leutenegger, A., Prum, B., Gènin, E., Verny, C., Lemainque, A., Clerget-Darpoux, F. & Thompson, E. A. (2003). Estimation of the Inbreeding Coefficient through Use of Genomic Data. *The American Journal of Human Genetics*, 73 (3): 516-523.

Lie, A. K. (2008). Setesdalsrykkja - 100 år etter. *Tidsskrift for Den norske legeforening*, 128: 2035.

McQuillan, R., Leutenegger, A., Abdel-Rahman, R., Franklin, C. S., Pericic, M., Barac-Lauc, L., Smolej-Narancic, N., Janicijevic, B., Polasek, O., Tenesa, A., et al. (2008). Runs of Homozygosity in European Populations. *The American Journal of Human Genetics*, 83: 359-372.

Moltke, I., Albrechtsen, A., Hansen, T. V., Nielsen, F. C. & Nielsen, R. (2011). A method for detecting IBD regions simultaneously in multiple individuals--with applications to disease genetics. *Genome Research*, 21 (7): 1168-1180.

Newberg, L. A. (2009). Error statistics of hidden Markov model and hidden Boltzmann model results. *BMC Bioinformatics*, 10: 212.

Ng, S. B., Buckingham, K. J., Lee, C., Bigham, A. W., Tabor, H. K., Dent, K. M., Huff, C. D., Shannon, P. T., Jabs, E. W., Nickerson, D. A., et al. (2010). Exome sequencing identifies the cause of a mendelian disorder. *Nature Genetics*, 42 (1): 30-35.

OMIM. Online Mendelian Inheritance in Man, OMIM. I: *McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD),* . Tilgjengelig fra: http://omim.org/ (lest 06.05.2014).

Pippucci, T., Benelli, M., Magi, A., Martelli, P. L., Magini, P., Torricelli, F., Casadio, R., Seri, M. & Romeo, G. (2011). EX-HOM (EXome HOMozygosity): a proof of principle. *Human Heredity*, 72: 45-53.

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A., Bender, D., Maller, J., Sklar, P., de Bakker, P. I., Daly, M. J., et al. (2007). PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. *The American Journal of Human Genetics*, 81 (3): 559-575.

Purcell, S. (2009). *PLINK v.1.07*.

The Python Tutorial. Python Software Foundation. Tilgjengelig fra: http://docs.python.org/2.7/tutorial/index.html (lest 09.03.14).

R Core Team. (2014). *R Language Definition*. DRAFT, version 3.1.0. Tilgjengelig fra: http://cran.r-project.org/manuals.html (lest 10.04.14).

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77 (2): 257-286.

Ross, S. M. (2007). *Introduction to Probability Models*. 9. utg. USA: Academic Press.

Surén, P., Grjibovski, A. & Stoltenberg, C. (2007). Inngifte i Norge - omfang og medisinske konsekvenser, Rapport 2007:2. Oslo: Nasjonalt folkehelseinstitutt.

TraitGenetics. *SNP markers*. Tilgjengelig fra: http://www.traitgenetics.com/en/index.php?option=com_content&task=view&id=33&Itemid=54 (lest 24.04.14).

Venables, W. N., Smith, D. M. & R Core Team. (2014). *An Introduction to R*. Notes on R: A Programming Environment for Data Analysis and Graphics, version 3.1.0: The R foundation for statistical Computing. Tilgjengelig fra: http://cran.r-project.org/manuals.html (lest 10.04.14).

Vigeland, M. D. (2014a). *FILTUS v-0.99-4*.

Vigeland, M. D. (2014b). *IBDsim: Simulation of chromosomal regions shared by family members*, R package version 0.9-3.

Vigeland, M. D. (2014c). *paramlink: Parametric linkage analysis in R.*, R package version 0.9-6.

Yang, J., Lee, T., Kim, J., Cho, M., Han, B., Lee, J., Lee, H., Cho, S. & Kim, H. (2013). Ubiquitous Polygenicity of Human Complex Traits: Genome-Wide Analysis of 49 Traits in Koreans. *PLOS Genetics*, 9 (3): e1003355.

Ziegler, A. & König, I. R. (2010). *A Statistical Approach to Genetic Epidemiology*. 2. utg. Darmstadt: Wiley-VCH Verlag GmbH & Co. KGaA.

Tillegg A - Programvarer

A.1 Plink

Det finnes tre egnede ROH detekteringsprogrammer, hvor Plink kommer best ut med tanke på IBD kartlegging (Howrigan et al. 2011). Det er derfor valgt å benytte dette programmet for å sammenligne hvor godt den skjulte markovmodellen i kapittel 3.3 detekterer autozygote områder i forhold til eksisterende programmer.

Plink er et kommandolinje-program som kan lastes ned fra *http://pngu.mgh.harvard.edu/~purcell/plink/*. Siste versjon er i skrivende stund v1.07 (Purcell 2009).

ROH metoden, *--homozyg,* detekterer homozygote markører i genomet basert på en sliding-window metode. Først designes et sliding-window ved å angi bredde, antall SNP og tillatte heterozygote- og manglende markører. Dersom vinduet er tilfredsstillende, klassifiseres det som et homozygot vindu. En gitt markør som dekkes av > 5% (standard verdi) overlappende homozygote vinduer er en IBD-kandidat. Dersom områder med IBD-kandidater oppfyller kriteriet for å bli klassifisert som et homozygot område, detekteres området som autozygot (Purcell et al. 2007).

Om Plink detekterer et område som autozygot eller ikke er avhengig av 9 parametere, hvor standard verdier er vist i tabell A.1-1.

Tabell A.1-1: Standard parametere for --homozyg metoden i Plink *(Purcell 2009).*

Plink-kode	**Standard parameterverdi**	**Definisjon**
--homozyg-window-kb	5000	Lengden på sliding-widow
--homozyg-window-snp	50	Antall markører i sliding-window
--homozyg-window-het	1	Antall heterozygote markører tillatt i et homozygot vindu
--homozyg-window-missing	5	Antall manglende genotyper tillatt i et homozygot vindu
--homozyg-window-threshold	0.05	Andelen homozygote vinduer som må overlappe for at en markør skal defineres homozygot.
--homozyg-snp	100	Minste antall markører i et homozygot område
--homozyg-kb	1000	Minimumslengde på homozygot område
--homozyg-density	50	1 SNP per 50 Kb
--homozyg-gap	1000	Hvis avstanden mellom to markører er mer enn 1000 Kb blir det homozygote område delt i to.

A.2 Filtus

Filtus er et program utviklet av Magnus Dehli Vigeland. Siste versjon er i skrivende stund v0.99-4 og kan lastes ned fra *http://folk.uio.no/magnusv/filtus.html*.

Filtus er et analyseverktøy som benytter data fra storskala-sekvensering. Ved hjelp av ulike filtreringsteknikker innsnevres antall interessante genvarianter. Programmet er spesielt egnet til å finne kausale genvarianter for monogene egenskaper. Analysen baserer seg på tilstedeværelse av genvarianter hos relaterte eller ikke-relaterte individer (Vigeland 2014a).

Foreløpig benytter Filtus Plink for å detektere autozygote områder. Et mål med denne oppgaven er å bidra til at en egen detekteringsalgoritme kan bli implementert i Filtus. Kriteriene er klare, algoritmen må håndtere autozygositetskartlegging mer tilfredsstillende enn Plink.

A.3 Python

Python er et objektorientert, høynivå programmeringsspråk med enkel syntaks. Dette gjør programmeringsspråket velegnet til portabel programutvikling (*The Python Tutorial*).

