Population Structure Tutorial

Wits H3A Bionet

April 2014

1 Setup

1. In this exercise we use the following standard programs
   - eigenstrat
   - admixture
   - CLUMPP
   - distruct
2. We will also use Wits wrapper scripts
   - runpca (bash shell)
   - popifyfam.py
   - evec2gp.py
   - cdg.py

   and a Java application *Genesis.java* with a wrapper script
3. Download the file `\http://www.bioinf.wits.ac.za/courses/gwas/pop.tar.gz`
   Then

   ```
   tar -xzf pop.tar.gz
   ```

   All files will be in the *pop* directory. Move into the directory and say

   ```
   sudo make install
   ```
4. You need to use the following data
   - The ALL.{bed,bim,fam} files.
   - group1.phe, group2.phe

   Make sure you understand this data.
5. **Note** that before running any population structure programs, data should be **pruned** so that remaining SNPs are not in LD with each other. For time reasons we don't do this now. However, this is a very small data set and you'll see the structure is not good – the results are not realistic.
6. Try to run `gnuplot`. If not install it

   ```
   apt-get install gnuplot-x11
   ```

2 PCA – Eigenstrat

1. Run Eigenstrat:

```
smartpca.perl  -i ALL.bed -a ALL.bim -b new.fam \
       -p ALL.pca -e ALL.eval -o ALL.pca  \
       -q NO -l ALL.log
```

2. This performs the PCA and produces several output files

   - ALL.log — the log file – read it and understand it
   - ALL.eval – the eigenvalues (look at it)

3. The most important one is the file with the ALL.pca.evec file.

   (a) The first line, commented. This the eigenvalues of the corresponding eigenvectors. Essentially these show the relative weightings of the corresponding principal components.

   (b) The remaining lines give, for each, individual their eigenvector, which we can interpret as a point in a high-dimensional space. Typically, we select a few of the dimensions for display. Commonly we look at PC 1 and PC 2, which are the first two columns but it may be necessary to display others to see more subtle population structure

4. If everything went well, you should have a PDF file called ALL.pca.pdf. View it.

5. Which PCs are significant? The *twstats* program is used. It takes as input your *eval* file and a file called *twtable*. This is a standard file distributed with EIGENSTRAT and is included here for convenience.

   ```
   twstats -t twtable -i ALL.eval
   ```

   Which eigenvectors are significant?

6. For some analyses and for drawing, we need to have the populations specified. Eigenstrat takes this from the fam file, but in practice often the fam file either has case/control/qt information or is missing. Also, you may wish to use different "population" labels at different times – in a realistic analysis you may wish to compare cases versus controls, and samples that were done in different batches as well as the population label.

   We have a script called *popifyfam.py* that creates a new fam file.

   Here is how you would run it. It matches information in the given *phe* file with the people mentioned in the ALL.fam file. You can specify which column of the *phe* file should be used. You can also specify what the new fam file should be called (if you don't specify an output file, output goes to standard output).

   ```
   popifyfam.py -h
   ```

   ```
   popifyfam.py  --popfname group2.phe --popcol 3 --output new.fam  ALL.fam
   ```

7. Now we rerun our analysis. But now we can't use the helper script because the fam file has a completely different name.

```
smartpca.perl  -i ALL.bed -a ALL.bim -b new.fam \
       -p ALL.pca -e ALL.eval -o ALL.pca  \
       -q NO -l ALL.log
```

8. View the PDF that was created.
9. Read the log file carefully.
10. Run the Genesis program: `genesis`

    Information and updates can be found at `http://www.bioinf.wits.ac.za/software/genesis/`

    This program is under development – come back soon!
11. Choose the PCA option and load ALL.pca.evec with group2.phe. Choose column 4.
    - Show different PCs
    - Change colours
    - Change labels, fonts
    - Find NA21525
    - Who is the leftmost individual?
    - Change to column 4 as the phenotype.
    - Hide the Tuscans (TSI) from the graph.
    - Export as PDF
12. Repeat the exercises above for the *comm-* data set, using the *comm.phe* file. I suggest you popify first – as the smartpca will take about 5 minutes to run.

Extra exercises

13. The rest of this section is for you to come back at the end of the day. It is an alternative to using *genesis*
14. Using the evec2gp program

    `python evec2gp.py --phe group1.phe ALL.pca.evec`

    This takes the vectors produced which give "objective" evidence of population structure plus evidence we give based on ascribed membership.

    It produces a gnuplot program.

    `gnuplot ALL.gp`
15. To produce a Postscript and PDF files, you do the following

```
python evec2gp.py --gp-term postscript --epstopdf --gp-output ALL.eps \
      --phe group1.phe ALL.pca.evec
gnuplot ALL.gp
```

3 Admixture

1. **Run admixture**

   ```
   admixture ALL.bed 4
   ```

2. This produces a .Q file that contains estimates for each person and a .P file that contains estimates for each SNP. Look at the data and understand it.

