AIAS Short Course: Methods for Analysing Complex Trait GWAS Data

Doug Speed and Bjarni Vilhjálmsson

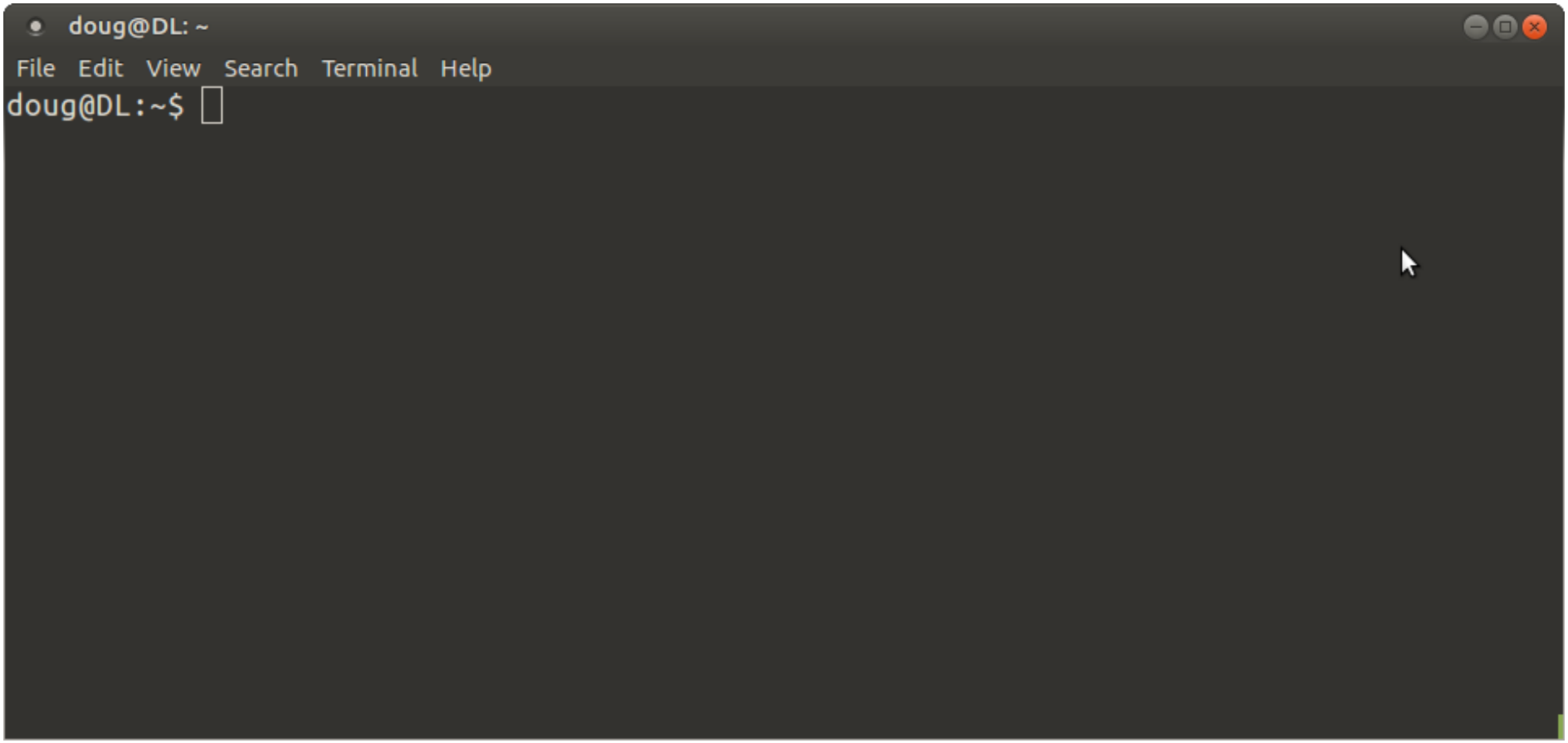
Aarhus University and QIAGEN

Thursday 22nd March 2018

www.ldak.org/short

Today's practical

We will be working in the terminal window (UNIX)



In Linux, will typically find in Applications / System Tools

In Macs go to Finder / Applications / Utilities

Today's practical

For the practicals, I suggest you make a new folder containing the data files and LDAK executable (either ldak5.linux or ldak5.mac).

Open the terminal and navigate to this folder (using `cd`). Check you can see the files using `ls`, and run ldak by typing `ldak5.linux` or `ldak5.mac`

```
doug@DL: ~/workdir
File  Edit  View  Search  Terminal  Help
doug@DL:~$ cd workdir/
doug@DL:~/workdir$
doug@DL:~/workdir$ ls
binary.pheno  hapmap.bim  human.bed  human.fam  mice.bed  mice.fam    multi.pheno
hapmap.bed    hapmap.fam  human.bim  ldak5.XXX  mice.bim  mice.pheno  quant.pheno
doug@DL:~/workdir$
doug@DL:~/workdir$ ./ldak5.XXX
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
LDAK - Software for obtaining Linkage Disequilibrium Adjusted Kinships and Loads More
Version 5.0 - Help pages at http://www.ldak.org
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

There are 0 pairs of arguments:

Each command requires a main argument:

--cut-weights <folder> - divide genome into sections ready for calculating weightss
--calc-weights <folder> - calculate weights for specified section
--join-weights <folder> - join up weights across sections
```

Working in the Terminal

There are many built-in UNIX commands, for example:

`cd` or `pwd` - change directory or print working directory

`ls` - list files

`cp` or `mv` - copy or move files

`mkdir` - make a directory

`head` or `less -S` or `more` - view tops of files

`rm` - remove files and folders (be careful)

`man` - display help files (e.g., `man ls`)

Note that a backslash (\) at the end a of line indicates the command spans multiple lines

PLINK

We will mainly be using my software LDAK, but there will also be a few scripts in PLINK and R

PLINK is the most popular genetics software. It has about 100 functions, and many developers. You will often hear people talk about data (or covariates or phenotypes) in "PLINK format"

The most recent version is PLINK 1.9, which is very, very fast
https://www.cog-genomics.org/plink2

There is a tutorial available on the old PLINK site
http://zzz.bwh.harvard.edu/plink/tutorial.shtml

R statistical programming language

I rarely use R to perform analysis, as it struggles with large datasets (although I'm told it is now much better). However, I think R is unrivalled when it comes to simple data manipulation and plotting

You start R by installing then typing "R" in a terminal; otherwise, there are GUI versions such as R Studio. It has its own syntax, similar (but slightly different) to, say, Matlab, Stata, S-PLUS. Some tutorials are available at:

https://www.r-project.org/

http://www.r-tutor.com/r-introduction

http://arrgh.tim-smith.us

We will only use a handful of R commands today

LDAK

Designed in the style of PLINK. For example, all options start with two dashes, if you want to provide data in binary PLINK format, you use `--bfile`, to select a subset of SNPs you use `--extract`, provide phenotypes with `--pheno`, perform linear regression with `--linear`, etc.

Has some similarity with GCTA — you merge kinship matrices using `--add-grm`, estimate variance components using `--reml`, and kinship matrices are stored in a compatible format.

LDAK performs mosts features of GCTA (not bivariate and some conditional analyses). LDAK has a number of additional features (e.g., computing SNP weights, merging datasets, profile scoring, thinning and clumping, as well as GBAT for set-based association testing, SumHer for summary stats heritability analysis, and MultiBLUP for prediction)

LDAK can also perform all the features of LDSC.

LDAK Commands

Call LDAK by typing the name of the executable (same for PLINK, GCTA)

All the scripts assume the executable is called `ldak5.XXX` so either edit the scripts, or rename the file (e.g., `mv ldak5.linux ldak5.XXX`)

```
./ldak5.XXX

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
LDAK - Software for obtaining Linkage Disequilibrium Adjusted Kinshi
Version 5.0 - Help pages at http://www.ldak.org
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

There are 0 pairs of arguments:

Each command requires a main argument:

--cut-weights <folder> - divide genome into sections ready for calcu
--calc-weights <folder> - calculate weights for specified section
--join-weights <folder> - join up weights across sections
--calc-weights-all <folder> - calculate weights for all sections in
```

LDAK Commands

LDAK options are provided in pairs. One main argument is required, which is followed by the name of the output file.

For example, suppose we want to compute statistics, we would use the main argument `--calc-stats`. The command might look like:

```
./ldak5.XXX --calc-stats output --bfile human
```

This is asking LDAK to read the data stored in Binary PLINK format with prefix `human`, then save the results to files with prefix `output`

Option pairs can be provided in any order, so would be equivalent to use

```
./ldak5.XXX --bfile human --calc-stats output
```

(This command computes MAF, call-rates and possibly info scores)

Data Formats

LDAK can accommodate almost any type of data. The formats are:

Binary PLINK (`--bfile`) - the most commonly used format, an efficient way to store hard genotypes (all values are 0, 1, 2 or NA). https://www.cog-genomics.org/plink/1.9/input#bed

Gen / Chiamo / Oxford Stat Format (`--gen`) - rows are predictors, columns are SNPs, can set the number of header rows and columns. Typically, used to provide state probabilities. http://www.stats.ox.ac.uk/~marchini/software/gwas/file_format.html. Can also store haplotype/phasing calls or dosage, etc. Files can be raw text or gzipped.

SP Format (`--sp`) - Simple text matrix, with a row for each predictor and a column for each individual. Value need not correspond to SNPs

SPEED Format (`--speed`) - binary version of SP Format

More info at http://www.ldak.org/file-formats/

Data files

For these practicals, we will use the files:

`human.bed`, `human.bim` and `human.fam` - 1000 Genomes Project SNP data for 424 humans, 3289 SNPs (Chromosomes 21 & 22)

`quant.pheno`, `binary.pheno`, `multi.pheno` and `covar.covar` - phenotypes and covariates for these

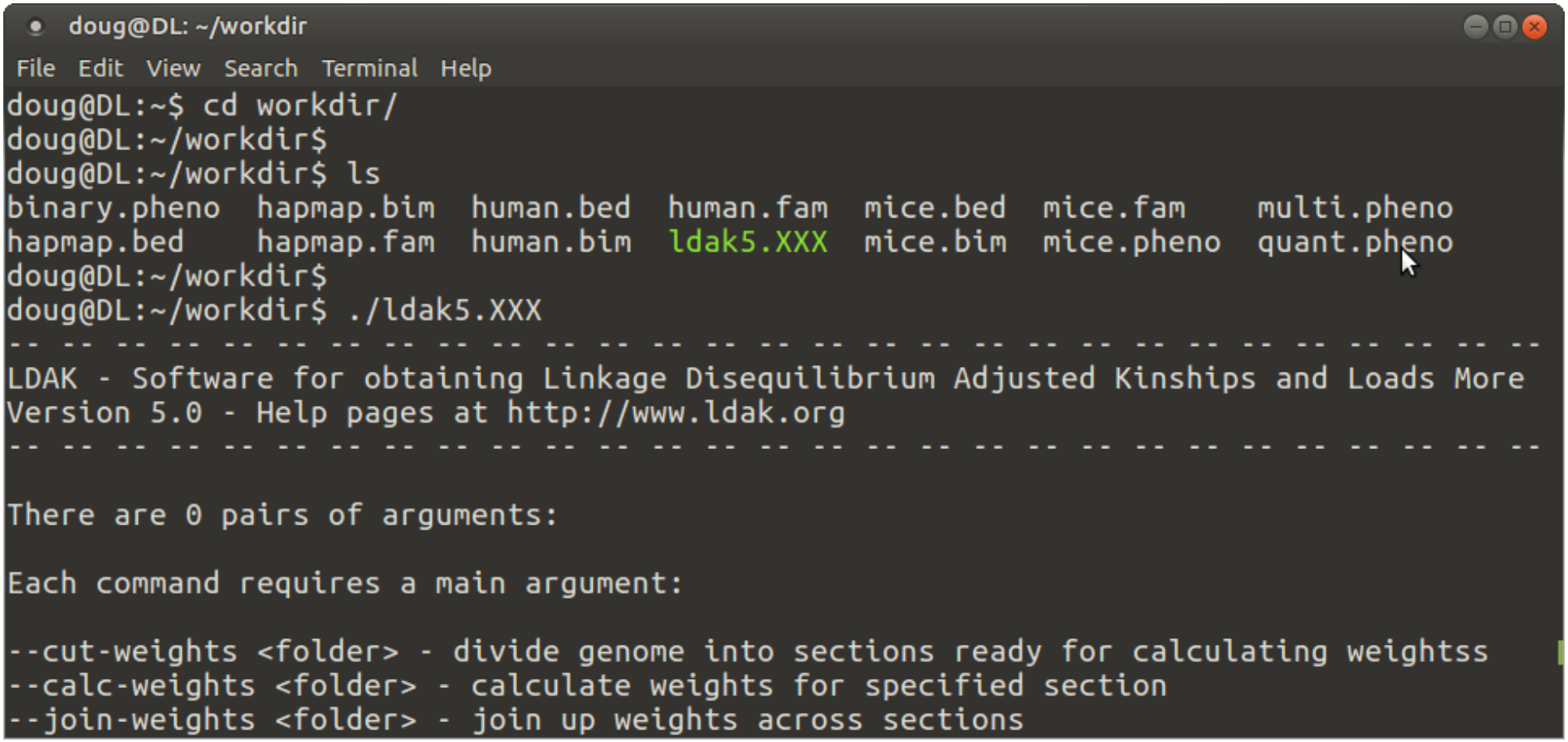
`hapmap.bed`, `hapmap.bim` and `hapmap.fam` - SNP data for 1184 humans, 1016 SNPs (from HapMap Project)

`mice.bed`, `mice.bim`, `mice.fam` and `mice.pheno` - SNP and phenotype data for 1940 mice, 2984 SNPs

Tip - to get number of samples/SNPs use `wc -l human.fam human.bim`

Ready to Start!

Open a terminal window and `cd` to your working directory. If you type `ls` you should see the data files; if you type `./ldak5.XXX` LDAK should start



If using R from terminal, I suggest you open two windows, one for LDAK, one for R.

Single-SNP association analysis

NOTE, ALWAYS DO QC BEFORE ANALYSIS.

We wish to test each of the 3 289 SNPs stored in the dataset `human` individually for association with the continuous phenotype contained within `quant.pheno` (using the model $Y = \alpha + \beta_j X_j$).