Python kan lastes ned fra den offisielle hjemmesiden, *https://www.python.org/.* I skrivende stund finnes det to versjoner av Python, Python 3.4.0 og Python 2.7.6. Utfyllende informasjon om valg av versjon og hvordan man installerer programmet er dokumentert på denne hjemmesiden. Her finnes også en offisiell Python tutorial.

Python er stadig under utvikling, da det er basert på delt kunnskap og felles forbedring mot et bedre programmeringsspråk.

A.4 R

R er et programmeringsspråk anvendt av statestikkere da det er velegnet for statistiske beregninger og grafiske framstillinger. Programmeringsspråket er objektorientert, da det er basert på programmeringsspråket S, utviklet i 1980 av Rick Becker, John M. Chambers og Allan Wilks (R Core Team 2014; Venables et al. 2014).

R kan lastes ned fra den offisielle hjemmesiden *http://cran.r-project.org/*. Den nyeste versjonen er i skrivende stund R-3.1.0. Fra den gitte hjemmesiden finnes det veiledende informasjon om nedlastning og installasjon for både Mac, Linux og Windows.

R er stadig under utvikling og forbedring av CRAN-miljøet, da det kontinuerlig utvikles nye pakker av brukere for brukere. Allikevel er de fleste R program skrevet for å løse gitte problemstillinger (Venables et al. 2014).

Tillegg B – Skript

Implementeringen i denne oppgaven er gjort på engelsk siden dette letter anvendelse for eventuell implementering i Filtus og annen publisering. Plottene har derimot norske figurtekster for å kunne bli anvendt i oppgaven.

B.1 Forward-backward algoritmen: «fwd_bwd.py»

```
#!/usr/bin/env python

"""
Forward-backward algorithm.

A script making use of HMM.py to calculate the most probable hidden state at a
given position t based on a observation sequence.
"""

__version__ = '0.1'
__author__ = "Kristina Stormo Gjøtterud"
__email__ = "kristina.stormo@gjotterud.no"

import HMM as hmm
from numpy import array
from math import exp, log
import sys

def FwdBwd(infile, error, a, f, logs):
    """
    Forward-backward algorithm to compute a posteriori probabilities.

    A local optimization to find the most probable hidden state at a given
    position t based on a observation sequence.
    Assuming known initiation-, transition- and emission probabilities.

    Input to the function:
    infile is a data frame with four columns:
    marker positions (CM), reference allele (al1),
    alternative allele (al2) and alternative allele frequency (Bfreq).

    error = the genotyping error-rate
    a = the distance between two states, exponential distributed
    f = inbreeding coefficient
    logs = 1 if calculations should be log transformed, 0 if not.

    Returning an array with two columns: the marker positions in cM and
    the a posteriori probabilities for the state to be IBD.
    """

    states = (1, 0) # 1 = IBD, 0 = not-IBD
    data = hmm.readfile(infile)
    n = len(data) # Number of observations
    Bfreqvec = hmm.frequencies(data)
    obsvec = hmm.observations(data)
    posvec = hmm.positions(data)
    distvec = hmm.distance(posvec)

    # Probability matrices
    I = hmm.initiation(f, logspace=logs)
    Evec = [hmm.emission(q, error, logspace=logs) for q in Bfreqvec]
```

```
    Tvec = [hmm.transition(d, a, f, logspace=logs) for  d in distvec]

    # The algorithm
    # For more details see chapter 3.3 in the master thesis of
    # Gjøtterud K. S, 2014, "Mapping of autozygous regions with
    # exome-sequencing: statistical methods and implementation"

    # Forward part #
    lfwd = [{}]

    # Initialization
    for j in states:
        lfwd[0][j] = I[j] + Evec[0][j][obsvec[0]]

    # Induction
    for t in range(1,n):
        lfwd.append({})
        for j in states:
           lfwd[t][j] = hmm.logsum([lfwd[t-1][i] + Tvec[t-1][i][j] +
                                    Evec[t][j][obsvec[t]]
                                    for i in states])

    # Termination
    lsum_fwd = hmm.logsum([lfwd[-1][j] for j in states])

    # Backward part #
    lbwd = [{} for x in range(n)]

    # Initialization
    for j in states:
        lbwd[n-1][j] = log(1)

    # Induction
    for t in reversed(range(n-1)):
        for j in states:
              lbwd[t][j] = hmm.logsum([lbwd[t+1][i] + Tvec[t][j][i] +
                                       Evec[t+1][i][obsvec[t+1]]
                                       for i in states])

    # A posteriori part #
    # A posteriori probabilities in log-space
    lprob = [{j : lfwd[t][j] + lbwd[t][j] - lsum_fwd for j in states}
            for t in range(n)]

    # A posteriori probabilities exp() transformed
    prob = [{j: exp(lprob[t][j]) for j in states} for t in range(n)]

    tab = array([[repr(posvec[x]).rjust(2), repr(prob[x][1]).rjust(3)]
                for x in range(n)])

    return tab

if __name__ == "__main__":
    if len(sys.argv) < 7:
        print "Syntax error. Too few arguments"
        sys.exit()
    infile = sys.argv[1]
    error = float(sys.argv[2])
    a = float(sys.argv[3])
    f = float(sys.argv[4])
    logs = sys.argv[5]
    outfile = sys.argv[6]

    hmm.writefile(FwdBwd(infile, error, a, f, logs), outfile)
```

B.2 Viterbi-algoritmen: «viterbi.py»

```
#!/usr/bin/env python

"""
Viterbi algorithm.

A script making use of HMM.py to calculate the most probable
path of hidden states given a sequence of observations.
"""

__version__ = '0.1'
__author__ = "Kristina Stormo Gjøtterud"
__email__ = "kristina.stormo@gjotterud.no"

import HMM as hmm
from numpy import array
from math import log
import sys

def viterbi(infile, error, a, f, logs):
    """
    Global optimization to find the state path behind observations.

    The Viterbi algorithm seek to find the most probable path of
    hidden states behind a given observation sequence.
    Assuming known initiation-, transition- and emission probabilities.

    Input for the function:
    Infile is a data frame of four columns:
    marker positions (CM), reference allele (al1),
    alternative allele (al2) and alternative allele frequency (Bfreq).

    error = the genotyping error-rate
    a = the distance between two states, exponential distributed
    f = inbreeding coefficient
    logs = 1 if calculations should be log-transformed, 0 if not.

    Returning an array with two columns: the marker positions in cM and
    the most probable hidden state at that position.
    """

    states = (1, 0) # 1 = IBD, 0 = not-IBD
    data = hmm.readfile(infile)
    n = len(data) # Number of observations
    Bfreqvec = hmm.frequencies(data)
    obsvec = hmm.observations(data)
    posvec = hmm.positions(data)
    distvec = hmm.distance(posvec)

    # Probability matrices
    I = hmm.initiation(f, logspace=logs)
    Evec = [hmm.emission(q, error, logspace=logs) for q in Bfreqvec]
    Tvec = [hmm.transition(d, a, f, logspace=logs) for d in distvec]

    # The algorithm #
    # For more details see chapter 3.3 in the master thesis of
    # Gjøtterud K. S, 2014, "Mapping of autozygous regions with
    # exome-sequencing: statistical methods and implementation"

    trellis = [{}] # store states and their probabilities of occurrence
    path = {} # store states

    # Initialization
```

```
    for j in states:
        trellis[0][j] = I[j]+ Evec[0][j][obsvec[0]]
        path[j] = [j]

    # Recursion
    for t in range(1,n):
        trellis.append({})
        newpath ={}
        for j in states:
            (prob, state) = max((trellis[t-1][i] + Tvec[t-1][i][j] +
                                 Evec[t][j][obsvec[t]], i)
                                 for i in states)
            trellis[t][j] = prob
            newpath[j] = path[state] + [j]

        path = newpath

     # Termination and backtracking
    (prob, state) = max((trellis[t][j], j) for j in states)

    return array(zip(posvec, path[state]))

if __name__ == "__main__":
     if len(sys.argv) < 7:
         print "Syntax error. Too few arguments"
         sys.exit()
     infile = sys.argv[1]
     error = float(sys.argv[2])
     a = float(sys.argv[3])
     f = float(sys.argv[4])
     logs = sys.argv[5]
     outfile = sys.argv[6]

     hmm.writefile(viterbi(infile, error, a, f, logs), outfile)
```

