Tartu Ülikool

Matemaatika-Informaatikateaduskond

Informaatika eriala

Egon Elbre

Efektiivsed genotüübi andmete analüüsimeetodid

Bakalaureusetöö

Juhendaja: J. Vilo, PhD

Autor: ..... "..." juuni 2010
Juhendaja: ..... "..." juuni 2010
Õppetooli juhataja: ..... "..." 2010

TARTU 2010

Sisukord

Terminid

Töös kasutatavaid termineid. Enamik seletusi kohandatud Mart Viikmaa "Klassikalise Geneetika Leksikoni"lehelt[1].

Alleel – üks kahest või mitmest alternatiivsest geenivariandist, mis asuvad samas lookuses.

Aluspaar – DNA ahelas kaks teineteist täiendavat nukleotiidi (adeniin - tümiin või guaniin - tsütosiin).

Peaalleel, Alamalleel – (ingl. k. *major allele, minor allele*) on sagedasem (peaalleel) ja vähem sagedasem (alamalleel) alleel dialleelsel geenil.

Dialleelsus – uuritava geeni esinemine kahe alleeli kujul populatsiooni geenifondis.

Diploidsus – liigiomase kromosoomikomplekti kahekordsus indiviidi (raku) kromosoomistikus.

DNA – e. desoksüribonukleiinhape säilitab pärilikkuse infot. DNA koosneb kahest keerdunud polümeerist, mille elementaarlülideks on nukleotiidid.

Fenotüüp – indiviidi (morfoloogiliste, füsioloogiliste, keemiliste, etoloogiliste, arenguliste) tunnuste vaadeldav kogum.

Geen – kromosoomi kindlas lookuses paiknev pärivustegur, mis määrab otse või kaudselt ühe või mitme tunnuse arengu.

Geenifond – e. genofond, populatsiooni kõigi isendite genotüüpides olevate geenide (alleelide) ja muude geneetiliste elementide kogum.

Geenisiire – geneetilise materjali vahetus populatsioonide või populatsiooni allosade vahel isendite migratsiooni ja ristumise teel.

Geenitriiv – geeni alleelide sageduse juhuslikud muutused populatsiooni järjestikustes põlvkondades.

Genotüüp – indiviidi (või raku) kogu geneetiline informatsioon või indiviidi (raku) geneetiliste lookuste alleelne koosseis.

Heterosügootsus, Homosügootsus – heterosügootsus on diploidse indiviidi genotüübi seisund, kus samas lookuses asuvad erinevad alleelid

(nt. Aa). Vastandub homosügootsusele, mille puhul neis lookustes on identsed alleelid (nt. AA või aa).

Looduslik valik – looduses toimuv protsess, mis seisneb kohasemate geneetiliste variantide edukamas üleelamises ja paljunemises ja vähemkohasete variantide eliminatsioonis.

Lookus – mingil viisil iseloomustatav kromosoomi- või DNA-molekuli lõik, milles paikneb kindel geen või mis tahes muu eristatav nukleotiidijärjestus (marker ehk SNP).

Marker – teadaolev geen või nukleotiidijärjestus, mille asukoht on DNA-s teada.

Mutatsioon – indiviidi geneetilise struktuuri muutus, mis pärandub tütarrakkudele või ka järglaspõlvkonna indiviididele ja põhjustab sageli muutunud fenotüübiga indiviidi tekke.

Nukleotiid – orgaaniline ühend, mis kujutab endast DNA ehituskivi. Nukleotiide on nelja tüüpi: A (adeniin), C (tsütosiin), G (guaniin), T (tümiin).

SNP (Single Nucleotide Polymorphism) – üksiku nukleotiidipaari variatsioon võrreldes teise kromosoomi või isendi DNA-ga.

Sissejuhatus

Geneetika on bioloogia teadusharu, kus uuritakse pärilike tunnuste kokkulangevusi ja erinevusi põlvkondade lõikes perekonnas, suguvõsas ja populatsioonis, samuti sarnasuste ja erinevuste põhjusi, tekkemehhanisme ning materiaalseid aluseid.

TÜ Eesti Geenivaramu on Eesti rahva tervise-, sugupuu- ja geeniandmete kogu. Neid andmeid kogutakse selleks, et tulevikus oleks võimalik senisest täpsemalt haigusi diagnoosida ja ravida ning ennustada haigestumise riske[2].

Soodumuse avastamiseks ja ravi tõhustamiseks tuleb leida seoseid genotüüpide andmekogust ning see võib palju aega võtta. Kõige lihtsam assotsiatsiooniuuring Geenivaramu andmetel, mis seisneb juhtumitumite ja kontrollgrupi alleelisageduste võrdlemisel χ^2 statistikuga 346110 markeril, ning see võib andmete halva esituse tõttu aega võtta 18 minutit. Ühekordse uuringu jaoks ei ole kiirus määrav, aga võimalikke seoseid ja kombinatsioone on palju. 100 samalaadse uuringu läbiviimine võtaks aega 30 tundi. Hetkel on Geenivaramu andmetes 2600 isikut aga neil on plaan koguda andmeid 50000 inimese kohta. Sellise koguse juures võtaks need 100 assotsiatsiooniuuringut 600 tundi.

Kiiruse juures mängivad suurt rolli andmed, mille mällu lugemine ja teisendamine sobivasse esitusse on aeganõudev, ja info pärimine.

Ülesande püstitus

Käesoleva töö eesmärgiks on luua Geenivaramu andmete esitamiseks uut tüüpi andmestruktuur ning algoritmid, mis võimaldavad nende andmetega kiiresti tööd teha – nimetagem need ühisterminiga "raamistik".

Töö esimeses osas anname ülevaate vajalikust bioloogilisest taustast.

Töö teises osas vaatame hetkel kasutusel olevaid andmeformaate ning nende puudujääke. Mõtiskleme erinevate andmestruktuuri võimaluste üle ning koostame nende põhjal sobiva struktuuri. Lõpuks vaatame kuidas näeksid sellel andmestruktuuril sagedamini kasutatavad info päringud.

Töö kolmandas osas vaatleme populatsioonigeneetika lihtsamaid uurimismeetodeid ning nende kasulikkust. Vaatame Hardy-Weinbergi tasakaalu arvutamist, mis näitab kui palju mõjuvad kindlale geenile evolutsiooni protsessid, ning assotsiatsiooniuuringut, kus otsitakse seoseid geenide ja haiguse vahel.

Töö neljandas osas anname ülevaate töö käigus loodud tarkvara teekidest ning nende kasutamisest. Lisaks toome lihtsustatud näite assotsiatsiooniuuringu koodist.

Viiendas osas mõõdame loodud programmide kiiruseid ning võrdleme neid Geenivaramus kasutusel oleva programmiga Plink[3].

Peatükk 1

Bioloogiline taust

Geneetika uurib pärilikkust ja varieeruvust organismides. Pärilikkusega on seotud paljud inimeste omadused – seal hulgas ka eelsoodumused haigustele. Kui me suudame leida päriliku haiguse põhjuse, siis on võimalik teha ennetavat tööd haiguse ära hoidmiseks või mõju vähendamiseks.

Pärilikkuse edasikandjaks on DNA. DNA ehk desoksüribonukleiinhape sisaldab geneetilisi instruktsioone erinevate rakkude ja raku osade valmistamiseks. DNA piirkondi, mis neid instruktsioone edasi kannavad, nimetatakse geenideks. Geeni kindlat esinemisvormi nimetatakse alleeliks ning tavaliselt on neid ühel positsioonil kaks.

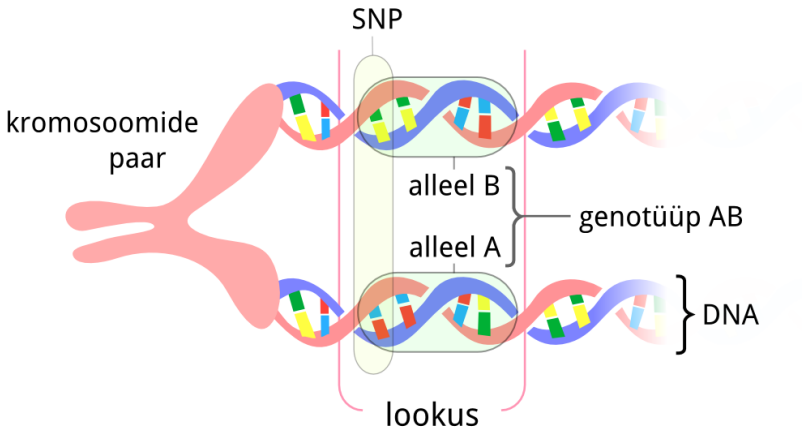
Täpsemalt koosneb DNA kahest pikast polümeerist, mis koosnevad nukleotiididest. Need kaks polümeeri on anti-paralleelsed st. polümeerides kohakuti esinevad nukleotiidid on alati kas adeniin ja tümiin või guaniin ja tsütosiin. Need kaks vastastikust nukleotiidi moodustavad aluspaari. Nende aluspaaride järjestusse on kodeeritud geneetiline informatsioon.