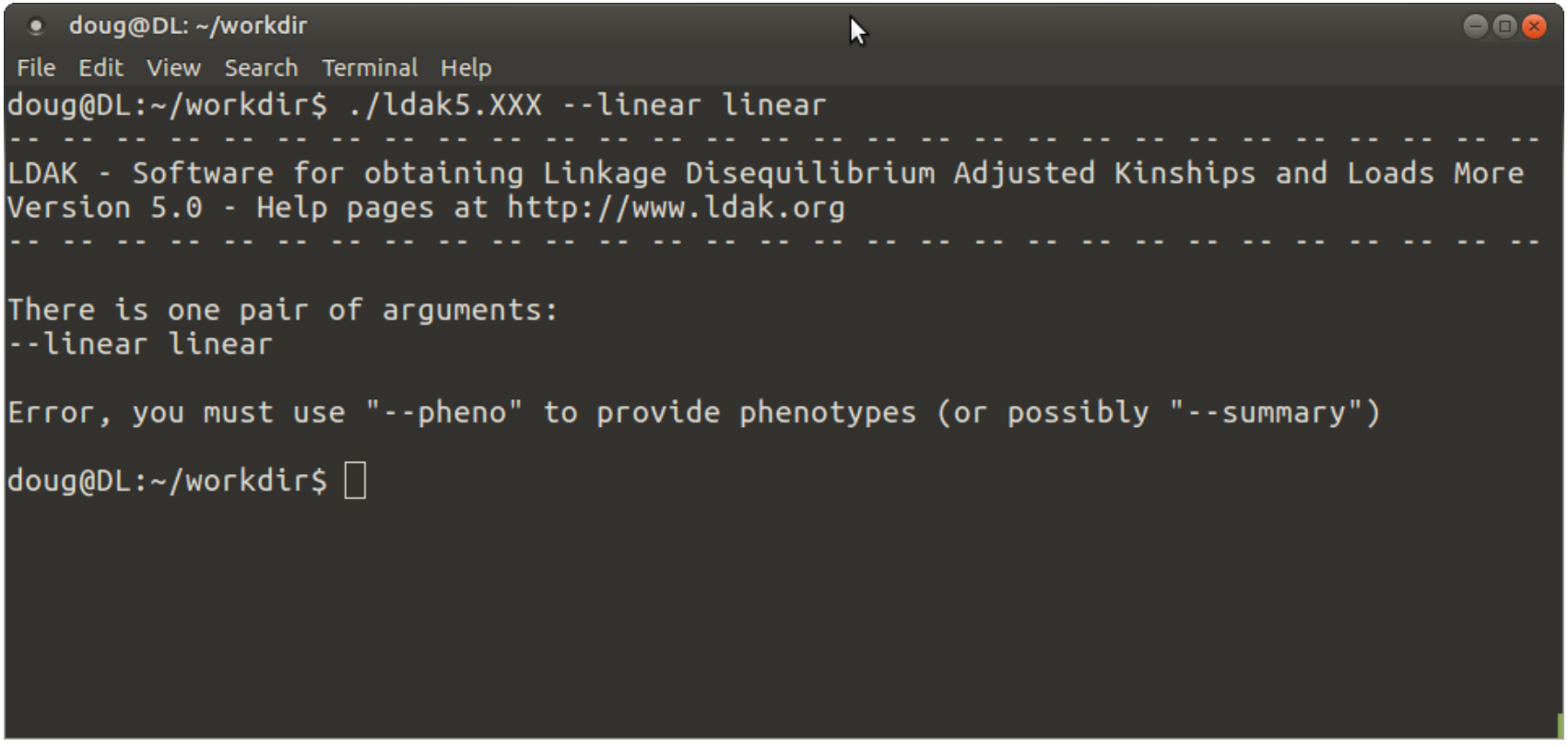
For this we use the main argument `--linear`

Each main argument requires different options. Documentation is provided at www.ldak.org, but typically the easiest way is to run LDAK using just the main argument, then it will tell you what options you require

So here we type

```
./ldak5.XXX --linear linear
```

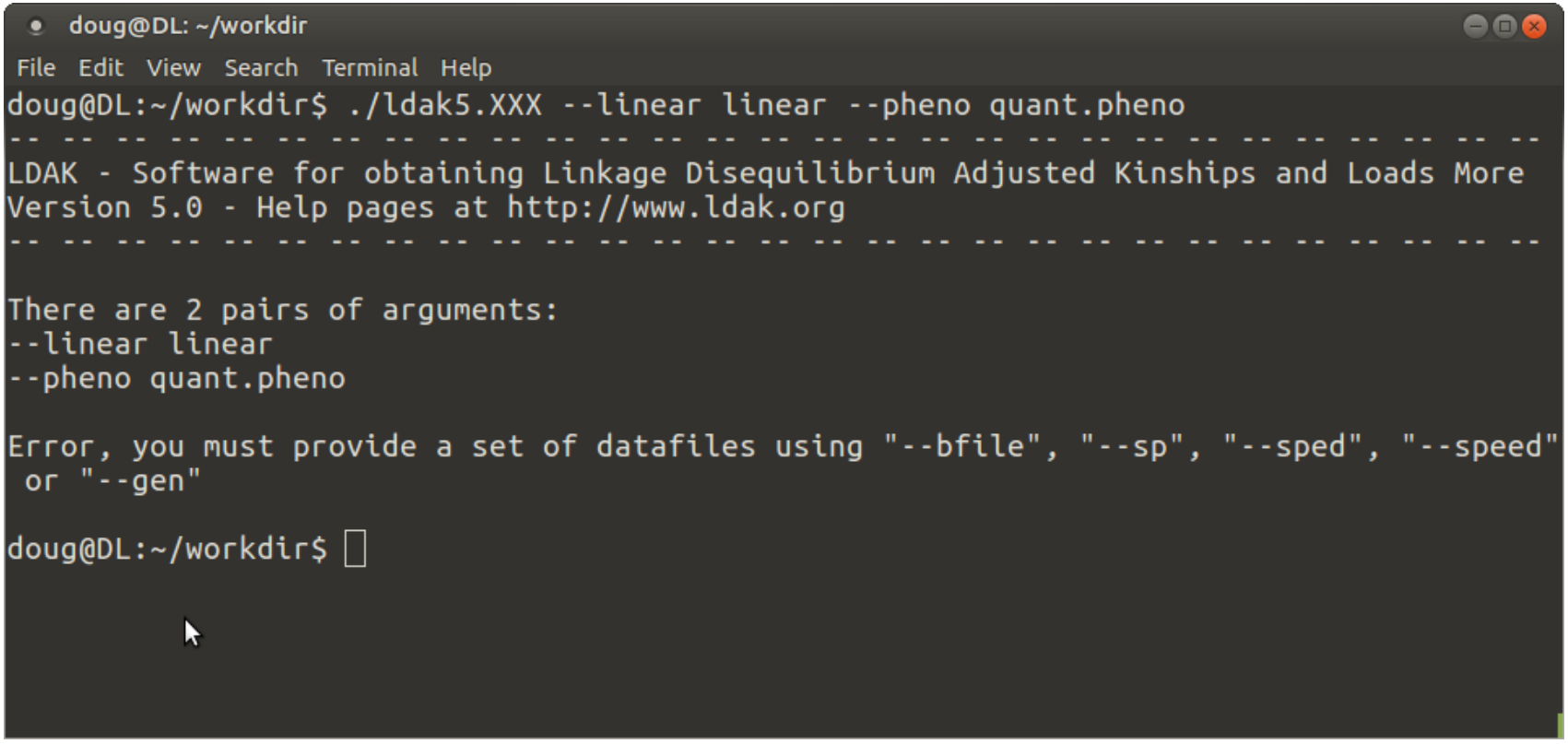
Single-SNP association analysis



First LDAK tells us we need to provide phenotypes, so we add the option `--pheno quant.pheno`

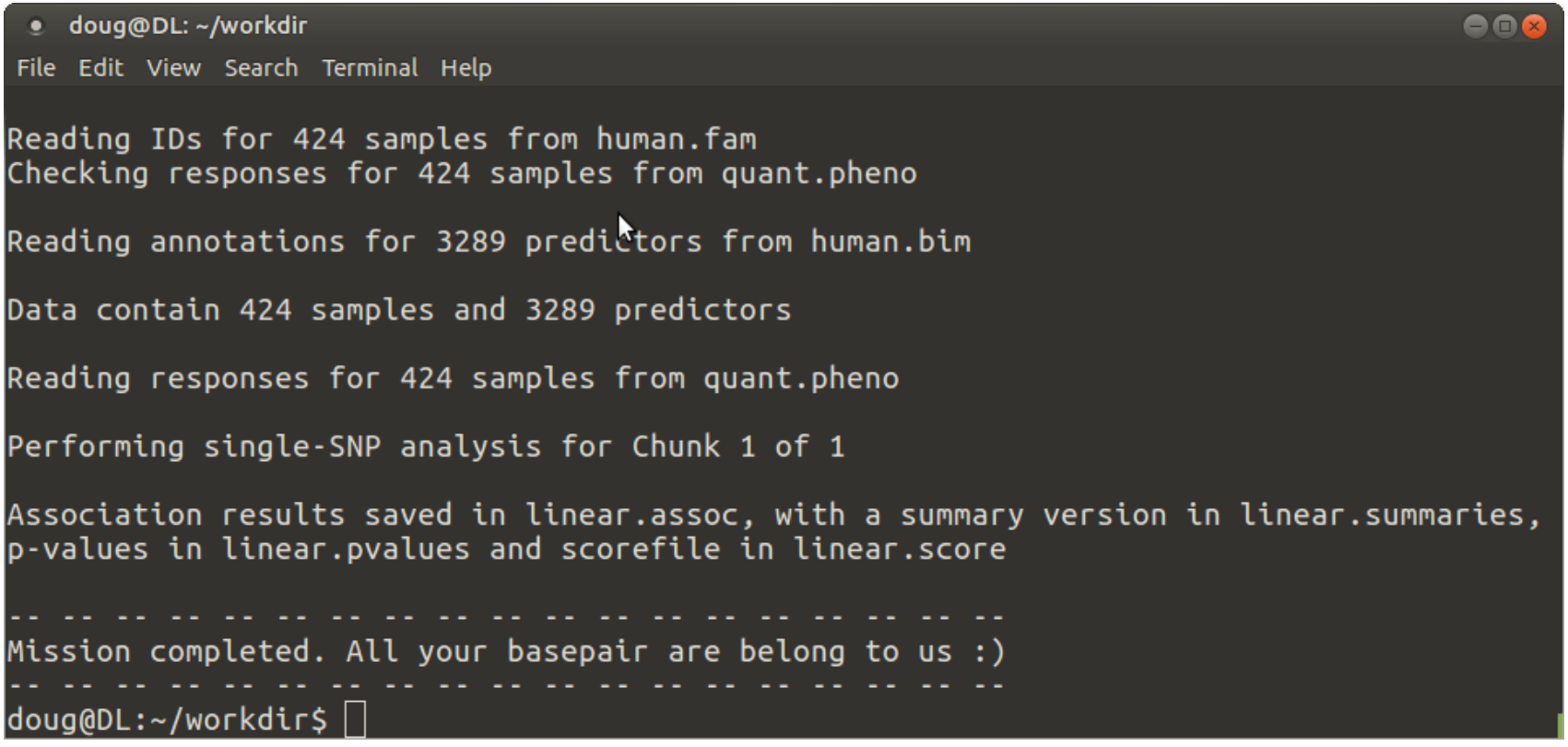
Tip - use the up arrow to return to previous commands

Single-SNP association analysis



Now LDAK tells us we need to provide datafiles, so we add the option `--bfile human`

Single-SNP association analysis



Vaersgo: the final command is

```
./ldak5.XXX --linear linear --pheno quant.pheno --bfile human
```

Single-SNP association analysis

Always pay attention to the screen output

It will tell you what options are required and why there are errors.

It will also update you on progress (as will the file `.progress`) and tell you where the results are saved.

For the linear regression, the main results are in `linear.assoc`. We will use `linear.summaries` for summary-statistic methods, `linear.pvalues` for clumping, while `linear.score` provides the polygenic risk score prediction model.

Single-SNP association analysis

Some extra options we might wish to use are `--keep` (to use only a subset of individuals), `--extract` (to use only a subset of predictors) and `--covar` (to include covariates, such as age, sex, population axes)

To achieve the same in PLINK, you would use

```
plink --linear --out linear_plink --bfile human --pheno quant.pheno
```

Note that by default PLINK ignores individuals without sex (add `--allow-no-sex` to continue).

Note, LDAK does not (yet) include logistic regression; to do this in PLINK use the option `--logistic`

Has everyone performed the single-SNP analysis?

Single-SNP association analysis

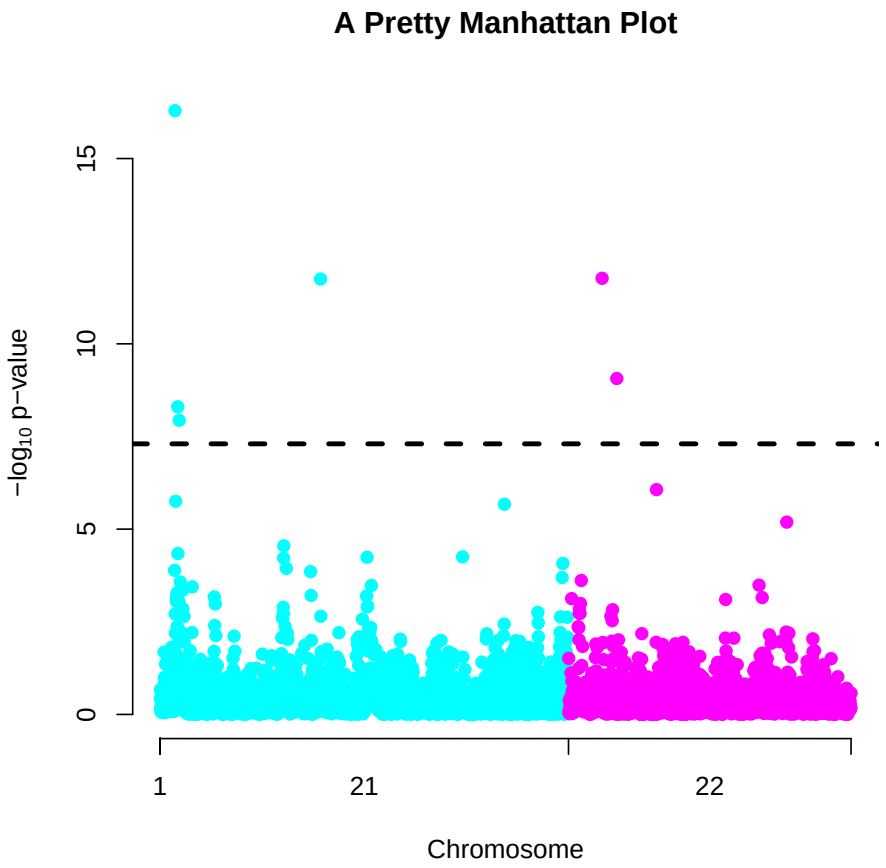
Let's read into R and find the top association

