Genome Informatics Workshop: An Introduction to GWAS in plink and R

Miles Benton & Rod Lea

July 25, 2015

Contents

Introduction **2**

Before we start . . . 2

Credit where credit is due . . . 2

Installing R and RStudio **2**

Required Software . . . 2

RStudio . . . 5

plink **5**

Setting up for this workshop . . . 6

plink file structure and data recoding . . . 7

GWAS: Genomewide Association Study **9**

Quality Control in plink (QC) . . . 9

Basic GWAS (association testing) . . . 12

Exploring association statistics . . . 12

Correcting for multiple-testing burden . . . 13

Genomic inflation factor and population structure . . . 13

Manhattan plotting . . . 14

Exploring population effects on association results . . . 15

Running a stratified association in plink . . . 16

Exploring population stratification further using PCA and MDS . . . 18

Adding in additional data . . . 24

Exploring genotype associations . . . 27

LD analysis and plotting . . . 34

Summary . . . 40

Appendix **41**

Introduction

In this workshop we will explore the basics of performing Genome-wide Association Studies (GWAS) using a combination of the software `plink` (http://pngu.mgh.harvard.edu/~purcell/plink/) and `R` (http://www.r-project.org/).

At a later stage (a separate workshop) we will also introduce the `GenABEL` package which offers additional functionality of genomic association analyses in R.

This workshop is written to be accessible to a range of people, from those that have never performed any command-line work, to those that just want a refresher. As such we have written it to include clear and straight forward outlines of all `R` and `plink` operations, most of the time stepping through these one line at at time.

Before we start

If you are attending this workshop in person it is likely you have also attending the Introduction to `R` and `RStudio` workshop the day before. If not (or if you just want a refresher) we are including a brief overview on obtaining and installing the required software for the current workshop. If you would like more detailed information, please feel free to access a detailed overview of `R` and `RStudio` which is available on Miles personal website, here. Miles is also working on converting the pdf version of this manual to an on-line interactive gitbook version, the beta version of this product is available here for those that are interested.

Credit where credit is due

We would like to thank all authors of the various pieces of software that are used throughout this workshop. Links and important information are included throughout the document should people be interested in contacting these individuals.

We would also like to thank Shaun Purcell, the original developer of `plink`, for an excellent piece of software. Also his 2008 GWAS tutorial inspired the design of this workshop, so we are grateful to him for making that freely available to modify and distribute (it made the job here a lot easier!).

Installing R and RStudio

Installation of `R` is fairly straight forward, although it differs slightly between platforms. All platforms have binaries (executable installers) available, and these are recommended for most users. If you want to build `R` from source you'll need the suitable compilers depending on your platform.

Required Software

Below are links to the R project and RStudio websites.

- R: www.r-project.org/
  - Windows: http://cran.r-project.org/bin/windows
  - Linux: http://cran.r-project.org/bin/linux
  - Mac OS X: http://cran.r-project.org/bin/macosx
- RStudio: http://rstudio.org/

Windows R installation

The bin/windows directory of a CRAN site contains binaries for a base distribution and a large number of add-on packages from CRAN to run on Windows XP or later on ix86 CPUs (including AMD64/Intel644 chips and Windows x64). You can locate the above directory via this link: http://cran.r-project.org/bin/windows/base/

Installation is via the installer – "R-3.2.2-win.exe" is current stable release. Once downloaded just double-click on the icon and follow the instructions. When installing on a 64-bit version of Windows the options will include 32- or 64-bit versions of `R` (and the default is to install both). You can uninstall `R` from the Control Panel or from the (optional) `R` program group on the Start Menu.

Installing R packages from source for Windows

Before starting the workshop you'll need to have the ability to build certain R packages from source. This requires special libraries, which most of the time aren't installed by default. To begin building R packages for Windows you'll need to install RTools.

The most recent version can be found here: http://cran.us.r-project.org/bin/windows/Rtools/

The current release is RTools33. It's as simple as downloading and installing, just follow the prompts.

Once RTools is installed you should be able to proceed with the workshop.

Linux R installation

At the time of writing there are four Linux distributions that have `R` binaries available; Debian, Redhat, Sues and Ubuntu. They are located here: http://cran.r-project.org/bin/linux. These binaries 'might' work under alternate distributions (likely in most cases, but your mileage may vary) - remember it's your system, so take care!

The `R` source code can be downloaded and compiled with little hassle on most Linux distros. A guide for compiling `R` can be found here: http://cran.r-project.org/doc/manuals/R-admin.pdf

Example installation under Debian/Ubuntu to install the complete `R` system, use:

```
sudo apt-get update
sudo apt-get install r-base
```

Installing R packages from source for Linux

If you are running Linux I'm assuming that you have your own working set up that you're happy with, so you don't need me to guide the way. If you're new to Linux you'll be pleasantly surprised to find that most popular distro's have the required libraries for building R and it's packages from source already installed. If you do have issues there is plenty of excellent documentation:

- R FAQ: https://cran.r-project.org/doc/FAQ/R-FAQ.html
- R Installation and Admin: https://cran.r-project.org/doc/manuals/r-release/R-admin.html

If you are completely in the woods, a good place to start is by installing the libraries that R is built from (these are the things you'll want installed if you are compiling R from source).

```
sudo apt-get install r-base-dev
```

Mac OS R installation

There is a nice FAQ on `R` for Mac OS X users which is frequently maintained and updated regularly. It can be found here: http://cran.r-project.org/bin/macosx/RMacOSX-FAQ.html

The 'bin/macosx' directory of a CRAN site contains binaries for OS X for a base distribution and a large number of add-on packages from CRAN to run on OS X 10.5/6/7.

The simplest way to install `R` on Mac is to use 'R-3.2.2.pkg': just double click on the icon.

Note from Miles: I apologise for the lack of Mac OS support throughout this document, I have never used a Mac before coming from a Windows start and am now very much entrenched in the Linux environment. I'll try my best to provide links to support for Mac users where possible.

Installing R packages from source for Mac OS X

The initial set up for Mac is a little more detailed, but should be fairly straight forward and once it's done you don't have to worry about it again.

Command Line Tools

To build R packages from source Macs require `command line tools` to be installed. To do this:

- Launch the Terminal, found in /Applications/Utilities/
- Type the following command string:

```
xcode-select --install
```

- Follow the prompts

Additional R libraries for package building on OS X

You will also need to install both additional libraries from here: https://cran.r-project.org/bin/macosx/tools/

If you are running OS X 10.5 or higher you'll want to download and install the following:

- gfortran - https://cran.r-project.org/bin/macosx/tools/gfortran-4.2.3.pkg
- tcltk - https://cran.r-project.org/bin/macosx/tools/tcltk-8.5.5-x11.pkg

... and then gfortran version from here (there are also a large list of optional packages for Mac here): http://r.research.att.com/libs/

Again, for OS X 10.5 or higher download and install the following:

- gfortran - http://r.research.att.com/libs/gfortran-4.8.2-darwin13.tar.bz2
- this needs to be extracted and installed in the correct location. You can do this by running the following code in the terminal:

```
sudo tar fvxz gfortran-4.8.2-darwin13.tar.bz2 -C /
```

... or, if you prefer, everything from the terminal:

```
curl -O http://r.research.att.com/libs/gfortran-4.8.2-darwin13.tar.bz2
sudo tar fvxz gfortran-4.8.2-darwin13.tar.bz2 -C /
```

Once you've performed the above steps you're ready to move on.

RStudio

> "RStudio is a new integrated development environment (IDE) for R. RStudio combines an intuitive user interface with powerful coding tools to help you get the most out of R."

`RStudio` is an IDE that works with the standard version of `R` that is freely available from CRAN. Like `R`, `RStudio` is available under a free software license. The goal of `RStudio` is to provide a powerful tool that is easy for beginners to grasp and at the same time offers increased productivity for advanced users.

We chose `RStudio` for this workshop as we believe it makes `R` easier to understand and operate, and allows an easier transition from the typical gui environment that most users are used to. It has features such as syntax highlighting and code completion that aid in the learning of the `R` language, and the object and history browser allow the user to see and understand how `R` is working behind the scenes – you can even convince yourself that your data is loaded by clicking on it and seeing a nice spread sheet like view!

There are of course many alternatives if you decide you don't like `RStudio`, or just feel like trying something different. A list of gui projects is available here: http://sciviews.org/_rgui. R Commander is apparently very good and will be quite familiar for people who have used SPSS, link here: http://socerv.mcmaster.ca/jfox/Misc/Rcmdr

Download and Installation

`RStudio` is available for Windows, Linux and Mac OS X (10.5+). The software can be downloaded from http://rstudio.org/download/desktop. This page will attempt to determine the best download for your system, but you may also choose from the list below.

Like `R`, there are pre-compiled binary install files available for most popular OS. If yours is not listed, and none of the binaries work, then you should download the source code and compile RStudio on your system.

There is also a nice option to download a zip/tarball version of RStudio. This version doesn't require installation, so if you plan to use it on a system where you don't have admin/root privileges this is the one you want to download. This version is also ideal to install on USB data sticks – giving you a portable version that you can travel with.

Note from Miles: RStudio is still in beta, thus you may come across an error or two. However updates are released frequently and the community are only too happy to help out. I have been running RStudio on Windows XP/7/Server 2008, Ubuntu and Debian with zero issues as yet. Current stable version is 0.99.473 – as of the 18th August 2015.

plink

> "PLINK is a free, open-source whole genome association analysis toolset, designed to perform a range of basic, large-scale analyses in a computationally efficient manner."

`plink` has long been one of the major pieces of software used for case/control GWAS analyses. It's lightweight design, ease of use and speed have made it widely popular. A recent update to the long-standing 2009 version (v1.07) has seen a dramatic shift in terms of capability and performance. The new 1.9 version of `plink` is capable of working with next-gen sequencing data while maintaining the speed of the previous version. Even though we will be dealing with a small number of SNPs generated from an array platform we will be using this new version in our workshop.

Version 1.9 software and documentation can be found here: https://www.cog-genomics.org/plink2

For reference, the 1.07 version of `plink` can be found here: http://pngu.mgh.harvard.edu/~purcell/plink/.

`plink` has excellent documentation in the form of an on-line manual found at the above links, we encourage everyone to get familiar with this. It is our opinion that the older 1.07 version manual is much more detailed and informative - so we suggest starting here for the basics. Some of the functionality has changed between versions, but the staples seem to have stayed the same. The documentation for 1.9 is fine and will hopefully get better as development continues.

Setting up for this workshop

Miles has made all the required files, code, and software available through his GitHub repository. This should hopefully make it reasonably straight forward for us to get up and running.

Download the archive

This is probably the easiest way for most people to get the required files. The repository can be found on Miles GitHub account, titled "Intro_to_GWAS", https://github.com/sirselim/Intro_to_GWAS.

If you point your browser to this address you find yourself in the "Intro_to_GWAS" repository. You should see an option on the right hand side to download the repository as a zip ('Download ZIP'). Download this file and then extract it to a suitable place on your system.

Clone into the repository

If you are familiar with git and GitHub you will be able to easily clone into the repository from your local machine. The below is how one would do this on Linux:

```
# clone into the Intro_to_GWAS repository
git clone https://github.com/sirselim/Intro_to_GWAS.git
```

Set Working Directory

First we need to ensure that our working directory is the base directory of the repository that we downloaded earlier.

```
getwd() # check working directory
```

This should be something like "/home/miles/GenomeInformatics/Intro_to_GWAS". If it is not, set your working directory to the 'Intro_to_GWAS/' directory (this is the file that you downloaded, or cloned, from GitHub).

Automated script for set up

In an attempt to make initial set up easy we have created an `R` script that will:

The script is provided to automatically:

- set up the correct directory structure for the workshop
- extract the example data to `example/`
- download an appropriate OS specific version of `plink`
- extract the `plink` binary to `bin/`
- download, build, install and load all R packages you'll need

With your working directory correctly set run the below line of code in R and you should be ready to go.

```
# source script
source('scripts/GWAS_workshop_setup.R')
```

Note: Windows users - if you encounter any errors at this stage try restarting RStudio and running `source('scripts/GWAS_workshop_setup.R')` again. In testing some Windows machines had issues with installing and then loading certain libraries, however upon a restart of RStudio these issues resolved.

If you are interested in the process behind the above automation you can find more detail in the **Appendix** at the end of this document.

