Module 4: PLINK Practical

Need help? doug.speed@ucl.ac.uk

Running PLINK

It is best to run PLINK (as well as GCTA, LDAK, IMPUTE2) in LINUX
http://pngu.mgh.harvard.edu/~purcell/plink/download.shtml

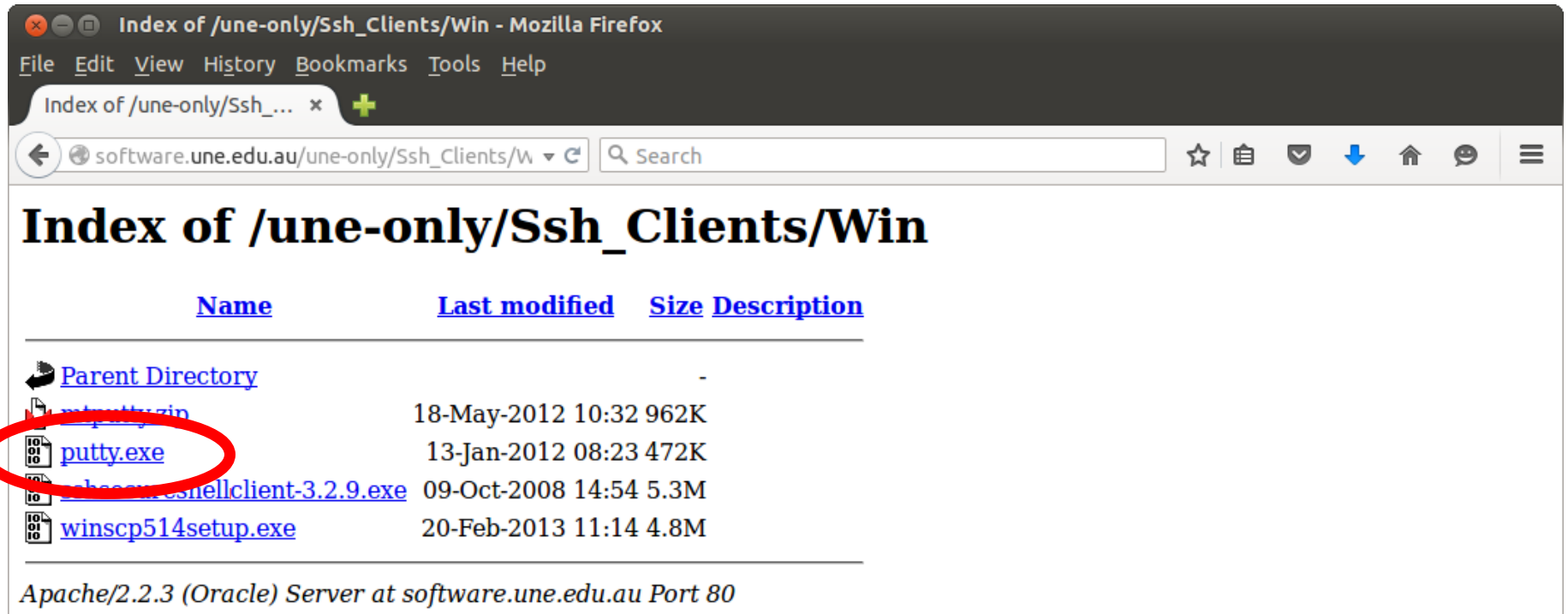
LINUX is the "third operating" system, an alternative to Windows and MAC. It is open source and comes in very many varieties (e.g., Ubuntu, Fedora, Scientific Linux)

If you already have a version of LINUX installed on your personal computer, we can use that. For PLINK (and possibly LDAK) you could alternatively use MAC. Otherwise, you can access Linux on the UNE Desktops

For this route, we will use programmes called putty and winSCP, which will allow us to access Hong's server within Windows