B.3 HMM: «HMM.py»

```
#!/usr/bin/env python

"""
HMM.

A script containing the function needed to create a HMM.
"""

__version__ = '0.1'
__author__ = "Kristina Stormo Gjøtterud"
__email__ = "kristina.stormo@gjotterud.no"

from math import e, exp, log

def readfile(infile):
    """ Reading a data frame.

    Needs to contain the following four columns: CM, al1, al2, Bfreq.
    """

    with open(infile, 'r') as fil:
        next(fil) # skip header
        data = [[str(x) for x in ln.split()]for ln in fil]
    return data

def observations(data):
    """ A vector of the observed events.

    Distinguish between homozygous for reference allele (0),
    heterozygous and homozygous for the alternative allele (1).
    """

    obs = []
    for i in range(len(data)):
        if data[i][1] == data[i][2] == '0':
                obs.append(1) # homozygous for reference allele
        elif data[i][1] == data[i][2] == '1':
                obs.append(2) # homozygous for alternative allele
        else:
                obs.append(0) # heterozygous
    return obs

def positions(data):
    """ A vector of the marker positions.

    Positions given in cM (assuming 1 cM = 1 Mb).
    """

    return [float(d[0]) for d in data]

def distance(pos):
    """ A distance vector of distances between markers."""

    return [y-x for x,y in zip(pos[:-1], pos[1:])]

def frequencies(data):
    """ Vector of B-allele frequencies. """

    return[float(d[3]) for d in data]

def logsum(v):
    """ Calculating the sum of a vector of logarithmic numbers."""
```

```
    a = max(v)
    return a + log(sum(exp(x-a) for x in v))

def transition(dist, a, f, logspace=0):
    """
    Compute transition probabilities for a HMM.

    Compute transition probabilities between hidden-states,
    when moving from time t to t+1.
    Genetic distance (cM) between the two markers are required.

    Assuming known parameters of a and f.
    If logspace = 1, calculations are log-transformed.

    Kay in outer- and inner dictionary: 0 = not-IBD, 1 = IBD.

    Return a dictionary containing the transition probabilities.
    """

    if logspace == 0:
        qk = e**(-a*dist)

        T = {
            1: {1: (1-qk)*f + qk, 0: (1-qk)*(1-f)},
            0: {1: (1-qk)*f, 0: (1-qk)*(1-f) + qk}
            }

    else:
        if dist == 0:
            dist = 1e-06

        qk = e**(-a*dist)
        lqk = -a*dist
        lc = log(1-qk)
        lf = log(f)
        lo = log(1-f)
        libd = lc + lf
        lnibd = lc + lo

        T = {
            1: {1: libd + log(1+exp(lqk-libd)), 0: lc + lo},
            0: {1: lc + lf, 0: lnibd + log(1+exp(lqk-lnibd))}
            }

    return T

def initiation(f, logspace=0):
    """
    Compute initiation probabilities for a HMM.

    Assuming that the probability for the first state being IBD or
    not-IBD is equivalent with the inbreeding-coefficient (f).

    If log-space = 1, calculations are log-transformed.

    Kay in dictionary: 0 = not-IBD, 1 = IBD.

    Return a dictionary containing the initiation probabilities.
    """

    if logspace==0:
        I = {1: f, 0: 1-f}
    else:
        I = {1: log(f), 0: log(1-f)}
```

```
    return I

def emission(q, error, logspace=0):
    """
    Compute emission probabilities for a HMM.

    This function takes as input three parameters:
    q = the minor allele frequency
    error = error-rate
    logspace = if 1, calculations are log-transfermed

    The output is an emission-matrix of probabilities between a given
    hidden-state and the observed event.

    Kay in outer dictionary: 0 = not-IBD, 1 = IBD.
    Kay in inner dictionary: 0 = heterozygous,
                             1 = homozygous for reference allele,
                             2 = homozygous for alternative allele.

    Return a dictionary containing the emission probabilities.
    """

    if logspace==0:
        p = 1-q

        E = {
            1: {1: (1-error)*p + error*p**2,
                0: error*2*p*q,
                2: (1-error)*q + error*q**2 },
            0: {1: p**2, 0: 2*p*q, 2: q**2}
            }

    else:
        if q == 1:
            q = 0.999999
        elif q == 0:
            q = 1e-06

        lp = log(1-q)
        lq = log(q)
        le = log(error)
        lne = log(1-error)

        E = {
            1: {1: logsum([lne + lp , le + 2*lp]),
                0: le + log(2) + lp + lq,
                2: logsum([lne + lq , le + 2*lq])},
            0: {1: 2*lp, 0: log(2) + lp + lq, 2: 2*lq}
            }

    return E

def writefile(data, outfile):
    """ Writing data, which must be an array, to a text-file."""

    with open(outfile, 'w') as fil:
        fil.write("\n".join(" ".join(map(str,x)) for x in data))
```