Testing `plink`

`plink` is a standalone program that runs from the command line. If you were running `plink` straight from the command-line this is a very basic example of what you would enter:

```
# basic plink execution
bin/plink --file example/wgas1 --out example/plink
```

To be able to use `plink` from within `R` we have to use the `system()` function. So to perform the example above we need to run the following within `R`:

```
# as plink is an external program commands need to be wrapped in the R system() function
system('bin/plink --file example/wgas1 --out example/plink')
# you should see the basic overview of plink functions
```

Is `plink` running in `R`? When running the above what do you see?

Hopefully upon running the above call you should see an overview output from `plink`. It will give you some basic information about the software itself, the arguments entered, detail on system resources and then information on the contents of our genotype data. In this example we see that `plink` reports that there are **228694** variants (SNPs) for **90** people. If this is all performing as expected we can move on.

plink file structure and data recoding

The PED/MAP files are a very common format for storing SNP genotype data, they are also referred to as linkage format (they are able to store pedigree information as well). The file structure for these files is important to understand, so we'll step through this below:

PED files

The PED file is a white-space (space or tab) delimited file: the first six columns are mandatory:

```
Family ID
Individual ID
Paternal ID
Maternal ID
Sex (1=male; 2=female; other=unknown)
Phenotype
```

Lets load the example plink PED file into R and explore is a little:

```
example.ped <- read.table('bin/toy.ped', head = F)
example.ped
```

FAMID	ID	PID	MID	SEX	pheno	A1	A2	B1	B2
1	1e+09	0	0	1	1	0	0	1	1
1	1e+09	0	0	1	2	1	1	1	2

MAP files

By default, each line of the MAP file describes a single marker and must contain exactly 4 columns:

```
chromosome (1-22, X, Y or 0 if unplaced)
rs# or snp identifier
Genetic distance (morgans)
Base-pair position (bp units)
```

Lets load the example MAP file into R and explore is a little:

```
example.map <- read.table('bin/toy.map', head = F)
example.map
```

chr	SNP	gpos	bp
1	rs0	0	1000
1	rs10	0	1001

For a more in-depth look into PED/MAP files refer to the online manual here.

Binary plink files

To save space and time, you can make a binary ped file (`*.bed`). This will store the pedigree/phenotype information in a separate file (`*.fam`) and create an extended MAP file (`*.bim`) (which contains information about the allele names, which would otherwise be lost in the BED file). To create these files use the command:

```
plink --file mydata --make-bed
```

which creates (by default):

```
plink.bed      ( binary file, genotype information )
plink.fam      ( first six columns of mydata.ped )
plink.bim      ( extended MAP file: two extra cols = allele names)
```

The .fam and .bim files are still plain text files: these can be viewed with a standard text editor. Do not try to view the .bed file however: it is a compressed file and you'll only see lots of strange characters on the screen...

For a more in-depth look into binary plink files refer to appropriate section of the online manual here.

We will now create binary `plink` files from the original ped/map data.

The following would be how the call is made from the command line:

```
# call plink to read example SNP data, convert to binary ped format
bin/plink --file example/wgas1 --out example/wgas2
```

To do this in R is exactly the same as before:

```
# convert the example data from ped/map to binary ped (much 'faster' format)
system('bin/plink --file example/wgas1 --out example/wgas2')
```

We now have binary plink files (bed, bim, fam) of our example data. This is where we will start with our GWAS analysis.

GWAS: Genomewide Association Study

Now that we've got the software installed and up and running we are ready to begin.

Quality Control in plink (QC)

PLINK will generate a number of standard summary statistics that are useful for quality control (e.g. missing genotype rate, minor allele frequency, Hardy-Weinberg equilibrium failures and non-Mendelian transmission rates). These can also be used as thresholds for subsequent analyses (described in the next section).

Initial QC is one of the most important aspects to consider before jumping head first into analysis. It is crucial to have a good understanding of your data and what it should 'look' like, i.e.:

- how many samples?
- how many males/females?
- number of cases/controls?
- total number of SNPs

. . . you get the point. If you notice something not quite right from the get-go, life is generally a lot easier!

We'll now step through a few of the common QC methods, for a more detailed list please refer to the Summary Statistics section of the plink manual.

Missing genotypes

To generate a list genotyping/missingness rate statistics:

```
system('bin/plink --bfile example/wgas2 --missing --out example/plink_qc')
```

This option creates two files:

```
plink.imiss
plink.lmiss
```

which detail missingness by individual and by SNP (locus), respectively. For individuals, the format is:

```
FID               Family ID
IID               Individual ID
MISS_PHENO        Missing phenotype? (Y/N)
N_MISS            Number of missing SNPs
N_GENO            Number of non-obligatory missing genotypes
F_MISS            Proportion of missing SNPs
```

For each SNP, the format is:

```
SNP               SNP identifier
CHR               Chromosome number
N_MISS            Number of individuals missing this SNP
N_GENO            Number of non-obligatory missing genotypes
F_MISS            Proportion of sample missing for this SNP
```

We can now load the summary files we have just created, allowing us to explore the data in R:

```
# load individual missing information
qc_missing_ind <- read.table('example/plink_qc.imiss', head = T)
# load SNP missing information
qc_missing_snp <- read.table('example/plink_qc.lmiss', head = T)
```

[**SELF TEST**] Using your knowledge of R can you explore the statistics we have just calculated? What can you conclude about the missingness in this study?

Test of missingness by case/control status

To obtain a missing chi-sq test (i.e. does, for each SNP, missingness differ between cases and controls?), use the option:

```
system('bin/plink --bfile example/wgas2 --test-missing --out example/plink_qc')
```

which generates a file

```
plink.missing
```

which contains the fields

```
CHR         Chromosome number
SNP         SNP identifier
F_MISS_A    Missing rate in cases
F_MISS_U    Missing rate in controls
P           Asymptotic p-value (Fisher's exact test)
```

The actual counts of missing genotypes are available in the plink.lmiss file, which is generated by the –missing option.

We can now load the summary file we have just created, allowing us to explore the data in R:

```
# load case/control missing information
qc_missing_study <- read.table('example/plink_qc.missing', head = T)
```

[**SELF TEST**] What can you conclude about the case/control missingness in this study?

Hardy-Weinberg Equilibrium

To generate a list of genotype counts and Hardy-Weinberg test statistics for each SNP, use the option:

```
system('bin/plink --bfile example/wgas2 --hardy --out example/plink_qc')
```

which creates a file:

```
plink.hwe
```

This file has the following format

```
SNP             SNP identifier
TEST            Code indicating sample
A1              Minor allele code
A2              Major allele code
GENO            Genotype counts: 11/12/22
O(HET)          Observed heterozygosity
E(HET)          Expected heterozygosity
P               H-W p-value
```

For case/control samples, each SNP will have three entries (rows) in this file, with TEST being either ALL, AFF (cases only) or UNAFF (controls only). For quantitative traits, only a single row will appear for each SNP, labelled ALL(QT).

Only founders are considered for the Hardy-Weinberg calculations – ie. for family data, any offspring are ignored.

We can now load the summary file we have just created, allowing us to explore the data in R:

```
# load HWE information
qc_hwe <- read.table('example/plink_qc.hwe', head = T)
```

[**SELF TEST**] What can you conclude about the HWE in this study?

Allele frequency

To generate a list of minor allele frequencies (MAF) for each SNP, based on all founders in the sample:

```
system('bin/plink --bfile example/wgas2 --freq --out example/plink_qc')
```

will create a file:

```
plink.frq
```

with five columns:

```
CHR       Chromosome
SNP       SNP identifier
A1        Allele 1 code (minor allele)
A2        Allele 2 code (major allele)
MAF       Minor allele frequency
NCHROBS   Non-missing allele count
```

We can now load the summary file we have just created, allowing us to explore the data in R:

```
# load allele freq information
qc_afreq <- read.table('example/plink_qc.frq', head = T)
```

[**SELF TEST**] What can you conclude about the allele frequency in this study?

Running QC on our data

We are going to perform our initial QC using fairly standard (default) settings, these will be:

- minor allele frequency > 0.05 (5%)
- genotyping rate > 0.05 (5%)
- missing individuals > 0.05 (5%)
- Hardy-Weinbery Equilibrium p>0.001

```
system('bin/plink --bfile example/wgas2 --maf 0.01 --geno 0.05 --mind 0.05
                  --hwe 0.001 --make-bed --out example/wgas3')
```

[**SELF TEST**] What does the above plink output tell you in terms of QC information?

Basic GWAS (association testing)

```
# now perform a basic association analysis
system('bin/plink --bfile example/wgas3 --assoc --out example/example_analysis')

# load the association results into R
assoc.results <- read.table('example/example_analysis.assoc', head = T)
```

Exploring association statistics

```
# examine the results
head(assoc.results)
```

```
##   CHR        SNP      BP A1     F_A     F_U A2  CHISQ      P     OR
## 1   1  rs3094315  792429  G 0.14890 0.08537  A 1.6840 0.1944 1.8750
## 2   1  rs4040617  819185  G 0.13540 0.08537  A 1.1110 0.2919 1.6780
## 3   1  rs4075116 1043552  C 0.04167 0.07317  T 0.8278 0.3629 0.5507
## 4   1  rs9442385 1137258  T 0.37230 0.42680  G 0.5428 0.4613 0.7966
## 5   1 rs11260562 1205233  A 0.02174 0.03659  G 0.3424 0.5585 0.5852
## 6   1  rs6685064 1251215  C 0.38540 0.43900  T 0.5253 0.4686 0.8013
```

```
tail(assoc.results)
```

```
##        CHR       SNP       BP A1     F_A     F_U A2   CHISQ       P     OR
## 179488  22 rs6151429 49353621  C 0.04167 0.02439  T 0.40530 0.52440 1.7390
## 179489  22 rs6009945 49379357  C 0.28120 0.46340  A 6.33100 0.01187 0.4531
## 179490  22 rs9616913 49405670  C 0.14580 0.06098  T 3.34000 0.06762 2.6290
## 179491  22  rs739365 49430460  C 0.45830 0.35370  T 2.00300 0.15700 1.5460
## 179492  22 rs6010063 49447077  G 0.42710 0.46340  A 0.23650 0.62680 0.8632
## 179493  22 rs9616985 49519949  C 0.03125 0.03659  T 0.03865 0.84410 0.8495
```

```
# explore p-values
summary(assoc.results$P)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000005 0.1950000 0.4486000 0.4635000 0.7254000 1.0000000
```

```
table(assoc.results$P <= 0.05)
```

```
##
##  FALSE   TRUE
## 166204  13289
```

[SELF TEST] What do the above statistics represent? If you are unsure you'll find this section of the plink manual very helpful: http://pngu.mgh.harvard.edu/~purcell/plink/anal.shtml#cc

Correcting for multiple-testing burden

In R

```
# create an adjusted p-value column
assoc.results$adj_P <- p.adjust(assoc.results$P, method = 'BH', n = nrow(assoc.results))
summary(assoc.results$adj_P)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06375 0.77170 0.89650 0.85590 0.96550 1.00000
```

```
table(assoc.results$adj_P <= 0.1)
```

```
##
##  FALSE   TRUE
## 179489      4
```

In plink

You can also apply adjustment in plink using the `--adjust` argument.

Genomic inflation factor and population structure

The genomic inflation factor (λ) is defined as the ratio of the median of the empirically observed distribution of the test statistic to the expected median, thus quantifying the extent of the bulk inflation and the excess false positive rate.

$$[\lambda = median(\chi^2)/0.456]$$
$$[\chi^2_{adjusted} = \chi^2/\lambda]$$

Genomic inflation factor (λ) and quantile–quantile (Q–Q) plots were used to compare the genome-wide distribution of the test statistic with the expected null distribution. Inflated (λ) values or residual deviations in the Q–Q plot may point to undetected sample duplications, unknown familial relationships, a poorly calibrated test statistic, systematic technical bias or gross population stratification.

We can calculate this manually in R from our association statistics:

```
# for CHISQ
z <- sqrt(assoc.results$CHISQ)
# for P value
# z <- qnorm(assoc.results$P/2)
## calculates lambda
lambda = round(median(z^2)/.454,3)
```

From our association statistics we conclude that our λ is **1.265**, thus we confirm genomic inflation (which we expected to see as we now know there is population stratification).

It should also be noted that `plink` calculates the genomic inflation factor when using the `--adjust` command in an association analyses. When the `--adjust` command is used, the log file records the inflation factor calculated for the genomic control analysis, and the mean chi-squared statistic (that should be 1 under the null).

Q-Q plotting

The Q–Q plot is a useful visual tool to mark deviations of the observed distribution from the expected null distribution.