   Also read the output that gets produced during and at the end of the run. . . What useful information is there?

3. Now, use the admixture `--cv` option to estimate which is the best value of K to use for this data. Having seen the PCA of the data, are you surprised?

4. **Viewing admixture** For these exercises, we'll use the *small* data set. We've already computed the small.*.Q files for various k values. Look at the phe file so that you understand what phenotypes are stored.

5. Run *genesis* using small.4.Q, small.fam and small.phe and a phenotype column of your choice.

   (a) Inspect the results.
   (b) Change the order of the populations
   (c) Change the colour of some of the populations. Find the individual *2477 NA20301*.
   (d) Show the chart vertically.
   (e) Now add the small.2.Q and small.5.Q charts.
   (f) Reorder the charts so that you have them in the order 2, 4, 5.
   (g) Colour the charts consistently.
   (h) Add some labels.
   (i) Export to PDF.

4 CLUMPP

First, we run admixture several times and then run CLUMPP.

1. The *runadmix.sh* script automates the process of running *admixture*. It's written in the bash shell language. Read through it to make sure you understand it. With automation comes great power, but also great power to to silly things! Why do we say `-s time`?

2. Modify the script so that it uses the ALL.bed file, and we have k values ranging from 2 to 4, and using 5 runs.

3. Run the script thus[1]: `./runadmix.sh`

[1]Linux revision: why do we say ./runadmix.sh? Why put the "./"? Remember that "." means the current directory. When you run a command, Un*x looks in your PATH directories for the program you want to run. However, it is considered good practice not to have the current working directory on your path. So scripts or programs that are in your current working directory will *not* be found. Hence, we tell the system where the script will be found – in the current working direcory.

4. This example gives a grossly misleading impression of how long it takes to do this. On real data set, you may take many hours or even days for this to run and it would be desirable to parallelise on a computer cluster.

5. Inspect the output and make sure you understand it.

6. Before running CLUMPP, we need to create the input for it and an appropriate *paramfile*. So we run cdg and then CLUMPP. Revise the overview in the notes.

7. We run the cdg script

```
cdg.py  -K 2 --glob "[0-9]*/" \
       --output all-2 --par_clumpp ALL
```

8. Inspect the paramfile that has been created – make sure you understand the key options. The important values are

   (a) DATATYPE should be 0
   (b) INDFILE the input file – should contain output of several admixture calls.
   (c) OUTFILE output file
   (d) MISCFILE log file – output file
   (e) K the number of clusters
   (f) C the number of individuals
   (g) R number of runs (number of data sets in the INDFILE)
   (h) M : method to be used 1, 2 or 3. From most accurate and expensive to least accurate and cheapest.
   (i) GREEDY_OPTION If you use 1 or 2 above, choose 2.

9. Run CLUMPP it takes the paramfile as input. `CLUMPP`

10. Identify and inspect the output file.

11. View the output file in Genesis.

5 Repetition

Repeat the exercises above with *comm-SYMCL*.

6 The cdg.py script

1. A manual is available

2. A helper script that creates the necessary things for CLUMPP and distruct.

```
python cdg.py -K 4 --glob "[0-2]/" --popfname group1.phe
       --par_clumpp --doclumpp ALL
```

This says look in the directories 0, 1, 2 for files named ALL.4.Q and use those as input to clumpp, and use the group1.phe file as population description. Typically you might have 100 runs.

3. The key output file is ALL.outfile which summarises result.

4. Now we can run `distruct`

```
python cdg.py -K 4 --glob "[0-2]/" --popfname group1.phe \
       --par_distruct --dodistruct --outputps ALL-admixture ALL
```

7 Distruct

1. You can download and install from `http://www.stanford.edu/group/rosenberglab/distruct.html`

2. DISTRUCT takes output from CLUMPP (or other programme) and some auxiliary files and creates very high quality pictures. It takes one file as input *drawparams*. These are the key values to change

   (a) INFILE_POPQ – for each (external) population average ancestry from ancestral populations

   (b) INFILE_INDIVQ – describes each individual

   (c) INFILE_LABEL_BELOW – a file that contains a list like this

```
5 YRI
7 LWK
9 MKK
12 SABS
2 HADZA
13 CHB
4 CHD
10 SAN
1 IMM
3 CEU
11 SANDAWE
6 JPT
8 GIH
```

   (d) INFILE_CLUST_PERM a file with the colours you want to use

```
1 orange
2 blue
3 yellow
4 green
5 light_purple
```

   (e) OUTFILE the name of the output file (it's a PostScript file)

(f) K number of clusters

(g) NUMPOPS number of pre-defined populations

(h) NUMINDS number of individuals