B.4 Simulerte data og validering av metoder: «simulations.R»

```
# version: '0.1'
# author: "Kristina Stormo Gjøtterud"
# email: "kristina.stormo@gjotterud.no"

# This script make a pedigree to simulate eksomedata for n simulations.
# The exomedata contains informations of true autozygous and
# not-autozygous regions for an individual in the given pedigree.
# Detection of true autozygous regions are performed by six methods:
# forward-backward algorithm, Viterbi algorithm and
# four Plink methods(plink0-plink3).
# The script calculates the power of the methods where figures illustrates
# results.

options("scipen" = 100)
options(stringsAsFactors = F)

source("validate.R")
require(plotrix)

# Parameters to be specified
simfi <- "simfile.txt"
plinkPATH <- "C:/windows/plink.exe"
pythonPATH <- "C:/Python27/python.exe"
a <- 0.063
f <- 0.0625
error <- 0.0001
threshold <- 0.5
nsim <- 500


#Marker positions(Mb) and Bfreq from a real exome
dat <- read.table("exome50_freqs.txt", header=T)

ped <- pedigree(1,7,8,1) # First-cousin

analyse_sim <- function(simfile, plinkpath, pythonpath, autsim,
                        error, a, f, threshold, chr=NULL, plot=F){
  # Make it feasible to compare different approaches.
  # autsim is the result from simulation() - a list containing:
  #     snp-, IBD- and map information.
  #
  # Return a list of 16 elements:
  # num_seg: number of autozygous segments in the simulation
  # len_seg: length of each autozygous segments in the simulation (Mb)
  # total_aut: total length of the autozygous segments (Mb)
  # L: length of the chromosome (Mb)
  # fbIBD, vitIBD, plink0IBD, plink1IBD, plink2IBD, plink3IBD:
  #     a list of TP-, FP-, FN-, TP- lengths (Mb)

  snps <- autsim$snps
  fb <- fwdbwd(simfile, pythonpath, error=error, a=a, f=f, logs=1)
  fb[,1] <- snps[,1] # from cM to Mb positions.
  fbIBD <- ranges(fb, threshold=threshold)

  vit <-  viterbi(simfile, pythonpath, error=error, a=a, f=f, logs=1)
  vit[,2] <- as.integer(vit[,2])
  vit[,1] <- snps[,1] # from cM to Mb positions.
  vitIBD <- ranges(vit)

  snp_matrix_plink <- snps[, c(1,3,4)]
  plink0 <- plink(snp_matrix_plink, plinkpath, window_kb=0,
                  window_snp=1, window_het=0, window_miss=0,
```

```
                  window_thres=1, homozyg_snp=10, homozyg_kb=0,
                  homozyg_dens=10000, homozyg_gap=1000000) # self defined
  plink1 = plink(snp_matrix_plink, plinkpath, window_kb=0,
                 window_snp=1, window_het=0, window_miss=0,
                 window_thres=1, homozyg_snp=10, homozyg_kb=500,
                 homozyg_dens=10000, homozyg_gap=1000000)  # self defined
  plink2 = plink(snp_matrix_plink, plinkpath, window_kb=0,
                 window_snp=50, window_het=1, window_miss=5,
                 window_thres=0.05, homozyg_snp=0, homozyg_kb=2500,
                 homozyg_dens=50, homozyg_gap=5000) # EX-HOM
  plink3 = plink(snp_matrix_plink, plinkpath) # default

  if(plot) { # compare trueIBD and detected IBD for all the approaches
    layout(matrix(c(1,1,1,2,1,1,1,2),nrow=2,ncol=4, byrow=T))
    plot(fb, type="l", lwd=2, ylim=c(-0.26, 1.05),
         xlab="Posisjon (Mb)", ylab="A posteriori-sannsynlighet",
         main=paste("Kromosom", chr))
    if(nrow(autsim$fasit) > 0) segments(autsim$fasit[,1], -0.04,
                                        autsim$fasit[,2], -0.04,
                                        col="green", lwd=3, lend=2)
    if(nrow(vitIBD) > 0) segments(vitIBD[,1], -0.08, vitIBD[,2], -0.08,
                                  col="blue", lwd=3, lend=2)
    if(nrow(plink0) > 0) segments(plink0[,1], -0.12, plink0[,2], -0.12,
                                  col="red3", lwd=3, lend=2)
    if(nrow(plink1) > 0) segments(plink1[,1], -0.16, plink1[,2], -0.16,
                                  col="orange", lwd=3, lend=2)
    if(nrow(plink2) > 0) segments(plink2[,1], -0.20, plink2[,2], -0.20,
                                  col="olivedrab4", lwd=3, lend=2)
    if(nrow(plink3) > 0) segments(plink3[,1], -0.24, plink3[,2], -0.24,
                                  col="purple", lwd=3, lend=2)


    plot(c(0,1), c(0,1), type="n", axes=F, xlab="", ylab="")
    legend("center", col=c("green", "black","blue", "red3", "orange",
                           "olivedrab4", "purple"),
           legend=c("fasit", "fwd-bwd", "Viterbi", "plink0", "plink1",
                    "plink2", "plink3"),
           lwd=2, y.inters=0.8, cex=1.2)  #fasit = true IBD
  }

  fasitIBD <- autsim$fasit # trueIBD
  numb_seg <- nrow(fasitIBD) # number of IBD-segments
  len_seg <- fasitIBD[,2] - fasitIBD[,1] # length of each IBD-seg (Mb)
  total_aut <- sum(len_seg) # total length of all IBD-segments (Mb)
  L <- attr(autsim$map, 'length_Mb') # length of chromosome (Mb)

list(numb_seg=numb_seg, len_seg=len_seg, total_aut = total_aut, L=L,
     fbIBD=fbIBD, vitIBD=vitIBD, plink0IBD=plink0,
     plink1IBD=plink1, plink2IBD=plink2, plink3IBD=plink3,
     fb = validateIBD(fasitIBD, fbIBD, L, summary=T),
     vit = validateIBD(fasitIBD, vitIBD, L, summary=T),
     plink0 = validateIBD(fasitIBD, plink0, L, summary=T),
     plink1 = validateIBD(fasitIBD, plink1, L, summary=T),
     plink2 = validateIBD(fasitIBD, plink2, L, summary=T),
     plink3 = validateIBD(fasitIBD, plink3, L, summary=T))
}

# Multiple simulations to validate approaches
simses <- lapply(1:nsim, function(i){
  # return a  list of 27 elements:
  # gen_numb_seg: number of IBD segments.
  # gen_len_seg: lengths of IBD segments (Mb).
  # gen_IBDprop: autozygous proportion.
  # Number of IBD segments, length of IBD segments and
```

```
#     autozygous proportion for fb, vit, and the four Plink methods.
# fb_validate, vit_validate, plink0_val, plink1_val, plink2_val,
#     plink3_val: TP, FP, FN, TN ratios.

genome <- lapply(1:22, function(i) { # 22 autosomes
  chr <- dat$CHROM == i
  Bfreq <- dat$Bfreq[chr]
  pos_Mb <- dat$POS[chr]/1e6 # from cM to Mb

  autsim <- simulation(simfi, ped, id=9, Bfreq=Bfreq, error=error,
                       pos_Mb=pos_Mb, remAA=T, chr=i, model="chi")
  analyse_sim(simfile=simfi, plinkpath=plinkPATH,
              pythonpath=pythonPATH, autsim, error=error, a=a, f=f,
              threshold=threshold, chr=i, plot=F)
})


# Collect information from all the 22 autosomes in each simulation
#(genome information)

# trueIBD information
gen_numb_seg <- sum(sapply(genome, function(chromres)
                           chromres$numb_seg))
gen_len_seg <- unlist(sapply(genome, function(chromres)
                             chromres$len_seg)) #(Mb)
gen_fasitIBD <- sum(sapply(genome, function(chromres)
                           chromres$total_aut), na.rm=T)# Total len
gen_L <- sum(sapply(genome, function(chromres) chromres$L)) # Genome
gen_fasit_nIBD <- gen_L - gen_fasitIBD # total not-IBD length (Mb)
fasitvec <- c(gen_fasitIBD, gen_fasit_nIBD,
              gen_fasitIBD, gen_fasit_nIBD)

# True autozygous proportion
gen_IBDprop <- round(gen_fasitIBD/gen_L, 4)*100


# estimatedIBD information
# x_tot: TP, FP, FN, TN (total length (Mb)),
# x_validate: TP, FP, FN, TN (ratios)

# fwd-bwd estimates
fbIBD_seg <- sum(sapply(genome, function(chromres)
                        nrow(chromres$fbIBD)))
fbIBD_len <- unlist(sapply(genome, function(chromres)
                           (chromres$fbIBD[,2]-chromres$fbIBD[,1])))
fbIBD_prop <- round(sum(fbIBD_len)/gen_L,4)*100
fb_tot <- rowSums(sapply(genome, function(chromres)
                         chromres$fb), na.rm=T)
fb_validate <- fb_tot/fasitvec


# Viterbi
vitIBD_seg <- sum(sapply(genome, function(chromres)
                         nrow(chromres$vitIBD)))
vitIBD_len <- unlist(sapply(genome, function(chromres)
                            (chromres$vitIBD[,2]-chromres$vitIBD[,1])))
vitIBD_prop <- round(sum(vitIBD_len)/gen_L,4)*100
vit_tot <- rowSums(sapply(genome, function(chromres)
                          chromres$vit), na.rm=T)
vit_validate <- vit_tot/fasitvec

# Plink0
plink0IBD_seg <- sum(sapply(genome, function(chromres)
                            nrow(chromres$plink0IBD)))
```

```
  plink0IBD_len <- unlist(sapply(genome, function(chromres)
                                 (chromres$plink0IBD[,2]-
                                  chromres$plink0IBD[,1])))
  plink0IBD_prop <- round(sum(plink0IBD_len)/gen_L,4)*100
  plink0_tot <- rowSums(sapply(genome, function(chromres)
                               chromres$plink0), na.rm=T)
  plink0_validate <- plink0_tot/fasitvec

  # Plink1
  plink1IBD_seg <- sum(sapply(genome, function(chromres)
                              nrow(chromres$plink1IBD)))
  plink1IBD_len <- unlist(sapply(genome, function(chromres)
                                 (chromres$plink1IBD[,2]-
                                  chromres$plink1IBD[,1])))
  plink1IBD_prop <- round(sum(plink1IBD_len)/gen_L,4)*100
  plink1_tot = rowSums(sapply(genome, function(chromres)
                              chromres$plink1), na.rm=T)
  plink1_validate = plink1_tot/fasitvec

  # Plink2
  plink2IBD_seg <- sum(sapply(genome, function(chromres)
                              nrow(chromres$plink2IBD)))
  plink2IBD_len <- unlist(sapply(genome, function(chromres)
                                 (chromres$plink2IBD[,2]-
                                  chromres$plink2IBD[,1])))
  plink2IBD_prop <- round(sum(plink2IBD_len)/gen_L,4)*100
  plink2_tot = rowSums(sapply(genome, function(chromres)
                              chromres$plink2), na.rm=T)
  plink2_validate = plink2_tot/fasitvec

  # Plink3
  plink3IBD_seg <- sum(sapply(genome, function(chromres)
                              nrow(chromres$plink3IBD)))
  plink3IBD_len <- unlist(sapply(genome, function(chromres)
                                 (chromres$plink03BD[,2]-
                                  chromres$plink3IBD[,1])))
  plink3IBD_prop <- round(sum(plink3IBD_len)/gen_L,4)*100
  plink3_tot = rowSums(sapply(genome, function(chromres)
                              chromres$plink3), na.rm=T)
  plink3_validate = plink3_tot/fasitvec

  val <-list(gen_numb_seg=gen_numb_seg, gen_len_seg=gen_len_seg,
             gen_IBDprop=gen_IBDprop,
             fbIBD_seg=fbIBD_seg, fbIBD_len=fbIBD_len,
             fbIBD_prop=fbIBD_prop,
             vitIBD_seg=vitIBD_seg, vitIBD_len=vitIBD_len,
             vitIBD_prop=vitIBD_prop,
             plink0IBD_seg=plink0IBD_seg, plink0IBD_len=plink0IBD_len,
             plink0IBD_prop=plink0IBD_prop,
             plink1IBD_seg=plink1IBD_seg, plink1IBD_len=plink1IBD_len,
             plink1IBD_prop=plink1IBD_prop,
             plink2IBD_seg=plink2IBD_seg, plink2IBD_len=plink2IBD_len,
             plink2IBD_prop=plink2IBD_prop,
             plink3IBD_seg=plink3IBD_seg, plink3IBD_len=plink3IBD_len,
             plink3IBD_prop=plink3IBD_prop,