To generate our Q-Q plots we will be using the `Haplin` package available from CRAN (here). This package produces really 'pretty' Q-Q plots (in our opinion), and includes useful information such as 95% Confidence Intervals. First we'll make sure the package is installed and available:

```
# install Haplin
install.packages('Haplin')
```

Then we'll load the `Haplin` package:

```
# load package
require("Haplin")
```

Now we can generate the Q-Q plot using the `pQQ()` function:

```
pQQ(assoc.results$P)
```

As indicated by our inflated λ calculated from above we see deviation from the expected in our Q-Q plot, confirming genomic inflation.

Manhattan plotting

A Manhattan plot is named for it's abstract similarity to the skyscrapers of the Manhattan skyline, and is a visual method of portraying GWAS results. Throughout the course of this workshop you'll get lots of practise making your own Manhattan plots. For this first one just keep in mind our inflated λ, so we now know to take this result with a large grain of salt!

To generate Manhattan plots we like to use a very useful package call `qqman`, the CRAN version is available from here, with the source code also available on GitHub. This package is created and maintained by Stephen Turner, who also has a really cool, informative blog called Getting Genetics Done that we advise everyone to check out.

We're going to download and install the latest version of qqman from GitHub:

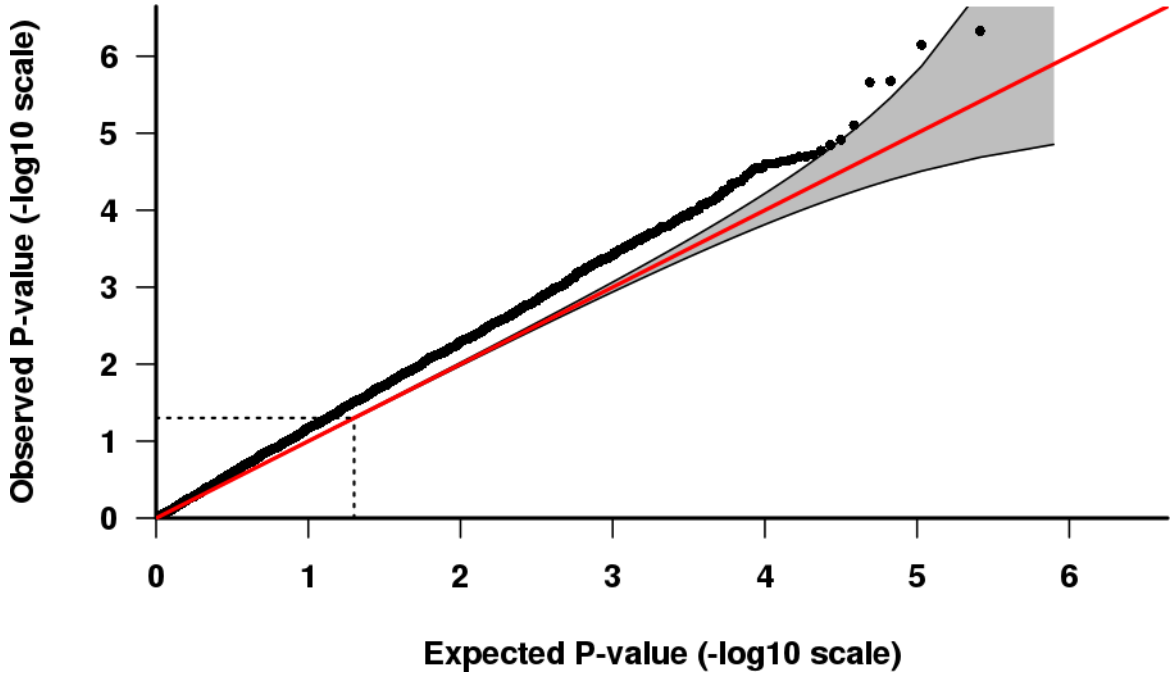

Figure 1: Q-Q plot of inital unadjusted plink association results with 95% CI.

```
# download the latest version of qqman from Stephen Turners GitHub repository
require("devtools") # used to install packages from source
install_github("stephenturner/qqman", ref="dev")
```

Now we load the `qqman` package:

```
# load package
require("qqman")
```

Now to create the Manhattan plot:

```
# generate a Manhattan plot of your results
manhattan(assoc.results)
```

We'd also like to note that Stephen's package also has the option to create Q-Q plots using the `qq()` function. Feel free to try it out with your results to date. What do you notice?

Exploring population effects on association results

The example data actually contains genotype and phenotype data from two distinct populations, a Chinese population and a Japanese population. This information can be provided to plink in the form of a covariate file - such a file has been provided for you in the for of the `pop.cov` file.

First lets load this into R and explore the numbers.

```
# load GWAS population information
pop.cov <- read.table('example/pop.cov')
```

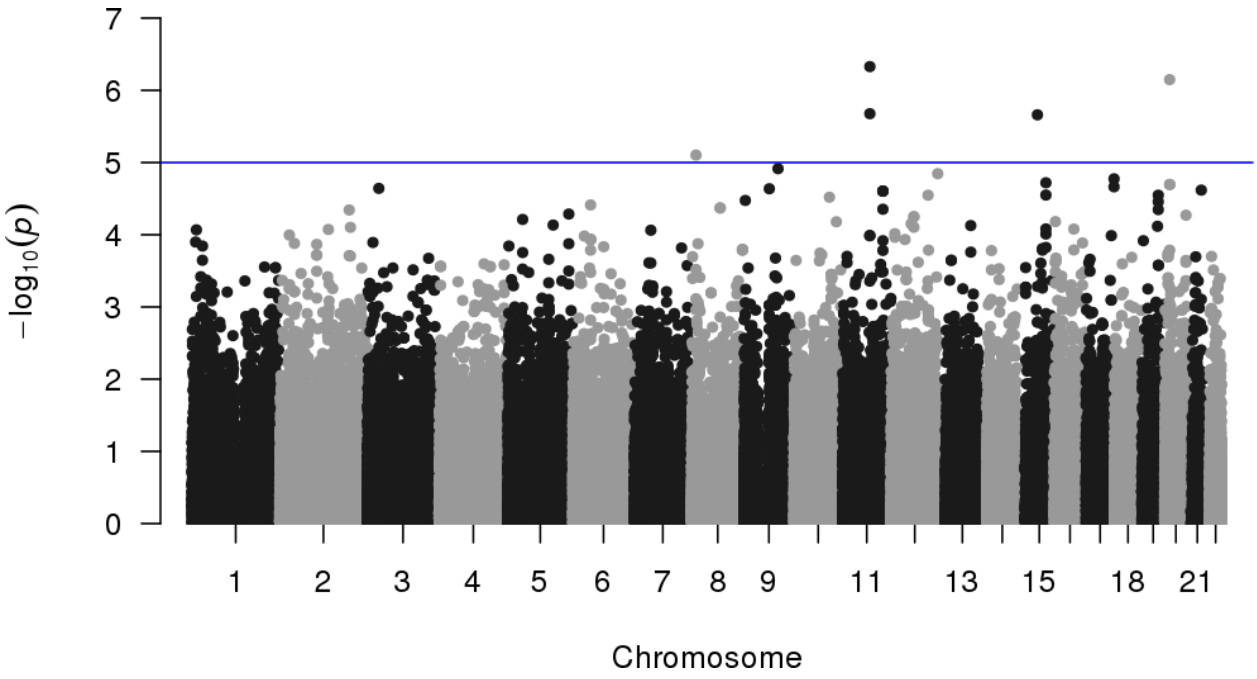

Figure 2: GWAS Manhattan plot of initial association results

```
# name the columns (family ID, individual ID, and population)
names(pop.cov) <- c('FAMID', 'IID', 'POP')
# assign levels to the population column
pop.cov$POP <- factor(pop.cov$POP, labels = c('Chinese', 'Japanese'))
# check population information
table(pop.cov$POP)
```

```
##
##  Chinese Japanese
##       45       45
```

Running a stratified association in plink

Now lets run a stratified analysis in plink which allows us to adjust for population effects when performing association tests.

```
# stratified GWAS in plink
system('bin/plink --bfile example/wgas3 --assoc --mh
                  --within example/pop.cov --out example/example_analysis_strat')

# read in the stratified results
assoc.strat <- read.table('example/example_analysis_strat.cmh', head = T)
```

Now we will calculate our genomic inflation factor:

```
# for CHISQ
z <- sqrt(assoc.strat$CHISQ)
# for P value
```

```
# z <- qnorm(assoc.results$P/2)
## calculates lambda
lambda.strat = round(median(z^2)/.454,3)
```

We see that λ is **1.019**, much better!

For completeness the Q-Q plot:

```
pQQ(assoc.strat$P)
```

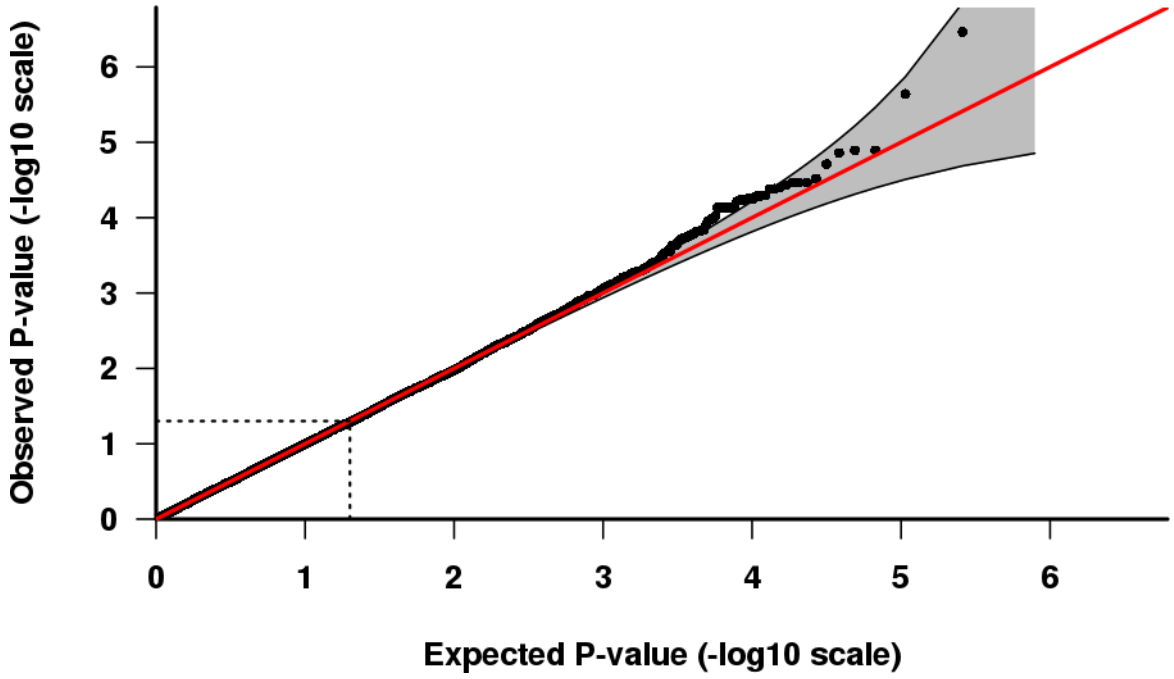

Figure 3: Q-Q plot of population adjusted plink association results with 95% CI.