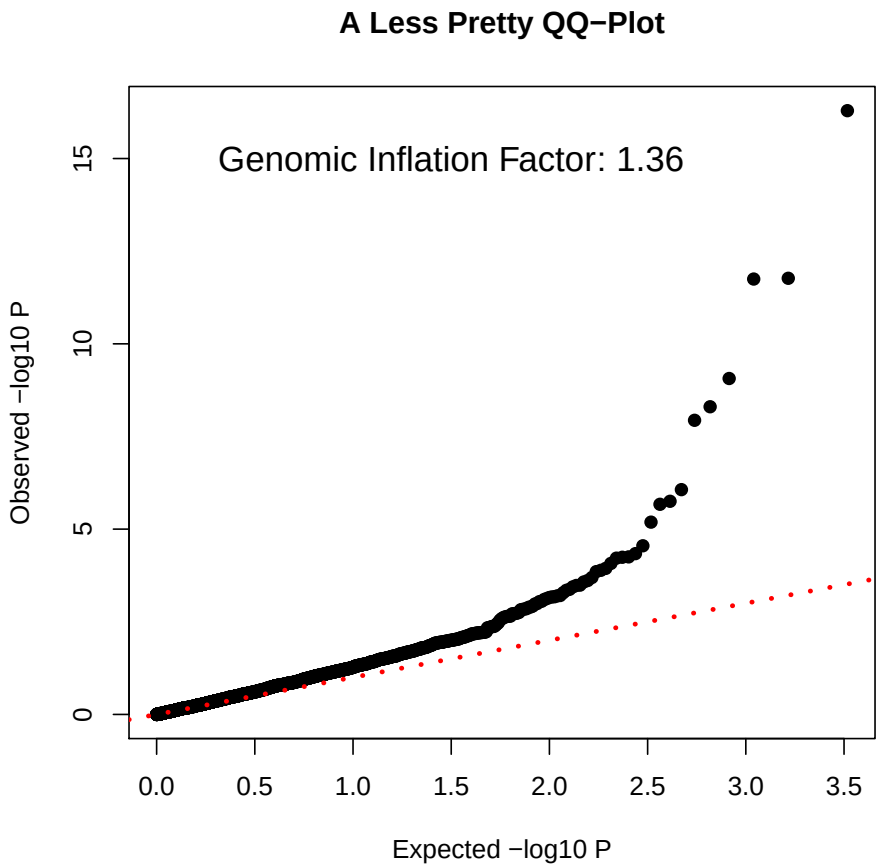
```
assoc=read.table("linear.assoc",head=T)

head(assoc,4)
  Chromosome Predictor Basepair A1 A2 Wald_Stat  Wald_P    Effect       SD
1         21 rs2260895 14642464  A  G  0.875011 0.38157  0.061757 0.070578
2         21 rs2821796 14649798  A  C -1.241588 0.21439 -0.085988 0.069256
3         21 rs1148737 14669931  G  A  0.435302 0.66334  0.030648 0.070406
4         21 rs1296972 14688825  T  C  0.208391 0.83492  0.014737 0.070718
  Heritability Effect_Liability SD_Liability Her_Liability  A1_Mean      MAF
1           NA               NA           NA            NA 0.884434 0.442217
2           NA               NA           NA            NA 0.974057 0.487028
3           NA               NA           NA            NA 0.969340 0.484670
4           NA               NA           NA            NA 0.983491 0.491745

which.min(assoc[,7])
[1] 71
> assoc[71,]
   Chromosome Predictor Basepair A1 A2 Wald_Stat     Wald_P   Effect       SD
71         21  rs389338 15603999  G  A  8.384245 5.1052e-17 0.710924 0.084793
   Heritability Effect_Liability SD_Liability Her_Liability  A1_Mean      MAF
71           NA               NA           NA            NA 0.360849 0.180425
```

Manhattan and QQ Plots

```
par(mfrow=c(1,2))

plot(-log10(assoc[,7]),col=assoc[,1],pch=19,xlab="Chromosome",
ylab=expression(paste("-log"[10]," p-value",sep="")),
main="A Pretty Manhattan Plot",axes=F)

marks=array(0,23);
for(i in 1:22){a=which(assoc[,1]==i);marks[i+1]=marks[i]+length(a);}
marks2=(marks[-1]+marks[-23])/2
axis(1,at=marks,rep("",23));axis(1,at=marks2,lab=1:22,tick=F)
axis(2);abline(h=-log10(5e-8),lwd=3,lty=2)

ord=order(assoc[,7]);obsP=assoc[ord,7]
N=nrow(assoc);expP=1:N/(N+1);
plot(-log10(expP),-log10(obsP),pch=19,xlab="Expected -log10 P",
ylab="Observed -log10 P",main="A Less Pretty QQ-Plot");
abline(a=0,b=1,col=2,lwd=3,lty=3)

gif=qchisq(median(assoc[,7]),1,lower=F)/qchisq(.5,1)
text(1.5,15,paste("Genomic Inflation Factor:",round(gif,2)),cex=1.3)
```

Single-SNP association analysis





Single-SNP association analysis

(Back in the terminal)

To identify genome-wide associated SNPs we can use `awk`

```
awk < linear.assoc '($7<5e-8){print $0}'
21 rs389338 15603999 G A 8.384245 5.1052e-17 0.710924 0.084793 NA NA NA NA
21 rs718673 15648392 G A 5.847790 4.9815e-09 0.550284 0.094101 NA NA NA NA
21 rs2822548 15656052 A C 5.706246 1.1550e-08 0.433156 0.075909 NA NA NA NA
21 rs2824534 19259311 G A -7.050404 1.7840e-12 -0.463614 0.065757 NA NA NA
22 rs4819979 17718119 T G -7.056939 1.7021e-12 -0.463612 0.065696 NA NA NA
22 rs5746419 17971883 C T 6.133227 8.6114e-10 0.618173 0.100791 NA NA NA NA
```

I find awk very useful for dealing with files too large to (easily) read into R.

For a quick guide, see www.ldak.org/awk.

Single-SNP association analysis

To identify the "independent loci", use the command `--thin-tops`; this will find then prune all SNPs with *p*-values below a specified threshold.

`--thin-tops` requires lots of options, but again, LDAK will walk you through them ...

```
./ldak5.XXX --thin-tops top --bfile human --pvalues linear.pvalues \
--cutoff 5e-8 --window-cm 1 --window-prune .2

cat top.in
rs389338
rs2824534
rs4819979
rs5746419

awk '(NR==FNR){arr[$1];next}($2 in arr){print $0}' top.in linear.assoc
21 rs389338 15603999 G A 8.384245 5.1052e-17 0.710924 0.084793 NA NA NA NA
21 rs2824534 19259311 G A -7.050404 1.7840e-12 -0.463614 0.065757 NA NA NA
22 rs4819979 17718119 T G -7.056939 1.7021e-12 -0.463612 0.065696 NA NA NA
22 rs5746419 17971883 C T 6.133227 8.6114e-10 0.618173 0.100791 NA NA NA NA
```

Preparing a GWAS

The steps are as follows:

1 - Sample and SNP quality control

2 - Exclude ethnic outliers

3 - Imputation

4 - More SNP quality control

5 - Exclude related individuals

Steps 1, 2 & 3 are fairly constant. Steps 4 & 5 depend on the aim of the GWAS; heritability analyses generally require much stricter quality control than single-SNP analysis, and are designed for unrelated individuals. Would also increase QC if studying a binary phenotype.

E.g., for a single-SNP analysis, I might use an info score threshold of >0.5 or >0.8, but for heritability analysis require info >0.99

Quality Control - Genotyping Accuracy

We want to remove samples and SNPs which are poorly genotyped.

For samples, common exclusion criteria are missingness and heterozygosity

Can also check SNP-sex matches clinical-sex.

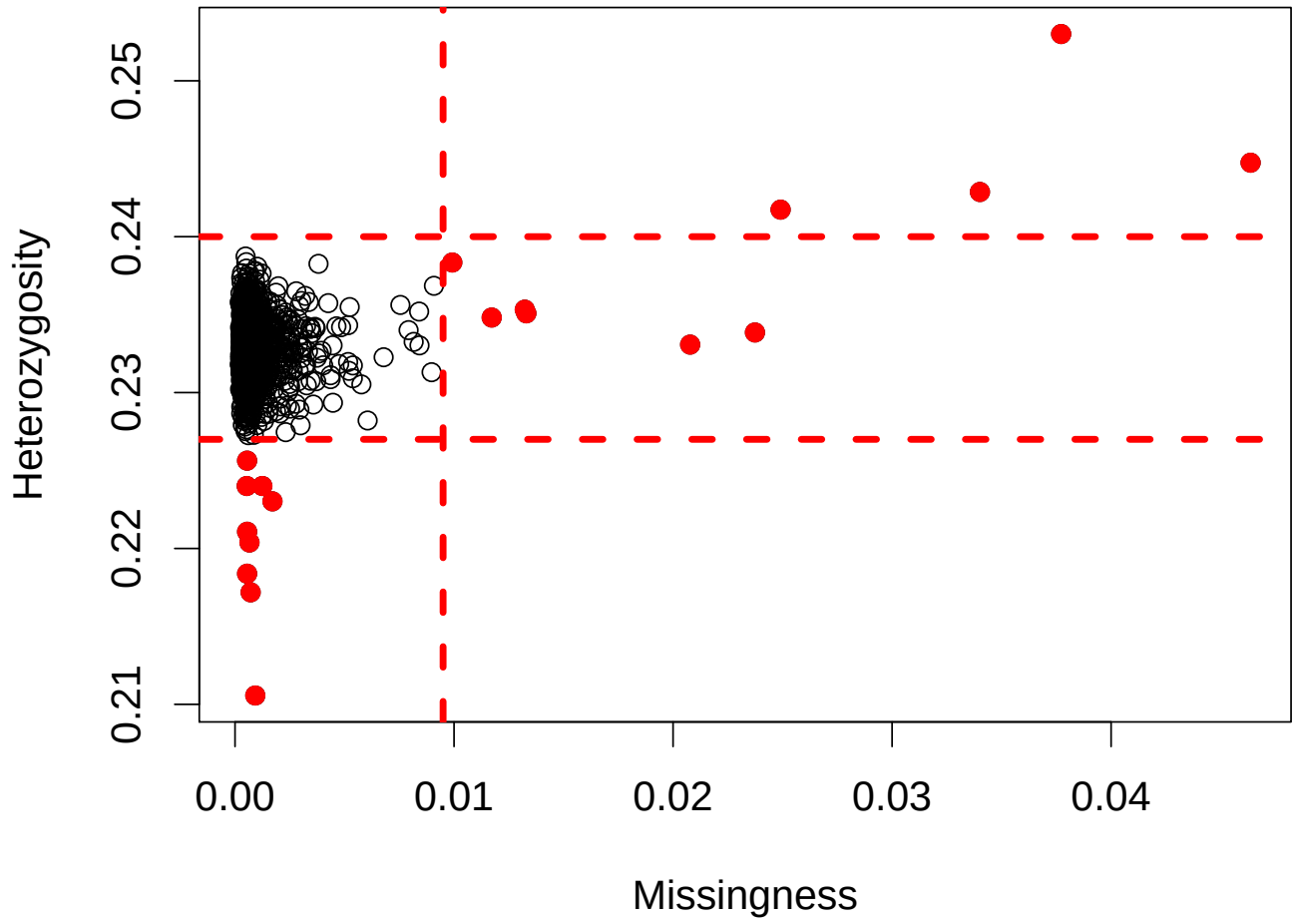
For SNPs, might use call-rate, HWE p-value and minor allele frequency

There are commonly-used thresholds (e.g., call-rate >0.95, MAF >0.01), but I prefer to decide thresholds by eye, by plotting and looking for a "cliff-edge"

Having identifed poor samples and SNPs, remake the data excluding those.

A good reference is Mike Weales Chapter on Quality Control: `link.springer.com/content/pdf/10.1007%2F978-1-60327-367-1_19.pdf`

Quality Control - Genotyping Accuracy



Could instead use more advanced methods - e.g., exclude points more than 3 SDs from the median; but in general I find the impact is slight

Quality Control - Ethnic Outliers

We typically wish to identify and exclude population outliers
(the exception is mixed-model association analysis)

People often use (in-sample) principal component analysis (PCA):

1 - Thin the SNPs, calculate a kinship matrix, perform PCA then plot

However, I prefer to project the data onto population axes derived from a diverse reference panel (e.g., HapMap):

1 - Find overlap of SNPs between dataset and reference panel

2 - Using the reference panel, calculate a kinship matrix, perform PCA and compute SNP loadings for the principal components

3 - Project the data onto these principal components, then plot

Quality Control - Ethnic Outliers

We typically wish to identify and exclude population outliers
(the exception is mixed-model association analysis)

People often use (in-sample) principal component analysis (PCA):

1 - Thin the SNPs, calculate a kinship matrix, perform PCA then plot

However, I prefer to project the data onto population axes derived from a diverse reference panel (e.g., HapMap):

1 - Find overlap of SNPs between dataset and reference panel

2 - Using the reference panel, calculate a kinship matrix, perform PCA and compute SNP loadings for the principal components

3 - Project the data onto these principal components, then plot

Standard Approach - Principal Component Analysis

```
#thin for LD
./ldak5.XXX --bfile human --thin prune \
--window-prune .2 --window-cm 1
```

This produces `prune.in` and `prune.out`

```
#compute (unweighted) kinships
./ldak5.XXX --calc-kins-direct prune --bfile human \
--ignore-weights YES --power -1 --extract prune.in \
--kinship-raw YES
```

The kinship matrix is stored with stem prune

```
#calculate principal components
./ldak5.XXX --pca prune --grm prune --axes 5
```

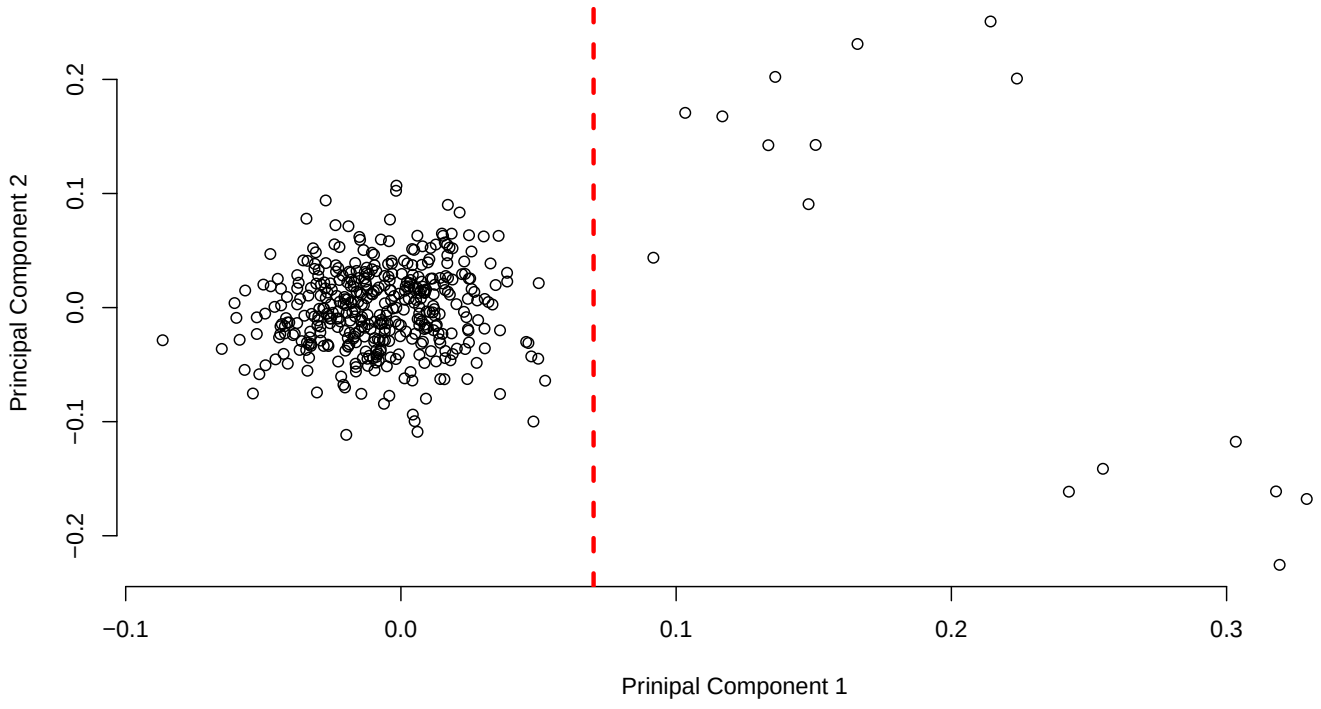
The axes and eigenvalues are stored in `prune.vect` and `prune.values`