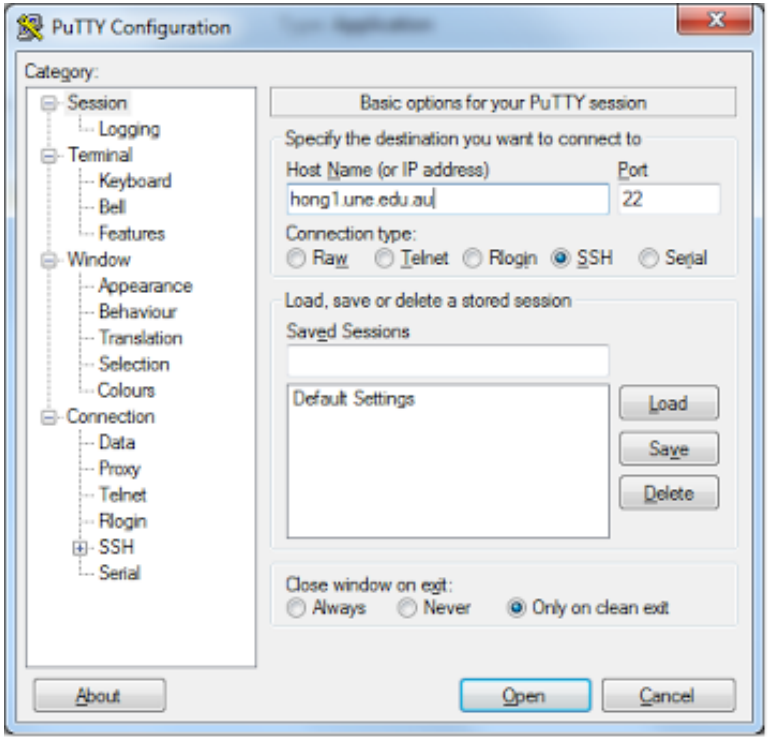
Visit http://software.une.edu.au/une-only/Ssh_Clients/Win/ and click on putty.exe

Installing PUTTY



CLICK TO DOWNLOAD
ONCE SAVED, CLICK TO OPEN

Installing PUTTY



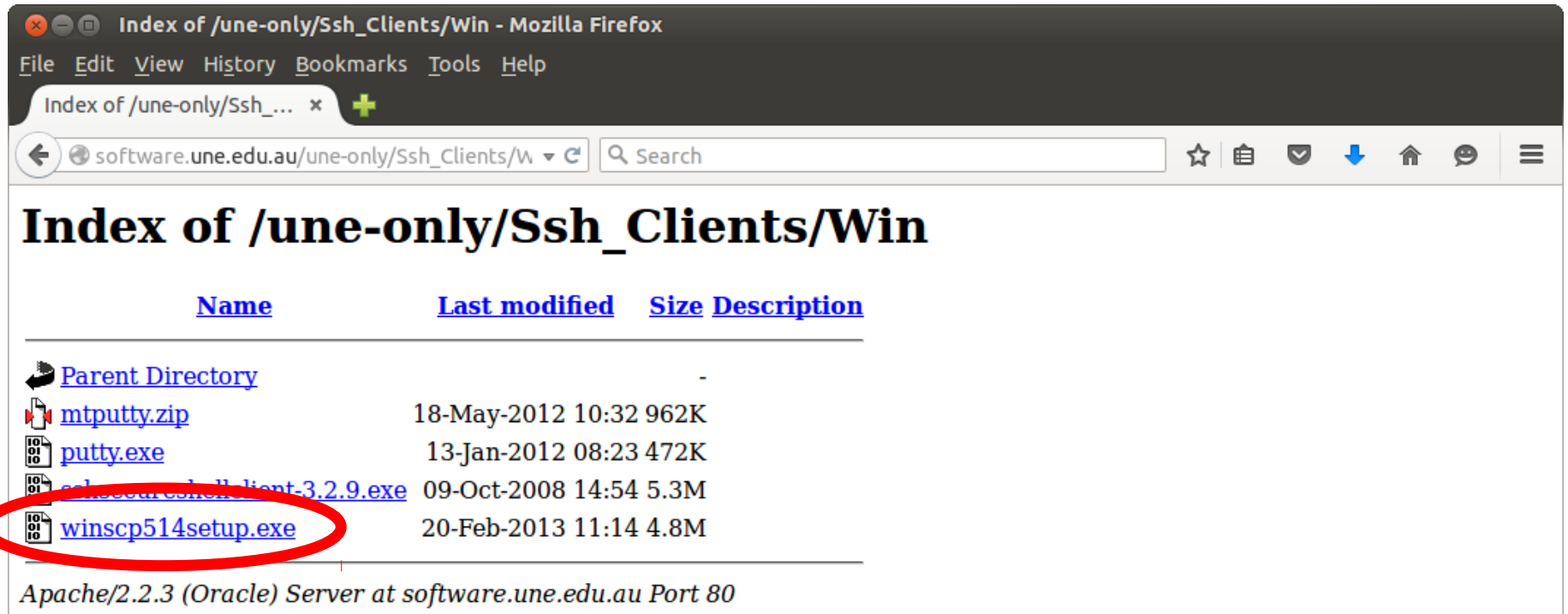
In Host Name, enter `hong1.une.edu.au` then press `Open`

