Introduction to `plink` tutorial

AIMS/H3A Bionet

April 2015

1 Set up

1.1. Create a directory `plinkex` for these exercises.

1.2. Unzip the sample data files into this directory.

1.3. Inspect the input files: `hapmap1.ped` and `hapmap1.map` so that you understand their contents.

This tutorial is a combination of mini-guide to PLINK and practical exercise. Some questions you are required to do practical work, and some are just showing you the features of PLINK.

Basic input

1.1. Now run `plink`

```
plink --file hapmap1
```

The `--file` option expects that there will be suitable ped/map files with the given base name. So in the above example, there must be files `hapmap1.ped` and `hapmap1.map`

1.2. Examine the output and make sure that you understand what you see. At this stage all you get are very basic statistical summaries of the data.

1.3. `plink` will typically output files to disk. If you don't specify an output name, it will use the name *plink*. In this case, a file called *plink.log* is created. You can change the basename with the `--out` option

(a) To send the log file to `hapmap1.log`, do the following

```
plink --file hapmap1 --out hapmap1
```

(b) As we explore other `plink` options, you will see that `plink` uses different suffixes (e.g., *frq*, *assoc*), all with the same base name, given by the `--out` option.

(c) Sometimes you may use the same input file with slightly different options and not want to over-write pre-existing files. The best approach is to use sensible names. But, when you are doing interactive work it may be hard to be rigorous. Another approach is to time-stamp each run — you can then always inspect the log files to be sure that the output you are using is produced with the correct options.

```
plink --file hapmap1 --out hapmap1-`date +"%Y%m%d-%H%M"`
```

(d) A slightly annoying feature of `plink` 1 is that each it is run, it checks for an update. On a slow network this sometimes causes delay and you can use the `--noweb` option to disable this. This not implemented in `plink` 2 "yet"

1.4. Now let's do some simple statistical analysis

```
plink --file hapmap1 --freq --out hapmap1
```

Look at the output file and make sure you understand it.

1.5. Now let's do some conversion to binary PED file

```
plink --file hapmap1 --make-bed --out hapmap1
```

1.6. You should see that a `.bed`, `.bim` and `.fam` files have been created. You won't be able to make sense of the first one, but the latter two you should understand.

1.7. To specify using binary ped format input rather than ped format, use the `--bfile` option. Let's try that, and at the same time compare the time it takes to process

```
time plink --file hapmap1 --freq
time plink --bfile hapmap1 --freq
```

How long does each run take? These are very small files by realistic GWAS standards. Note that in both cases, the files *plink.frq* and *plink.log* are created. This means that the second call over-writes the first call – OK here, but in some cases disastrous, so try to get into good habits about using the `--out` option with appropriate names.

Also inspect the relative sizes

```
du -sk hapmap1*
```

1.8. Given a bed/bim/fam comibination, how can you work out the number of SNPs, people, cases, controls.

Recoding

1.9. Data can be transformed into other formats using the table below.

Format	Input option	Output option
PED/MAP (ACGT)	`--file`	`--recode`
BED/BIM/FAM	`--bfile`	`--make-bed`
TPED/TFAM	`--tfile`	`--recode --transpose`
LGEN/MAP/FAM	`--lfile`	`--recode-lgen`

Note that for the PED format, alleles can be encoded as ACGT or 1234. The `--alleleACGT` and `--allele1234` options can be used to do conversion – you have to use the `--recode` or `--make-bed` too.

1.10. Exercise: Recode the *small.ped/map* files to ACGT coding. Recode the *small.ped/map* files into transposed and long formats. Make sure you understand what you see. Convert to binary PED format.