Note that as we are only interested in broad sources of genetic variation, it suffices to prune (meaning we don't have to worry about SNP weightings)

Standard Approach - Principal Component Analysis

Having computed the PCAs, we can plot and identify outliers

```
datapca=read.table("prune.vect")
plot(datapca[,3:4],xlab="PC 1",ylab="PC 2")
abline(v=.07,col=2,lwd=3,lty=2)
```



Population Projections from Reference Panel

```
#get overlap between human and hapmap, then thin hapmap
awk '(NR==FNR){arr[$2];next}($2 in arr){print $2}' \
human.bim hapmap.bim > overlap.txt
./ldak5.XXX --thin thin --bfile hapmap \
--window-prune .2 --window-kb 1000 --extract overlap.txt

#compute kinships, pcas and loadings
./ldak5.XXX --calc-kins-direct hapmap --bfile hapmap \
--extract thin.in --ignore-weights YES --power -1
./ldak5.XXX --pca hapmap --grm hapmap
./ldak5.XXX --calc-pca-loads hapmap --bfile hapmap \
--grm hapmap --pcastem hapmap

#project human onto these loadings
./ldak5.XXX --calc-scores project --bfile human \
--scorefile hapmap.load --allow-flips YES --power 0
```

Tip - remove ambiguous SNPs (A & T or C & G) to avoid strand errors

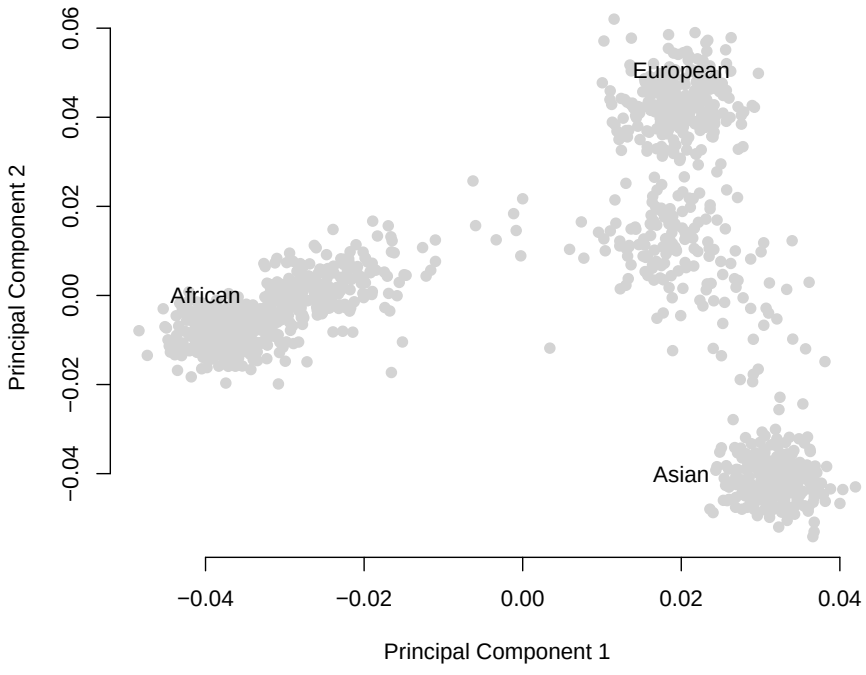
Quality Control - Ethnic Outliers

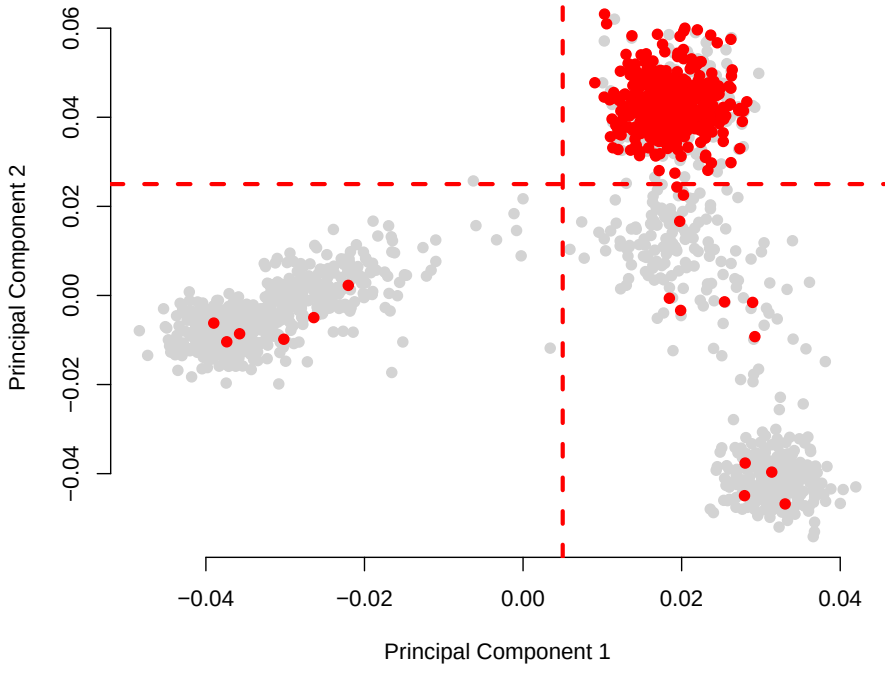
Plot the HapMap PCs, project onto these our data, then identify outliers

Save a list of individuals to keep to goodpop.keep

```
happca=read.table("hapmap.vect")
proj=read.table("project.profile",head=T)

par(mfrow=c(1,2));
plot(happca[,3:4],col="light grey",pch=19,xlab="Proj 1",ylab="Proj 2")
text(c(-.04,.02,.02),c(0,.05,-.04),c("African","European","Asian"))
plot(happca[,3:4],col="light grey",pch=19,xlab="Proj 1",ylab="Proj 2")
points(proj[,c(5,7)],col=2,pch=19)

abline(v=.005,col=2,lwd=3,lty=2)
abline(h=.025,col=2,lwd=3,lty=2)

keep=which(proj[,5]>.005&proj[,7]>.025)
#write.table(proj[keep,1:2],"goodpop.keep",row=F,col=F,quote=F)
```

Quality Control - Ethnic Outliers



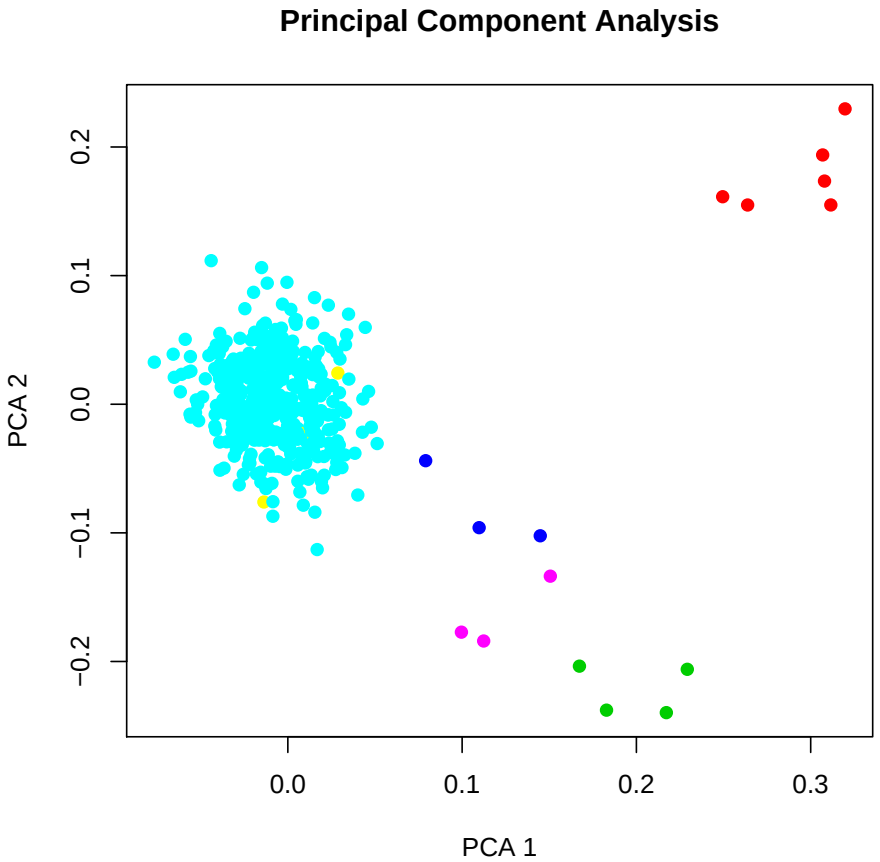


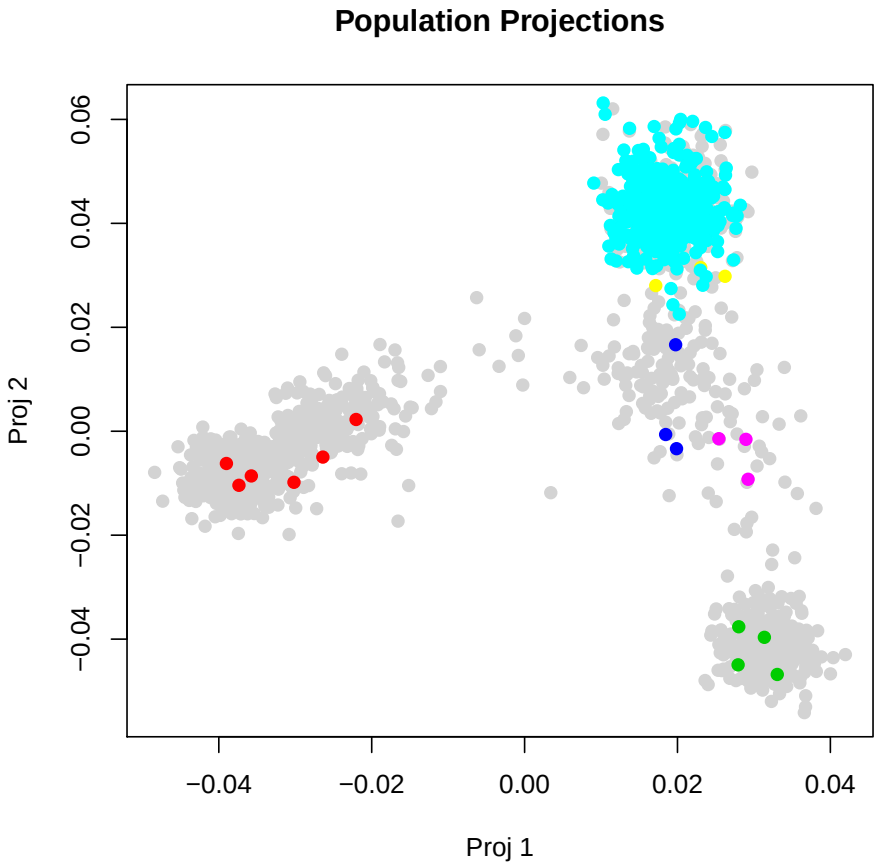
Note that when using a "proper" number of individuals, the clusters will be much tighter. Also, for convenience, here I used HapMap (ftp://ftp.ncbi.nlm.nih.gov/hapmap/genotypes/2009-01_phaseIII/plink_format), but better to use 1000 Genomes Project (see www.ldak.org/reference-panel)

Why do I prefer Population Projections?

```
datapca=read.table("prune.vect")
happca=read.table("hapmap.vect")
proj=read.table("project.profile",head=T)

sites=rep(1,nrow(proj))
a=grep("AFR",proj[,2]);sites[a]=2
a=grep("EAS",proj[,2]);sites[a]=3;
a=grep("SAS",proj[,2]);sites[a]=4
a=grep("EUR",proj[,2]);sites[a]=5;
a=grep("AMR",proj[,2]);sites[a]=6
a=grep("FIN",proj[,2]);sites[a]=7;

par(mfrow=c(1,2))
plot(datapca[,3:4],col=sites,pch=19,xlab="PCA 1",ylab="PCA 2")
plot(happca[,3:4],col="light grey",pch=19,xlab="Proj 1",ylab="Proj 2")
points(proj[,c(5,7)],col=sites,pch=19)
```

Why do I prefer Population Projections?