DNA on rakkudes keerdunud kromosoomi ning inimesel on kokku 23 kromosoomipaari. Seda, et kromosoomid on paarides, nimetatakse diploidsuseks.

Kromosoomipaari samas lookuses olevaid alleele nimetatakse kokku genotüübiks. Eristatakse homosügootset ja heterosügootset genotüüpi. Homosügootsus tähendab, et mõlemas kromosoomis on samasugune alleel. Heterosügootsus tähendab, et kromosoomi lookustes on alleeli erinevad variandid.

Üldisemalt kasutatakse genotüübi mõistet ka organismi kogu geneetilise informatsiooni tähistamiseks. Genotüüp määrab organisimi omadusi.

SNP ehk üksik eristuv aluspaar on DNA ühe nukleotiidi erinevus teisest kromosoomist või teise inimese DNA-st. Inimesel on ligikaudu 3 millionit SNP-i ning nendest on tingitud 90% inimeste erinevusest. SNP-d, mis asuvad geneetilist informatsiooni edasikandvas osas, võivad suurema tõenäosusega mõjutada organismi – seega on nad väga head uurimisobjektid.



Joonis 1.1: Kromosoom, DNA, lookus, genotüüp ja alleel

Peatükk 2

Andmed ja tehted

Andmestruktuur on andmete esitus arvuti mälus ja andmeformaat ehk failiformaat on andmestruktuuri esitus failis. Kui meil on andmeformaat ja andmestruktuur erinevad, siis põhjustab andmete sisselugemine ja teisendamine üleliigseid kulutusi, mida saab vältida. Seega on kasulik kirjutada andmestruktuur otse faili. Nii saame andmeid esitada kompaktsemalt ning andmete sisselugemisel ei pea neid teisendama.

Raamistiku idee seisneb optimiseeritud failipõhise andmestruktuuri loomisel ja hiljem sellele operatsioonide implementeerimisel. Üks tähtsamatest operatsioonidest genotüübi andmetel on alleelide ja genotüüpide absoluutse sageduse (edaspidi lihtsalt sagedus) leidmine, seega tuleb sageduse leidmine teha võimalikult kiireks.

Andmete käsitlemisel on oluline, et saaks efektiivselt töödelda palju andmeid korraga. Tähtis on pidada silmas ka protsessori mälu kasutamist. Ulrich Drepper demonstreeris, et mälust eelhaaramine ja L1 ning L2 vahemälu kasutamine võib kiirendada operatsioone 20 korda[4]. Intel Pentium M andmete põhjal võib L1 vahemälust ja põhimälust info küsimise kiiruse erinevus olla isegi 80 korda[4]. Sarnane efekt on ka otsustamisloogikal – Igor Ostrovsky näitas, et kood, milles on hästi ennustatavad `if` laused, võib töötada halvast ennustamisest 6 korda aeglasemalt[6]. Seega üritame vältida suvalisest kohast mälu küsimist ning kasutada hästi ennetatavaid `if` lauseid (ehk lauseid, mis

on peamiselt tõesed).

2.1 Kasutusel olevaid formaate

Andmetes on meil vaja esitada isiku kohta käivat informatsiooni (näiteks isiku ID, perekonna ID ja sugu), markerite kohta käivat informatsiooni (näiteks markeri ID) ning genotüüpe.

Geenivaramus kasutatakse genotüübi andmete esitamiseks PED formaati. Vaatame lisaks TPED ja BED (programmist Plink[3]) failide ülesehitust ning nende häid ja halbu külgi.

PED

PED failis on igas reas ühe isiku info koos tema genotüübi infoga. Info väljad on eraldatud tabulatsioonidega. Faili esimesed kuus veergu on perekonna ID, indiviidi ID, isa ID, ema ID, sugu ja fenotüüp. Fenotüüp on kirjeldatud numbritega `0` - teadmata, `1` - omadus ei esine, `2` - omadus esineb. Ülejäänud veerud on genotüübid, mis on eraldatud tabulatsioonidega. Üks genotüüp on kirjeldatud kahe alleeliga tavaliselt grupist `A`, `C`, `G` või `T`, kuid esitamine nendega ei ole kohustuslik.

PED failil on lisaks MAP fail, mis kirjeldab markereid. Väljadeks on failis: kromosoom, SNP-i või markeri identifikaator, geneetiline kaugus eelmisest markerist (ühik morgan, mis vastab umbes 10,000-le aluspaarile) ning aluspaari asukoht (ühik bp ehk aluspaari).

PED faili headus seisneb selles, et teda on võimalik vaadata lihtsas tekstiredaktoris ning samal kujul andmete agregeerimiseks piisab ühe faili teise otsa lisamisest.

PED failil on aga üks suur probleem - tal puudub indeks. Kindlale isikule positsioneerimine on keerukas, kuna väljades esinevate andmete pikkus ei pruugi olla sama. Teine probleem on kindla markeri analüüsimine. Kuna indiviidi pole võimalik leida, pole ka kindla markeri leidmine lihtne.

```
1 1 0 0 1  0  G G  2 2  C C
1 2 0 0 1  0  A A  0 0  A C
1 3 1 2 1  2  0 0  1 2  A C
2 1 0 0 1  0  A A  2 2  0 0
2 2 0 0 1  2  A A  2 2  0 0
2 3 1 2 1  2  A A  2 2  A A
```

Joonis 2.1: Näidisfaili PED sisu

```
1 snp1 0 1
1 snp2 0 2
1 snp3 0 3
```

Joonis 2.2: Näidisfaili MAP sisu

TPED

TPED on PED faili pööratud kuju. TPED failis on esitatud ridades markerid ning veergudes inimesed. Inimeste kohta käiv info on tõstetud TFAM laiendiga faili.

TPED faili esimesed neli veergu on kromosoom, SNP-i või markeri identifikaator, geneetiline kaugus eelmisest markerist ning aluspaari asukoht. Ülejäänud veergudes on inimeste genotüübid vastavalt TFAM failis kirjeldatud järjekorrale. TFAM faili kuus veergu on perekonna ID, indiviidi ID, isa ID, ema ID, sugu ja fenotüüp.

```
1 snp1 0 1  G G  A A  0 0  A A  A A  A A
1 snp2 0 2  2 2  0 0  1 2  2 2  2 2  2 2
1 snp3 0 3  C C  A C  A C  0 0  0 0  A A
```

Joonis 2.3: Näidisfaili TPED sisu

Sarnaselt PED failile on esimesel neljal väljal esitatud info erinevate pik-

```
1 1 0 0 1  0
1 2 0 0 1  0
1 3 1 2 1  2
2 1 0 0 1  0
2 2 0 0 1  2
2 3 1 2 1  2
```

Joonis 2.4: Näidisfaili TFAM sisu

kusetega seega on ka TPED-s raske markeritele positsioneerida. Markeri põhine analüüs on siiski lihtsam, sest kõik vajalik informatsioon on ühel real.

BED

BED ehk binaarne PED faili esitus pakib andmed kokku ning kirjeldab alleelid ja isikute kohta käiva info eraldi failis (vastavalt BIM ja FAM).

BIM faili esimesed 4 veergu on sama, mis MAP failil. BIM failis on lisaks veel peaalleel ja alamalleel. Fail FAM on PED faili kuus esimest veergu ehk perekonna ID, indiviidi ID, isa ID, ema ID, sugu ja fenotüüp.

```
1 snp1 0 1 G A
1 snp2 0 2 1 2
1 snp3 0 3 A C
```

Joonis 2.5: Näidisfaili BIM sisu

Kuna enamikel geenidel on ainult kaks alleeli siis on BED failis genotüübid esitatud kahe alleeli kaudu. Üks neist tähistab peaalleeli ja teine alamalleeli.

BED faili esimesed kaks baiti identifitseerivad failitüübi. Kolmas bait näitab kas veergudes on SNP-d (`1`) või inimesed (`0`). Bitikodeering on genotüüpidel vastavalt: `00` - homosügoot (peaalleel), `11` - homosügoot (alamalleel), `01` - heterosügoot, `10` - teadmata. Ühte baiti salvestatakse neli genotüüpi.