Now we can generate a new Manhattan plot which we should have more confidence in.

```
# generate a Manhattan plot of the stratified results
manhattan(assoc.strat)
```

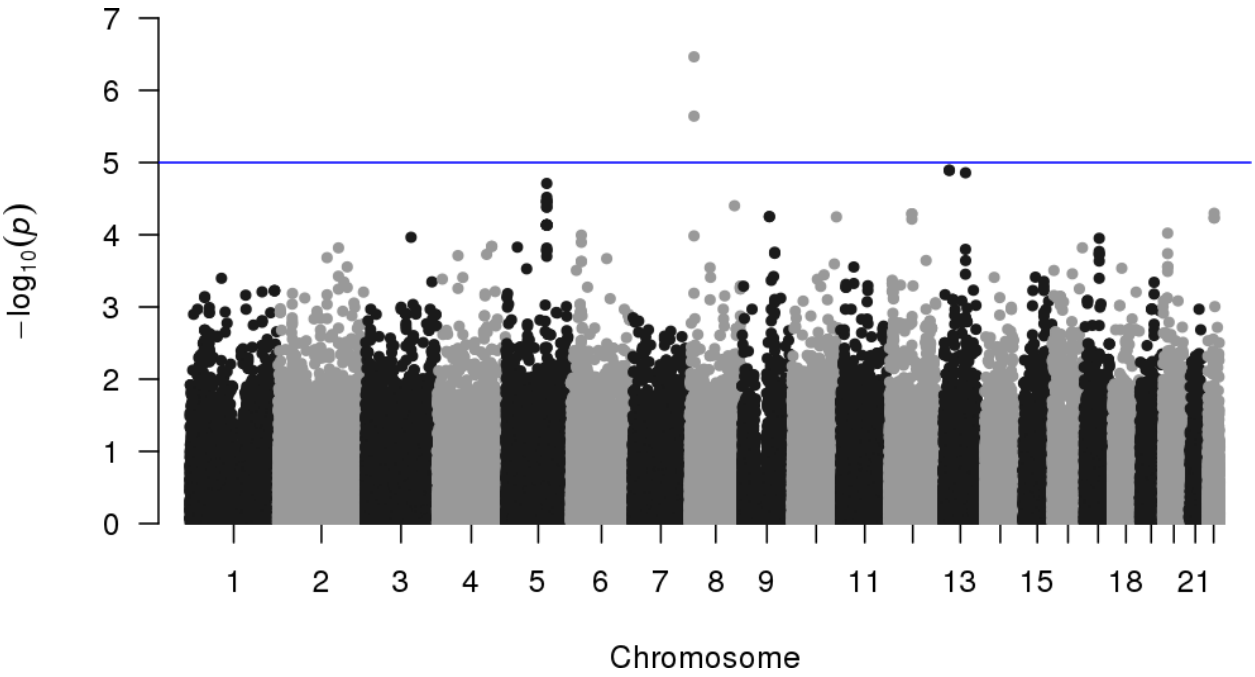

Figure 4: GWAS Manhattan plot of stratified association results. In this test we have accounted for population effects.

Exploring population stratification further using PCA and MDS

This section covers a more in-depth look at identifying and visualising population stratification within a cohort. We will be using a combination of different R packages:

- `gdsfmt` bioconductor
  - *"This package provides a high-level R interface to CoreArray Genomic Data Structure (GDS) data files, which are portable across platforms and include hierarchical structure to store multiple scalable array-oriented data sets with metadata information. It is suited for large-scale datasets, especially for data which are much larger than the available random-access memory."*
- `SNPRelate` bioconductor
  - *"SNPRelate is also designed to accelerate two key computations on SNP data using parallel computing for multi-core symmetric multiprocessing computer architectures: Principal Component Analysis (PCA) and relatedness analysis using Identity-By-Descent measures."*

Download and install the GitHub versions of both packages:

```
# gdsfmt
install_github("zhengxwen/gdsfmt")
# SNPRelate
install_github("zhengxwen/SNPRelate")
```

For the sake of this workshop the gds file has already been created from the `plink` genotype data we were using earlier. Now to load the packages and open a link to our gds file (genotype data):

```
# load required packages
require("gdsfmt")
```

```r
require("SNPRelate")
# load the gds file (genotype data of the same cohort as before, just in gds format)
genofile <- snpgdsOpen('example/test.gds')

# genotype info for the first 5 SNPs of the first 3 individuals
read.gdsn(index.gdsn(genofile, "genotype"), start = c(1,1), count = c(5,3))
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
## [3,]    1    1    0
## [4,]    0    0    1
## [5,]    1    1    0
```

```r
# perform the PCA
# set a random seed
set.seed(1500)
# Try different LD thresholds for sensitivity analysis
snpset <- snpgdsLDpruning(genofile, ld.threshold = 0.4)
```

```
## SNP pruning based on LD:
## Excluding 0 SNP on non-autosomes
## Excluding 0 SNP (monomorphic: TRUE, < MAF: NaN, or > missing rate: NaN)
## Working space: 89 samples, 179510 SNPs
##  Using 1 (CPU) core
##  Sliding window: 500000 basepairs, Inf SNPs
##  |LD| threshold: 0.4
## Chromosome 1: 32.56%, 4966/15250
## Chromosome 2: 32.24%, 4680/14516
## Chromosome 3: 33.26%, 3956/11894
## Chromosome 4: 35.51%, 3482/9805
## Chromosome 5: 32.37%, 3758/11608
## Chromosome 6: 31.26%, 3590/11483
## Chromosome 7: 34.27%, 3163/9230
## Chromosome 8: 32.51%, 3137/9650
## Chromosome 9: 33.56%, 2926/8719
## Chromosome 10: 30.65%, 3387/11052
## Chromosome 11: 30.76%, 3081/10015
## Chromosome 12: 34.20%, 3167/9260
## Chromosome 13: 35.86%, 2207/6154
## Chromosome 14: 35.05%, 2045/5835
## Chromosome 15: 36.02%, 2053/5699
## Chromosome 16: 34.34%, 2129/6200
## Chromosome 17: 39.13%, 1880/4804
## Chromosome 18: 36.38%, 1857/5104
## Chromosome 19: 41.94%, 1223/2916
## Chromosome 20: 33.31%, 1679/5041
## Chromosome 21: 36.98%, 915/2474
## Chromosome 22: 37.31%, 1045/2801
## 60326 SNPs are selected in total.
```

```r
# Get all selected snp id
snpset.id <- unlist(snpset)
```

```
# Run PCA
pca.snp <- snpgdsPCA(genofile, snp.id = snpset.id, num.thread = 2)

## Principal Component Analysis (PCA) on SNP genotypes:
## Excluding 119184 SNPs on non-autosomes
## Excluding 0 SNP (monomorphic: TRUE, < MAF: NaN, or > missing rate: NaN)
## Working space: 89 samples, 60326 SNPs
##  Using 2 (CPU) cores
## PCA: the sum of all working genotypes (0, 1 and 2) = 2177894
## PCA: Wed Aug 19 17:04:27 2015    0%
## PCA: Wed Aug 19 17:04:27 2015    100%
## PCA: Wed Aug 19 17:04:27 2015    Begin (eigenvalues and eigenvectors)
## PCA: Wed Aug 19 17:04:27 2015    End (eigenvalues and eigenvectors)

# variance proportion (%)
pc.percent <- pca.snp$varprop*100
head(round(pc.percent, 2))

## [1] 1.77 1.26 1.24 1.23 1.22 1.21

# make a data.frame (used for plotting)
tab.pca <- data.frame(sample.id = pca.snp$sample.id,
                      EV1 = pca.snp$eigenvect[,1],    # the first eigenvector
                      EV2 = pca.snp$eigenvect[,2],    # the second eigenvector
                      pop = as.factor(c(rep('Chinese', 45), rep('Japanese', 44))),
                      stringsAsFactors = FALSE)
# view the head of this data.frame
head(tab.pca)

##   sample.id         EV1          EV2     pop
## 1   NA18524 -0.10588863  0.095591513 Chinese
## 2   NA18526 -0.10388360 -0.002753589 Chinese
## 3   NA18529 -0.09525804  0.003146420 Chinese
## 4   NA18532 -0.08771483 -0.133993565 Chinese
## 5   NA18537 -0.10406642  0.064147394 Chinese
## 6   NA18540 -0.09847035  0.022680463 Chinese

plot(tab.pca$EV2, tab.pca$EV1, xlab="eigenvector 2", ylab="eigenvector 1",
     col = ifelse(tab.pca$pop == 'Chinese', 'blue', 'orange'), pch = 19)
legend("bottomleft", legend = levels(tab.pca$pop), pch = 19,
       col = c('blue', 'orange'), bty = 'n')
```

We can explore additional principal component scores:

```
lbls <- paste("PC", 1:4, "\n", format(pc.percent[1:4], digits = 2), "%", sep = "")
pairs(pca.snp$eigenvect[,1:4], col = ifelse(tab.pca$pop == 'Chinese', 'blue', 'orange'),
      labels = lbls)
```

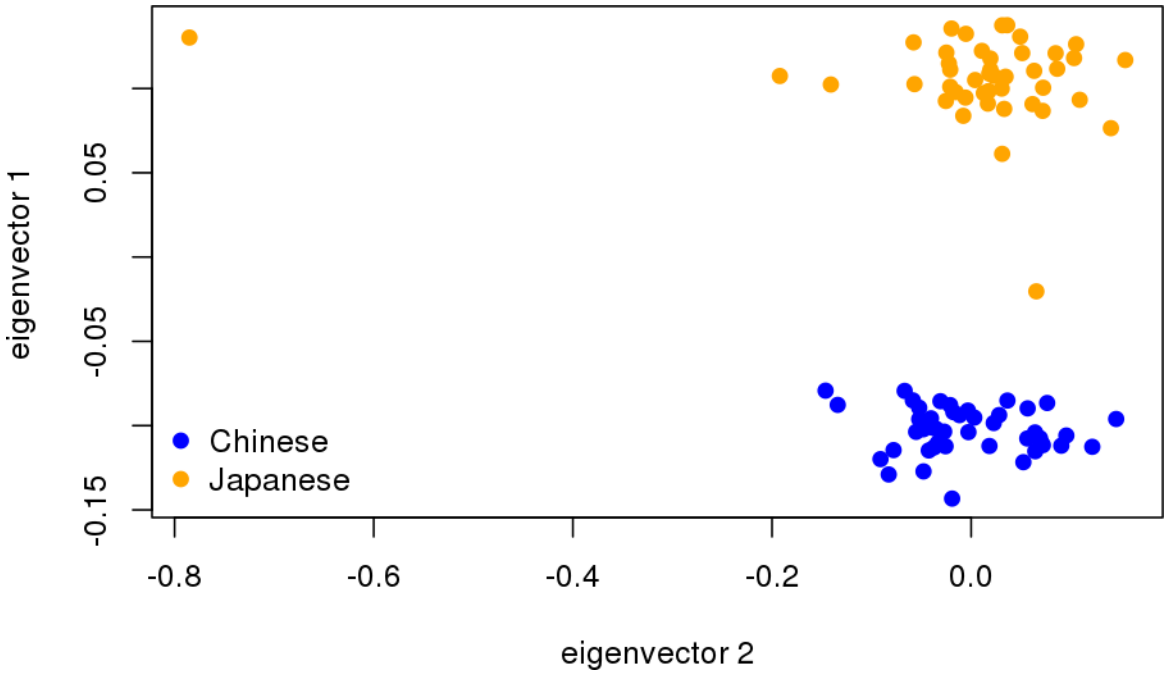

Figure 5: PCA plot of LD-pruned SNP data showing population stratification.

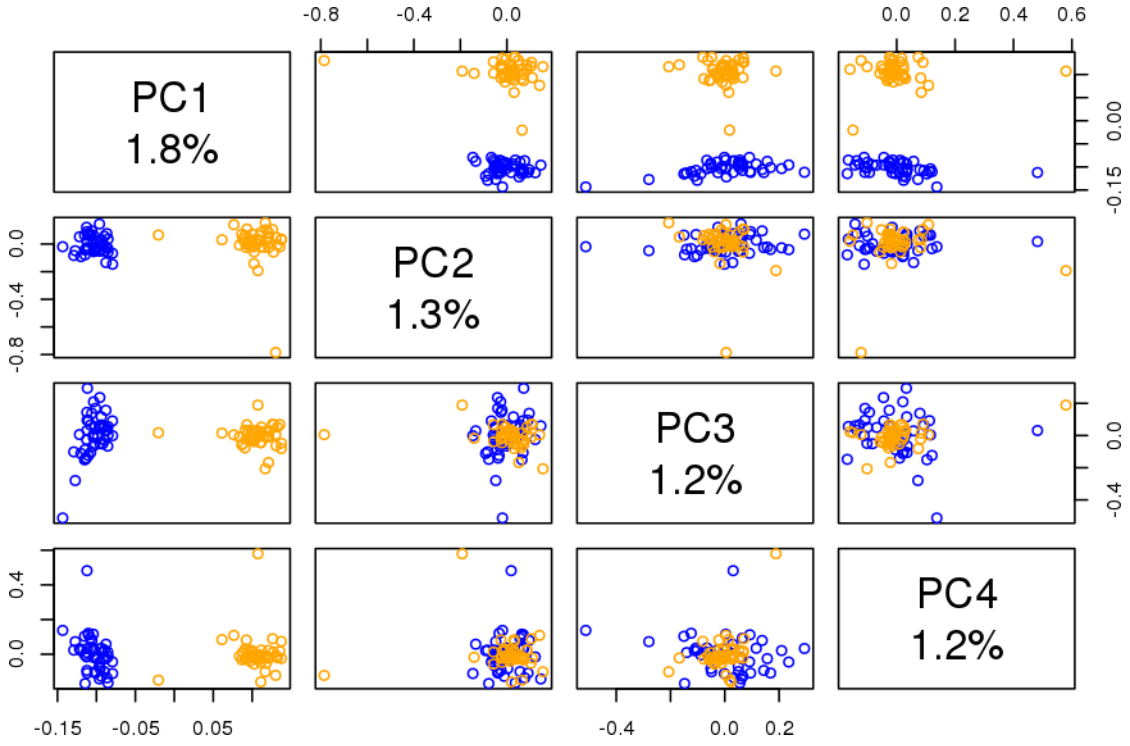

Figure 6: Multiple PCA plots showing the relationship between the first 4 components.