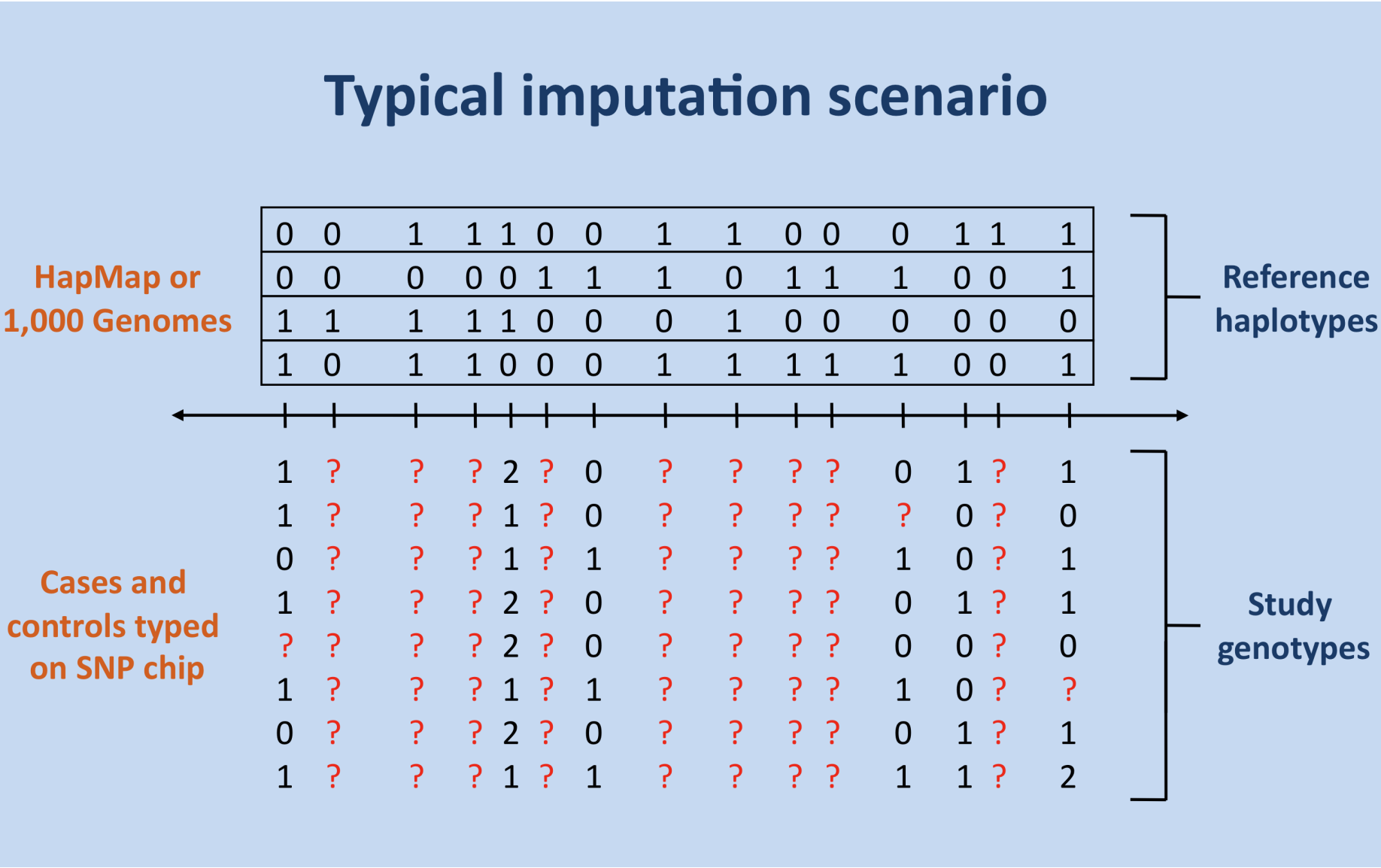




Imputation

We have now identified a set of homogenous, well-genotyped individuals; now is the time to impute. Note that imputation is not just useful for increasing coverage, but also a very effective form of quality control.

Imputation


Typical imputation scenario
HapMap or 1,000 Genomes
0 0 1 1 1 0 0 1 1 0 0 0 1 1 1
0 0 0 0 0 1 1 1 0 1 1 1 0 0 1
1 1 1 1 1 0 0 0 1 0 0 0 0 0 0
1 0 1 1 0 0 0 1 1 1 1 1 0 0 1
Reference haplotypes
Cases and controls typed on SNP chip
1 ? ? ? 2 ? 0 ? ? ? ? 0 1 ? 1
1 ? ? ? 1 ? 0 ? ? ? ? ? 0 ? 0
0 ? ? ? 1 ? 1 ? ? ? ? 1 0 ? 1
1 ? ? ? 2 ? 0 ? ? ? ? 0 1 ? 1
? ? ? ? 2 ? 0 ? ? ? ? 0 0 ? 0
1 ? ? ? 1 ? 1 ? ? ? ? 1 0 ? ?
0 ? ? ? 2 ? 0 ? ? ? ? 0 1 ? 1
1 ? ? ? 1 ? 1 ? ? ? ? 1 1 ? 2
Study genotypes

Imputation

We have now identified a set of homogenous, well-genotyped individuals; now is the time to impute. Note that imputation is not just useful for increasing coverage, but also a very effective form of quality control.

Full scripts for self-imputation (using SHAPEIT and IMPUTE2) are provided at www.ldak.org/protocol. However, this is a laborious and computationally intensive process so if you are allowed, it is much easier and more accurate to use an imputation server (e.g., Sanger or Michigan).

After imputation, you should re-perform SNP quality control. For heritability analysis, I would by default exclude SNPs which (in any cohort) have info score <0.99 (both for imputed and genotyped SNPs) or (expected) MAF <0.01

You can use dosages, but if using strict QC, easier to convert to hard genotypes.

Covariates

We want to create a covariate file to use in subsequent regressions.

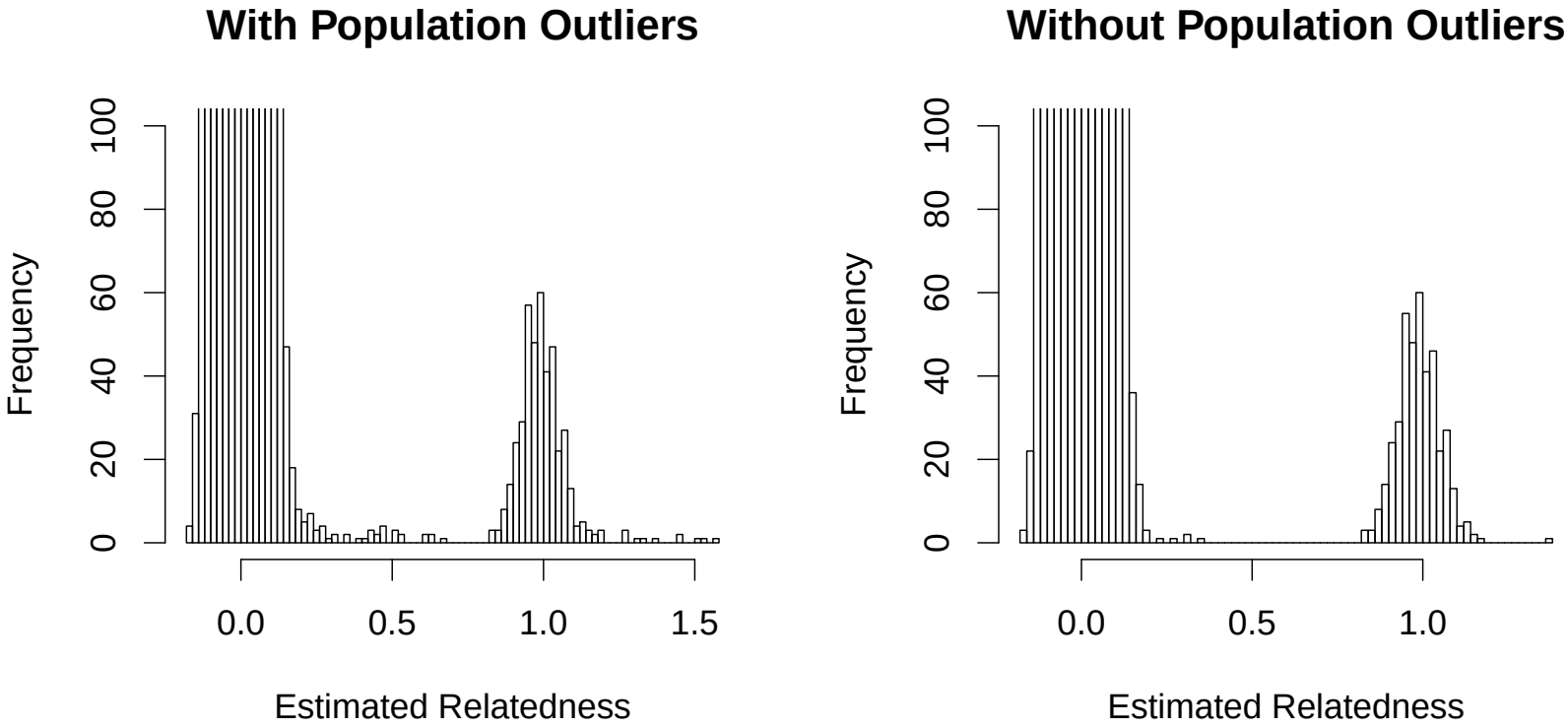
I generally include as covariates sex plus 10 principal components (5 from in-sample PC, 5 from population projections).

Note that usually I would repeat the PCA using the imputed data

```
#can get sex from Column 5 of the fam file
fam=read.table("human.fam")

#use match() to ensure samples are alighned
m1=match(fam[,1],datapca[,1])
m2=match(fam[,1],proj[,1])

covar=cbind(fam[,c(1,2,5)],datapca[m1,3:7],proj[m2,c(5,7,9,11,13)])
#write.table(covar,"covar.covar",row=F,col=F,quote=F)
```

Quality Control - Remove Relatedness

To estimate relatedness we can use the kinship matrix computed when performing PCA (i.e., from thinned SNPs). Make sure to first remove ethnic outliers



Notice that allelic correlation kinships are distributed around expected values (e.g., 1/2 for full-sibs, 0 for "unrelateds"), and there are values above 1 and below 0. (For proper sample sizes, spread will be smaller)

Quality Control - Remove Relatedness

The general assumption is if we find identical individuals, these are accidental duplicates, so we want to remove one of each pair. We can do this using filter:

```
./ldak5.XXX --filter dups --grm prune --keep goodpop.keep --max-rel .8
```

(Of course, duplicates may be twins, so check clinical data if possible)

If our aim is single-SNP analysis, we normally want to remove "close relatedness"; a typical cut-off is 0.185, half-way between half-sibs and cousins:

```
./ldak5.XXX --filter close --grm prune --keep goodpop.keep --max-rel .375
```

(This is not necessary if performing mixed-model association analysis)

Note that imputation benefits from having relatedness

Quality Control - Remove Relatedness

For heritability analyses, we generally require unrelated individuals, so we filter based on estimated relatedness

As a threshold, can use an arbitrary value (say 0.025 or 0.05) or second cousins (1/32=0.03125), but I prefer to use the negative of the smallest value observed (the default in LDAK)

```
./ldak5.XXX --filter filter --grm prune --keep goodpop.keep
```

In this (unrealistic example), the minimum kinship is -0.176; using this threshold results in removal of 10 individuals, so that 396 remain (stored in `filter.keep`)

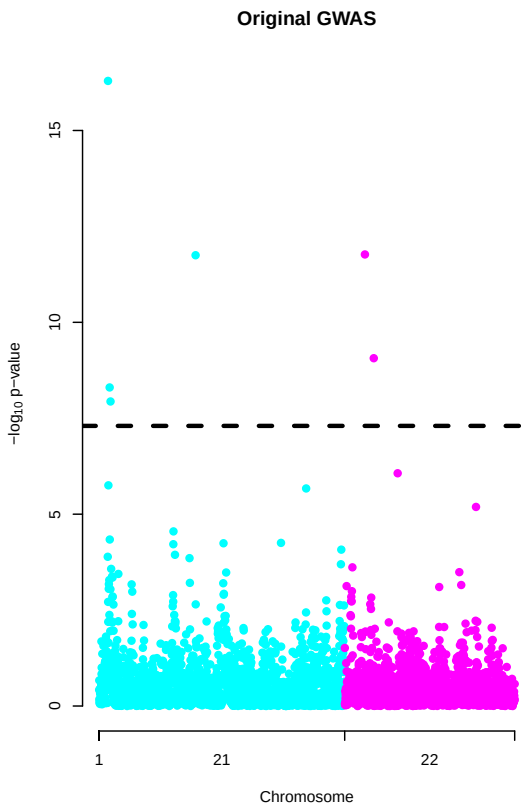
Tip - if you have missing phenotypic values, then add `--pheno` to ensure individuals with missing values are removed first

Conditional association analysis

Earlier, we performed single-SNP analysis using the command

```
./ldak5.XXX --linear linear --pheno quant.pheno --bfile human
```

(could now add `--covar covar.covar` to include covariates)



We now wish to condition on the most strongly associated SNP (rs389338)

Conditional association analysis

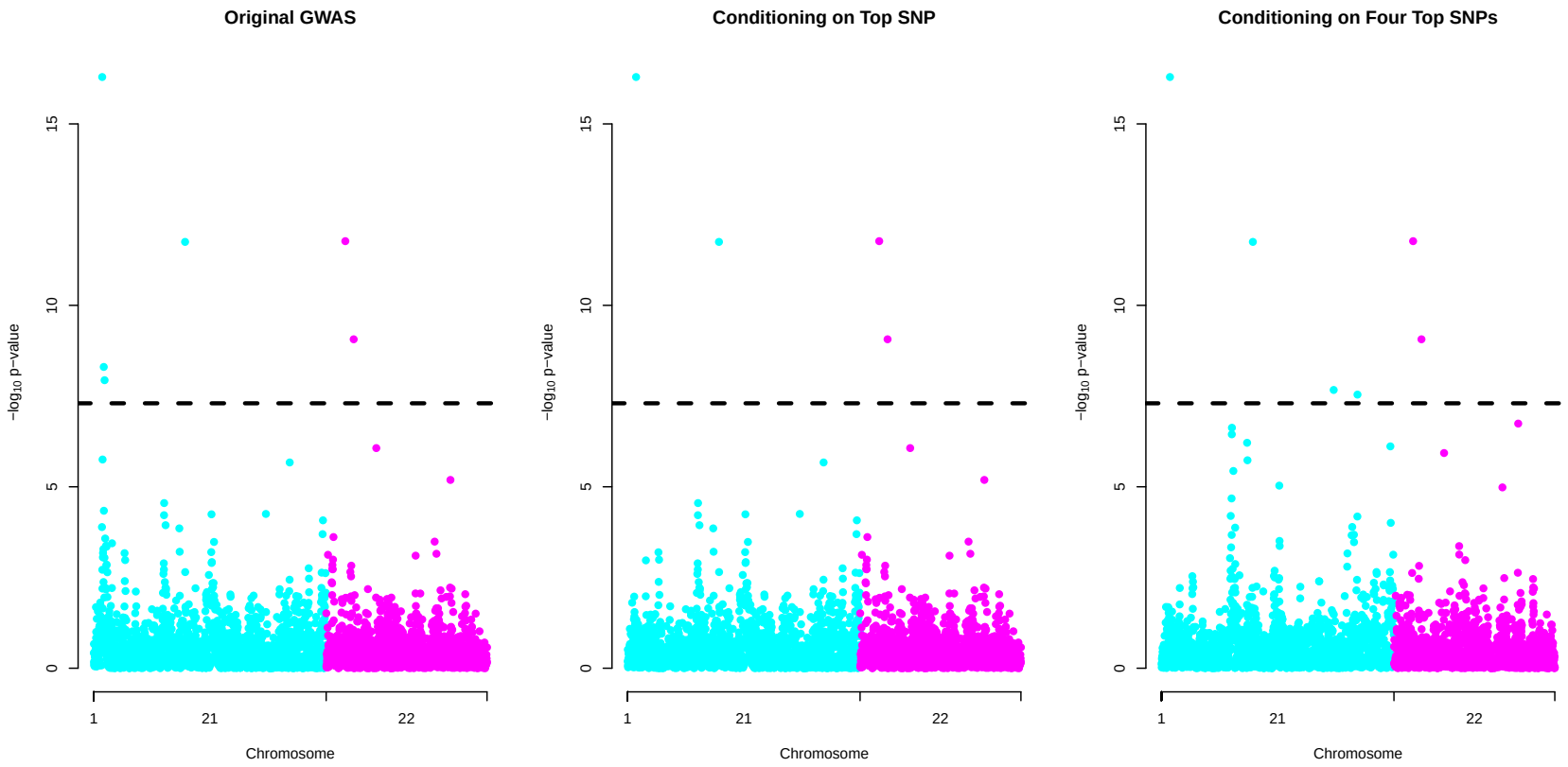
Save this SNP to a file, then add the option `--top-preds`:

```
echo "rs389338" > top.snps
./ldak5.XXX --linear linear_cond --pheno quant.pheno --bfile human \
--top-preds top.snps
```