When asked to login, type `asc2016` then press Enter

When promted for your password, type `feb01` and press Enter

Make a home directory which you will use for the course - be creative!
e.g., `mkdir doug`

Installing winSCP



CLICK TO DOWNLOAD
ONCE SAVED, CLICK TO OPEN

Host: hong1.une.edu.au
Username: asc2016
Password: feb01

Running Commands in UNIX

When you run a built-in command in UNIX (using a "bash" shell), you are actually calling an executable. E.g., when you type `ls` this is equivalent to typing `/bin/ls`

This works because bash automatically adds `/bin/` to the front of commands (and if that fails, `/usr/bin/`, and so on)

When running PLINK on your own LINUX computer, it is easiest to save a copy of the PLINK executable in `/bin/`, then you can run it (from any folder) with commands such as

```
plink --bfile <data> --freq --out <output>
```

For this practical, you will run PLINK from the folder you have created (for me, this is called doug), so your PLINK commands will look like

```
../plink --bfile <data> --freq --out <output>
```

or `/home/asc2016/plink --bfile <data> --freq --out <output>`

Running Commands in UNIX

We should be using PLINK 1.9 https://www.cog-genomics.org/plink2

In the PLINK commands, diagonal brackets indicate bits you should change. So to get the above command to work, you might actually type, e.g.,

```
../plink --bfile testdata --freq --out stats
```

which would work if your data are stored in the files `testdata.bed`, `testdata.bim` and `testdata.fam`

Remember, that all options in PLINK are preceeded by two dashes, `--`

When using R, commands are preceeded by a closed diagonal bracket, >

Do not type >, this just represents the command prompt

A backwards slash, \ at the end of a line, means the command spans multiple lines. You can type this, or ignore and use just one line

Data Files

The files for this practical are saved in a zip file in the folder module4

unzip (a copy of) this FROM YOUR HOME FOLDER

```
cd
cd doug
unzip ../module4/mod4_files.zip
```

If you are using your own computer, you can obtain a copy of the zip file using `sftp`

```
sftp asc2016@hong1.une.edu.au
feb01
get module4/mod4_files.zip
```

Which you can then unzip

Data Files

Now in your home folder, you will find genotype data (with prefixes geno1, geno2, genoX) for 258 HapMap individual and 5 k SNPs (as well as a few hapmap files). Note, to find the number of individuals or SNPs, you could use bash commands like

```
wc -l geno1.fam
wc -l geno1.bim geno2.bim genoX.bim
wc -l geno[12X].bim
```

> *"The International HapMap Project is a multi-country effort to identify and catalog genetic similarities and differences in human beings. Using the information in the HapMap, researchers will be able to find genes that affect health, disease, and individual responses to medications and environmental factors"*

http://hapmap.ncbi.nlm.nih.gov

Check List

If all has gone to plan you should:

- Have `putty` installed and configured, which allows you to "ssh" into Hong's server, hong1.une.edu.au (username asc2016, pw feb01). This is where you will run `PLINK, GCTA, LDAK and IMPUTE2`
- Have `winSCP` installed and configured, which allows you to copy files from Hong's server to your Desktop (which you can then read into R), and files from your Desktop to Hong's server
- Have a copy of the slides, either from Julius' website `http://jvanderw.une.edu.au/AGSCcourse.htm`, or from the module2 folder on Hong's server

If so, you may now begin :)

1 - Merge genotypes

Use --bmerge to merge genotypes for Chromosomes 1 & 2

Use --merge-list to merge genotypes for Chromosomes 1, 2 & X

(save with prefix geno12X)

https://www.cog-genomics.org/plink2/data#merge

1 - Merge genotypes

```
../plink --bfile geno1 --bmerge geno2 --make-bed --out geno12

echo "geno1
geno2
genoX" > list.txt
../plink --merge-list list.txt --make-bed --out geno12X
```