Parallel coordinates plot for the top principal components:

```
# install the MASS package (if needed)
install.packages('MASS')


require("MASS")

datpop <- factor(tab.pca$pop)
parcoord(pca.snp$eigenvect[,1:16], col = datpop)
```

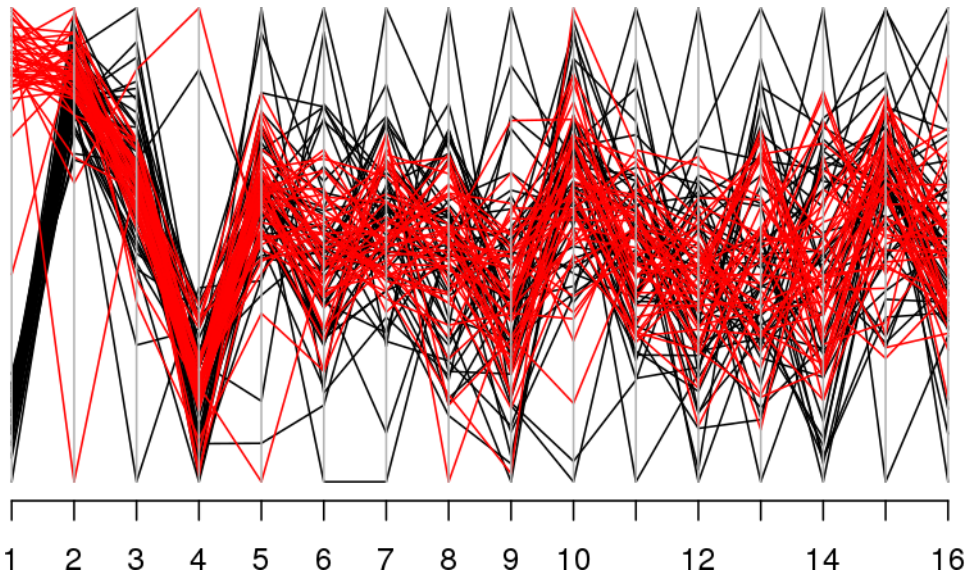

Figure 7: Parallel coordinates plot for the top principal components.

```
pop_code <- datpop
```

Identity by state (IBS) Analysis

For the n individuals in a sample, `snpgdsIBS()` can be used to create a n×n matrix of genome-wide average IBS pairwise identities:

```
# generate ibs matrix
ibs <- snpgdsIBS(genofile, num.thread = 2)

## Identity-By-State (IBS) analysis on SNP genotypes:
## Excluding 0 SNP on non-autosomes
## Excluding 0 SNP (monomorphic: TRUE, < MAF: NaN, or > missing rate: NaN)
## Working space: 89 samples, 179510 SNPs
##  Using 2 (CPU) cores
## IBS: the sum of all working genotypes (0, 1 and 2) = 7375697
## IBS: Wed Aug 19 17:04:28 2015    0%
## IBS: Wed Aug 19 17:04:28 2015    100%
```

To perform multidimensional scaling analysis on the n×n matrix of genome-wide IBS pairwise distances:

```
loc <- cmdscale(1 - ibs$ibs, k = 2)
x <- loc[, 1]; y <- loc[, 2]
race <- as.factor(pop_code)
# create plot
plot(x, y, xlab = "", ylab = "", pch = 19,
     col = ifelse(tab.pca$pop == 'Chinese', 'blue', 'orange'))
legend("topright", legend = levels(race), text.col = c('blue', 'orange'), bty = 'n')
```

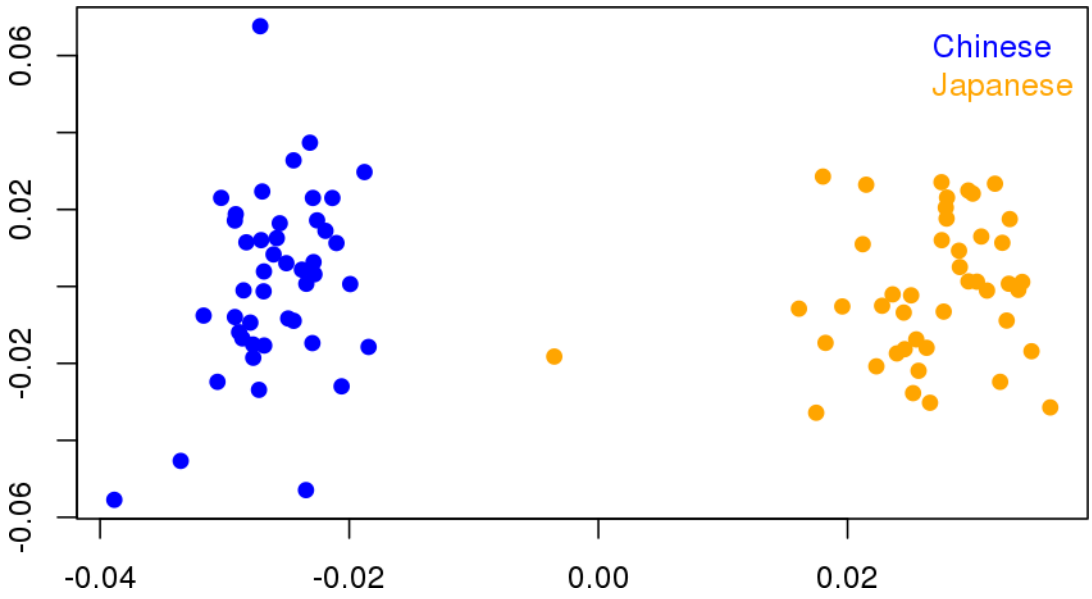

Figure 8: MDS plot of IBS matrix shows clear seperation of both populations.

Adding in additional data

Imagine that you obtained the results above and performed some targeted genotyping around the Chr 8 SNPs of interest. We are able to merge that data back into the original GWAS set and rerun our analyses.

Merging data in plink

```
# plink data merge
system('bin/plink --bfile example/wgas3 --merge example/extra.ped example/extra.map
                  --make-bed --out example/followup')
```

Rerun stratified association analysis

```
# stratified analysis
system('bin/plink --bfile example/followup --assoc --mh
                  --within example/pop.cov --out example/followup_analysis_strat')

# load data into R
followup.results <- read.table('example/followup_analysis_strat.cmh', head = T)
```

Association results and Manhattan plot of 'new' dataset

Again we will calculate our genomic inflation factor:

```
# for CHISQ
z <- sqrt(followup.results$CHISQ)
# for P value
# z <- qnorm(followup.results$P/2)
## calculates lambda
lambda.add = round(median(z^2)/.454,3)
```

We see that λ is **1.019**, the same as before - we only added a few extra SNPs so this isn't a surprise.

Now the Q-Q plot:

```
pQQ(followup.results$P)
```

. . . and finally the Manhattan plot:

```
# Manhattan plot
manhattan(followup.results)
```

We can also explore the top 'hits' in our results:

```
head(followup.results[order(followup.results$P),], n = 10)
```

```
##        CHR        SNP        BP A1    MAF A2 CHISQ         P       OR
## 85075    8  rs7835221  12878098  G 0.4045  A 57.54 3.305e-14  0.01914
## 85077    8 rs11204005  12895576  A 0.4775  G 25.99 3.432e-07  0.09950
## 85079    8  rs2460338  12914531  G 0.4775  C 22.35 2.277e-06  0.09967
```

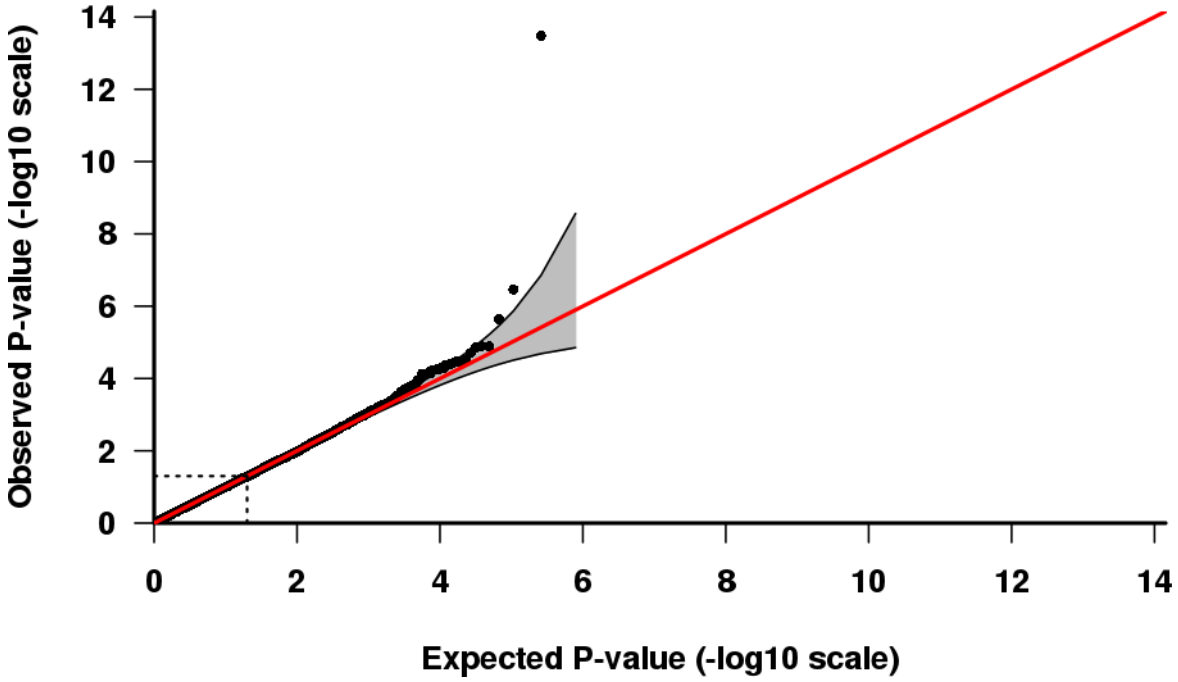

Figure 9: Q-Q plot of ammended population adjusted plink association results with 95% CI.