This will include the top SNP as a (fixed-effect) covariate in the regression (this is equivalent to making a covariate file containing the SNP values, then using `--covar` instead of `--top-preds`)

Previously, we searched for independent loci by pruning; conditional analysis is a more effective (but slower) way to do this

Conditional association analysis



We see that when we condition on rs389338, the two nearby SNPs are no longer significant. Could then also condition on the next top SNP (by adding to the file top.snps), and so on

Mixed-model association analysis

The standard linear model for testing SNP j is $Y = \alpha + \beta_j X_j$

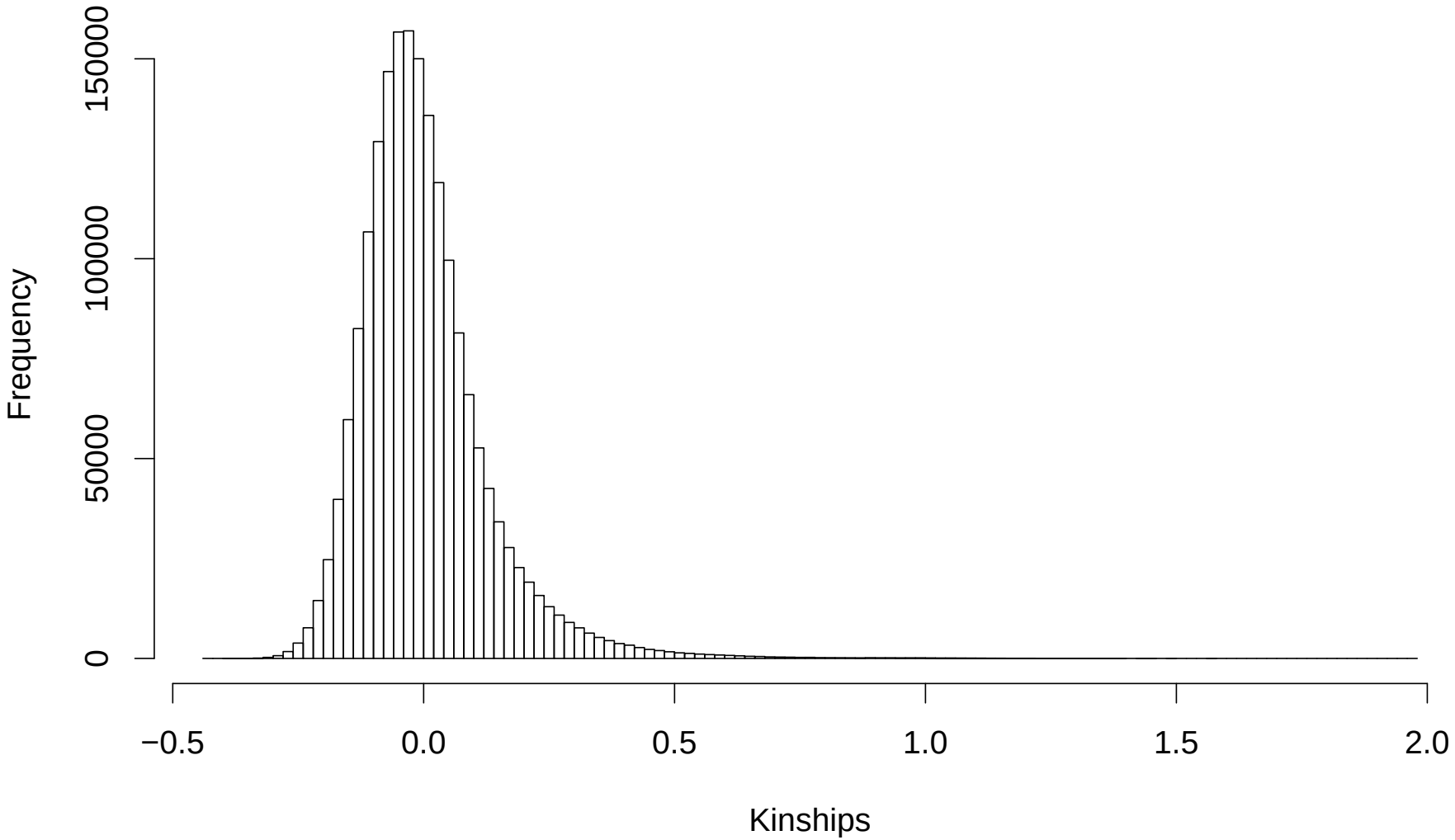
Mixed-model linear regression extends this to $Y = \alpha + \beta_j X_j + g$
where $g_i \sim \mathbb{N}(0, K\sigma_g^2)$ and K is a kinship matrix

The original idea was that including g allows for long-range correlations due to relatedness and population structure. But recently, it has been realised that g will also capture the contribution of other causal variants, so in effect we are testing SNP j using a multi-SNP model.

Mixed-model association analysis is also implemented in EMMA, FastLMM, GEMMA and GCTA.

Mixed-model association analysis

The mice data contain large amounts of relatedness (all individuals are derived from just 8 founders)

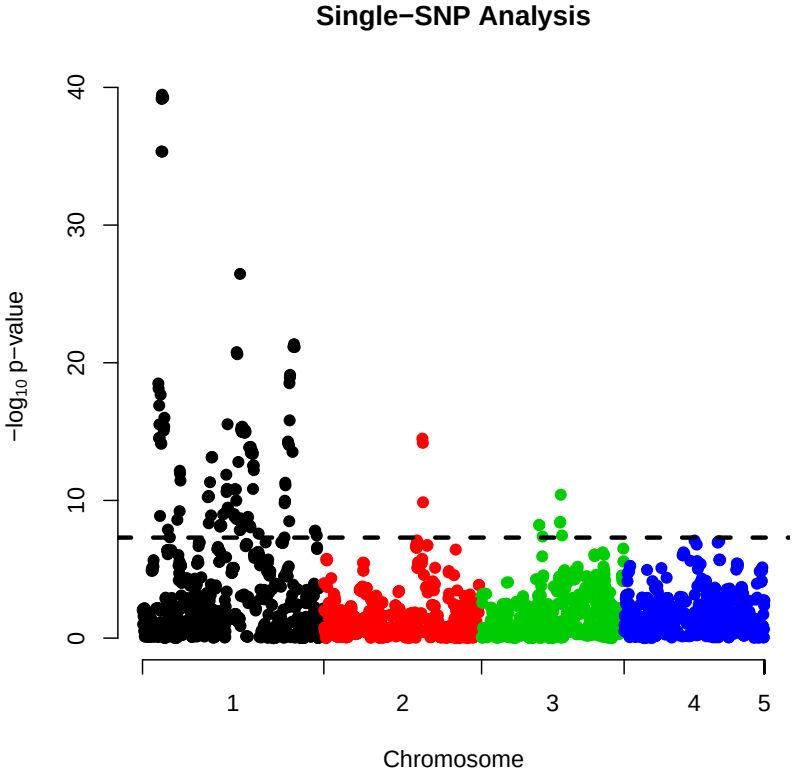


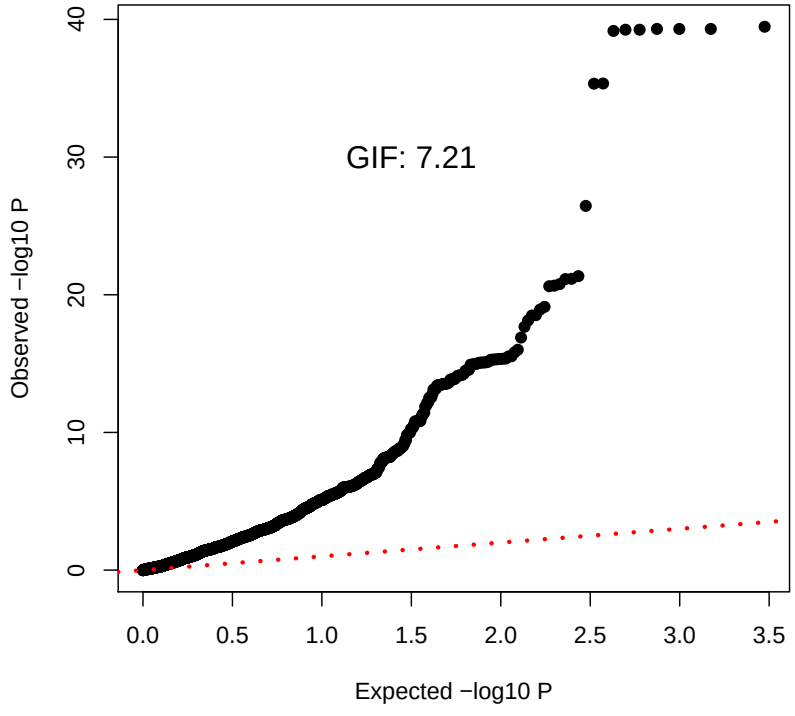
Mixed-model association analysis

The mice data contain large amounts of relatedness (all individuals are derived from just 8 founders)

Standard single-SNP analysis will give near meaningless results

```
./ldak5.XXX --linear stand --pheno mice.pheno --bfile mice
```

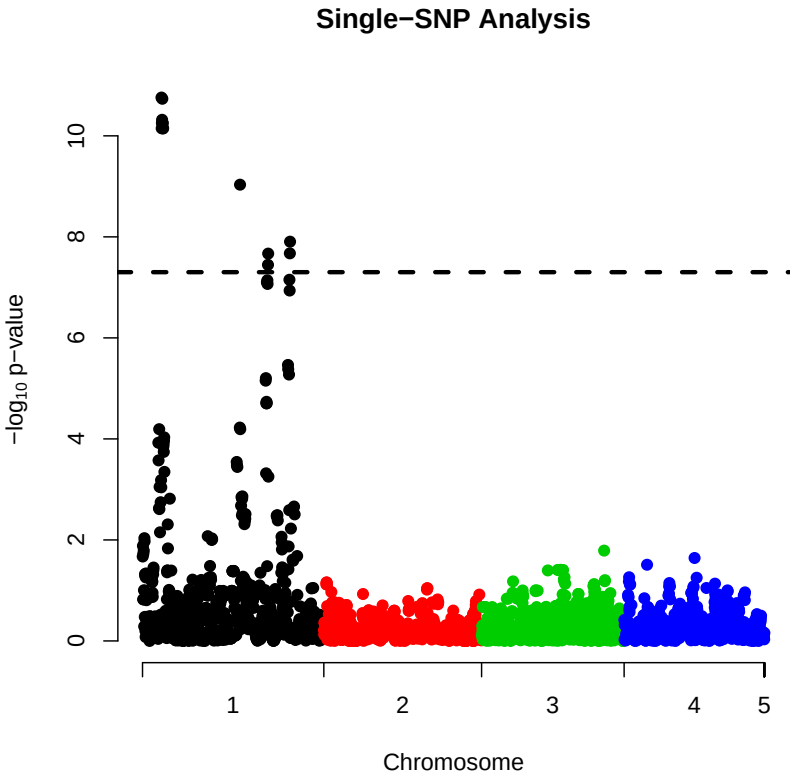


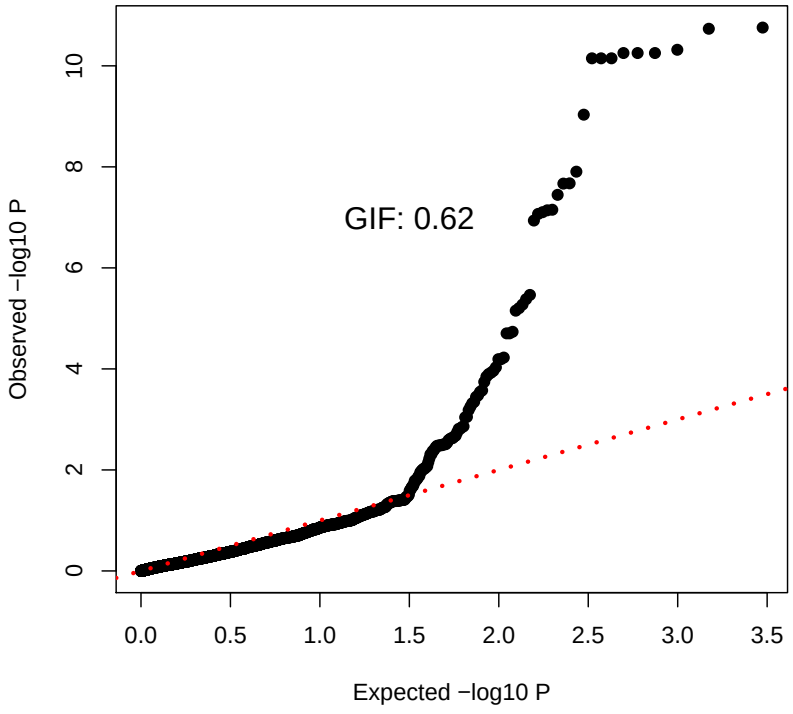


Mixed-model association analysis

However, the results are dramatically improved by adding a kinship matrix

```
./ldak5.XXX --calc-kins-direct kinsm --bfile mice \
--ignore-weights YES --power -1
./ldak5.XXX --linear mm --pheno mice.pheno --bfile mice --grm kinsm
```





Estimating SNP Heritability

To estimate (total) SNP Heritability, we assume:

$$
\begin{aligned}
Y = \alpha & \\
& + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 \\
& + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} \\
& + \beta_{15} X_{15} + \beta_{16} X_{16} + \beta_{17} X_{17} + \beta_{18} X_{18} + \beta_{19} X_{19} + \beta_{20} X_{20} + \beta_{21} X_{21} \\
& + \beta_{22} X_{22} + \beta_{23} X_{23} + \beta_{24} X_{24} + \beta_{25} X_{25} + \beta_{26} X_{26} + \beta_{27} X_{27} + \beta_{28} X_{28} \\
& + \ldots + \beta_m X_m \\
& + e,
\end{aligned}
$$

with $\beta_j \sim \mathbb{N}(0, w_j\ [f_j(1-f_j)]^{-0.25}\sigma_g^2/W)$ and $e \sim \mathbb{N}(0, \sigma_e^2)$