             fb_validate=fb_validate, vit_validate=vit_validate,
             plink0_val=plink0_validate, plink1_val=plink1_validate,
             plink2_val=plink2_validate, plink3_val=plink3_validate)

  lapply(val, cat, "\n", file="validate.txt", append=T)
  return(val)
})

# Collect information from nsim simulations
```

```
# trueIBD information
simses_numb_seg <- sapply(simses, function(simres) simres$gen_numb_seg)
simses_len_seg <- sapply(simses, function(simres) simres$gen_len_seg)
simses_sumlen <- sapply(simses_len_seg, function(simres) sum(simres))
simses_len <- unlist(sapply(simses, function(simres)
                            simres$gen_len_seg))
simses_IBDprop <- t(sapply(simses, function(simres)
                           simres$gen_IBDprop))
simses_mean_IBDprop <- round(mean(simses_IBDprop),2)

#estimatedIBD information
# x_rates: nrow=nsim, columns=TPR, FPR, FNR, TNR

# bwd-fwd
simses_fbIBD_seg <- sapply(simses, function(simres) simres$fbIBD_seg)
simses_fbIBD_len <- unlist(sapply(simses, function(simres)
                                  simres$fbIBD_len))
simses_fbIBD_prop <- sapply(simses, function(simres) simres$fbIBD_prop)
fb_rates <- t(sapply(simses, function(simres) simres$fb_val))

# Viterbi
simses_vitIBD_seg <- sapply(simses, function(simres) simres$vitIBD_seg)
simses_vitIBD_len <- unlist(sapply(simses, function(simres)
                                   simres$vitIBD_len))
simses_vitIBD_prop <- sapply(simses, function(simres)
                                   simres$vitIBD_prop)
vit_rates <- t(sapply(simses, function(simres) simres$vit_val))

# Plink0
simses_plink0IBD_seg <- sapply(simses, function(simres)
                               simres$plink0IBD_seg)
simses_plink0IBD_len <- unlist(sapply(simses, function(simres)
                                      simres$plink0IBD_len))
simses_plink0IBD_prop <- sapply(simses, function(simres)
                                simres$plink0IBD_prop)
plink0_rates <- t(sapply(simses, function(simres) simres$plink0_val))

# Plink1
simses_plink1IBD_seg <- sapply(simses, function(simres)
                               simres$plink1IBD_seg)
simses_plink1IBD_len <- unlist(sapply(simses, function(simres)
                                      simres$plink1IBD_len))
simses_plink1IBD_prop <- sapply(simses, function(simres)
                                simres$plink1IBD_prop)
plink1_rates <- t(sapply(simses, function(simres) simres$plink1_val))

# Plink2
simses_plink2IBD_seg <- sapply(simses, function(simres)
                               simres$plink2IBD_seg)
simses_plink2IBD_len <- unlist(sapply(simses, function(simres)
                                      simres$plink2IBD_len))
simses_plink2IBD_prop <- sapply(simses, function(simres)
                                simres$plink2IBD_prop)
plink2_rates <- t(sapply(simses, function(simres) simres$plink2_val))

# Plink3
simses_plink3IBD_seg <- sapply(simses, function(simres)
                               simres$plink3IBD_seg)
simses_plink3IBD_len <- unlist(sapply(simses, function(simres)
                               simres$plink3IBD_len))
simses_plink3IBD_prop <- sapply(simses, function(simres)
                                simres$plink3IBD_prop)
plink3_rates <- t(sapply(simses, function(simres) simres$plink3_val))
```

```
# nrow=nsim, columns=fb, vit, plink0, plink1, plink2, plink3
tpr <- sapply(list(fb = fb_rates[,1],
                   vit = vit_rates[,1],
                   plink0 = plink0_rates[,1],
                   plink1 = plink1_rates[,1],
                   plink2 = plink2_rates[,1],
                   plink3 = plink3_rates[,1]),
              function(i) i)

fpr <- sapply(list(fb = fb_rates[,2],
                   vit = vit_rates[,2],
                   plink0 = plink0_rates[,2],
                   plink1 = plink1_rates[,2],
                   plink2 = plink2_rates[,2],
                   plink3 = plink3_rates[,2]),
              function(i) i)

fnr <- sapply(list(fb = fb_rates[,3],
                   vit = vit_rates[,3],
                   plink0 = plink0_rates[,3],
                   plink1 = plink1_rates[,3],
                   plink2 = plink2_rates[,3],
                   plink3 = plink3_rates[,3]),
              function(i) i)

tnr <- sapply(list(fb = fb_rates[,4],
                   vit = vit_rates[,4],
                   plink0 = plink0_rates[,4],
                   plink1 = plink1_rates[,4],
                   plink2 = plink2_rates[,4],
                   plink3 = plink3_rates[,4]),
              function(i) i)



#########################################################################
#  Figure: 20 Boxplot of TPR, FPR, FNR and TNR for all the approaches  #
#########################################################################
x11()
par(mar=c(3,4,3,1)+0.1)
layout(matrix(c(rep(1,4),rep(c(2,2,3,3),2), rep(c(4,4,5,5),2)),
              nrow=5, ncol=4, byrow=T))
# Info
plot(c(-1,1), c(-1,1), type="n", axes=F, xlab="", ylab="")
box()
text(0,0, cex=1,
     labels=c(paste("Deteksjon av autozygote områder\n",
                    "Antall simuleringer =", nsim,
                    "Feilrate =", error)))

#TPR
boxplot(tpr, ylim=c(0,1), pch=20, cex=0.8,
        ylab="Sannsynlighet", main="Sann positiv rate")
points(colMeans(tpr), pch=3, cex=0.5, col="red")

#TPR
boxplot(fpr, ylim=c(0,1), pch=20, cex=0.8,
        ylab="Sannsynlighet", main="Falsk positiv rate")
points(colMeans(fpr), pch=3, cex=0.5, col="red")

#TPR
boxplot(fnr, ylim=c(0,1), pch=20, cex=0.8,
        ylab="Sannsynlighet", main="Falsk negativ rate")
points(colMeans(fnr), pch=3, cex=0.5, col="red")
```

```
#TPR
boxplot(tnr, ylim=c(0,1), pch=20, cex=0.8,
        ylab="Sannsynlighet", main="Sann negativ rate")
points(colMeans(tnr), pch=3, cex=0.5, col="red")

#########################################################################
#        Figure: Distribution of number of segments per genome          #
#########################################################################
# true IBD
x11()
hist(simses_numb_seg, breaks=30,
     xlim=c(0,ceiling(max(simses_numb_seg)/5)*5),
     col="cornsilk3", border="cornsilk4",
     xlab="Totalt antall autozygote områder per genom",
     ylab="Frekvens",
     main="Histogram over antall autozygote områder",
     sub=paste("\nAntall simuleringer =", nsim), cex.sub=0.8)

# Comparison of approaches
x11()
multhist(list(simses_fbIBD_seg, simses_vitIBD_seg,
              simses_plink0IBD_seg, simses_plink1IBD_seg,
              simses_plink2IBD_seg, simses_plink0IBD_seg),
         breaks=10, ylim=c(0,500),
         col=c("black", "blue", "red3", "orange", "olivedrab4",
               "purple"),
         xlab="Totalt antall autozygote omrÃ¥der per genom",
         ylab="Frekvens",
         main="Histogram over antall autozygote områder",
         sub=paste("\nAntall simuleringer =", nsim,
                   ", Feilrate =", error), cex.sub=0.8)
legend("topright", col=c("black","blue", "red3", "orange",
                         "olivedrab4", "purple"),
       legend=c("fwd-bwd", "Viterbi", "plink0", "plink1", "plink2",
                "plink3"),
       lwd=2, y.inters=0.8, cex=0.8)


#########################################################################
#       Figure: Distribution of autozygosity ratio per genome           #
#########################################################################
# trueIBD
x11()
par(mar=c(6,4,4,1)+0.1)
hist(simses_IBDprop, breaks=35,
     xlim=c(0, ceiling(max(simses_IBDprop)/5)*5),
     col="cornsilk3", border="cornsilk4",
     xlab="Total andel autozygositet per genom (%)",
     ylab="Frekvens",
     main="Histogram over andel autozygositet",
     sub=paste("\nGjennomsnittlig andel =", simses_mean_IBDprop, "%",
               ", Antall simuleringer =", nsim), cex.sub=0.8)


# Comparison of approaches
x11()
multhist(list(simses_fbIBD_prop, simses_vitIBD_prop,
              simses_plink0IBD_prop, simses_plink1IBD_prop,
              simses_plink2IBD_prop, simses_plink0IBD_prop),
         breaks=9, ylim=c(0,500),
         col=c("black","blue", "red3", "orange", "olivedrab4",
               "purple"),
         xlab="Total andel autozygositet per genom (%)",
```

```
         ylab="Frekvens",
         main="Histogram\nSammenligner andel autozygositet for de
              ulike metodene",
         sub=paste("\nGjennomsnittlig sann andel =",
                   simses_mean_IBDprop, "%", ", Antall simuleringer =",
                   nsim, ", Feilrate =", error), cex.sub=0.8)
legend("topright", col=c("black","blue", "red3", "orange", "olivedrab4",
                         "purple"),
       legend=c("fwd-bwd", "Viterbi", "plink0", "plink1", "plink2",
                "plink3"),
       lwd=2, y.inters=0.8, cex=0.8)


#######################################################################
#     Figure: Distribution of segments length for nsim as a whole     #
#######################################################################
# trueIBD
x11()
dens <- density(simses_len)
hist(simses_len, freq=F, col="cornsilk3", border="cornsilk4",
     breaks=100, ylim=c(0,0.08),
     xlab="Lengde av autozygot område (Mb)",
     ylab="Tetthet",
     main="Lengdefordeling over autozygote områder",
     sub=paste("\nAntall simuleringer =", nsim), cex.sub=0.8)
lines(density(simses_len), lwd=2, col="red2")

# Comparison of approaches
x11()
plot(density(simses_len), type="l",lwd=2, col="green",
     ylim=c(0,0.4),
     xlab="Lengden av autozygote områder (Mb)",
     ylab="Tetthet",
     main="Tetthetsfordeling av
           autozygote områder for de ulike metodene",
     sub=paste("\nAntall simuleringer =", nsim, ", Feilrate =", error),
     cex.sub=0.8)
if(length(simses_fbIBD_len) > 0) lines(density(simses_fbIBD_len),
                                       lwd=2, col="black")
if(length(simses_vitIBD_len) > 0) lines(density(simses_vitIBD_len),
                                        lwd=2, col="blue")
if(length(simses_plink0IBD_len) > 0) lines(density(simses_plink0IBD_len),
                                           lwd=2, col="red3")
if(length(simses_plink1IBD_len) > 0) lines(density(simses_plink1IBD_len),
                                           lwd=2, col="orange")
if(length(simses_plink2IBD_len) > 0) lines(density(simses_plink2IBD_len),
                                           lwd=2, col="olivedrab4")
if(length(simses_plink3IBD_len) > 0) lines(density(simses_plink3IBD_len),
                                           lwd=2, col="purple")
legend("topright", col=c("green", "black", "blue", "red3", "orange",
                         "olivedrab4", "purple"),
       legend=c("fasit IBD", "Fwd-Bwd", "Viterbi", "Plink0", "plink1",
                "plink2", "plink3"),
       lwd=2, y.inters=0.8, cex=0.8)
```