BED võtab sellise pakkimise tõttu märgatavalt vähem ruumi võrreldes

```
1 1 0 0 1  0
1 2 0 0 1  0
1 3 1 2 1  2
2 1 0 0 1  0
2 2 0 0 1  2
2 3 1 2 1  2
```

Joonis 2.6: Näidisfaili FAM sisu

```
|-faili tüüp-------------| |--genotüüpide andmed----------|
01101100 00011011 00000001 11011100 00001111 11100111 ....
```

Joonis 2.7: Näidisfaili BED sisu (binaarne)

PED ja TPED failidega. Täpsemalt faili ülesehitusest võib vaadata Plink'i juhendist[7].

2.2 Andmestruktuuri ülesehitus

Laiemalt vaadates on genotüübi andmete esitamiseks kaks võimalust: ridades inimesed ja veergudes markerid või ridades markerid ja veergudes inimesed. Kuna tegeleme peamiselt inimeste gruppidega ja analüüsime markereid eraldi siis tasub asetada markeri informatsioon mälus võimalikult kokku ehk esitame ridades markerid.

Genotüüpide loendamisel peab iga genotüübi loendamiseks olema muutuja. Esitades kõik genotüüpide variandid ühel real, peaks igal sammul loendajat vahetama. Selle vältimiseks esitame iga genotüübi omal real ning kirjeldame vastavas veerus ainult seda, kas selline genotüüp on vastaval inimesel olemas või mitte.

Selline andmete esitus vajab ruumi $m * (a + 1) * i$, kus m on markerite arv, a on genotüüpide arv markeril ning i indiviidide arv. Genotüüpide arvule lisatud 1 on teadmata genotüüpide jaoks. Tavaliselt on inimeste genotüübid

Indeks		
Marker	Genotüüp	Genotüübi esinemised
Marker 1	AA	10101010010000100100010...
Marker 1	AB	01010100001101011000101 1...
Marker 1	BB	00000001000010000011000 0...
Marker 2	AA	01010010010101101101000 1...
Marker 2	AB	10101101101010010010111 0...
...	...	...

Joonis 2.8: Genotüüpide esinemised inimestel

märgitud nelja alleeliga (`A`, `C`, `G` ja `T`), mis annab 9 erinevat kombinatsiooni (kui lugeda samadest alleelidest koosnevad heterosügoodid võrdseteks). Kuna on teada, et enamus SNP-del on tegemist ainult kahe alleeliga, siis on võimalik andmeid pakkida, jättes alles ainult vajalikud genotüübid.

Näiteks tähistame need kaks alleeli vastavalt A ja B-ga. Selleks, et pärast teada saada, milliste alleelidega tegelikult tegemist on, võime hoida neid markerite indeksis. Kahel alleelil on võimalik kolm kombinatsiooni (`AA`, `AB`, `BB`), mis kokkuvõttes võtab 2.5 korda vähem ruumi.

2.3 Operatsioonid

Genotüüpide esinemise reast vastava genotüübi koguarvu leidmiseks tuleb kõik ühed ära loendada. Arvestades lisaks seda, et võimalikult palju andmeid üheaegselt töödelda, on mõistlik bitte ühes arvus koos hoida. Kasutame antud juhul s-bitiseid arve ehk esitame ühes arvus s inimest. Tähistame neid grupeeritud inimesi tähega a_i, milles on esitatud inimesed $i * s$ kuni $(i + 1) * s - 1$. Sellest arvust bittide loendamiseks saame kasutada operatsiooni *bitcount* ehk bitisageduse leidmine, mis osadel arhitektuuridel on juba sisseehitatud. Alternatiivseid bitisageduse leidmise meetodeid võib vaadata lisast C.

Indeks		
Marker	Genotüüp	Genotüübi esinemised
Marker 1	AA	$a_1, a_2, a_3, a_4, \ldots, a_n$
Marker 1	AB	$b_1, b_2, b_3, b_4, \ldots, b_n$
...	...	...

Joonis 2.9: Grupeeritud genotüüpide esinemised

Vastava genotüübi sageduse saab arvutada

$$f(\texttt{Marker 1, AA}) = bc(a_1) + bc(a_2) + bc(a_3) + bc(a_4) + \ldots$$
$$f(\texttt{Marker 1, AB}) = bc(b_1) + bc(b_2) + bc(b_3) + bc(b_4) + \ldots$$

kus f on sagedus ja bc on positiivsete bittide sagedus arvus.

Selline loendamise algoritm on väga lihtne kui teeme massiivid $[a_1, a_2, \ldots, a_n]$ ja $[b_1, b_2, \ldots, b_n]$. Loomulikult võib kõiki arve hoida ka ühes massiivis.

Algoritm 2.1: Genotüübi sageduse arvutamine

```
// Sisend: andmed[]
i = 0;
kokku = 0;
while ( i < andmed.pikkus ){
    kokku = kokku + bitcount ( andmed[i] );
    i++;
};
return kokku;
```

2.3.1 Sageduse leidmine valimiga

Kui infot on vaja ainult kindlate isikute kohta, saab tekitada sobiva bitifiltri. Ülesehituselt on bitifilter samasugune nagu markeri massiiv, ent iga bitt tähistab, kas isik kuulub valimisse või mitte. Vastav algoritm selle jaoks on:

Algoritm 2.2: Genotüübi sageduse arvutamine valimiga

```
// Sisend: andmed[], valim[]
i = 0;
kokku = 0;
while ( i < andmed.pikkus ){
    kokku = kokku + bitcount ( andmed[i] && valim[i] );
    i++;
};
return kokku;
```

2.3.2 Sageduse leidmine pakitud valimiga

Loomulikult tasub väikeste valimite korral esitada filter pakitud kujul. Selleks tasuks jätta vahele filtri osad, mis on 0-d. Filtri õigeks positsioneerimiseks tuleb lisada aadress. Filtri arvude jada, mis on täiendatud aadressi ja arvude kogusega, nimetame filtrigrupiks.

Näiteks, kui meil on pakkimata kujul filter:

$$[a_1, a_2, 0, 0, 0, 0, a_7, a_8].$$

Pakitud filtri tekitamiseks lisame algusaadressi ja järjest olevate filtrite arvu ning filtrid:

$$[1, 2, a_1, a_2 \dots].$$

Otsime järgmise filtrigrupi ning kordame protsessi:

$$[1, 2, a_1, a_2, 7, 2, a_7, a_8].$$

Sellise esitusega jätame tegemata valimi iga nulli juures kaks mälust pärimist, ühe ”JA” tehte ja ”bitcount” operatsiooni.

Kohandatud algoritm näeb välja:

Algoritm 2.3: Genotüübi sageduse arvutamine pakitud valimiga

```
// Sisend: andmed[], valim[]
// Valimi kuju:
// { aadress i, valimite arv n,
//       valim i+1, valim i+2 ... valim i+n-1 }*
aadress = 0; // algaadress andmetes
vi = 0; // indeks valimis
kokku = 0;
while ( vi < valim.pikkus ){
    aadress = valim[vi]; vi++;
    aadress_viimane = aadress + valim[vi]; vi++;
    for (; aadress < address_viimane ; aadress++){
        kokku = kokku +
                bitcount ( andmed[aadress] && valim[vi] );
        vi++;
    };
};
return kokku;
```

Kuna korduse lõpetamine ja alustamine võtab ka aega, siis tasub leida hea nullide arv, mida tasub vahelt ära jätta. Hetkel rakendame siiski kõige lihtsamat varianti ehk jätame kõik nullid vahelt ära.

2.3.3 Sageduse leidmine mitme valimiga

Selleks, et mitut valimit korraga loendada, ühendame mitu filtrit omavahel, ning igale filtrile lisame identifikaatori. Võime järjestada filtri sektsioonid aadressi järgi, et teha väiksemaid hüppeid mälus.

Näiteks, kui meil on pakitud filtrid:

$$[1, 2, a_1, a_2, 7, 2, a_7, a_8] \quad ja \quad [4, 3, a_4, a_5, a_6]$$

Selleks, et neid ühendada lisame igale filtrigrupile identifikaatorid (tähistame

neid A ja B – andmetes hoiame siiski arve):

$$[1, A, 2, a_1, a_2, \dots] \qquad [4, B, 3, a_4, a_5, a_6]$$
$$[1, A, 2, a_1, a_2, 7, A, 2, a_7, a_8] \qquad [4, B, 3, a_4, a_5, a_6]$$

Hakkame valimitest võtma filtrigruppe. Võtame alati väikseima aadressiga filtrigrupi.