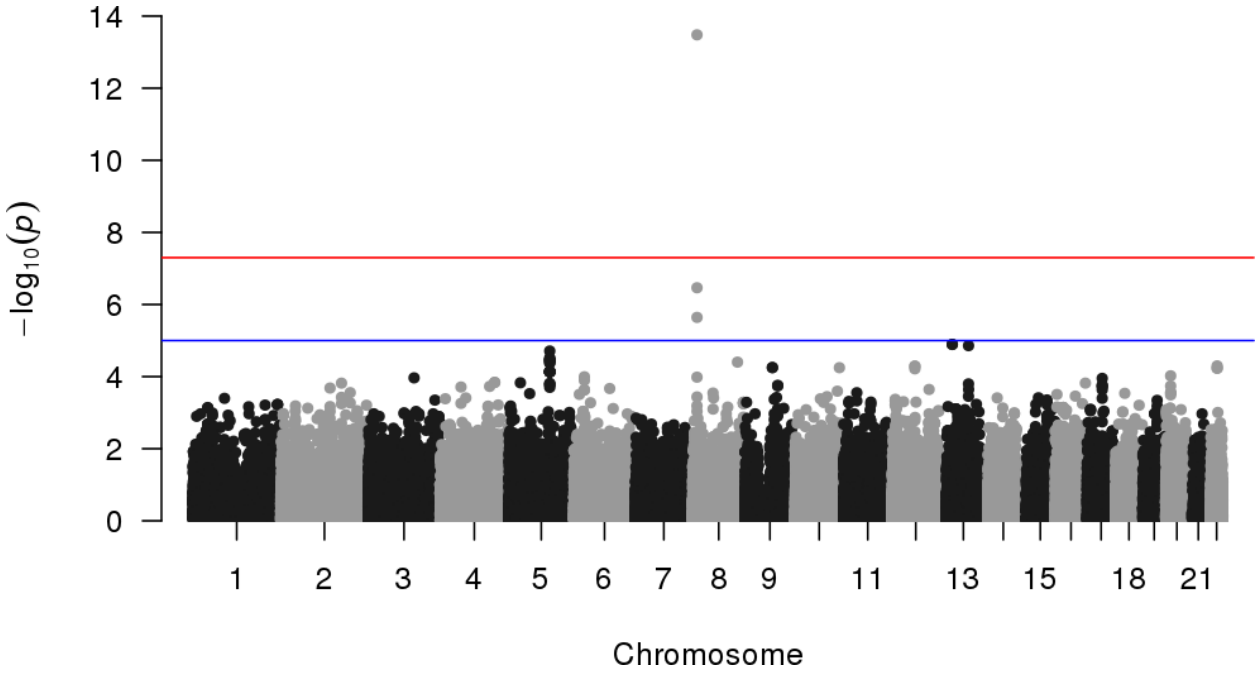

Figure 10: Updated Manhattan plot showing with additional genotype data for the Chr 8 region of interest.

```
## 133850  13  rs4941815  35207635  C 0.4663  T 19.04 1.280e-05  6.53600
## 133852  13  rs4943327  35209062  T 0.4663  C 19.04 1.280e-05  6.53600
## 136481  13  rs9531117  80316559  T 0.2159  C 18.89 1.386e-05  7.13300
## 57992    5   rs839220 113956183  C 0.1910  T 18.24 1.949e-05 12.19000
## 57990    5   rs373386 113942503  C 0.2706  G 17.40 3.033e-05  8.69300
## 57984    5   rs444800 113925807  G 0.2640  C 17.17 3.424e-05  8.34400
## 57986    5   rs454540 113926349  A 0.2640  C 17.17 3.424e-05  8.34400
##            SE      L95     U95
## 85075  0.7713 0.004222  0.0868
## 85077  0.5164 0.036160  0.2738
## 85079  0.5605 0.033220  0.2990
## 133850 0.4643 2.631000 16.2400
## 133852 0.4643 2.631000 16.2400
## 136481 0.5345 2.502000 20.3300
## 57992  0.6368 3.499000 42.4700
## 57990  0.5456 2.984000 25.3300
## 57984  0.5363 2.917000 23.8700
## 57986  0.5363 2.917000 23.8700
```

Exploring genotype associations

More detailed analysis of association results using `plink`. In this section we will cover the following:

- advanced population-specific effects (allele freq, etc.)
- examine genotype models
- explore sex-specific effects

Population-specific effects

So now that we have established that there are 2 populations in our data set we can examine whether this association varies between them. When using the Cochran-Mantel-Haenszel test, we can request an additional Breslow-Day test for heterogeneous odds ratios between strata. Following this, we will use two alternate approaches that use different statistical methods to answer the same question (i.e is the effect different between Chinese and Japanese individuals?)

The previous analyses showed that the SNP rs11204005 was the most highly associated when using the stratified CMH test before adding the additional genotype data. With the addition of the extra data we saw that rs7835221 was now the most significantly associated SNP. We are going to extra both of these SNPs and further investigate the effects of population stratification on their association.

```
# extract SNPs
system('bin/plink --bfile example/followup --recode
                  --snps rs11204005,rs7835221 --out example/tophit')
```

First we will examine genotyping rate and Hardy-Weinberg equilibrium for rs11204005 and rs7835221

```
# plink assoc
system('bin/plink --file example/tophit --hardy
                  --all --missing --out example/results')

# load results
results.lmiss <- read.table('example/results.lmiss', head = T)
results.hwe <- read.table('example/results.hwe', head = T)
# explore
results.lmiss
```

```
##   CHR        SNP N_MISS N_GENO F_MISS
## 1   8  rs7835221      0     89      0
## 2   8 rs11204005      0     89      0
```

```
results.hwe
```

```
##   CHR        SNP  TEST A1 A2     GENO O.HET. E.HET.        P
## 1   8  rs7835221   ALL  G  A 21/30/38 0.3371 0.4818 0.004593
## 2   8  rs7835221   AFF  G  A  1/11/36 0.2292 0.2342 1.000000
## 3   8  rs7835221 UNAFF  G  A  20/19/2 0.4634 0.4036 0.462300
## 4   8 rs11204005   ALL  A  G 21/43/25 0.4831 0.4990 0.831900
## 5   8 rs11204005   AFF  A  G  4/23/21 0.4792 0.4373 0.741600
## 6   8 rs11204005 UNAFF  A  G  17/20/4 0.4878 0.4497 0.736000
```

Repeat stratified CMH test for association with disease for rs11204005 and rs7835221 with Breslow-Day test for heterogeneity

```
##   CHR        SNP       BP A1    MAF A2 CHISQ         P      OR     SE
## 1   8  rs7835221 12878098  G 0.4045  A 57.54 3.305e-14 0.01914 0.7713
## 2   8 rs11204005 12895576  A 0.4775  G 25.99 3.432e-07 0.09950 0.5164
##        L95    U95 CHISQ_BD   P_BD
## 1 0.004222 0.0868  0.08109 0.7758
## 2 0.036160 0.2738  1.01900 0.3128
```

```
# plink assoc
system('bin/plink --file example/tophit --assoc
                  --within example/pop.cov --bd --out example/results')
# load results
results <- read.table('example/results.cmh', head = T)
# explore
results
```

Repeat stratified test for association with disease for rs11204005 and rs7835221 using a different approach (partitioning effects into total, between and within strata)

```
##    CHR        SNP A1 A2     F_A    F_U N_A N_U  TEST   CHISQ DF         P
## 1    8  rs7835221  G  A      NA     NA  NA  NA TOTAL 32.0600  2 1.095e-07
## 2    8  rs7835221  G  A      NA     NA  NA  NA ASSOC 31.9500  1 1.584e-08
## 3    8  rs7835221  G  A      NA     NA  NA  NA HOMOG  0.1082  1 7.422e-01
## 4    8  rs7835221  G  A 0.06522 0.6739  22  68     1 14.7400  1 1.236e-04
## 5    8  rs7835221  G  A 0.16670 0.9000  74  14     2 17.3200  1 3.162e-05
## 6    8 rs11204005  A  G      NA     NA  NA  NA TOTAL 19.3300  2 6.362e-05
## 7    8 rs11204005  A  G      NA     NA  NA  NA ASSOC 18.5800  1 1.628e-05
## 8    8 rs11204005  A  G      NA     NA  NA  NA HOMOG  0.7435  1 3.885e-01
## 9    8 rs11204005  A  G 0.19570 0.6014  22  68     1  9.8930  1 1.659e-03
## 10   8 rs11204005  A  G 0.36670 0.9000  74  14     2  9.4320  1 2.132e-03
##         OR
## 1       NA
## 2       NA
## 3       NA
## 4  0.03376
## 5  0.02222
## 6       NA
## 7       NA
## 8       NA
## 9  0.16120
## 10 0.06433
```

```
# plink assoc
system('bin/plink --file example/tophit --assoc
                  --within example/pop.cov --homog --out example/results')
# load results
results <- read.table('example/results.homog', head = T)
# explore
results
```

Repeat test for association with disease for rs11204005 and rs7835221 using a different approach (logistic regression, including population as a covariate)

```
##   CHR        SNP       BP A1 TEST NMISS        OR   STAT         P
## 1   8  rs7835221 12878098  G  ADD    89   0.01237 -4.633 3.603e-06
## 2   8  rs7835221 12878098  G COV1    89 198.50000  3.680 2.331e-04
## 3   8 rs11204005 12895576  A  ADD    89   0.06667 -4.330 1.489e-05
## 4   8 rs11204005 12895576  A COV1    89  79.15000  4.680 2.871e-06
```

```
# plink assoc
system('bin/plink --file example/tophit --assoc --logistic
                  --covar example/pop.cov --out example/results')
# load results
results <- read.table('example/results.assoc.logistic', head = T, as.is = T)
# explore
results
```

Explicitly test for between-population heterogeneity using logistic regression allowing for an interaction effect

```
##   CHR        SNP       BP A1     TEST NMISS       OR    STAT        P
## 1   8  rs7835221 12878098  G      ADD    89   0.0080 -1.6590 0.097030
## 2   8  rs7835221 12878098  G     COV1    89 146.2000  2.1290 0.033290
## 3   8  rs7835221 12878098  G ADDxCOV1    89   1.3580  0.1607 0.872300
## 4   8 rs11204005 12895576  A      ADD    89   0.2918 -0.6450 0.518900
## 5   8 rs11204005 12895576  A     COV1    89 319.1000  2.6550 0.007936
## 6   8 rs11204005 12895576  A ADDxCOV1    89   0.3366 -0.7811 0.434800
```

```
# plink assoc
system('bin/plink --file example/tophit --assoc --logistic
                  --covar example/pop.cov --interaction
                  --out example/results')
# load results
results <- read.table('example/results.assoc.logistic', head = T)
# explore
results
```

The above analyses suggest that the association is equally present in both populations (make a note of what the precise results are that suggest this). Next, we can ask the more basic question of whether allele frequency (not the odds ratio for association) differs between the two groups. This involves using the population label as the phenotype of an association test rather than as a covariate.

Explicitly test whether allele frequency for rs11204005 and rs7835221 differs between populations

```
##   CHR        SNP       BP A1    F_A    F_U A2   CHISQ       P     OR
## 1   8  rs7835221 12878098  G 0.2841 0.5222  A 10.4700 0.00121 0.3631
## 2   8 rs11204005 12895576  A 0.4545 0.5000  G  0.3685 0.54380 0.8333
```

```
# plink assoc
system('bin/plink --file example/tophit --assoc
```

```
                --pheno example/pop.cov --out example/results')
# load results
results <- read.table('example/results.assoc', head = T)
# explore
results
```

Explicitly test whether allele frequency for rs11204005 and rs7835221 differs between populations, allowing for association with disease

```
# plink assoc
system('bin/plink --file example/tophit --logistic --pheno example/pop.cov
                --covar example/followup.fam --covar-number 4 --out example/results')

# load results
results <- read.table('example/results.assoc.logistic', head = T)
# explore
results
```

```
##   CHR        SNP       BP A1 TEST NMISS      OR  STAT         P
## 1   8  rs7835221 12878098  G  ADD    89   5.849 2.245 2.476e-02
## 2   8  rs7835221 12878098  G COV4    89 212.000 3.715 2.033e-04
## 3   8 rs11204005 12895576  A  ADD    89   4.466 2.727 6.400e-03
## 4   8 rs11204005 12895576  A COV4    89  73.770 4.708 2.498e-06
```

These results would suggest that the frequency does indeed differ (again, make a note of exactly why this is).

Genotype models

For simplicity in this Practical, we will ignore the effect of population for subsequent exercises. This would not be advised with real data, as in this case, we in fact know that both allele frequency and disease rate differ between populations. It would therefore normally be important to perform analysis within-population or to include population as a covariate.

The previous association statistics were all based on allelic models (that each extra copy of the risk allele increases risk equally). We can also ask whether specific genotype configurations (heterozygotes versus homozygotes) have specific risk profiles.

Test genotypic models for rs11204005 and rs7835221

```
system('bin/plink --file example/tophit --model --cell 1 --out example/results')
# load results
results.mod <- read.table('example/results.model', head = T)
# explore
results.mod
```

```
##    CHR        SNP A1 A2    TEST     AFF   UNAFF CHISQ DF         P
## 1    8  rs7835221  G  A    GENO 1/11/36 20/19/2 49.50  2 1.783e-11
## 2    8  rs7835221  G  A   TREND   13/83   59/23 48.17  1 3.906e-12
## 3    8  rs7835221  G  A ALLELIC   13/83   59/23 62.64  1 2.485e-15
## 4    8  rs7835221  G  A     DOM   12/36    39/2 44.44  1 2.623e-11
## 5    8  rs7835221  G  A     REC    1/47   20/21 26.75  1 2.320e-07
## 6    8 rs11204005  A  G    GENO 4/23/21 17/20/4 19.39  2 6.171e-05
## 7    8 rs11204005  A  G   TREND   31/65   54/28 19.35  1 1.087e-05
## 8    8 rs11204005  A  G ALLELIC   31/65   54/28 19.97  1 7.882e-06
## 9    8 rs11204005  A  G     DOM   27/21    37/4 12.65  1 3.755e-04
## 10   8 rs11204005  A  G     REC    4/44   17/24 13.46  1 2.434e-04
```

Test genotypic models for rs11204005 and rs7835221 using logistic regression

```
system('bin/plink --file example/tophit --logistic
                  --genotype --out example/results')

# load results
results.log <- read.table('example/results.assoc.logistic', head = T)
# explore
results.log
```

```
##   CHR        SNP       BP A1 TEST NMISS      OR  STAT         P
## 1   8  rs7835221 12878098  G  ADD    89   5.849 2.245 2.476e-02
## 2   8  rs7835221 12878098  G COV4    89 212.000 3.715 2.033e-04
## 3   8 rs11204005 12895576  A  ADD    89   4.466 2.727 6.400e-03
## 4   8 rs11204005 12895576  A COV4    89  73.770 4.708 2.498e-06
```

Test genotypic models for rs11204005 and rs7835221 using logistic regression with an alternate genotypic coding

```
system('bin/plink --file example/tophit --logistic --genotypic
                  --hethom --out example/results')
```

```
# load results
results.hethom <- read.table('example/results.assoc.logistic', head = T)
# explore
results.hethom
```

```
##   CHR        SNP       BP A1     TEST NMISS       OR   STAT         P
## 1   8  rs7835221 12878098  G      HOM    89 0.002778 -4.686 2.786e-06
## 2   8  rs7835221 12878098  G      HET    89 0.032160 -4.195 2.732e-05
## 3   8  rs7835221 12878098  G GENO_2DF    89       NA 26.310 1.932e-06
## 4   8 rs11204005 12895576  A      HOM    89 0.044820 -3.987 6.681e-05
## 5   8 rs11204005 12895576  A      HET    89 0.219000 -2.428 1.518e-02
## 6   8 rs11204005 12895576  A GENO_2DF    89       NA 15.900 3.528e-04
```

These analyses suggest that the effect is an allele-dosage one, rather than showing dominant or recessive non-additivity.

