Introduction to `plink` tutorial

H3A Bionet

April 2014

1 Set up

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

1.2. Copy any sample data files into this directory.

1.3. Data is in `/data/plinkex`. NB: need sudo to write to this.

1.4. Look at hapmap1.* to familiarise yourself with the data.

1.5. We are using plink 1.07 in this lab. If you have used plink before and would like to try the new version of plink, execute the command `use_plink_19`. To go back to using plink 1.07, execute the command `use_plink_107`

2 Input and conversion

2.1. 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`

2.2. Examine the output and make sure that you understand what you see. At this stage all you get are basic statistical summaries of the data.[1]

2.3. `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.

[1] Note that in `plink1.9a`, the behaviour is different – no statistical output is given and rather a conversion is made to binary output.

(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` 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. However, usually you can just ignore

2.4. 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.

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

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

2.6. 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.

2.7. 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*
```

2.8. Aside: Executing `plink` tells you how many cases and controls there are. Using the `cut` and `grep` commands, do the same.

Recoding

2.9. 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.

2.10. 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.

2.11. `plink1.9a` also supports conversion from VCF format.

3 Slicing, dicing, . . .

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

3.1. **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 meaningful values.**)

```
plink --bfile small --snps rs9729550,rs3813196 --recode --out vsmall
```

You can also remove a range of physically close SNPs

```
plink --bfile small --snps rs307347-rs745910  --recode --out vsmall
```

These SNPs are close together on chromosome 1 – all the SNPs between these two will be included in the output file (look at the map file so that you can be sure you understand what you are doing).

3.2. 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, in the following way (choose your own input file and create an appropriate *snplist.txt*).

```
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.

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

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

3.5. **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
```

NB: create an appropriate vips.txt file

3.6. 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.

Extracting based on group membership

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

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

3.8. 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 as an *alternate phenotype* file. The `--pheno` option allows this to be specified. The phenotype file has at least 3 columns – more is optional. The format is

FID IID PHENO1 PHENO2 ....

We'll mainly use this in association testing, but it can be used elsewhere.

3.9. A special case of the phenotype file is a *cluster* file which only has one phenotype column.

3.10. 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 --filter group2.phe AFR --mfilter 4 --make-bed --out yri
```

Using the hapmap1 data file, create bed/bim/fam files for the Japanese participants.

3.11. There is also more general way of extracting out both SNPs and individuals based on attribute sets. See section 6.20 of the manual. This is a very powerful feature.

Selecting based on criteria

The exercises here give general formats of instructions to be run. Apply your mind to the data in front of you and give

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

3.2. –mind excludes individuals with missing genotype data above the given rate

```
plink --bfile data --mind 0.02 --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.

Using the hapmap1 file: how many people would be removed if you chose 1%, 2%, 4% and 5% as the quality cut-off?

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

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

3.4. 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
  ```

  These exercises look at the mechanics – we'll discuss the principles later. There are more advanced features we're not looking at: see `--qual-geno-scores`, `--qual-scores`

3.5. 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?

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

3.7. 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.

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

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

- Use the cut command to create a cluster file from hapmap1.phe and then compute frequencies per population.

3.9. 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.

3.10. We've seen the `--freq` option already – it tells us about the MAF.

3.11. *Hardy-Weinberg Equilibrium*. The `--hardy` option computes the genotype counts[2] 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.

3.12. Checking sex: to check that your data is consistent with respect to sex, use the `--check-sex` option – this checks the genetic data is consistent with the specified data in column 5 of the fam or ped file.

```
plink --bfile data --check-sex
```

4 . . . merging, mixing

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

4.2. 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
```

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

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

```
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
```

4.4. Exercise: in the directory `hmpops` are small extracts from HapMap data. Combine the three African data sets (YRI, LWK, MKK) into a bed format AFR.

Do this twice:

- Once using two *bmerge* commands.
- Once using the *merge-list* command.

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

An 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.

4.6. A second problem may be that you combine data sets where the map files differ in physical and genetic distance. The `--update-map` and `--update-cm` options can be used (look in the Data Management section of the manual).

4.7. A third, 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.

4.8. Read the manual on the merging option. Real care must be taken with merging. This makes automating the process of merging when you are merging data that comes from different sources.

4.9. **Exercise:** Merge the YRI and JPT data sets into a file YJ. Fix all problems so that no warnings occur. This will involve:

- Updating the map file so that there is consistency of physical position.
- Flipping any SNPs that can need and can be.
- Noting, investigating and removing and SNPs which may still need to be.

5 Miscellaneous/Advanced

5.1. Annotation. Association files are the result of plink analysis and show which SNPs are associated with the phentoype. We'll look at association testing later, but here is a useful feature to annotate the results so as to understand it. You need

- An association file: look at *example.assoc*
- A gene list: *gene_list.txt*
- A SNP attribute list: *snp_attrib.txt*

`plink` will combine all three for you:

```
plink --annotate example.assoc attrib=snp_atrrib.txt ranges=gene_list.txt
```

This isn't a feature in `plink1.9a` (but you can keep both binaries on your system).

5.2. Relatedness: `plink` has several features for testing relatedness. These rely on the samples being homogeneous.

5.3. IBD – identity by descent.

```
plink --file mydata --genome
```

This produces pairwise estimations of participants. If you have n individuals there will be n^2 rows, so this is something that should be done with care. Typically you would do this with a very small data set, or to use the `--rel-check` option for family based studies.

5.4. Inbreeding coefficients: the `--het` option computes inbreeding coefficients – which is computed by comparing the observed and expected number of homozygote positions. This would typically be done using pruned data sets.

5.5. Defining sets. Many `plink` tests support sets of SNPs. These are specified in a SET file, of the following format

```
SET_1
rs18728
rs393893
rs97373
rs7873
END

REGA rs817278 rs67363 rs7873873 rs1933333 END
```

Each set started with a name and finishes with "END". The SNPs are separated by space, tab or new line.

5.6. Use the `plink` epistatis option to see if there is any epistatic effect between the following SNPs

```
rs2980319
rs2511734
rs2511735
rs2905037
rs2980319
rs6672353
rs2905036
rs4951864
rs11240777
rs4245756
rs2980290
rs10751454
rs12726447
rs4475691
```

at the p=0.01, p=0.1, p=0.5, p=0.8 levels.

5.7. Investigate the new version of `plink`: `https://www.cog-genomics.org/plink2`