B.5 Funksjoner til simulerte data og validering av metoder: «validate.R»

```
# version: '0.1'
# author: "Kristina Stormo Gjøtterud"
# email: "kristina.stormo@gjotterud.no"

# Functions to simulations.R

source("autozyg_simulation5.R")
require(IBDsim)
require(paramlink)

options("scipen"=100)
options(stringsAsFactors = F)

viterbi <- function(simfile, pythonpath, error, a, f, logs=0){
  # Run python to estimate IBD regions with HMM - Viterbi algorithm
  # Return the most probable estimated state-path

  outvit <- "viterbi.txt"

  kommando <- paste(pythonpath, "viterbi.py",
                    simfile, error, a, f, logs, outvit,
                    sep=" " )

  system(kommando)

  return(read.table(outvit)) # Estimated state-path
}

fwdbwd <- function(simfile, pythonpath, error, a, f, logs=0){
  # Run python to estimate a probabilities with HMM -
  # forward-backward algorithm.
  # Return estimated a posteriori probabilities for each
  # marker position.

  outFB <- "fwdbwd.txt"

  kommando <- paste(pythonpath, "fwd_bwd.py",
                    simfile, error, a, f, logs, outFB,
                    sep=" " )

  system(kommando)

  return(read.table(outFB)) # estimated a a probabilities
}

pedfile <- function(snp_matrix, file_prefix){ #snp_matrix(MB, AL1, AL2)
  # Create a .ped file.
  # One-line, white-space delimited file

  patient <- c(1,1,1,2) # family ID, Individual ID, sex, affected phenotype
  alleles <- as.numeric(t(snp_matrix[,2:3])) + 1 # Genotypes
  pedf <- c(patient, alleles)

  file <- paste(file_prefix, "ped", sep=".")
  writeLines(paste(pedf, collapse=" "), file)

  invisible(pedf)
}

mapfile <- function(snp_matrix, file_prefix){ #snp_matrix(MB, AL1, AL2)
  # Create a .map file.
  # Returns a matrix of 3 columns: Chr, snpID, BpPos
```

```
  n <- nrow(snp_matrix)
  Chr <- rep(1, n)
  snpID <- 1:n
  BpPos <- snp_matrix$MB * 1e6
  map <- cbind(Chr, snpID, BpPos)

  file = paste(file_prefix, "map", sep=".")
  write.table(map, file=file, row.names=F, col.names=F)

  invisible(map)
}

plinkscript <- function(scriptname, file_prefix,
                        window_kb=5000, window_snp=50, window_het=1,
                        window_miss=5, window_thres=0.05,
                        homozyg_snp=100, homozyg_kb=1000,
                        homozyg_dens=50, homozyg_gap=1000){
  # Create a script containing parameters to be used running plink

  param <- c(paste("--file", file_prefix),
                  "--noweb ",
                  "--no-parents",
                  "--map3",
                  paste("--homozyg-window-kb", window_kb),
                  paste("--homozyg-window-snp", window_snp),
                  paste("--homozyg-window-het", window_het),
                  paste("--homozyg-window-missing", window_miss),
                  paste("--homozyg-window-threshold", window_thres),
                  paste("--homozyg-kb", homozyg_kb),
                  paste("--homozyg-snp", homozyg_snp),
                  paste("--homozyg-density", homozyg_dens),
                  paste("--homozyg-gap", homozyg_gap),
                  paste("--out", file_prefix))

  writeLines(param, scriptname, sep="\n")
}

plink <- function(snp_matrix, plinkpath, ... ){
  # Run plink to detect autozygous regions.
  # Calls pedfile, mapfile and plinkscript.
  # Returns the IBD-regions in a matrix. Columns: start, end.

  scriptname = "plinkscript.txt"
  file_prefix = "_plink_"

  pedfile(snp_matrix, file_prefix)
  mapfile(snp_matrix, file_prefix)
  plinkscript(scriptname, file_prefix, ...)

  # Run plink with parameters produced by plinkscript
  system(paste(plinkpath, "--script", scriptname),
         show.output.on.console=F)

  # Read .hom file produced by plink
  hom <- read.table(paste(file_prefix, ".hom", sep=""), header=T)

  # IBD ranges
  plinkIBD <- cbind(hom$POS1, hom$POS2) / 1e6 # convert to MB
  colnames(plinkIBD) <- c("start", "end")

  return(plinkIBD)
}
```

```
ranges <- function(estfile, threshold=NA){
  # Returns the IBD-regions in a matrix. Columns: start, end.

  # rearrange column trueIBD in tab
  n <- nrow(estfile)
  up <- c("", estfile[,2][-n])
  down <- c(estfile[,2][-1], "")

  # IBD ranges
  if (is.na(threshold)){
    start <- estfile[,1][estfile[,2] == 1 & up != 1]
    end <- estfile[,1][estfile[,2] == 1 & down != 1]
  } else {
    start <- estfile[,1][estfile[,2] >= threshold & up < threshold]
    end <- estfile[,1][estfile[,2] >= threshold & down < threshold]
  }

  return(cbind(start,end))
}

validateIBD <- function(trueIBD, estimatedIBD, L, summary=F){
  # If summary: Returns a list of four matrices:
  #             total length of TP-, FP-, FN-TN segments:
  # Else: Return a list of four matrices:
  #       TP-, FP-, FN- and TN segments, with columns: start, end.

  I <- IBDsim:::rangeIntersect
  C <- IBDsim:::rangeCompl

  TP <- I(list(trueIBD, estimatedIBD))
  FP <- I(list(C(trueIBD, L),estimatedIBD))
  FN <- I(list(C(estimatedIBD, L),trueIBD))
  TN <- I(list(C(trueIBD, L), C(estimatedIBD, L)))

  if(summary) {
    TP=sum(TP[,2]-TP[,1])
    FP=sum(FP[,2]-FP[,1])
    FN=sum(FN[,2]-FN[,1])
    TN=sum(TN[,2]-TN[,1])
    return(c(TP=TP, FP=FP, FN=FN, TN=TN))
  }

  return(list(TP=TP, FP=FP, FN=FN, TN=TN))
}

pedigree <- function(consanguinity, father, mother, noffs){
  # Set pedigree with paramlink

  cp <- cousinPed(consanguinity)
  cp <-addOffspring(cp, father=father, mother=mother, noffs=noffs)
  return(cp)
}

simulation <- function(simfile, pedigree, Bfreq, error, id,
                       SNPdist=NULL, pos_Mb=NULL, remAA=F, L=0,
                       chr=NULL, model="haldane"){
  # Simulate exome-data.
  # Returning a list:  snps (data frame of 6 columns:
  #                         MB, CM, al1, al2, Bfreq, trueIBD),
  #                    fasit (matrix of 2 columns: start, end),
  #                    map (list of chromosome positions,
  #                         with attr(,"length_Mb"))

  # Simulate IBD pattern
```

```
  map = if(L > 0) uniformMap(cM=L) else decode.map[[chr]]
  simulate <- autozyg.sim(pedigree, id, map=map, model=model)
  trueIBD <- ibd.fasit(simulate)$ibd

  # Simulate genotypes
  fakeIBD = runif(rbinom(1, 2, prob=0.5), min=0.5, max=1.5)
  snps <- SNPgeno.sim(simulate, SNPdist, pos_Mb, Bfreq, error, fakeIBD=fakeIBD)
  snps <- snps[!is.na(snps$CM),]
  if(remAA)
    snps <- removeAA(snps, allelA=0)

  # Write table
  write.table(snps[2:5], simfile, quote=F, row.names=F )

  return(list(snps=snps, fasit=trueIBD, map=map))
}
```