Slicing, dicing, merging, mixing

1.11. There is often a need to extract parts of `plink` files, or to merge files together. We now explore this.

1.12. **Extracting only entries for particular SNPs**. To extract one or a few SNPs from a file you can use the `--snp` option, which takes a single SNP id as an argument. To extract a few SNPs, use the `--snps` option, which takes a list of 1 or more comma-delimited SNP ids. (**NB! This command below has *example* SNP ids — look at your bim or map files to pick meaninful values.**)

```
plink --bfile small --snps rs7727602,rs307347 --recode --out vsmall
```

You can also remove a range of physically close SNPs

```
plink --bfile small --snps rs9729550-rs2842925  --recode --out vsmall
```

These SNPs are close together on chromosome 1 – all the SNPs between these two will be included in the output file

1.13. These commands are only useful for a few SNPs. To extract many SNPs, put the SNP IDs into a file and use the `--extract` sequence.

```
plink --bfile dataf --extract snplist.txt --make-bed --out extract
```

Note how you must use `--extract` in conjunction with a recode or make-bed command.

1.14. The `--exclude` option does the inverse

1.15. There are also options for extracting based on chromosome `--chr` and between particular positions, and randomly sampling (see manual)

1.16. **Extracting named individuals**. The `--keep` and `--remove` take file names as arguments. The corresponding files should contain the individuals – each on a line with family and individual ID. For example, in the file vip.txt we might have

```
HCB193 1
HCB194 1
HCB195 1
HCB196 1
HCB197 1
HCB198 1
```

and then say

```
plink --bfile small --keep vips.txt --make-bed  --out see
```

1.17. You can combine options too:

```
plink --bfile small --keep vips.txt --extract candidatesnps.lst --make-bed  --out see
```

This extracts out the details of a few individuals and a few of their SNPs and stores the extracts in a new file.

1.18. We shall come back to extraction based upon criteria rather than on naming in a while.

Statistics

1.19. The `--missing` option makes `plink` compute how many of the calls are missing. In the output file with the `.imiss` suffix, for each individual we are told what how many SNP calls are missing and what the missing rate is. In the file with the `.lmiss` file, we see listed for each SNP, the number of individuals for which the call for that SNP is missing.

1.20. Exercise: Find ten SNPs in the *hapmap1* file with the highest missing rate. Hint: after running `plink`, use the Linux sort, head and cut commands

```
plink.imiss | sort -k 6 -n -r | head -n 20 | cut -b 5-20,37-50
```

1.21. The `--test-missing` option makes `plink` see whether the missing rate in cases and controls are significantly different. A χ^2 test is used.

This a very important test to use in principle – would there be any SNPs in this file that you would exclude on this basis in this file.

In practice, we are often very strict about the missing rates and filter out any SNPs that have a significant missing rate – hence relatively few SNPs will fail this `--test-missing` test since we've already excluded any with significant missing rate. But one should use this as a sanity test.

1.22. We've seen the `--freq` option already – it tells us about the MAF. We'll see in point 1.42 that we can also get per-group frequencies.

1.23. *Hardy-Weinberg Equilibrium*. The `--hardy` option computes the genotype counts[1] and the H-W p-value. Note that by default this only computes the HW score for those individuals who don't have parents in the set.

Selecting based on criteria

1.24. These questions will be covered in detail in Shaun's section. Just read over

1.25. Earlier, we saw we could extract or remove based on SNP ID or individual ID. Now we look at removing based on various criteria.

1.26. –mind excludes individuals with missing genotype data above the given rate. (This is an example – apply your mind to your data and choose appropriate file names and parameters)

```
plink --bfile data --mind 0.04 --make-bed --out data-i96
```

This creates a new set of binary ped files *data-i96* that contain all individuals from the *data* file who have at least a 96% call rate.

The IDs of the individuals removed are put in the *.irem* file.

1.27. The `--geno` option removes SNPs that have less than the specified call rate

```
plink --bfile data --geno 0.04 --make-bed --out data-s96
```

[1] i.e., the number homozygous in the minor allele, the number heterozygous, the number homozygous in the major allele

1.28. Exercise: what is the difference, in principle, between

- `plink --bfile data --geno 0.04 --mind 0.04 --make-bed --out data-s96`
- `plink --bfile data --mind 0.04 --geno 0.04 --make-bed --out data-s96`
- ```
  plink --bfile data --mind 0.04 --make-bed --out 1
  plink --bfile 1 --geno 0.04 --make-bed --out 2
  ```
- ```
  plink --bfile data --geno 0.04 --make-bed --out 1
  plink --bfile 1 --mind 0.04 --make-bed --out 2
  ```
- ```
  plink --bfile data --mind 0.1 --make-bed --out 1
  plink --bfile 1 --geno 0.04 --make-bed --out 2
  plink --bfile 2 --mind 0.04 --make-bed --out 3
  ```
  You should see that you must decide whether you wish to maximise the number of individuals or SNPs that are left.