Valim A	Valim B	Tulemus
$[1, A, 2, a_1, a_2, 7, A, 2, a_7, a_8]$	$[4, B, 3, a_4, a_5, a_6]$	[]
$[7, A, 2, a_7, a_8]$	$[4, B, 3, a_4, a_5, a_6]$	$[1, A, 2, a_1, a_2]$
$[7, A, 2, a_7, a_8]$	[]	$[1, A, 2, a_1, a_2, 4, B, 3, a_4, a_5, a_6]$
[]	[]	$[1, A, 2, a_1, a_2, 4, B, 3, a_4, a_5, a_6, 7, A, 2, a_7, a_8]$

Ja tulemuseks on valim:

$$[1, A, 2, a_1, a_2, 4, B, 3, a_4, a_5, a_6, 7, A, 2, a_7, a_8]$$

Algoritm sageduse leidmiseks näeb välja:

Algoritm 2.4: Genotüübi sageduse arvutamine mitme valimiga

```
// Sisend: andmed[], valim[]
// { aadress i, valimi id, maskide arv n,
//       mask i+1, mask i+2 ... mask i+n-1}*
aadress = 0; // aadress andmetes
vi = 0; // indeks valimis
kokku[] = 0;
while ( vi < valim.pikkus ){
    aadress = valim[vi]; vi++;
    id = valim[vi]; vi++;
    aadress_viimane = aadress + valim[vi]; vi++;
    id_kokku = kokku[id]
    for (; aadress < address_viimane ; aadress++){
        id_kokku = id_kokku +
                   bitcount ( andmed[aadress] && valim[vi] );
        vi++;
    };
    kokku[id] = id_kokku;
};
return kokku;
```

Peatükk 3

Populatsioonigeneetika

Populatsioonigeneetika on geneetikaharu, mis uurib mingi populatsiooni geneetilisi seaduspärasusi evolutsiooniprotsesside (looduslik valik, geenitriiv, mutatsioon ja geenisiire) mõju alleelide sagedustele. Populatsioonigeneetika peamisteks uurimisobjektideks on populatsiooni geenid ja nende alleelide sagedused. Populatsioonigeneetikas leiavad laialdast rakendust mitmed statistilised uurimismeetodid ja valemid[8].

3.1 Hardy-Weinbergi taskaal

Hardy-Weinbergi tasakaal väidab, et alleelide ja genotüüpide sagedused püsivad stabiilsena läbi aja – ehk on tasakaalus –, välja arvatud juhul, kui esinevad segavad protsessid. Hardy-Weinbergi tasakaal on väga harva esinev looduses, sest väljaspool laboratooriume esineb kindlasti üks või mitu evolutsiooniprotsessi. See teoreetiline tasakaal on seis, mille suhtes on võimalik geneetilisi muutusi mõõta[8].

Alleelide sagedused säilivad üle põlvkondade juhul kui järgnevad tingimused on täidetud:

1. paljunemine on juhuslik ehk eelistus partnerite suhtes puudub;
2. mutatsioonid puuduvad;

3. migratsioon puudub;
4. populatsioon on lõpmatult suur;
5. surve mingi omaduse omandamiseks puudub;

Kõrvalekalle Hardy-Weinbergi tasakaalust näitab evolutsiooniprotsesside olemasolu vastaval geenil. Kõrvalekalded paljudel geenidel võib näidata ka viga genotüpiseerimisel.

Kõige lihtsamal juhul dialleelses lookuses saab tasakaalu väljendada:

$$p^2 + 2pq + q^2 = 1$$

kus p ja q on vastavate alleelide sagedused.

Kui me teame alleelide sagedusi siis me võime ennustada genotüüpide sagedusi. Näiteks teades alleelide A ja B sagedusi:

$$p = f(A)$$
$$q = f(B)$$

saame nende põhjal ennustada genotüüpide AA, AB (AB ja BA loeme ekvivalentseteks) ja BB sagedused:

$$f(AA) = p^2$$
$$f(AB) = 2pq$$
$$f(BB) = q^2$$

Kui meil on ühes lookuses üle kahe alleeli, siis saame üldistada valemit

$$(p_1 + p_2 + \cdots + p_i)^2 = 1$$

mis ennustab kõikide homosügootsete genotüüpide jaoks sagedusi

$$f(A_iA_i) = p_i^2$$

ning heterosügootsete jaoks

$$f(A_iA_j) = 2p_ip_j$$

Kui me teame tegelikke ja oodatavaid genotüüpide sagedusi siis me saame arvutada kõrvalekalde sellest teoreetilisest tasakaalu punktist. Tavaliselt

kasutatakse selleks Pearsoni χ^2 testi.

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

kus O on tegelikud väärtused ning E on ennustatud väärtus. Kui genotüüpide arv on väike, siis võib olla vajalik kasutada selle asemel Fisheri täpset testi[9].

Näiteks oletame, et me teame genotüüpide sagedusi $MM = 165$, $MN = 562$, $NN = 339$ ja teame kogu genotüüpide arvu $K = 1066$. Nendest saame arvutada alleelide M ja N suhtelised sagedused:

$$f(M) = \hat{p} = \frac{2 * MM + MN}{2K} = \frac{2 * 165 + 562}{2 * 1066} = 0.4184$$

$$f(N) = \hat{q} = 1 - \hat{p} = 0.5816$$

Genotüüp	Sagedus (S)	Oodatav (O)	$S - O$
MM	165	$K * \hat{p}^2 = 1066 * (0.4184)^2 = 186.61$	-21.6
MN	562	$K * 2\hat{p}\hat{q} = 1066 * 2(0.4184)(0.5816) = 518.80$	43.2
NN	339	$K * \hat{q}^2 = 1066 * (0.5816)^2 = 360.58$	-21.6

Arvutame χ^2 statistiku:

$$\chi^2 = \frac{(-21.6)^2}{181.61} + \frac{43.2^2}{518.80} + \frac{(-21.6)^2}{360.58} = 7.46$$

Lisaks läheb meil vaja p-väärtuse arvutamiseks vabadusastet. Vabadusaste näitab mitu muutujat võivad muutuda üksteisest sõltumatult.

Vabadusastme arvutamiseks on valem:

$$\text{vabadusaste} = \text{genotüüpe} - \text{alleele} \qquad [9]$$

Antud juhul $v = 3 - 2 = 1$ – kolm genotüüpi ja kaks alleeli.

P-väärtus $\chi^2 = 7.46$ on 0.01 ja 0.001 vahel. Seega võime öelda, et kõrvalekalle tasakaalust on oluline[9].

3.2 Assotsiatsiooniuuring

Assotsiatsiooniuuring seisneb lookuse alleelide või genotüüpide sageduste erinevuste leidmisel kahe eri grupi vahel (enamasti haiged juhtumigrupis ning terved kontrollgrupis)[10].

Kõige lihtsam assotsiatsiooniuuring on võrrelda kontrollgrupi ja haigete (või mingi omadusega) grupi alleelide või genotüüpide sagedusi. Kontrollgrupis on isikud, kelle kohta on teada, et neil puudub vastav omadus või haigus. Kontrollgrupi võib koostada ka juhuslikest isikutest populatsioonis.

Genotüüpide sagedused võivad erineda asukohtade ja rahvuste vahel. Kui kontrollpopulatsioon ei ole hästi valitud võib leida palju vale-positiivseid seoseid.

Gruppide sageduste võrdlemiseks saab kasutada χ^2 testi.

Vaatame näidet, kus meil on teada alleelide sagedused:

Grupp	Alleel A	Alleel B	Kokku
Juhtum	332	164	496
Kontroll	230	262	492
Kokku	562	426	988

Oletame, et meil on kokku N inimest ja teame alleeli suhtelist sagedust $f(A)$, siis oodatav inimeste arv, kellel see alleel on, on $O(A) = f(A) * N$. Alleeli suhtelise sageduse saame leida: $f(A) = A/N$, kus A on alleeli sagedus.

Grupp	Alleel	Sagedus (S)	Oodatav (O)	$(S-O)^2/O$
Juhtum	A	332	282.137	8.812
Juhtum	B	164	213.863	11.626
Kontroll	A	230	279.862	8.884
Kontroll	B	262	212.138	11.720
			Kokku	41.041

Vabadusastme saame arvutada: vabadusaste = (ridade arv - 1)∗(veergude arv - 1). Arvutame χ^2 statistikule vastava p-väärtuse ning tulemus väiksem kui 0.0000001 [11].