Sex-specific effects

Next, in the same manner as we tested for between-population heterogeneity, we can ask whether the effect varies between males and females. We do this first by performing sex-specific analyses; second, by including sex as a covariate in a logistic regression model.

Test for association specific in males

```
# explore association in males
system('bin/plink --file example/tophit --filter-males --logistic --out example/plink')
# read in results
male.assoc <- read.table('example/plink.assoc.logistic', head = T)
```

Have a look at the association results, what do you see?

CHR	SNP	BP	A1	TEST	NMISS	OR	STAT	P
8	rs7835221	12878098	G	ADD	44	0.07673	-3.390	0.0006996
8	rs11204005	12895576	A	ADD	44	0.21060	-2.823	0.0047620

Note: Can also include other covariates here (population, etc).

Test for association specific in females

```
# explore association in females
system('bin/plink --file example/tophit --filter-females --logistic --out example/plink')
# read in results
female.assoc <- read.table('example/plink.assoc.logistic', head = T)
```

CHR	SNP	BP	A1	TEST	NMISS	OR	STAT	P
8	rs7835221	12878098	G	ADD	45	0.01977	-3.505	0.0004563
8	rs11204005	12895576	A	ADD	45	0.20860	-2.829	0.0046660

Test for different effects in males versus females

```
system('bin/plink --file example/tophit --sex --logistic --out example/plink')
# read in results
sex.assoc <- read.table('example/plink.assoc.logistic', head = T)
```

CHR	SNP	BP	A1	TEST	NMISS	OR	STAT	P
8	rs7835221	12878098	G	ADD	89	0.04261	-4.9390	7.841e-07
8	rs7835221	12878098	G	SEX	89	1.30300	0.4091	6.824e-01
8	rs11204005	12895576	A	ADD	89	0.20960	-3.9970	6.428e-05
8	rs11204005	12895576	A	SEX	89	0.78210	-0.5088	6.109e-01

Note: The –sex command adds sex as a covariate (0 = males, 1 = females).

These results suggest no sex differences in the nature of the association. Again, make a note of the exact supporting statistical evidence for this. What are the odds ratios in males and females?

LD analysis and plotting

```
# load required packages
require("genetics")
require("LDheatmap")
require("RColorBrewer")
```

First we will need to isolate the region of interest from the genotype data. We can do this several ways, for now we will concentrate of using `plink` for this purpose.

There are multiple ways to extract just specific SNPs for analysis; this section describes options that use the command-line directly; the next section describes other methods that read a file containing the information.

Based on a single chromosome (–chr)

To analyse only a specific chromosome use:

```
plink --file data --chr 8
```

```
head(followup.results[order(followup.results$P),])
```

```
##        CHR        SNP       BP A1    MAF A2 CHISQ         P      OR     SE
## 85075    8  rs7835221 12878098  G 0.4045  A 57.54 3.305e-14 0.01914 0.7713
## 85077    8 rs11204005 12895576  A 0.4775  G 25.99 3.432e-07 0.09950 0.5164
## 85079    8  rs2460338 12914531  G 0.4775  C 22.35 2.277e-06 0.09967 0.5605
## 133850  13  rs4941815 35207635  C 0.4663  T 19.04 1.280e-05 6.53600 0.4643
## 133852  13  rs4943327 35209062  T 0.4663  C 19.04 1.280e-05 6.53600 0.4643
## 136481  13  rs9531117 80316559  T 0.2159  C 18.89 1.386e-05 7.13300 0.5345
##             L95     U95
## 85075  0.004222  0.0868
## 85077  0.036160  0.2738
## 85079  0.033220  0.2990
## 133850 2.631000 16.2400
## 133852 2.631000 16.2400
## 136481 2.502000 20.3300
```

Based on a range of SNPs (–from and –to)

To select a specific range of markers (that must all fall on the same chromosome) use, for example:

```
plink --bfile mydata --from rs7835221 --to rs2460338
```

Based on single SNP (and window) (–snp and –window)

Alternatively, you can specify a single SNP and, optionally, also ask for all SNPs in the surrounding region, with the –window option:

```
plink --bfile mydata --snp rs7835221 --window 100
```

which extracts only SNPs within +/- 100kb of rs7835221, our most significantly associated SNP on chr 8.

We are going to use the last option to extract a ~100kb region around our most significantly associated hit on chr 8:

```
# extract 200kb window around peak on chr 8
system('bin/plink --bfile example/followup --snp rs7835221 --window 150
                  --recode --transpose --out example/chr8_ldregion')
```

Load the extracted SNP data into R:

```
snp.data <- read.table('example/chr8_ldregion.tped', head = F, as.is = T)
sample.data <- read.table('example/chr8_ldregion.tfam', head = F, as.is = T)
```

Create genotype files from this data for LD mapping:

```
# generate distance
snpDist <- snp.data$V4
#
snp.data <- snp.data[c(2, 5:ncol(snp.data))]
rownames(snp.data) <- snp.data$V2
snp.data <- snp.data[-c(1)]
#
snp.data[snp.data == "0"] <- NA
# snp.clean <- snp.data[ , colSums(is.na(snp.data)) == 0] snp.data <-
# snp.clean
snp.collapse <- NULL
#
for (i in seq(1, ncol(snp.data), 2)) {

    n <- i + 1
    snp.collapse <- rbind(snp.collapse, paste(as.character(snp.data[[i]]), as.character(snp.data[[n]]),
        sep = "/"))

}
#
colnames(snp.collapse) <- rownames(snp.data)
snp.collapse <- as.data.frame(snp.collapse)
rownames(snp.collapse) <- sample.data$V1
snp.collapse[snp.collapse == "NA/NA"] = NA
snp.collapse <- na.omit(snp.collapse)
#
example.geno <- snp.collapse
#
for (i in colnames(example.geno)) {

    geno <- as.factor(example.geno[[i]])
    geno <- genotype(geno)

    example.geno[[i]] <- geno

}
# the first 45 rows are Chinese
CH.snps <- snp.collapse[grep("CH", rownames(snp.collapse)), ]
#
CH.geno <- CH.snps
#
for (i in colnames(CH.geno)) {
```

```
    geno <- as.factor(CH.geno[[i]])
    geno <- genotype(geno)

    CH.geno[[i]] <- geno

}
# the rest are Japanese
JP.snps <- snp.collapse[grep("J", rownames(snp.collapse)), ]
#
JP.geno <- JP.snps
#
for (i in colnames(JP.geno)) {

    geno <- as.factor(JP.geno[[i]])
    geno <- genotype(geno)

    JP.geno[[i]] <- geno

}
#

# genetate the LDheatmap
rgb.palette <- colorRampPalette(rev(c("white", "cadetblue")), space = "rgb")
snp.list <- c("rs7835221", "rs11204005", "rs2460338")

# ldhm <- LDheatmap(example.geno, genetic.distances = snpDist, distances =
# 'physical', LDmeasure = 'r', SNP.name = colnames(example.geno), title =
# 'Pairwise LD', flip = F, text = T, color = rgb.palette(20))

ldhm.all <- LDheatmap(example.geno, genetic.distances = snpDist, distances = "physical",
    LDmeasure = "r", SNP.name = snp.list, title = "Pairwise LD (both populations)",
    flip = F, text = T, color = rgb.palette(20), name = "ldheatmap")
LDheatmap.marks(ldhm.all, 8, 10, gp = gpar(cex = 2), pch = "*")
# highlight our cluster of interesting SNPs
LDheatmap.highlight(ldhm.all, 7, 12, fill = "NA", col = "grey25", lwd = 1, lty = 2)


# LD plot - Chinese
ldhm.CH <- LDheatmap(CH.geno, genetic.distances = snpDist, distances = "physical",
    LDmeasure = "r", SNP.name = snp.list, title = "Pairwise LD (CH)", flip = F,
    text = T, color = rgb.palette(20))
LDheatmap.marks(ldhm.CH, 8, 10, gp = gpar(cex = 2), pch = "*")
# highlight our cluster of interesting SNPs
LDheatmap.highlight(ldhm.all, 7, 12, fill = "NA", col = "grey25", lwd = 1, lty = 2)


# LD plot - Japanese
ldhm.JP <- LDheatmap(JP.geno, genetic.distances = snpDist, distances = "physical",
    LDmeasure = "r", SNP.name = snp.list, title = "Pairwise LD (JP)", flip = F,
    text = T, color = rgb.palette(20))
LDheatmap.marks(ldhm.JP, 8, 10, gp = gpar(cex = 2), pch = "*")
# highlight our cluster of interesting SNPs
LDheatmap.highlight(ldhm.all, 7, 12, fill = "NA", col = "grey25", lwd = 1, lty = 2)
```

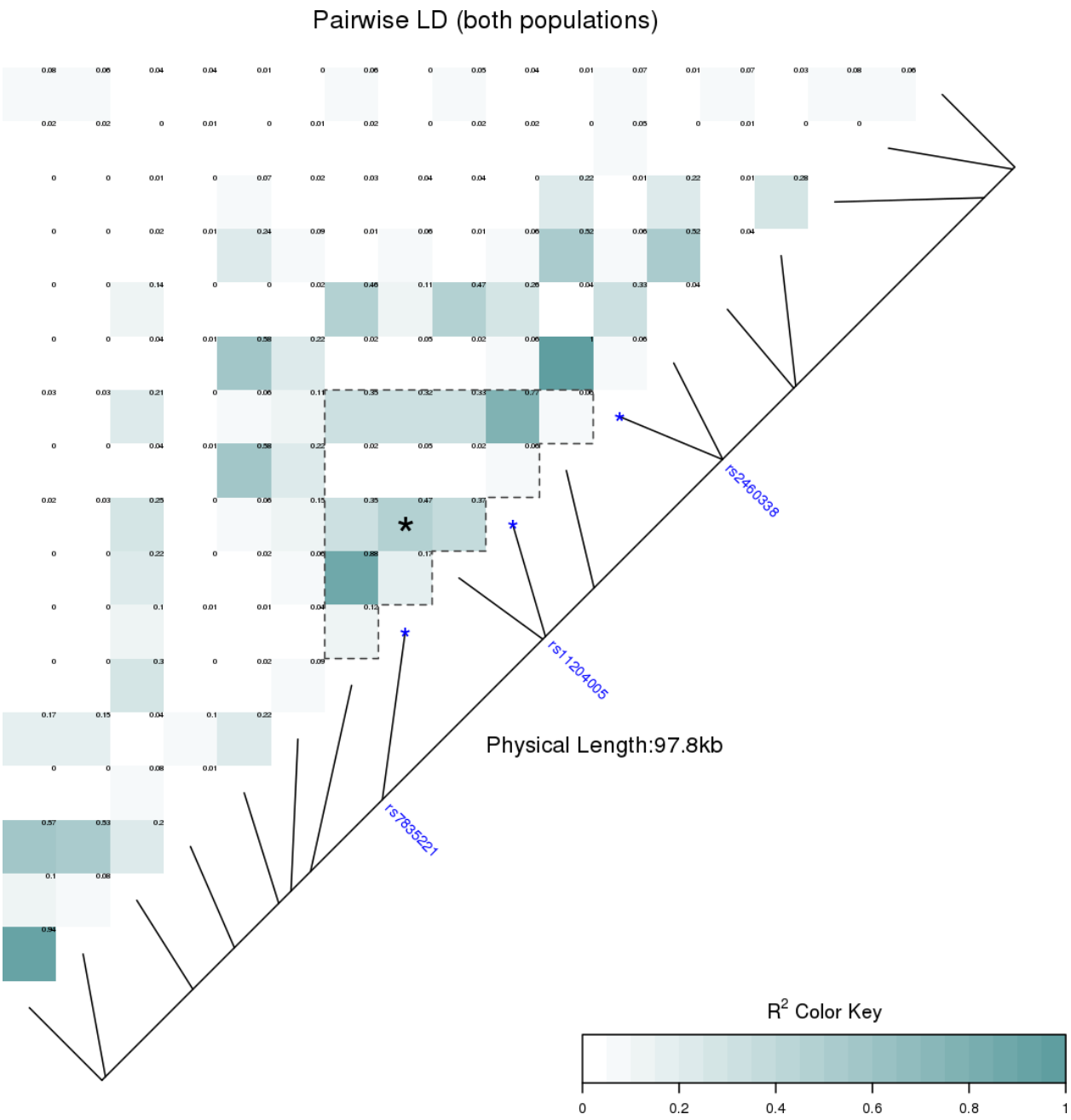

Figure 11: LD plot of chr 8 region for both populations combined.