1.29. The `--maf` option can be used to remove SNPs with a MAF below a certain rate. Take care since the MAF is notional: what if the MAF rate 0.96?

1.30. You can use the `--hwe` option to remove SNPs which fail Hardy-Weinberg (see s. 6.4).

1.31. Note that with these options, you may not necessarily want to output the filtered file. For example, you could do the following

```
plink --bfile data  --mind 0.04 --freq --out see
```

This computes the MAF for the *data* file after cleaning it. But the cleaned version itself is not stored.

Merging files

1.32. Merging files is easy – some of the time. However, you must make sure it makes sense to do so.

1.33. The `--merge` option can be used to merge files in ped/map format. The following are all examples. Note you have to specify the ped and map file names.

```
plink --file dataA --merge dataB.ped dataB.map --recode --out new

plink --file dataA --merge dataB.ped dataB.map --make-bed --out new

plink --bfile dataA --merge dataB.ped dataB.map --recode --out new

plink --bfile dataA --merge dataB.ped dataB.map --recode --make-bed new
```

1.34. The `--bmerge` option can be used to merge files in bed/bim/fam format. All three files must be specified. Here are examples:
```

```
plink --bfile dataA --bmerge dataB.bed dataB.bim dataB.fam --recode --out new
```


```
plink --bfile dataA --bmerge dataB.bed dataB.bim dataB.fam --recode --out new
```


1.35. Exercise: in the directory `hmpops` are small extracts from HapMap data. Combine the two European data sets into a binary ped file EUR. Combine the three African data sets, the CEU and JPT data sets into a BED/BIM/FAM file set AFR.

1.36. There are often complications in merging caused by differences in the data.

A once uncommon problem is caused by the fact that `plink` only handles biallelic alleles. Although this is true of most alleles there are some alleles that are tri-allelic. If you take a population where the alleles are A and C, and merge it with a population where the alleles are A and G, `plink` cannot handle this. You have to discard the SNP or not merge.

A more common problem is that the data sets may not consistently agree on whether the forward or reverse strand were sampled. So one data set may see the SNP's alleles as A/C and the other may see it is G/T.

If you try to merge two files and there are SNPs that disagree, the merge fails and a new file with the *.missnp* suffix is produced containing the names of the SNPs which failed. If the reason they failed is because the data sets sampled different strands this is easy to fix. Use the `--flip` option to flip the SNPs and try merge again. However, you must take care that this is the reason merging failed. A typical merge operation would be

```
plink --bfile A --bmerge B.bed B.bim B.fam --make-bed --out AB

# oops this fails -- plink complains and produces a file called
# AB.missnp

mv AB.missnp flipsnp1.txt

# I rename because a subsequent plink may overrite AB.missnp

# now flip only the problem snps

plink --bfile A  --flip flipsnp1.txt --make-bed --out A0

# try merge again

plink --bfile A0 --bmerge B.bed B.bim B.fam --make-bed --out AB
```

The above commands will take care of any flipped SNPs. But you may either have triallelic or problem SNPs left and the last merge command fails. If there are only a few SNPs, you can probably safely exclude by doing

```
mv AB.missnp problem.snp
plink --bfile A0 --exclude problem.snp --make-bed A1
plink --bfile A1 --bmerge B.bed B.bim B.fam --make-bed --out AB
```

However, if there are many SNPs that mismatch, you need to look at the data very carefully.

1.37. Read the manual on the merging option. Real care must be taken with merging.

1.38. Exercise: in the directory `exercise` are extracts from various data sets. Combine the three 1000 Genomes data sets (CEU, and YRI) into one. [This is different to the data set from Question 1.35 because you will have to do SNP flipping]

Extracting based on group membership

1.39. Handy ways of extracting out groups are to use the filter options