Peatükk 4

Implementatsioon

Teegid ja programmid on implementeeritud D-keeles. Programmeerimise keele valikul oli kõige tähtsamad kiirus ja koodi arusaadavus. Kiiruse poolelt on lisaks veel võrreldavad C ja C++, sest D-keele jaoks on olemas liides GNU ja LLVM kompilaatoritele. C ja C++ keele probleemiks on arusaadavus. Teekide omavaheline kasutamine on implementeeritud läbi makrode ning seetõttu on veaallikaks kompileerimisel. Lisaks päiste ja koodi eraldamine raskendab koodi haldamist. D-keel on piisavalt sarnane C++'ga, seega on vajadusel kood võimalik lihtsalt ümber kirjutada.

D-keele valikul oli tagajärg, millega ei osanud arvestada – nimelt D-keele 1.0 linux-i versioonil puudub tugi üle 2GB suuruste failidele. Seetõttu võivad programmid mitte töötada sisend või väljund andmetega, mis on suuremad kui 2GB. D2.0-l on see probleem lahendatud aga tema spetsifikatsioon ei ole veel lõplik ning tal puudub täielik tugi vabavara kompilaatoritelt (GDC ja LDC).

Failiformaatide ja programmide kirjeldused leiab lisadest A ja B.

4.1 Teegid

Raamistik koosneb kahest põhiteegist ning kahest abiteegist.

Koodis kasutame lühendeid: `mi` - markeri indeks, `gi` genotüübi indeks,

`pi` inimese indeks ja `grpi` grupi indeks. Need näitavad soovitud elemendi indeksit.

gmap.gmap

Peamine klass on `GMap`, millega saab sisse lugeda GMAP, MIDX, GIDX ja PIDX andmed. Samuti on tal kirjeldatud meetodid genotüüpide sageduste leidmiseks.

Algoritm 4.1: GMap objekti meetodite ja parameetrite näited

```
// andmete laadimine
GMap gmap = new GMap();
gmap.load("andmed");

// laetud markerite kogus ja nimed
ulong  markerite_arv = gmap.markers.count;
string markeri_nimi  = gmap.markers.names[ mi ];

// laetud genotyypide arv ja nimed
ulong  genotyypide_arv = gmap.genotypes.count;
string genotyybi_nimi  = gmap.genotypes.names[ gi ];

// kui gmap.markers.mm_mode, siis peaalleel ja alamalleeli
//   saab k2tte vastavalt:
string pea_alleel  = gmap.markers.major[ mi ];
string alam_alleel = gmap.markers.minor[ mi ];

// laetud inimeste arv ning identifikaatorid
ulong  inimeste_arv = gmap.people.count;
string inimese_id = gmap.people.individual_id[ pi ];

// k6ikidel markeritel sageduste leidmine
ulong[][] andmed = gmap.countAllMarkers();
ulong kogus = andmed[ mi ][ gi ];
```

gmap.mask

See teek hoiab endas valimeid hoidvaid klasse `GMapMaskSingle`, mis hoiab ühte valimit, ja `GMapMaskMultiple`, mis võib hoida rohkem valimeid. Valimitega saab piirata loendamist kindlale isikute grupile.

GMapMaskSingle ja GMapMaskMultiple kasutamise näide

```
// valimite tekitamine
GMapMaskSingle maskA =
    new GMapMaskSingle( [1,2,5], "GruppA" );
GMapMaskSingle maskB =
    new GMapMaskSingle( [7,8,11], "GruppB" );
// mitmese valimi tekitamine
GMapMaskMultiple maskC = new GMapMaskMultiple(maskA);
// valimite yhendamine yheks mitmeseks valimiks
maskC = maskC.merge(maskB);

// grupi nime leidmine
string alam_grupi_nimi = maskC.names[ grpi ];

// genotyypide sageduste leidmine valimiga:
ulong[][] andmed  = gmap.countAllMarkers(maskA);

// Mitme grupi genotyypide sageduste leidmine korraga
ulong[][][][] andmed3 = gmap.countAllMarkers(maskC);
```

gmap.statistics, gmap.utils

Need teegid hoiavad abifunktsioone, mis teisendavad, loevad sisse andmeid ning arvutavad. gmap.statistics moodulis on kirjeldatud p-väärtuse ja χ^2 statistiku arvutamiseks abifunktsioone.

Algoritm 4.2: gmap.statistics ja gmap.utils kasulikke meetodeid

```
// gmap.statistics
// abiobjekt sageduste teisendamiseks
GMapStatistics stat = new GMapStatistics( gmap );

// loendame esimese markeri genotyypide sagedused
ulong[] gt_kogus = gmap.countFromPosition( 0 );

// teisendame genotyypide sagedused alleelide sagedusteks
ulong[] alleelide_kogus = stat.alleleCount( gt_kogus );

// teisendame genotyypide sagedused alleelide suhtelisteks
//   sagedusteks
real[]  alleelide_sagedus = stat.allelePercentage( gt_kogus );

// gmap.utils
// valimite lugemine PSET failist
GMapMaskMultiple maskA = maskFromFile("filter.pset");
```

4.2 Assotsiatsiooniuuring

See näitab kõige lihtsamat viisi kuidas teeke kasutades on võimalik teha assotsiatsiooniuuringu programm.

Assotsiatsiooniuuringu koodinäide

```
module gmapassocsimp;

import std.stdio;
import gmap.gmap, gmap.mask,
       gmap.statistics, gmap.utils;

int main(string[] args){
  // loeme sisse andmed
  GMap map = new GMap();
  map.load("andmed");
  // teeme genotyybi sageduste teisendamiseks objekti
  GMapStatistics stat = new GMapStatistics(map);

  // loeme sisse grupid
  GMapMaskMultiple mask = maskFromFile("grupid.pset", map);

  // leiame genotyypide sagedused k6ikidele gruppidele
  ulong[][][][] sagedused = map.countAllMarkers(mask);

  // v2ljastame tulemuse p2ise
  writefln("MARKER\tCHISQ\tP\t%s\t%s",
               stat.alleles, stat.alleles);
  for (uint i = 0; i < map.markers.count; i++){
    // leiame kontrollgrupi alleelide
    //    absoluutse ja suhtelise sageduse
    ulong[] kontroll = stat.alleleCount(sagedused[i][0]);
    real[]  kontroll_f = countToFrequency(kontroll);

    // leiame juhtumgrupi alleelide abs. ja suhtelise sageduse
    ulong[] juhtum = stat.alleleCount(sagedused[i][1]);
    real[]  juhtum_f = countToFrequency(juhtum);

    // leiame hii ruudu juhtumi ja kontrolli vahel
    //   ning leiame vabadusastme
```

```
    int  vabadusAste;
    real chi = caseControlChiSquare(juhtum, kontroll,
                                    vabadusAste);

    // leiame p
    real p = pFromChiSquare(chi, vabadusAste);

    // v2ljastame tulemuse
    writefln("%s\t%.4f\t%.4f\t%.4s\t%.4s",
                map.markers.names[i], chi, p,
                kontroll_f, juhtum_f);
  }
  return 1;
}
```

Peatükk 5

Võrdlus

Mõõtsime meie kirjutatud programmide ja Plink[3] kiirusi erinevate failiformaate kasutades. Plink'ga meie programmid täisfunktsionaalsuses veel võistelda ei saa, aga suudavad võistelda kiiruses. Järelduste tegemisel arvestasime, et võimaluste arv võib mõjuda halvasti programmi kiirusele. Võrdluseks valisime programmi Plink, sest see on hetkel kasutusel Eesti Geenivaramu projektis.

Testisime kiirusi erinevate failiformaatidega. Meie programmid kasutasid failiformaate GMAP ja pakitud GMAP (ehk formaat kus on üleliigsed genotüübid eemaldatud). Plink'i testisime PED, TPED ja BED failiformaatidega. Edaspidi tabelites nimetame ainult failiformaate. (Pakitud GMAP-i tähistame GMAP*-ga.)

Teste jooksutasime Geenivaramu andmetel, kus on 346110 markerit ja 2620 isikut. Kokku on seal mahult andmeid 3629MB.

Teisendamise teste jooksutasime kaks korda ning valisime iga testi kiireima aja sekundi täpsusega. TPED ja PED failide jaoks jooksutasime analüüse samuti kaks korda ning valisime kiireima aja. GMAP, GMAP* ja BED failidega tegime analüüse kümme korda ning võtsime keskmise aja.