B.6 Simulering av autozygote områder: «autozyg_simulation5.R»

```
# auther: Magnus Dehli Vigeland
require(IBDsim)

# Some useful (but currently not public) functions in the
# IBDsim package
uniformMap = IBDsim:::uniformMap
loadMap = IBDsim:::loadMap
sap.finder = IBDsim:::sap.finder
getAlleles = IBDsim:::.getAlleles

rangeCompl = IBDsim:::rangeCompl  # the complement of a range matrix
rangeIntersect = IBDsim:::rangeIntersect  # the intersection of a list
                                          # of range matrices

decode.map = loadMap("decode")

autozyg.sim = function(ped, id, map, model="haldane") {
    # map: map = uniformMap(cM=L) gives a single chromosome of length
    #            L cM, where 1cM = 1Mb
    #      map = loadMap("decode")[[i]] gives chromosome i of the
    #            Decode map
    # model: either 'haldane' or 'chi'
    a = IBDsim(ped, sims=1, map=map, model=model, verbose=F)
    list(sim=a[[1]][[1]][[id]], map=map)
}


ibd.fasit = function(autsim) { # autsim = output from autozyg.sim
    sim = autsim$sim
    map = autsim$map
    L = attr(map, 'length_Mb')
    y = structure(list(sim), chromosome=attr(map, "chromosome"),
                  length_Mb=L)
    ibd = sap.finder(y, list('2'=1))[,2:3, drop=F]
    nibd = rangeCompl(ibd, L)
    rownames(nibd) = NULL
    list(ibd=ibd, nibd=nibd)
}

# Verbatim copy (but cm <-> mb) of cm2phys in IBDsim
phys2cm <-
function(Mb_locus, mapmat) { # mapmat matrise med kolonner 'Mb' og 'cM'
      last = mapmat[nrow(mapmat), ]
      nontriv = Mb_locus >= 0 & Mb_locus <= last[['Mb']]
      res = numeric(length(Mb_locus))
      res[!nontriv] <- NA
      mb <- Mb_locus[nontriv]
      interv = findInterval(mb, mapmat[, 'Mb'], all.inside=TRUE)
                          #.findInterval_quick not allowed
      res[nontriv] = mapmat[interv, 'cM'] +
                  (mapmat[interv+1, 'cM'] - mapmat[interv, 'cM']) *
                  (mb - mapmat[interv, 'Mb'])/(mapmat[interv+1, 'Mb']
                                               - mapmat[interv, 'Mb'])
      res
}

SNPgeno.sim = function(autsim, SNPdist=NULL, pos_Mb=NULL, Bfreq,
                       error_rate=0, alleles=c(0,1), fakeIBD=numeric()){
    # autsim: output from autozyg.sim
    # SNPdist: Denne er som før, men settes lik NULL ved bruk av pos_Mb
    # pos_Mb: vektor med fysiske markør-posisjoner.
    #         Settes lik NULL ved bruk av SNPdist
```

```
# Bfreq: Kan være enten et konstant tall, eller en vektor med samme
#        lengde som pos_Mb
# error_rate: som før
# alleles: vektor av lengde 2.
#
# Output: data frame med 5 kolonner MB, CM, al1, al2 og trueIBD

sim = autsim$sim
map = autsim$map
L = attr(map, 'length_Mb')
if(is.null(pos_Mb)) {
    stopifnot(!is.null(SNPdist) && length(SNPdist)==1)
    pos_Mb = seq(0, L, by=SNPdist)
}
nSNP = length(pos_Mb)


if (length(Bfreq)==1)
    Bfreq = rep(Bfreq, nSNP)
else
    stopifnot(length(Bfreq) == nSNP)

trueAlleles = getAlleles(sim, pos_Mb) # true alleles (in child)
trueIBD = trueAlleles[1, ]==trueAlleles[2, ]
status = as.character(trueIBD) # may change if fakeIBD

#map founder alleles to SNP alleles for all markers
nAllel = max(sim[[1]][,2], sim[[2]][,2])
snp_founders = rbinom(nSNP*nAllel, size=1, prob=Bfreq) # sample 0/1
                                                       # vector
dim(snp_founders) = c(nSNP, nAllel)

# map the child's true alleles to SNP alleles
snp_allele1 = snp_founders[cbind(1:nSNP, trueAlleles[1,])]
snp_allele2 = snp_founders[cbind(1:nSNP, trueAlleles[2,])]
snps = cbind(snp_allele1, snp_allele2)

# insert fake IBD segments at random positions with
# lengths (Mb) as indicated
for (fakeMb in fakeIBD) {
    fake_start = runif(1, min=0, max=max(0, L-fakeMb))
                                        # if L < fake_Mb, start at 0.
    fake_stop = fake_start + fakeMb
    snps_in_region = (pos_Mb > fake_start) & (pos_Mb < fake_stop)
    if (!any(snps_in_region))
        next
    a1 = rbinom(sum(snps_in_region), size=1,
                prob=Bfreq[snps_in_region])
    snps[snps_in_region, ] = c(a1,a1)
    status[snps_in_region & !trueIBD] = "FAKE"
}

# errors: The parameter 'error_rate' is the prob of switching the
# second allele.
if (error_rate>0) {
    switch = which(1 == sample.int(2, size=nSNP,
                                   prob=c(error_rate, 1-error_rate),
                                   replace=T))
    snps[switch, 2] = 1 - snps[switch, 2]
}

# convert 1/0 into 0/1
swap = snps[,1] > snps[,2]
snps[swap, ] = snps[swap, 2:1]
```

```
    if(!all(alleles==c(0,1)))
        snps[] = alleles[snps+1]

    # estimate cM positions (sex averaged)
    pos_cM = (phys2cm(pos_Mb, map$male) +
              phys2cm(pos_Mb, map$female))/2

    snpsim = data.frame(MB=pos_Mb, CM=pos_cM, snps, Bfreq=Bfreq,
                        trueIBD=status) # organize output
    names(snpsim)[3:4]=c('al1','al2')
    snpsim
}

removeAA = function(snpsim, allelA)
    subset(snpsim, al1!=allelA | al2!=allelA)
```