```
--filter-cases
--filter-controls
--filter-males
--filter-females
--filter-founders
--filter-nonfounders
```

1.40. There is also a general `--filter` option which uses the same format as a phenotype file. For example, in the `hmpops` directory is a fileset ALL which contains the data from all the groups. Suppose I have this fileset but not the other files, and now I want to extract out the Yoruban data. I could say

```
plink --bfile ALL --filter group1.phe YRI --make-bed --out yri
```

or to produce a file with all the African data I could say

```
plink --bfile ALL --mfilter group2.phe AFR 4 --make-bed --out yri
```

1.41. There is also more general way of extracting out both SNPs and individuals based on attribute sets. See section 4.20 of the manual. This is a very powerful feature.

1.42. Another example use of these files is to compute frequencies per population. For example

```
plink --bfile ALL --freq --within group1.phe
```

Association testing

1.43. We now finally get to association testing. All the other preparatory commands are crucial to ensure good quality data. You must get into the habit of reading the log files after each operation as it is too easy to skip errors that only manifest themselves much later.

1.44. The `--assoc` option can be used to do association testing

```
plink --bfile hapmap1 --assoc --out hapmap1
```

This produces the *.assoc* file.

1.45. Try the example to understand what the output means. Use Linux commands to find the most significant SNPs.

1.46. Adding the `--ci X` option also produces the X% confidence level for the ORs.

1.47. Using the `--adjust` option *as well* produces some of the standard scores adjusted for multiple testing. This is in the *.assoc.adjusted* file.

1.48. Using the `--mperm X` option with the `--assoc` option performs permutation testing. This is described in 11.3. Essentially, the EMP2 column found in the *.assoc.mperm* file is the key column to look at. Typically, a cut-off of 0.05 would be considered reliable.

Alternative phenotypes

1.49. Usually the sixth column of the PED or BIM file is taken as the phenotype. However, often it is desirable to put the phenotype in another file in PHE format. The `--pheno` option allows this to be specified

```
plink --bfile data --pheno cascon.phe --assoc
```

This would perform an association test on the *data* file, using the 3rd column of the *cascon.phe* file as the phenotype column.

1.50. A more general case the phenotype file could have more then 3 columns – the `--mpheno` option allows the column number to be specified.

```
plink --bfile data --mpheno cascon.phe --assoc:
```

Note that the phenotype must be encoded by a number – but see the `--make-pheno` option below.

1.51. We'll now apply this to the idea of comparing two populations to see which SNPs are significantly different. We are not doing a case/control study, but rather distinguish the populations using the phenotype value – for example, we could use a value of 1 to mean TSI and a value of 2 to mean CEU. However, it's safer to use a meaningful code like TSI or CEU. This is where the `--make-pheno` option is useful. It takes two parameters —

a file name and a string. The file should be in PHE format except you can have string labels.

```
plink --bfile data --make-pheno pop.phe TSI --assoc
```

The file pop.phe labels each individual – treat the ones labelled TSI as the case, and the others as the control.

1.52. Exercise: In the `hmpops` directory, merge the TSI and CEU files into a EUR file and then find any significantly different SNPs.

Take care!!!! Read the log files.

Linkage disequilibrium

1.53. `plink` has various routines to deal with linkage disequilibrium

1.54. The most basic is the one to compute LD between two named SNPs

```
plink --bfile YRIs --ld rs10000092 rs10000096
```

1.55. Exercise: consider the LD in different populations

1.56. The `--indep-pairwise` option allows us to *prune* SNPs that are in LD. What this does is scan through the genome and try, in each window, to select one SNP that represents the others in the window. It takes three arguments — the length of the sliding window, how much the window is shifted at each step, and the r^2 value which is used.

For example

```
plink --bfile hapmap --indep-pairwise 50 10 0.1 --out check
```

- The file check.prune.in is the list of SNPs that should be used as representative of the data
- The file check.prune.out is the list of SNPs that could be filtered out.
- The `--extract` or `--exclude` options can be used to appropriately.

1.57. Now prune the AFR data from question 1.35 above. Call the new data set *afrprune*.