5.1 Andmete teisendamine

Andmete teisendamine, PED failist sobivasse formaati, võib kaua aega võtta, kuid teisendamine on ühekordne – seega ei pea see olema väga kiire.

Tulemus	**Aeg**	**Suurus**
PED		3629MB
GMAP	09m20s	1249MB
GMAP*	10m17s	448MB
TPED	27m32s	3637MB
BED	17m27s	228MB

Tabel 5.1: PED faili teisendamine

Näeme, et BED failiformaat on kõige väiksem, mis ei ole üllatav, sest iga genotüübi esitamiseks kasutatakse kahte bitti.

5.2 Andmete analüüsimine

Genotüübi sageduste leidmine on üks tähtsaim operatsioon ning see mõjutab tugevalt ülejäänud algoritmide kiirusi. Hardy-Weinberg testib lisaks analüüsimise kiirusi. Assotsiatsiooniuuring testib mitme valimiga sageduse leidmist.

Fail	Sageduste leidmine	Hardy-Weinberg	Assotsiatsiooniuuring
gmap	00m32s	00m29s	00m56s
gmap*	00m25s	00m26s	00m37s
ped	17m18s	17m33s	21m37s
tped	10m52s	11m42s	11m45s
bed	01m14s	01m42s	01m44s

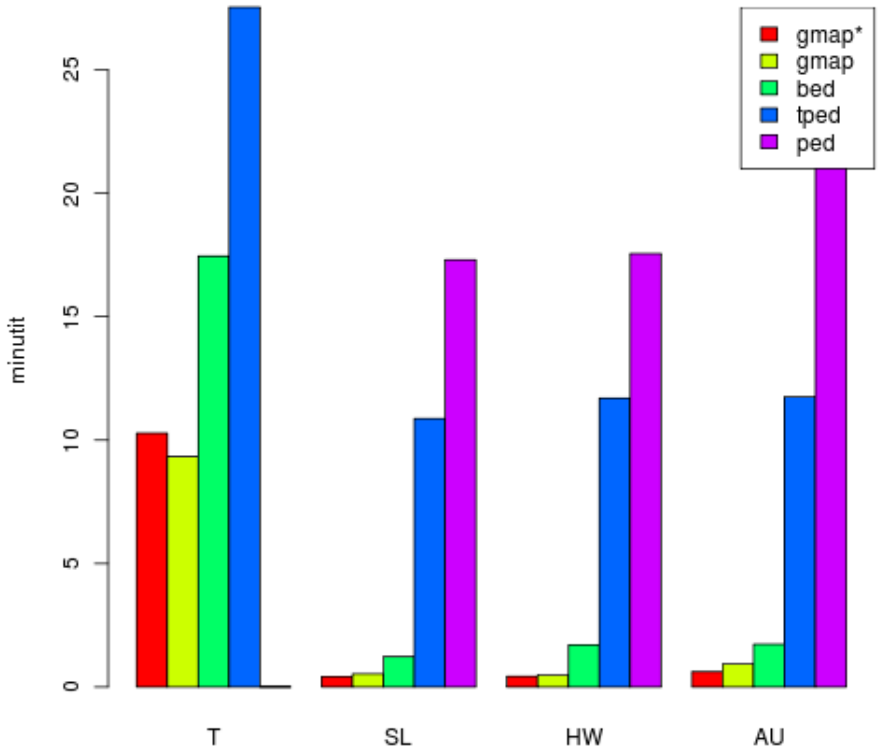
Tabel 5.2: PED ja TPED failide puhul on tegemist kahe testi minimaalse ajaga. GMAP, GMAP* ja BED failidel on märgitud 10 testi keskmine aeg.

Nagu näha teeb töö kõige kiiremini ära GMAP. Sellele üsna lähedal on tavalisel kujul GMAP. Nendevaheline suhteline kiiruse erinevus (vastavalt

88%, 80%, 63%) on üllatav, sest pakitud ja pakkimata failide suuruse erinevus on kolmekordne.

BED failiformaadi kasutamine on keskmiselt 2-3 korda aeglasem GMAP*'st. Kahjuks ei ole erinevus nii suur, et saaks kindlalt öelda, et see on andmeesituse tõttu.

Kõige aeglasemaks osutusid PED ja TPED failiformaadid – seega on õige meie algne arvamus nende kasutamise kiiruste kohta.



Joonis 5.1: Programmide tööaegade võrdlus. `T` - Teisendamine, `SL` - Sageduse leidmine, `HW` - Hardy-Weinberg ja `AU` - Assotsiatsiooniuuring.

Peatükk 6

Kokkuvõte

Käesolev bakalaureusetöö uurib erinevaid andmeformaate ja -struktuure genotüüpide andmete esitamiseks. Töö tulemusena valmistasime GMap teegid ja programmid, mis võimaldavad efektiivsemalt töötada genotüüpide andmetega. Võrdlesime nende kiirusi Geenivaramus kasutatava programmiga Plink. Testimine näitas, et binaarformaadi kasutamine parandab märgatavalt programmi kiirust. Kõikidel testidel olid meie programmid kiiremad Plink'st. Kasutades Plink'i binaarformaati oli programmi tööajad lähedased meie programmide aegadele, seega ei saanud kindlalt väita, et kiiruse erinevus (võrreldes selle formaadiga) on andmeesituse ja algoritmide tõttu.

Tehtud töö edasiarenguna tuleks arendada populatsioonigeneetika raskemaid ja keerukamaid algoritme. Teiseks tasub raamistikule lisada lõimede kasutamine. Kolmanda edasiarendusena pakume välja liidese interaktiivsesse skriptimiskeelde (näiteks R, LUA või Python), et uusi algoritme oleks võimalik lihtsamalt kirjutada ja testida.

Effective algorithms on genotype dataset

Egon Elbre

Bachelor thesis [6 ECTS]

Abstract

Genotype datasets are usually very large and they are expected to grow rapidly. Data size and format will start affecting speed of programs, therefore it is neccesary to have a fast framework and data representation and structure to get the best performance.

This paper discusses how simplifying the datasets and algorithms can have an improvement on program execution speed. We developed a data format (GMAP), framework (GMap) and programs for analysing genotype datasets.

We compared the speed of programs with Plink using different file formats. Testing showed that we can get a large improvement in performance using binary file format (such as our GMAP and Plink's BED) instead of text-based format (such as PED and TPED). Also, we show that our programs work faster than Plink, yet we could not say definitively if this is due to our data format or our algorithm implementation.

Kirjandus

[1] M. Viikmaa, "Klassikalise geneetika leksikon." http://lepo.it.da.ut.ee/~martv/genolex.html, 1998. [viimati kontrollitud 2. juuni 2010].

[2] "Eesti Geenivaramu." http://www.geenivaramu.ee, 2010.

[3] S. Purcell, "PLINK v1.07." http://pngu.mgh.harvard.edu/purcell/plink/, 2010.

[4] U. Drepper, "What every programmer should know about memory," 2007.

[5] I.Õstrovsky, "Gallery of Processor Cache Effects." http://igoro.com/archive/gallery-of-processor-cache-effects/, 2010. [viimati kontrollitud 31. mai 2010].

[6] I.Õstrovsky, "Fast and slow if-statements: branch prediction in modern processors." http://igoro.com/archive/fast-and-slow-if-statements-branch-prediction-in-modern-processors/, 2010. [viimati kontrollitud 31. mai 2010].

[7] Purcell, S. and Neale, B. and Todd-Brown, K. and Thomas, L. and Ferreira, M.A.R. and Bender, D. and Maller, J. and Sklar, P. and de Bakker, P.I.W. and Daly, M.J. and Sham, P.C., "PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses.," *Am J Hum Genet*, vol. 81, no. 3, pp. 559–75, 2007.

[8] Wikipedia, "Population Genetics." http://en.wikipedia.org/wiki/Population_genetics, 2010. [viimati kontrollitud 1. juuni 2010].

[9] M. B. Hamilton, *Population Genetics*, pp. 13–28. Wiley-Blackwell, 2009.

[10] Wikipedia, "Genetic Association." http://en.wikipedia.org/wiki/Genetic_association, 2010. [viimati kontrollitud 31. mai 2010].

[11] C. for Integrated Genomic Medical Research, "The χ^2 test." http://slack.ser.man.ac.uk/theory/association_chi.html, 2004. [viimati kontrollitud 2. juuni 2010].

[12] G. S. Manku, "Puzzle: Fast bit counting." http://gurmeetsingh.wordpress.com/2008/08/05/fast-bit-counting-routines/, 2008.