R Basics

R data types:

Arrays: Y=array(0,dim=100) or Y=array(0,100) or Y=rep(0,times=100) or Y=1:100 or Y=c(1,2,3,4)

Matrices: X=matrix(0,nrow=10,ncol=20) or X=matrix(0,10,20) or X=matrix(1:200,nrow=10)

Use sq. brackets to change elements: Y[4]=1 X[1,2]=2 X[1:4,3]=3

R has a lot of built-in functions:

mean(Y) var(Y) summary(X) cov(X,Y) cbind(Y,Y)

nrow(X) dim(X) length(Y) head(Y)

sample(Y,size=10,rep=F) sample(1:100,10,T)

table(Y) which(Y=1)

Set Operations:

union(1:4,c(4,5)

setdiff(1:4,c(4,5))

intersect(1:4,c(4,5))

You can make your own functions:

```
my_function_name=function(arguments_required)
{
#    code goes in here
}
```

2 - Filter Individuals by Heterozygosity and Missingness

Use `--indep-pairwise`, `--maf` and `--autosome` to prune common SNPs on Chromosomes 1 & 2

Using these pruned SNPs, use `--missing` and `--het` to compute individual missingness and heterozygosity

Read these into R (`read.table`), create histograms of the missing and heterozygosity rates (`hist`), plot missingness against heterozygosity (`plot`), identify indviduals to remove (`which`), and save those to a file called `keepind.fam` (write.table with options `row.names=F`, `col.names=F` and `quote=F`)

2 - Filter Individuals by Heterozygosity and Missingness

```
../plink --bfile geno12X --indep-pairwise 50 10 .2 --out paut \
--maf .01 --autosome
../plink --bfile geno12X --missing --out stats \
--extract paut.prune.in
../plink --bfile geno12X --het --out stats --extract paut.prune.in
```

```
> miss=as.matrix(read.table("stats.imiss",head=T))
> het=as.matrix(read.table("stats.het",head=T))
> missf=as.numeric(miss[,6])
> hetf=1-as.numeric(het[,3])/as.numeric(het[,5])

> par(mfrow=c(1,2))
> hist(missf,xlab="Missingness");hist(hetf,xlab="Heterozygosity")

> plot(missf,hetf,xlab="Missingness",ylab="Heterozygosity")
> abline(v=.007,col=2);abline(h=c(.365,.28),col=2)
> keep=which(missf<.007&hetf<.37&hetf>.28)
> lines(missf[keep],hetf[keep],col=2,pch=19,type="p")
> write.table(miss[keep,1:2],"keepind.fam",row=F,col=F,quote=F)
```

3 - Infer Sex of Individuals from SNP Genotypes

Use --indep-pairwise, --maf and --chr to prune common SNPs on Chromosome X

Using these pruned SNPs, use --check-sex to estimate Chromosome X heterozygosity

Read these into R and plot estimated heterozygosity. Make an array containing SNP sex (1=male, 2=female) and compare these to sex provided in the fam file (table)

Resave the genotypes (with prefix genonew) using only the individuals retained in 2) and with correct sex (–update-sex)

3 - Infer Sex of Individuals from SNP Genotypes

```
../plink --bfile geno12X --indep-pairwise 50 10 .2 --out pX \
--maf .01 --chr X

../plink --bfile geno12X --extract pX.prune.in --check-sex \
--out sex

> sex=as.matrix(read.table("sex.sexcheck",head=T))
> plot(sex[,6],col=as.numeric(sex[,4])+1,pch=19,
ylab="Estimated Heterozygosity",xlab="Individual")

> newsex=array(0,nrow(sex))
> aa=which(as.numeric(sex[,6])>.8);newsex[aa]=1;newsex[-aa]=2
> #newsex2=2-I(as.numeric(sex[,6])>.8)
> table(sex[,4],newsex)
> write.table(cbind(sex[,1:2],newsex),"all.sex",row=F,col=F,quot=F)

../plink --bfile geno12X --make-bed --keep keepind.fam \
--update-sex all.sex --out genonew
```

4 - Filter SNPs Based on CallRate, MAF and HWE

Use --missing, --freq and --hardy (and datafiles with prefix genonew)

Read these into R and make histograms. Save those with CR $>$ .99, MAF $>$ 0.01 and HWE $P < 10^{-3}$

Note, these thresholds are for demonstration: often they will be too strict in practice.