Then $Y \sim \mathbb{N}(\alpha, K\sigma_g^2 + I\sigma_e^2)$, where $K = \frac{XX^T}{W}$

(w_j are SNP weights, f_j frequencies and $W = \sum w_j\ [f_j(1-f_j)]^{-0.25}$)

Estimating SNP Heritability from raw data

To estimate the total proportion of phenotypic variance explained by (common) SNP genotypes, there are three steps:

A - Calculate SNP weights (the w_j)

B - Calculate a weighted kinship matrix (K)

C - Estimate σ_g^2 and σ_e^2 via REML or Haseman Elston Regression

A - Calculate SNP Weights

The simple way to do this is using `--cut-weights` and `--calc-weights-all`:

```
./ldak5.XXX --bfile human --cut-weights sections \
--keep goodpop.keep
./ldak5.XXX --bfile human --calc-weights-all sections \
--keep goodpop.keep
```

Normally, this will finish within a few hours. But if you prefer you can parallelize (note your loop will normally be from 1 to 22)

```
for j in {21..22}; do
./ldak5.XXX --bfile human --cut-weights sections$j \
--keep goodpop.keep --chr $j
./ldak5.XXX --bfile human --calc-weights-all sections$j \
--keep goodpop.keep --chr $j
done

cat sections{21..22}/weights.short > weights.merge
```

B - Calculate a Weighted Kinship Matrix

For this, we use `--calc-kins-direct`:

```
./ldak5.XXX --calc-kins-direct kins --bfile human \
--weights sections/weights.all --power -.25 --kinship-raw YES
```

`--power` corresponds to the parameter α in the slides

(Note - normally no need for `--kinship-raw YES`)

Again, this will typically finish within a few hours, but can also be parallelized (see later)

To create a GCTA kinship matrix, use:

```
./ldak5.XXX --calc-kins-direct kins --bfile human \
--ignore-weights YES --power -1
```

C - Estimate variance components

First we consider the phenotype `quant.pheno` and use REML

```
./ldak5.XXX --reml quant --grm kins \
--pheno quant.pheno --keep filter.keep
```

Remember to include `--keep` so that we only use unrelated individuals. The main output file is `quant.reml`, but there are many additional files

Alternatively, we can use Haseman Elston regression

```
./ldak5.XXX --he quant_he --grm kins \
--pheno quant.pheno --keep filter.keep
```

As the phenotype is quantitative, we would expect the two estimates of heritability to match (here they are 0.74 and 0.72)

Note, large-effect SNPs should be included as top predictors (www.ldak.org/protocol)

Multiple Phenotypes

The file `multi.pheno` contains 10 traits. We could analyze using

```
for m in {1..10}; do
./ldak5.XXX --reml multi$m --grm kins \
--pheno multi.pheno --keep filter.keep --mpheno $m
done
```

However, it is more efficient (and simpler) to use

```
./ldak5.XXX --reml multi --grm kins \
--pheno multi.pheno --keep filter.keep --mpheno -1
```

... and even more efficient to use

```
./ldak5.XXX --decompose kins --grm kins --keep filter.keep
```

```
./ldak5.XXX --reml multi --grm kins --eigen kins \
--pheno multi.pheno --keep filter.keep --mpheno -1
```

This is particularly useful when testing gene expressions (20 000+ traits)

Binary traits

`binary.pheno` contains a case-control phenotype. We can analyse this exactly the same as if it were a quantitative trait, in which case our heritability estimates are on the "oberved scale"

```
./ldak5.XXX --reml binary --grm kins \
--pheno binary.pheno --keep filter.keep
```

However, generally it is preferable to convert to the "liability scale" (see next slide), as then we can directly compare estimates across differently-ascertained studies. To do this, we add `--prevalence`:

```
./ldak5.XXX --reml binary --grm kins \
--pheno binary.pheno --keep filter.keep --prevalence 0.5
```

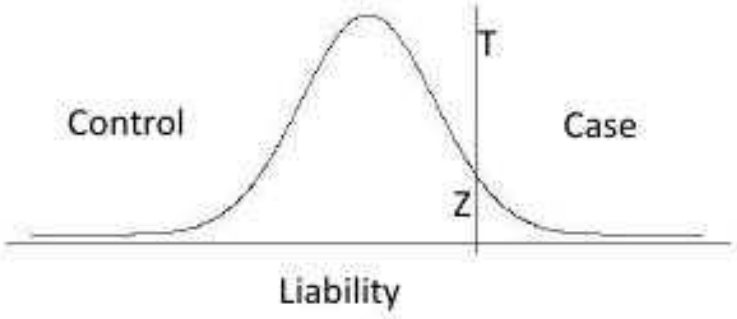
Tip - usually the prevalence is not known with certainty, so it is good practice to consider a few (perhaps double and half your best-guess?)

Liability Model for binary traits

We suppose each individual has a liability, L (which we can not observe)

We only observe whether L above or below a threshold T

T determined by disease prevalence π



If we knew L, we could perform REML with L (instead of Y) and get an estimate of the heritability on the liability scale directly

But with L unknown, we instead analyse the phenotype pretending it is continuous, then use the following transformation:

$$h^2_{Liab} = h^2_{SNP} \frac{\pi^2(1-\pi)^2}{P(1-P)z^2}$$

where π is the prevalence, P the ascertainment, and z is the "height of the standard normal distribution" at the liability threshold, T

Haseman Elston Regression for binary traits

Golan, Lander and Rosset (PNAS 2014) argue that Haseman Elston should be used for binary traits. Their suggestion has not been widely adopted

One reason is that while straightforward to include covariates in REML:

```
./ldak5.XXX --reml binary2 --grm kins --covar covar.covar \
--pheno binary.pheno --keep filter.keep --prevalence 0.5
```

with HE you must first "regress the covariates out of the kinships"

```
./ldak5.XXX --adjust-grm kins2 --grm kins --covar covar.covar
./ldak5.XXX --he binary2_he --grm kins2 --covar covar.covar \
--pheno binary.pheno --keep filter.keep --prevalence 0.5 \
--kinship-details NO
```

As many (most) of the interesting traits are binary diseases, this is an important issue. Additionally, HE is computationally efficient, whereas REML struggles for over 50 000 samples

Genomic Partitioning

We now use the model

$$
\begin{aligned}
Y = \alpha & \\
& + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 \\
& + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} \\
& + \beta_{15} X_{15} + \beta_{16} X_{16} + \beta_{17} X_{17} + \beta_{18} X_{18} + \beta_{19} X_{19} + \beta_{20} X_{20} + \beta_{21} X_{21} \\
& + \beta_{22} X_{22} + \beta_{23} X_{23} + \beta_{24} X_{24} + \beta_{25} X_{25} + \beta_{26} X_{26} + \beta_{27} X_{27} + \beta_{28} X_{28} \\
& + \ldots + \beta_{500\,000} X_{500\,000} \\
& + e
\end{aligned}
$$

with $\beta_j \sim \mathbb{N}(0, w_j p_j \sigma^2_{g1} / W_1)$ and $\beta_j \sim \mathbb{N}(0, w_J p_j \sigma^2_{g2} / W_2)$

Then $Y \sim \mathbb{N}(\alpha, K_1 \sigma^2_{g1} + K_2 \sigma^2_{g2} + I \sigma^2_e)$, where $K_1 = \frac{X_1 X_1^T}{W_1}$ and $K_2 = \frac{X_2 X_2^T}{W_2}$

Instead of estimating only total h^2_{SNP}, we can now estimate h^2 of RED SNPs and h^2 of BLUE SNPs

Genomic Partitioning

Instead of one genome-wide kinship matrix, we need one kinship for each SNP subset. E.g., we can divide h^2_{SNP} by chromosome:

```
./ldak5.XXX --calc-kins-direct chr21 --bfile human \
--weights sections/weights.all --power -.25 --chr 21

./ldak5.XXX --calc-kins-direct chr22 --bfile human \
--weights sections/weights.all --power -.25 --chr 22
```

Now we do multi-grm REML (or HE) using `--mgrm` instead of `--grm`

```
echo "chr21
chr22" > mlist.txt
./ldak5.XXX --reml split --mgrm mlist.txt --covar covar.covar \
--pheno quant.pheno --keep filter.keep

tail -n 5 split.reml
```

Genomic Partitioning

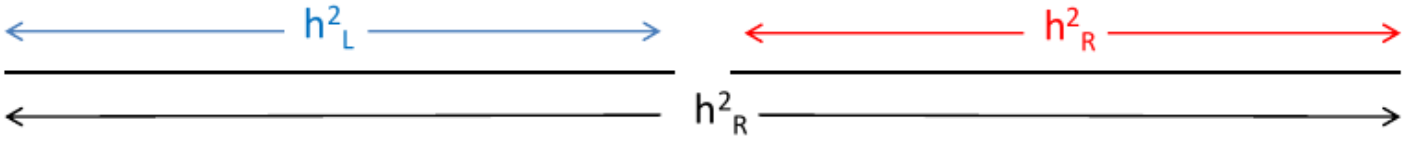
It is also an effective way to test for inflation due to population structure

Divide the SNPs X into left half X_1 and right half X_2
(e.g. X_1 contains SNPs in Chr 1-8; X_2 contains those in Chr 9-22)

Estimate total h^2 mixed model with $K = XX^T / N$

Estimate left half h^2_L mixed model with $K_1 = X_1X_1^T / N_1$

Estimate right half h^2_R mixed model with $K_2 = X_2X_2^T / N_2$



If cryptic relatedness is not inflating estimates, then would expect $h^2 = h^2_L + h^2_R$

In which case, estimates of $h^2_L + h^2_R$ from the joint model $Y \sim N(\alpha, K\sigma_g^2 + K\sigma_g^2 + I\sigma_e^2)$

should match those from individual models
$Y \sim N(\alpha, K\sigma_g^2 + I\sigma_e^2)$ and $Y \sim N(\alpha, K\sigma_g^2 + I\sigma_e^2)$

Genomic Partitioning

So in our example (treating Chr 21 as the "left" and Chr 22 as the "right"):

```
./ldak5.XXX --reml chr21 --grm chr21 \
--covar covar.covar --pheno quant.pheno --keep filter.keep

./ldak5.XXX --reml chr22 --grm chr22 \
--covar covar.covar --pheno quant.pheno --keep filter.keep

grep Her_A {quant,chr21,chr22}.reml
quant.reml:Her_ALL 0.587314 0.093651 3325.66 176.600875 28.160064
chr21.reml:Her_ALL 0.386968 0.086060 230.32 1680.134404 373.652549
chr22.reml:Her_ALL 0.406302 0.083042 179.10 2268.608374 463.669627
```

So here, $40.6 + 38.9 = 79.2 > 58.7$, (which if the SDs were smaller) would indicate inflation (can do same analysis with HE)

Genomic Partitioning

Suppose now we wish to divide the genome into 3 subsets. Let the files part1, part2, part3 detail the SNPs in each partition:

```
./ldak5.XXX --cut-kins parts --partition-prefix part \
--partition-number 3 --bfile human

for j in {1..3}; do
./ldak5.XXX --calc-kins parts --partition $j \
--bfile human --weights sections/weights.all --power -.25
done

./ldak5.XXX --reml divide --mgrm parts/partition.list \
--covar covar.covar --pheno quant.pheno --keep filter.keep
```

Tip - when performing REML (or HE) with very many kinship matrices, consider adding `--mem-save YES`; this will read kinships on-the-fly, which is far more memory-efficient

Genomic Partitioning

A similar approach can be used to parallelize when calculating kinships for large data. First divide the genome into partitions:

```
./ldak5.XXX --cut-kins parts2 --bfile human --by-chr YES
```

Could replace `--by-chr YES` with (say) `--partition-length 1000000`

Then compute kinships for each partition

```
for j in {1..2}; do
./ldak5.XXX --calc-kins parts2 --partition $j \
--bfile human --weights sections/weights.all --power -.25
done
```

Finally, join up the kinships

```
./ldak5.XXX --join-kins parts2
```

Genomic Partitioning

We now use the model

$$
\begin{aligned}
Y = \alpha & \\
& + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 \\
& + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} + \beta_{11} X_{11} + \beta_{12} X_{12} + \beta_{13} X_{13} + \beta_{14} X_{14} \\
& + \beta_{15} X_{15} + \beta_{16} X_{16} + \beta_{17} X_{17} + \beta_{18} X_{18} + \beta_{19} X_{19} + \beta_{20} X_{20} + \beta_{21} X_{21} \\
& + \beta_{22} X_{22} + \beta_{23} X_{23} + \beta_{24} X_{24} + \beta_{25} X_{25} + \beta_{26} X_{26} + \beta_{27} X_{27} + \beta_{28} X_{28} \\
& + \ldots + \beta_{500\,000} X_{500\,000} \\
& + e
\end{aligned}
$$

with $\beta_j \sim \mathbb{N}(0, w_j p_j \sigma^2_{g1} / W_1)$ and $\beta_j \sim \mathbb{N}(0, w_J p_j \sigma^2_{g2} / W_2)$

Then $Y \sim \mathbb{N}(\alpha, K_1 \sigma^2_{g1} + K_2 \sigma^2_{g2} + I \sigma^2_e)$, where $K_1 = \frac{X_1 X_1^T}{W_1}$ and $K_2 = \frac{X_2 X_2^T}{W_2}$

Instead of estimating only total h^2_{SNP}, we can now estimate h^2 of RED SNPs and h^2 of BLUE SNPs

Genomic Partitioning

Instead of one genome-wide kinship matrix, we need one kinship for each SNP subset. E.g., we can divide h^2_{SNP} by chromosome:

```
./ldak5.XXX --calc-kins-direct chr21 --bfile human \
--weights sections/weights.all --power -.25 --chr 21

./ldak5.XXX --calc-kins-direct chr22 --bfile human \
--weights sections/weights.all --power -.25 --chr 22
```

Now we do multi-grm REML (or HE) using `--mgrm` instead of `--grm`

```
echo "chr21
chr22" > mlist.txt
./ldak5.XXX --reml split --mgrm mlist.txt --covar covar.covar \
--pheno quant.pheno --keep filter.keep

tail -n 5 split.reml
```

Genomic Partitioning

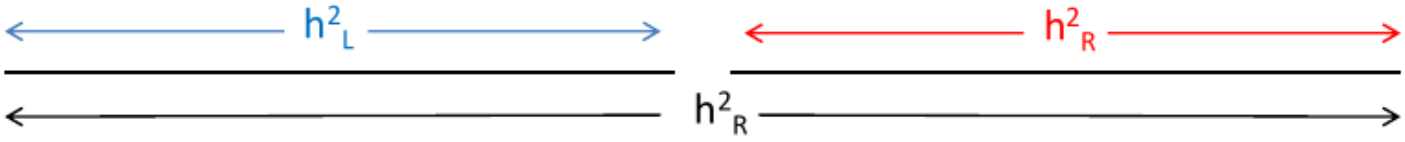
It is also an effective way to test for inflation due to population structure