Lisa A

Failid

Teegid ja programmid kasutavad sisendiks ja väljundiks peamiselt faile. Põhifailid on hädavajalikud GMap teegi ja programmide tööks, sest need sisaldavad genotüübi andmeid. Väljund- ja lisafailid on kas programmide poolt tekitatavad failid või argumentides kasutatavad failid.

A.1 Põhifailid

`<nimi>.gmap`	genotüübid
`<nimi>.gidx`	genotüüpide indeks
`<nimi>.midx`	markerite indeks
`<nimi>.pidx`	inimeste indeks

GMAP

GMAP on binaarne fail, mis sisaldab iga isiku jaoks markeri ja genotüüpide informatsiooni. Fail on 64bitiste arvude massiiv. Asukohad failis on kirjeldatud MIDX-ga ja GIDX-ga. Täpsemalt ülesehitusest on peatükis 2.2.

GIDX

GIDX esimesel real on erinevate genotüüpide arv, ülejäänud ridadel on genotüüp ning tema järjekord markeri alguse suhtes.

```
4
0 0    0
A A    1
A B    2
B A    2
B B    3
```

Joonis A.1: Faili näide.gidx sisu

MIDX

MIDX fail on on laiendatud MAP fail – sinna on lisatud igale markerile esimese genotüübi massiivi asukoht GMAP failis (baitides). Lisaks võib olla veel kirjeldatud peaalleel ning alamalleel.

```
3
1   Mark0   0   2   0   A   G
1   Mark1   0   4   88  C   T
1   Mark2   0   6   176 A   T
```

Joonis A.2: Näidisfaili MIDX sisu

PIDX

PIDX faili esimesel real on isikute arv. Ülejäänud read on PED faili kuus esimest rida, millele on ette lisatud isikute järjekorra number.

```
2
1 1 0 0 1  0
1 2 0 0 1  0
```

Joonis A.3: Näidisfaili PIDX sisu

A.2 Väljund- ja lisafailid

`<nimi>.ped`	genotüüpide lähtefail
`<nimi>.map`	markerite lähtefail
`<nimi>.gt`	genotüüpide seosed
`<mask>.pset`	isikute grupid
`<nimi>.frq*`	sageduse failid
`<nimi>.hwb`	Hardy-Weinbergi test
`<nimi>.asca`	assotsiatsiooniuuring

PED ja MAP

PED ja MAP faili ülesehitus on kirjeldatud peatükis 2.1. PED, MAP ja GT failid konverteeritakse GMAP, MIDX, PIDX ja GIDX failideks.

GT

GT fail kirjeldab, millised genotüübid on PED failis, ning millised genotüübid on samaväärsed. Esimene rida GT failis peab olema teadmata genotüübi `0 0` jaoks. Iga rida tähistab erinevat genotüüpi. Kui ühel real on mitu genotüüpi tabulatsiooniga eraldatud, siis loetakse neid võrdseteks genotüüpideks ja käsitletaksegi edaspidi samaväärsetena.

PSET

PSET fail kirjeldab inimeste gruppe. Iga grupp algab nimega, millele järgneb tühikute või enteritega eraldatud inimeste järjekorranumbrid (need

```
0 0
A A
A B     B A
B B
```

Joonis A.4: Näidisfaili GT sisu

on kirjeldatud PIDX failis) ning lõppeb `END` tekstiga. Ühes failis võib olla mitu gruppi.

```
kontroll
    1
    7
END
haiged
    2  6  10 11 12
END
```

Joonis A.5: Näidisfaili PSET sisu

FRQA ja FRQG

FRQA ja FRQG failid on sageduse failid. FRQA sisaldab alleelide sagedusi ja FRQG fail sisaldab genotüüpide sagedusi. Esimeses väljas on kromosoom, teises väljas marker ja kolmandas väljas on sagedused. Faili esimesel real on kirjeldatud alleelide/genotüüpide järjekord sageduste jaoks.

```
CHR     SNP     [A,C,G,T]
1       Mark0   [0.45,0.55,0,0]
1       Mark1   [0,0,0.3,0.7]
1       Mark2   [0,0,0.375,0.625]
```

Joonis A.6: Näidisfaili FRQA sisu

HWB

HWB fail sisaldab Hardy-Weinbergi testi tulemusi. Esimeses veerus on marker, teises χ^2 väärtus, kolmandas p-väärtus, neljandas genotüüpide nimed ning viiendas genotüüpide sagedused.

```
SNP     CHI     p       GT              GT(c)
Mark0   0.9940  0.3188  [C C,C G,G G]   [231,501,239]
Mark1   0.0307  0.8610  [C C,C G,G G]   [233,483,256]
Mark2   8.6144  0.0033  [C C,C G,G G]   [233,533,209]
```

Joonis A.7: Näidisfaili HWB sisu

ASCA

ASCA failis on assotsiatsiooniuuringu tulemused. Esimeses veerus on kirjas marker, teises χ^2 väärtus, kolmandas p-väärtus, neljandas kontrollgrupi sagedused ja viiendas juhtumite sagedused. Sageduste järjekord on kirjeldatud esimeses reas.

```
MARKER  CHISQ   P       [A,C,G,T]               [A,C,G,T]
Mark0   3.5390  0.0599  [0,0.5173,0.4827,0]     [0,0.4744,0.5256,0]
Mark1   1.7851  0.1815  [0,0.5021,0.4979,0]     [0,0.4716,0.5284,0]
Mark2   0.3088  0.5784  [0,0.5052,0.4948,0]     [0,0.5178,0.4822,0]
```

Joonis A.8: Näidisfaili ASCA sisu

Lisa B

Programmid

Kõikidel programmidel saab täpsemat infot käivitades neid lipuga `-h`.

gmapconvert

```
Sisend     : <sisend>.ped <sisend>.map <sisend>.gt
Väljund    : <väljund>.gmap <väljund>.midx <väljund>.gidx <väljund>.aidx
Käivitamine: gmapconvert -i <sisend> <väljund>
```

Programm gmapconvert teisendab PED ja MAP failid GMAP, AIDX, GIDX ja MIDX formaati ümber.

ALL fail peab sisaldama kõiki genotüüpe, mida on PED failis kasutatud.

gmappack

```
Sisend     : <sisend>.gmap <sisend>.midx <sisend>.gidx <sisend>.aidx
Väljund    : <väljund>.gmap <väljund>.midx <väljund>.gidx <väljund>.aidx
Käivitamine: gmappack -i <sisend> <väljund>
```

Programm pakib genotüübi andmed kokku kasutades teadmist, et enamustel markeritel on ainult kaks alleeli. Kõik markerid, mis seda tingimust ei täida, jäetakse välja ning nende nimed kirjutatakse logifaili. Programm laiendab MIDX faili peaalleeli ja alamalleelidega. Genotüüpi failis tähistatakse peaalleeli A-ga ning alamalleeli B-ga. Sisend ja väljund ei tohi olla sama nimega.

gmapfreq

```
Sisend     : <nimi>.gmap <nimi>.midx <nimi>.gidx <nimi>.aidx (<mask>.pset)
Väljund    : <nimi>.frqa <nimi>.frqg
Käivitamine: gmapfreq (-c) (-g <mask>.pset) <nimi>
```

Programm arvutab andmetest genotüüpide ja alleelide protsendi. Kui programmile on antud kaasa lipp `-c`, siis väljastab programm allelide ja genotüüpide arvu.

Programmile võib lisaks kaasa anda isikute grupid PSET failiga. Sel juhul tekib iga grupi jaoks `<nimi>.<grupp>.frqa` ja `<nimi>.<grupp>.frqg`.

gmaphardyweinberg

```
Sisend     : <nimi>.gmap <nimi>.midx <nimi>.gidx <nimi>.aidx
Väljund    : <nimi>.hwb
Käivitamine: gmaphardyweinberg <nimi>
```

Programm viib läbi peatükis 3.1 kirjeldatud Hardy-Weinbergi testi ning annab tulemuseks HWB faili.

gmapassoc

```
Sisend     : <nimi>.gmap <nimi>.midx <nimi>.gidx <nimi>.aidx <mask>.pset
Väljund    : <nimi>.asca
Käivitamine: gmapassoc -g <mask>.pset <nimi>
```

Programm viib läbi peatükis 3.2 kirjeldatud assotsiatsioniuuringu.