B.7 Reelt eksom: «real_exome.R»

```
# version: '0.1'
# author: "Kristina Stormo Gjøtterud"
# email: "kristina.stormo@gjotterud.no"

# Detection of autozygous regions are performed by three methods:
# forward-backward algorithm and two Plink methods, plink0 and plink1.

source("validate.R")
options(stringsAsFactors = F)

# Parameters to be specified
pythonPATH = "C:/Python27/python.exe"
plinkPATH = "C:/windows/plink.exe"
outfile = "exom.txt"
error = 0.001
a= 0.063
f = 0.0625

# real exome-data with masked markers
data <- read.table('exome_pass.csv', header=T, sep="\t")
data <- data[which(is.na(data$X1000g2010nov_ALL)==F),]

genome <- lapply(1:22, function(i) { # 22 autosomes
  chr <- data$CHROM == i
  Bfreq <- data$X1000g2010nov_ALL[chr]
  CM <- data$POS[chr]/1e12
  MB <- CM*1e6 # from cM to Mb

  gt <- strsplit(substr(data$VALUES[chr], 1, 3), "/")
  gt <- lapply(gt, function(i) as.integer(i))
  al1 <- sapply(gt, function(i) i[1])
  al2 <- sapply(gt, function(i) i[2])

  aut <- data.frame(MB=MB, CM=CM, al1=al1, al2=al2, Bfreq=Bfreq)
  list(aut)
})
```

Norges miljø- og
biovitenskapelige
universitet

Postboks 5003
NO-1432 Ås
67 23 00 00
www.nmbu.no