Divide the SNPs X into left half X_1 and right half X_2
(e.g. X_1 contains SNPs in Chr 1-8; X_2 contains those in Chr 9-22)

Estimate total h^2 mixed model with $K = XX^T / N$

Estimate left half h^2_L mixed model with $K_1 = X_1X_1^T / N_1$

Estimate right half h^2_R mixed model with $K_2 = X_2X_2^T / N_2$



If cryptic relatedness is not inflating estimates, then would expect $h^2 = h^2_L + h^2_R$

In which case, estimates of $h^2_L + h^2_R$ from the joint model $Y \sim N(\alpha, K\sigma_g^2 + K\sigma_g^2 + I\sigma_e^2)$

should match those from individual models
$Y \sim N(\alpha, K\sigma_g^2 + I\sigma_e^2)$ and $Y \sim N(\alpha, K\sigma_g^2 + I\sigma_e^2)$

Genomic Partitioning

So in our example (treating Chr 21 as the "left" and Chr 22 as the "right"):

```
./ldak5.XXX --reml chr21 --grm chr21 \
--covar covar.covar --pheno quant.pheno --keep filter.keep

./ldak5.XXX --reml chr22 --grm chr22 \
--covar covar.covar --pheno quant.pheno --keep filter.keep

grep Her_A {quant,chr21,chr22}.reml
quant.reml:Her_ALL 0.722953 0.087979 409.42 1765.811077 214.889148
chr21.reml:Her_ALL 0.376547 0.086172 230.32 1634.888996 374.140442
chr22.reml:Her_ALL 0.388731 0.082445 179.10 2170.501026 460.337952
```

So here, $37.6 + 38.9 = 76.5 > 72.2$, (which if the SDs were smaller) would suggest slight inflation (can do same analysis with HE)

Genomic Partitioning

Suppose now we wish to divide the genome into 3 subsets. Let the files part1, part2, part3 detail the SNPs in each partition:

```
./ldak5.XXX --cut-kins parts --partition-prefix part \
--partition-number 3 --bfile human

for j in {1..3}; do
./ldak5.XXX --calc-kins parts --partition $j \
--bfile human --weights sections/weights.all --power -.25
done

./ldak5.XXX --reml divide --mgrm parts/partition.list \
--covar covar.covar --pheno quant.pheno --keep filter.keep
```

Tip - when performing REML (or HE) with very many kinship matrices, consider adding `--mem-save YES`; this will read kinships on-the-fly, which is far more memory-efficient

Genomic Partitioning

A similar approach can be used to parallelize when calculating kinships for large data. First divide the genome into partitions:

```
./ldak5.XXX --cut-kins parts2 --bfile human --by-chr YES
```

Could replace `--by-chr YES` with (say) `--partition-length 1000000`

Then compute kinships for each partition

```
for j in {1..2}; do
./ldak5.XXX --calc-kins parts2 --partition $j \
--bfile human --weights sections/weights.all --power -.25
done
```

Finally, join up the kinships

```
./ldak5.XXX --join-kins parts2
```

Estimating SNP Heritability from summary statistics

To estimate the total proportion of phenotypic variance explained by (common) SNP genotypes, there are three steps:

A - Calculate SNP weights (from reference panel)

B - Calculate a tagging file (from reference panel))

C - Estimate variance components using `--sum-her`

Estimating SNP heritability using SumHer

First you must compute a tagfile, which provides the expected heritability tagged by each SNP (under the chosen heritability model).

If we assume the GCTA Model (each SNP expected to contribute one unit of heritability), then SNP j is expected to tag $\sum_k r_{j,k}^2$ units, where the sum is over all SNPs within (say) 1 cM (or 1 000 kb):

```
./ldak5.XXX --calc-tagging tag1 --bfile human --window-cm 1 \
--ignore-weights YES --power -1
```

```
head tag1.tagging
Predictor A1 A2 Neighbours Tagging Weight MAF Info Exp_Heritability
rs2260895 A G 76 6.171 1.0000 0.442217 1.0000 1.0000 6.1709
rs2821796 A C 76 6.215 1.0000 0.487028 1.0000 1.0000 6.2147
rs1148737 G A 78 6.823 1.0000 0.484670 1.0000 1.0000 6.8233
rs1296972 T C 79 6.943 1.0000 0.491745 1.0000 1.0000 6.9434
```

Note: normally you would compute tagging from an INDEPENDENT well-matched reference panel (for convenience, here we use the same data)

Estimating SNP heritability using SumHer

First you must compute a tagfile, which provides the expected heritability tagged by each SNP (under the chosen heritability model).

Under the LDAK Model, the expected heritability tagged by SNP j is $\sum_k w_j \ [f_j(1-f_j)]^{-0.25} r_{j,k}^2$:

```
./ldak5.XXX --calc-tagging tag2 --bfile human --window-cm 1 \
--weights sections/weights.all --power -.25
```

Calculating tagging under the GCTA Model can take very long, so you can divide by chromosome

```
./ldak5.XXX --calc-tagging tag3.21 --bfile human --window-cm 1 \
--ignore-weights YES --power -1 --chr 21
./ldak5.XXX --calc-tagging tag3.22 --bfile human --window-cm 1 \
--ignore-weights YES --power -1 --chr 22

echo -e "tag3.21.tagging\ntag3.22.tagging" > join.txt
./ldak5.XXX --join-tagging tag3 --taglist join.txt
```

Estimating SNP heritability using SumHer

Then you estimate SNP heritability using `--sum-hers`:

```
./ldak5.XXX --sum-hers t1 --summary linear.summaries --tagfile tag1.tagging
./ldak5.XXX --sum-hers t2 --summary linear.summaries --tagfile tag2.tagging
```

`t1.hers` and `t2.hers` provide estimates of SNP heritability (in practice, we should exclude large-effect SNPs)

By default, SumHer assumes no confounding. To obtain LDSC estimate, compute taggings under the GCTA Model then add `--intercept YES`:

```
./ldak5.XXX --sum-hers t1b --summary linear.summaries \
--tagfile tag1.tagging --intercept YES
```

But if concerned about confounding, we instead advise adding `--genomic-control YES` (because in our view, genomic control is the biggest form of confounding).

```
./ldak5.XXX --sum-hers t2b --summary linear.summaries \
--tagfile tag2.tagging --genomic-control YES
```

Estimating enrichment using SumHer

When computing the tagfile, we must now calculate the expected contributions from each SNP Subset. Under the LDAK Model, we use:

```
./ldak5.XXX --calc-tagging tag4 --bfile human --window-cm 1 \
--weights sections/weights.all --power -.25 \
--partition-number 3 --partition-prefix part
./ldak5.XXX --sum-hers t4 --summary linear.summaries \
--tagfile tag4.tagging
```

```
cat t4.hers
Component Heritability Her_SD Size Mega_Intensity Int_SD
Her_P1 0.390589 0.124531 74.38 5250.984281 1674.164985
Her_P2 0.207188 0.064762 66.91 3096.461792 967.872297
Her_P3 0.309131 0.129546 99.77 3098.567808 1298.499558
Her_Top 0.000000 NA NA NA NA
Her_ALL 0.906908 0.190654 241.06 3762.152054 790.896633
```

```
cat t4.enrich
Component Share SD Expected Enrichment SD
Enrich_P1 0.430682 0.109371 0.308569 1.395740 0.354447
Enrich_P2 0.228456 0.073840 0.277570 0.823056 0.266021
Enrich_P3 0.340862 0.114937 0.413861 0.823616 0.277718
```

Bivariate Analysis

Trait 1: $Y_1 = \alpha_1 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_N X_N + e_1$
$= \alpha_1 + g_1 + e_1$

Trait 2: $Y_2 = \alpha_2 + \gamma_1 X_1 + \gamma_2 X_2 + \ldots + \gamma_N X_N + e_2$
$= \alpha_2 + g_2 + e_2$

Now interested in the correlation between genetic effects:
$\rho = cor(g_1, g_2)$.

Or equivalently can think of the average correlation between effect sizes:
$\rho = cor(\beta_j, \gamma_j)$

GCTA and Bivariate Analysis

LDAK and GCTA have nearly the same syntax, except in GCTA, the main arguments tend to be "solo", and so it is necessary to use `--out` to specify output files. The command for REML is

```
gcta --reml --out quantgcta --grm kins \
--pheno quant.pheno --keep filter.keep
```

To estimate the genetic correlation between the first two traits in `multi.pheno` use

```
gcta --reml-bivar 1 2 --out bivar --grm kins \
--pheno multi.pheno --keep filter.keep
```

```
grep G bivar.hsq
V(G)_tr1 0.624334 0.128917
V(G)_tr2 0.664430 0.125075
C(G)_tr12 -0.183146 0.089852
V(G)/Vp_tr1 0.589101 0.093452
V(G)/Vp_tr2 0.660432 0.089452
rG -0.284357 0.139200
```

Gene-Based Association Testing (GBAT)

anns.txt contains annotations for genes (on Chromosomes 21 & 22)

```
./ldak5.XXX --cut-genes genes --bfile human \
--genefile anns.txt --ignore-weights YES --pvalues linear.pvalues

./ldak5.XXX --calc-genes-reml genes --bfile human \
--ignore-weights YES --power -.25 --pheno quant.pheno

./ldak5.XXX --join-genes-reml genes

res=read.table("genes/remls.all",head=T)
dets=read.table("genes/genes.details",head=T,skip=2)

plot(-log10(res[,12]),col=res[,2],pch=4,xlab="Gene",
ylab="-log10 P",ylim=c(0,10))
points(-log10(dets[,11]),pch=19)
legend(100,9,leg=c("Single-SNP","Gene-Base"),pch=c(19,4),cex=1.3)
```

Tip - use --gene-buffer and --overlap to change how genes are cut

Prediction

A prediction model is defined as a list of SNPs and effect sizes (referred to by LDAK as a "scorefile"). Prediction methods differ in how they estimate effect sizes

The most basic prediction method is polygenic risk scores; this uses effect sizes estimates from single-SNP analysis

```
Predictor A1 A2 Centre ALL P<0.1 P<0.01 P<0.001 P<0.0001 P<0.00001 I
rs2260895 A G 0.884434 0.061757 0.000000 0.000000 0.000000 0.000000
rs2821796 A C 0.974057 -0.085988 -0.000000 -0.000000 -0.000000 -0.0(
rs1148737 G A 0.969340 0.030648 0.000000 0.000000 0.000000 0.000000
rs1296972 T C 0.983491 0.014737 0.000000 0.000000 0.000000 0.000000
rs2821821 C T 0.858491 -0.021104 -0.000000 -0.000000 -0.000000 -0.0(
rs2257596 T C 0.929245 -0.028179 -0.000000 -0.000000 -0.000000 -0.0(
```

Prediction

To apply a prediction model, use `--calc-scores`:

```
./ldak5.XXX --calc-scores poly --bfile human \
--scorefile linear.score --pheno quant.pheno --power 0

Reading IDs for 424 samples from human.fam
Reading annotations for 3289 predictors from human.bim
Data contain 424 samples and 3289 predictors

Reading details for 3289 predictors from linear.score
Reading responses for 424 samples from quant.pheno

Calculating scores for Chunk 1 of 1
Profiles saved in poly.profile, with an adjusted version in poly.pr

Correlations between scores and phenotype (saved in poly.cors):
0.79873, 0.77574, 0.70198, 0.68740, 0.69825, 0.64488 and 0.56888
```

(Note, these values are invalid, as using same data to train and test!)

Improving polygenic risk scores

Changing the *p*-value threshold:

```
awk '(NR==1 && $7<0.005){print $2, $4, $5, $14, $8}' \
linear.assoc > new.poly
```

LD-clumping

```
./ldak5.XXX --thin clump --bfile human --pvalues linear.pvalues \
--window-cm 1 --window-prune .2

awk '(NR==FNR){arr[$1];next}(FNR==1 || $1 in arr){print $0}' \
clump.in linear.score > clump.score

./ldak5.XXX --calc-scores poly2 --bfile human \
--scorefile clump.score --pheno quant.pheno --power 0
```

Performing BLUP (Best Linear Unbiased Prediction)

REML also estimates g_i, the breeding value for each individual (`quant.indi.blp`). The BLUP estimates of effect sizes (BLUEs), stored in the vector β, satisfy $X\beta = g$. These are obtained using

```
./ldak5.XXX --reml quant --grm kins \
--pheno quant.pheno --keep filter.keep
```

```
./ldak5.XXX --calc-blups quant --bfile human --grm kins \
--pheno quant.pheno --keep filter.keep --remlfile quant.reml
```

We can do the same with results from genomic partitioning

```
./ldak5.XXX --calc-blups split --bfile human --mgrm mlist.txt \
--pheno quant.pheno --keep filter.keep --remlfile split.reml \
--covar covar.covar
```

The main results are stored in `quant.blup`; can use these to predict phenotypes with `--calc-scores`

MultiBLUP Cross-Validation Example

```
./ldak5.XXX --cut-folds cv --bfile human --keep filter.keep --num-folds 10

chunk=1
./ldak5.XXX --cut-genes fold$chunk --bfile human --chunks-bp 75000 \
--weights sections/weights.all

./ldak5.XXX --calc-genes-reml fold$chunk --bfile human \
--weights sections/weights.all --power -0.25 --pheno quant.pheno \
--keep cv.train$chunk

./ldak5.XXX --join-genes-reml fold$chunk --bfile human \
--sig1 0.00001 --sig2 0.01

./ldak5.XXX --sub-grm fold$chunk/region.0 --bfile human --grm kins \
--extract fold$chunk/region.0 --keep cv.train$chunk
```

MultiBLUP Cross-Validation Example

```
R=`awk < fold$chunk/region.number '{print $1}'`
./ldak5.XXX --reml fold$chunk/out --bfile human \
--grm fold$chunk/region.0 --weights sections/weights.all --power -0.25 \
--pheno quant.pheno --keep cv.train$chunk \
--region-number $R --region-prefix fold$chunk/region.

./ldak5.XXX --calc-blups fold$chunk/out --bfile human \
--grm fold$chunk/region.0 --remlfile fold$chunk/out.reml \
--keep cv.train$chunk

./ldak5.XXX --calc-scores fold$chunk/out --bfile human --power 0 \
--scorefile fold$chunk/out.blup --keep cv.test$chunk --pheno quant.pheno
```

Finally, can check performance in R

```
res=read.table("fold1/out.profile",head=T)
cor(res[,3],res[,5])
```

The END

There are so many more scripts at www.ldak.org

These include full protocols for estimating SNP heritability from raw data at www.ldak.org/protocol

www.ldak.org/sumher has scripts for using SumHer with "proper" data.

Also, we have just put two papers on Bioarxiv
www.biorxiv.org/content/early/2018/03/13/280784 and
www.biorxiv.org/content/early/2018/03/19/284976

The supplements provide full scripts for applying SumHer (and LDSC) to real data.

If you have any questions after the course, email me on doug@aias.au.dk