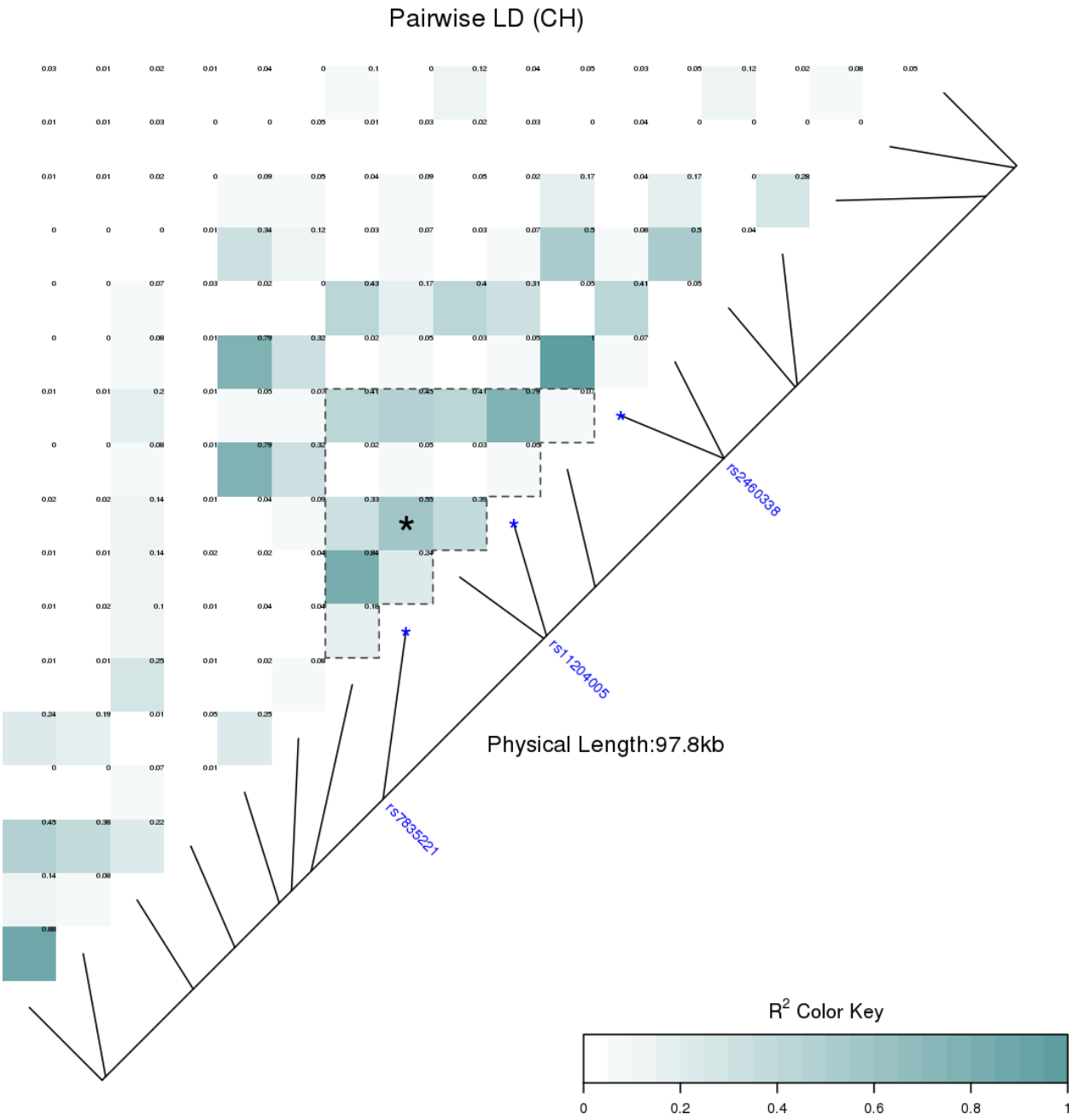

Figure 12: LD plot of chr 8 region for the Chinese population.

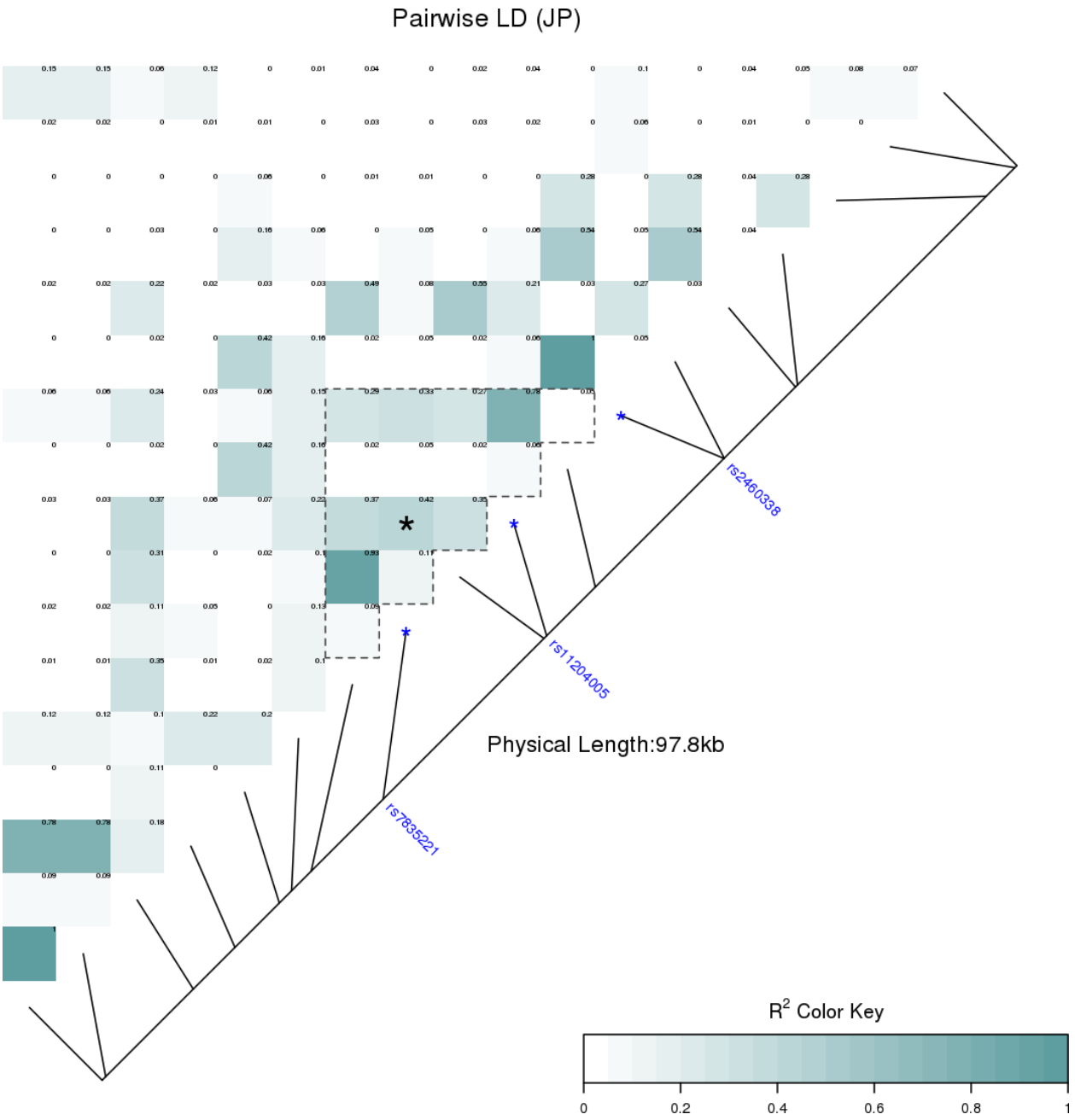

Figure 13: LD plot of chr 8 region for the Japanese population.

Summary

Thanks for attending.

Well, we hope you enjoyed the workshop, learnt a little something and don't have too much of a sore head. All going well we hope to run more workshops in the near future.

- Miles and Rod

Appendix

The following code is just to detail the steps that were automated for your interest.

```
#!/usr/bin/Rscript

# unzip example data into example/
unzip('example/example_data.zip', exdir = 'example')
cat('\n', 'Data extracted', '\n')

# detect OS
os.type <- Sys.info()['sysname']
# download the latest OS specific version of plink 1.9
ifelse(os.type == 'Linux',
       download.file('http://www.cog-genomics.org/static/bin/plink150805/plink_linux_x86_64.zip',
                     'bin/plink_linux_x86_64.zip'),
       ifelse(os.type == 'Windows',
              download.file('http://www.cog-genomics.org/static/bin/plink150805/plink_win64.zip',
                            'bin/plink_win64.zip'),
              download.file('http://www.cog-genomics.org/static/bin/plink150805/plink_mac.zip',
                            'bin/plink_mac.zip')))
cat('\n', 'PLINK downloaded', '\n')

# identify the zip file
plink.zip <- list.files('bin/', pattern = '.zip', full.names = T)
# unzip this file to the bin/ directory within the current working directory
unzip(plink.zip, exdir = 'bin')
# some mac systems seem to require the plink binary to be flagged as executable
if(os.type == 'Darwin') system('chmod +x bin/./plink')

cat('\n', 'PLINK extracted and installed in current working directory bin/', '\n')

# set up R environment (packages) for workshop
cat('\n', 'Checking packages and installing those required for this workshop...', '\n')

# download, install and load required bioconductor packages
# run this before base CRAN packages as Haplin depends on these
cat("\n", "Installing required bioconductor packages...", "\n")
#
# Install function for bioconductor packages
# should allow detection of already installed packages
bioc.packages <- function(x){
  x <- as.character(match.call()[[2]])
  if (!require(x, character.only = TRUE)){
    biocLite(pkgs = x)
    require(x, character.only = TRUE)
  }
}
#
source("http://bioconductor.org/biocLite.R")
bioc.packages("GenABEL")
#
cat("\n", "...Done...", "\n")
```

```
# Install function for CRAN packages
# should allow detection of already installed packages
packages <- function(x){
  x <- as.character(match.call()[[2]])
  if (!require(x, character.only = TRUE)){
    install.packages(pkgs = x, repos = "http://cran.r-project.org")
    require(x, character.only = TRUE)
  }
}

# use the above to download, install and load CRAN packages
cat("\n", "Installing required CRAN packages...", "\n")
#
packages(devtools) # used to install packages from source and GitHub
packages(MASS)
packages(Haplin)
packages(genetics)
packages(LDheatmap)
packages(RColorBrewer)
#
cat("\n", "...Done...", "\n")

# required packages from GitHub repositories
# NOTE: certain software will need to be installed to build the following
# further instructions will be available in the GitHub README
cat("\n", "Retrieving and installing Github pacakges...", "\n")
# download, install and load qqman
install_github("stephenturner/qqman", ref="dev")
require("qqman")
# download, install and load gdsfmt
install_github("zhengxwen/gdsfmt")
require("gdsfmt")
# download, install and load SNPRelate
install_github("zhengxwen/SNPRelate")
require("SNPRelate")
#
cat("\n", "...Done. You are ready to start the workshop...", "\n")
# END
```