4 - Filter SNPs Based on CallRate, MAF and HWE

```
../plink --bfile genonew --missing --out stats2

../plink --bfile genonew --freq --out stats2

../plink --bfile genonew --hardy --out stats2

> miss=as.matrix(read.table("stats2.lmiss",head=T))
> freq=as.matrix(read.table("stats2.frq",head=T))
> hardy=as.matrix(read.table("stats2.hwe",head=T))

> missf=as.numeric(miss[,5])
> freqf=as.numeric(freq[,5])
> hardyp=as.numeric(hardy[,9])
```

4 - Filter SNPs Based on CallRate, MAF and HWE

```
> par(mfrow=c(3,1))
> hist(1-missf,n=100,xlab="CallRate",main="")
> abline(v=.9,col=2,lwd=3)
> hist(freqf,n=100,xlab="MAF",main="")
> abline(v=.01,col=2,lwd=3)
> hist(-log10(hardyp),n=100,xlab="-log10(P) from HWE Test")
> abline(v=-6,col=2,lwd=3)

> par(mfrow=c(3,1))
> hist(1-missf,n=100,xlab="CallRate",main="",ylim=c(0,250))
> abline(v=.99,col=2,lwd=3)
> hist(freqf,n=100,xlab="MAF",main="",ylim=c(0,250))
> abline(v=.01,col=2,lwd=3)
> hist(-log10(hardyp),n=100,xlab="HWE -log10(P)",ylim=c(0,250))
> abline(v=3,col=2,lwd=3)
```

5 - Check for Population Outliers

hapmap.load1 and hapmap.load2 contain two "population projections", constructed from HapMap data (will see how to make these in a later Module). Each file contains a linear combination of 1802 SNPs designed to best elucidate the main axes of genetic variation; the major sources of variation will typically correspond to population differences. (Note: would normally have 30-80 000 SNPs)

The files `hapmap.bed`, `hapmap.bim` and `hapmap.fam` contain genotypes for these SNPs for all individuals. Use `--score` to project these data onto the two population projections. Read the two resulting .profile files into R and plot, with points coloured by site (available in relationships_w_pops_121708.txt)

Extract the individuals in genonew.fam (match) and replot with these individuals highlighted (use lines or points). Identify population outliers and save the IDs of non-outliers to keeppop.fam. Resave the genotypes (to genofinal) using only these individuals (–keep)

5 - Check for Population Outliers

```
../plink --score hapmap.load1 --bfile hapmap --out load1
../plink --score hapmap.load2 --bfile hapmap --out load2

> load1=as.matrix(read.table("load1.profile",head=T))
> load2=as.matrix(read.table("load2.profile",head=T))
> rel=as.matrix(read.table("relationships_w_pops_121708.txt",he=T))

> mm=match(load1[,2],rel[,2])
> summary(mm)
> length(unique(mm))
> sites=unique(rel[,7])
> cols=match(rel[mm,7],sites)

> fam=as.matrix(read.table("geno1.fam"))
> mm2=match(fam[,2],load1[,2])
```

5 - Check for Population Outliers

```
> par(mfrow=c(1,2))
> plot(load1[,6],load2[,6],col=cols,xlab="Pop Axes 1",
ylab="Pop Axes 2")
> plot(load1[,6],load2[,6],col="light grey",xlab="Pop Axes 1"
,ylab="Pop Axes 2")
> lines(load1[mm2,6],load2[mm2,6],cex=1.5,type="p",col=2,lwd=3)

> abline(v=5e-6,col=2,lwd=3)
> abline(h=0,col=2,lwd=3)
> keepb=which(as.numeric(load1[,6])>5e-6&as.numeric(load2[,6])>0)
> keep=intersect(keepb,mm2)

> lines(load1[keep,6],load2[keep,6],cex=1.5,type="p",col=3,lwd=3)
> write.table(load1[keep,1:2],"keeppop.fam",row=F,col=F,quote=F)

../plink --bfile genonew --make-bed --keep keeppop.fam \
--hwe 1e-3 --maf 0.01 --geno 0.01 --out genofinal
```