Failis `<mask>.pset` kirjeldatud esimest gruppi käsitletakse kontrollgrupina ning teist gruppi juhtumitena.

gmapgenerate

```
Sisend     : <nimi>.gt
Väljund    : <väljund>.ped <väljund>.map
Käivitamine: gmapgenerate -s <markereid>x<inimesi> -g <nimi>.gt <väljund>
Näide      : gmapgenerate -s 10000x1000 -g naide.gt naide
```

Programm genereerib suvaliselt PED ning MAP failid. Genotüübid võetakse GT failist. Iga GT failis kirjeldatud genotüübil on võrdne võimalus sattuda suvalisse positsiooni PED failis. See tähendab, et geneeritud fail ei kajasta sisult reaalseid PED ja MAP faile. Genereeritud failid sobivad ainult kiiruse testimiseks.

gmaprandpheno

```
Sisend     : <nimi>.gmap <nimi>.midx <nimi>.gidx <nimi>.aidx <mask>.pset
Väljund    : <väljund> <väljund>.pset
Käivitamine: gmaprandpheno -i <nimi> <väljund>
```

Programm tekitab suvaliselt indiviidide fenotüübi faili. `<väljund>`-i veerud on perekonna id, isiku id ja fenotüüp (`0` - teadmata, `1` - omaduseta, `2` omadusega). Tõenäosus, et tuleb fenotüüp `0` on 1%. Ülejäänutel on võimalus 49.5%.

Lisa C

Arvu bitisagedus

Arvu bitisageduse leidmise meetodeid on erinevaid. Toome siin välja kaks kiiremat meetodit, mis on kasutusel ka GMap teegis, Gurmeet Singh Manku artiklist[12]. Bitisageduse leidmise algoritmid on kirjeldatud 64-bitiste arvude jaoks.

Binaararvu tähistamiseks kasutame ülajoont (näiteks $\overline{11} = 3$). Selline tähistus aitab paremini näidata bittide mõttelist grupeerimist jättes alles arvu piirid (näiteks $\overline{11\ 00}\quad\overline{01\ 01}$). Meetodite kirjeldamiseks läheb meil vaja lisaks bititehteid, mis on kirjeldatud tabelis C.1.

Tehe	Sümbol	Näide
Arvude liitmine	$+$	$\overline{1001} + \overline{0101} = \overline{1110}$
Loogiline "JA"	$\wedge$	$\overline{1001} \wedge \overline{0101} = \overline{0001}$
Loogiline "VÕI"	$\vee$	$\overline{1001} \vee \overline{0101} = \overline{1101}$
Bitinihe paremale	$\triangleright$	$\overline{1001} \triangleright 1 = \overline{0100}$

Tabel C.1: Bititehted

C.1 Paralleelne loendamine

Paralleelne bittide loendamine seisneb lihtsal põhimõttel – jaotame bitid paaridesse ning liidame nad omavahel. Pärast seda jaotame tekkinud summad paaridesse ning liidame need omavahel. Kordame seda protsessi seni kuni oleme kõik bitid omavahel kokku liitnud.

Näiteks on meil binaararv $\overline{11010010}$. Jaotame bitid paaridesse:

$$\overline{11\ 01\ 00\ 10}$$

Selleks, et paarides olevaid arve kokku liita peame nad üksteisest eraldama. Seda saame teha bitimaskidega $\overline{10\ 10\ 10\ 10}$ ja $\overline{01\ 01\ 01\ 01}$. Õigete bittide kokkuliitmiseks tuleb eespool olevaid bitte nihutada ühe koha võrra paremale. Kokkuliitmine näeb välja nii:

$$\begin{aligned}(\overline{11\ 01\ 00\ 10} \wedge \overline{10\ 10\ 10\ 10}) \triangleright 1 + \overline{11\ 01\ 00\ 10} \wedge \overline{01\ 01\ 01\ 01} &= \\ \overline{10\ 00\ 00\ 10} \triangleright 1 + \overline{01\ 01\ 00\ 00} &= \\ \overline{01\ 00\ 00\ 01} + \overline{01\ 01\ 00\ 00} &= \\ \overline{10\ 01\ 00\ 01}\end{aligned}$$

Summade eraldamiseks kasutame bitimaske $\overline{11\ 00\ 11\ 00}$ ja $\overline{00\ 11\ 00\ 11}$. Seekord tuleb esimesi bitte nihutada kahe koha võrra, sest summad on kirjeldatud kahe bitiga.

$$\begin{aligned}(\overline{10\ 01\ 00\ 01} \wedge \overline{11\ 00\ 11\ 00}) \triangleright 2 + \overline{10\ 01\ 00\ 01} \wedge \overline{00\ 11\ 00\ 11} &= \\ \overline{10\ 00\ 00\ 00} \triangleright 2 + \overline{00\ 01\ 00\ 01} &= \\ \overline{00\ 10\ 00\ 00} + \overline{00\ 01\ 00\ 01} &= \\ \overline{00\ 11\ 00\ 01}\end{aligned}$$

Nüüd eraldame nende gruppide summad bitimaskidega $\overline{1111\ 0000}$ ja $\overline{0000\ 1111}$

ning nihutame nelja koha võrra.

$$(\overline{0011\ 0001} \wedge \overline{1111\ 0000}) \triangleright 4 + \overline{0011\ 0001} \wedge \overline{0000\ 1111} =$$
$$\overline{0011\ 0000} \triangleright 4 + \overline{0000\ 0001} =$$
$$\overline{0000\ 0011} + \overline{0000\ 0001} =$$
$$\overline{0000\ 0100}$$

Tulemuseks on bittide sagedus $\overline{00000100} = 4$.

Algoritm C.1: Bitisageduse leidmine paralleelse loendamisega

```
private {
    ulong m1  = 0x5555555555555555L; // 0101...
    ulong m2  = 0x3333333333333333L; // 00110011..
    ulong m4  = 0x0f0f0f0f0f0f0f0fL; // 4 nulli,  4 yhte ...
    ulong m8  = 0x00ff00ff00ff00ffL; // 8 nulli,  8 yhte ...
    ulong m16 = 0x0000ffff0000ffffL; // 16 nulli, 16 yhte ...
    ulong m32 = 0x00000000ffffffffL; // 32 nulli, 32 yhte
}

static byte bitcount_parallel( ulong x ){
    x = (x & m1 ) + ((x >>  1) & m1 );
    x = (x & m2 ) + ((x >>  2) & m2 );
    x = (x & m4 ) + ((x >>  4) & m4 );
    x = (x & m8 ) + ((x >>  8) & m8 );
    x = (x & m16) + ((x >> 16) & m16);
    x = (x & m32) + ((x >> 32) & m32);
    return x;
}
```

C.2 Sageduse leidmine 16-bitiste arvude bitisagedustabeli põhjal

See meetod on eelnevast lihtsam, aga vajab rohkem mälu ning eelnevat ülesseadmist. Me koostame 16-bitiste arvude tabeli, kus igale arvule vastab tema bitisagedus. Rohkemate bittidega arvu bitisageduse leidmiseks teeme arvu 16-bitisteks osadeks, leiame tabelist nendele osadele vastavad bitisa-

gedused ning liidame nende sagedused. See tabel tuleb tekitada programmi käivitamisel mingi teise bitiloendamise meetodiga.

Kuna seda algoritmi kasutatakse meie teegis palju, siis on see implementeeritud assemblerkeeles.

Algoritm C.2: Bitisageduse arvutamine 16-bitiste arvude tabeli põhjal

```
byte[1 << 16] bitcount16;
static this() {
    for(uint x = 0; x < bitcount16.length; x++)
        bitcount16[x] = bitcount_parallel(x);
}
\\ ekvivalentne kood k6rgemas keeles
static uint bitcount_kood( ulong x ){
    return bitcount16[x & 0xFFFF] +
           bitcount16[(x >> 16) & 0xFFFF] +
           bitcount16[(x >> 32) & 0xFFFF] +
           bitcount16[(x >> 48) & 0xFFFF];
}

static uint bitcount( ulong x ){
    asm{naked                   ;
        xor EAX, EAX            ;
        xor ECX, ECX            ;
        mov CX,  [ESP + 4]      ;
        add AL,  bitcount[ECX]  ;
        mov CX,  [ESP + 6]      ;
        add AL,  bitcount[ECX]  ;
        mov CX,  [ESP + 8]      ;
        add AL,  bitcount[ECX]  ;
        mov CX,  [ESP + 10]     ;
        add AL,  bitcount[ECX]  ;
        ret 8                   ;
    }
}
```

Lisa D

Lisatud CD lähtekoodiga

Bakalaureusetööga on kaasas CD, mis sisaldab programmide ja teekide lähtekoodi ning testimisskripte. Täpsem informatsioon CD sisu ja kompileerimise kohta on failis `README`.