Genome-wide Association Study using GenABEL

The Rockefeller University, New York

Copyrighted (c) 2017 Gao Wang and Suzanne Leal

Purpose

In this exercise, you will learn commands to carry out association tests using the R\(^1\) and the R library \texttt{GenABEL}\(^2\) in the presence of population stratification. You will be working with a pre-cleaned dataset from your PLINK exercise containing \(\sim250\) individuals each genotyped for \(\sim6,500\) SNPs and also phenotyped for both a binary and quantitative phenotype.

Methodology

You will use multidimensional scaling (MDS), principal component analysis (PCA) and genomic admixture control for detecting and adjusting for population stratification in your analysis.

Resources

For additional information on this exercise please refer to 3.3, 3.4, 5.1, 5.3, 5.4, 5.5 and 7.1 of the June 19, 2013 version GenABEL tutorial. The analysis on the binary trait partially overlaps with the PLINK exercise, with which you may compare your results and output.

This exercise is based on GenABEL v. 1.8-0. Complete result output could be found at the end of this exercise.

GenABEL tutorial: \url{http://www.genabel.org/sites/default/files/pdfs/GenABEL-tutorial.pdf}

Data files

You are provided with a dataset consisting of 5 files. The \texttt{*.ped}, \texttt{*.map} and \texttt{*.phen} files are slightly different files from the ones you have worked with in your PLINK GWAS exercise. Please keep in mind that the data has been 'cleaned', which is a necessary requirement for all analysis below. There are two additional phenotype files (\texttt{*.praw}) formatted for GenABEL input.

We will be working with GenABEL, which use the \texttt{*.tped} and \texttt{*.tfam} format files rather than the \texttt{*.ped} and the \texttt{*.map} files you used for the PLINK exercise. The following PLINK commands will allow you to re-format your input; you can copy these commands from the code files provided.

```
plink --file GWAS_clean4 --pheno pheno.phen --pheno-name Aff --transpose --recode --out gwa_gabel
```

```
plink --file GWAS_clean4 --pheno pheno.phen --pheno-name systolic --transpose --recode --out gwa_gabel_qtl
```

The two \texttt{*.praw} files have been formatted from the \texttt{*.tfam} files you just generated. Such re-formatting could be done with simple shell scripts. Please take a look at the files to check that you understand how the information is coded. Please read the PLINK documentation for \texttt{*.tfam} file specification if you are not familiar with it.
**Q1: Compare gwa_gabel.praw with gwa_gabel.tfam. How is information sex and disease coded in both files?**

**Answer:**

Now we are ready to use GenABEL for GWAS. The individual tests are generally the same as were previously introduced in the R exercise, but with GenABEL the whole genome can be analyzed quickly while easily correcting for population structure and other confounders.

1. **A Preliminary GWAS Analysis**

For this preliminary study, we ignore the presence of population structure. Please carry-out these commands in R, and copy-paste the following to load GenABEL and our data files:

```r
library(GenABEL)# load files to GenABEL
convert.snp.tped(tped = "gwa_gabel_qtl.tped", tfam = "gwa_gabel_qtl.tfam", out = "gwa_gabel_qtl.raw", strand = "u")
g.dat <- load.gwaa.data(phen = "gwa_gabel_qtl.praw", gen = "gwa_gabel_qtl.raw", force = T)
```

Now you have created a GenABEL object ‘g.dat’ that contains all information in your original data set. To view members in that data object:

```r
slotNames(g.dat)
slotNames(g.dat@gtdata)
colnames(g.dat@phdata)
```

It is always necessary to become familiar with your data before engaging into analysis. The following code provides you a starting point. The histogram generated provides visualization of the distribution and relative frequency of the quantitative values, accompanied by a rug plot showing distinct values on the X-axis.

```r
# sample size
sample.size <- g.dat@gtdata@nids

# number of SNPs
snps.total <- g.dat@gtdata@nsnps

print(c(sample.size, snps.total))

# Trait
summary(g.dat@phdata$disease)

hist(g.dat@phdata$disease, main="Quantitative Phenotype data summary", xlab = "Systolic pressure", freq = F, breaks=20, col="gray")
rug(g.dat@phdata$disease)
```
Tip

You may also want to look at other information such as mean and variance of trait, percentage of males/females, missing data, etc. For more detailed data information summary, please read documentation for descriptives.trait() and descriptives.marker() functions.

Q2: What is the sample size, and how many SNPs have been genotyped in this study? What type of trait will we analyze (binary or continuous)?

Answer:

Let us begin our analysis to try to detect an association with the quantitative trait using the scan.glm() function, which scans the whole genome using the glm() function. Note that this test will be very slow because in GLMs parameters are estimated numerically via MLE method. However, we introduce this function because it allows you to add interaction terms (gene-gene, gene-environment or any interactions) the same manner as you used the glm() function in R.

test.snp <- scan.glm('disease ~ CRSNP', family = gaussian(), data = g.dat)
names(test.snp)
alpha <- 5e-8
test.snp$snpnames[test.snp$pldf < alpha]
test.snp$pldf[test.snp$pldf < alpha]
From the test, you find that for the entire “genome”, only `rs4571722` is associated with the quantitative trait with a p-value of 2E-17.

We can also use a score test, the `qtscore()` function, which is efficient. For quantitative traits, it is a score test for the linear regression model.

---

**Note**

A “warning message” pops up in R, when the estimated genomic inflation factor (GIF) is lower than 1. You can use `quiet = T` argument in `qscore()` to turn it off.

```r

```test.qt <- qtscore(disease, data = g.dat, trait = "gaussian")

```

slotNames(test.qt)
names(test.qt$results)
head(results(test.qt))
test.qt@lambda
descriptives.scan(test.qt)
row.names(results(test.qt))[results(test.qt)$P1df < alpha]
results(test.qt)$P1df[results(test.qt)$P1df < alpha]
results(test.qt)$Pc1df[results(test.qt)$Pc1df < alpha]
```

For this test SNP `rs4571722` has p-value 9E-14.

**Task 1:** Please find P1df (p-value for the 1 df test), chi2.1df (1 df test), P1df (p-value for the 1 df test corrected for inflation factor) and lambda (genomic inflation factor) on your screen. These R objects contain information for the 1df test (additive model). Information for the 2df score test (genotypic model, i.e. effects of AB and BB estimated against reference, AA) are also provided but for the text below we will only be discussing the 1df test. You could extract information from an R object with the ‘$’ symbol, as demonstrated above.

In publications we often display our result (the 1 df p-value in our case) with a QQ plot and a Manhattan plot. The QQ plot is very useful for diagnostic purposes: it plots the $-\log_{10}$ of the observed p-values vs. the $-\log_{10}$ of the expected p-values (under null). If all points fall on the diagonal line, then none of the p-values are smaller than would be expected when there is no association. If we have many p-values which are smaller than expected ($-\log_{10}$ values on top the line) this is probably not because we have many true associations, but because there is a problem with the data, for example population substructure/admixture. The Manhattan plot is very nice to visualize the findings in chromosomal order, it also helps to see if there are multiple associations within a small genomic region.

We will now make a QQ and Manhattan plot of the p-values which have not been adjusted (P1df).

```r

```

# QQ plot
obs <- sort(results(test.qt)$P1df)
ept <- ppoints(obs)

plot(-log10(ept), -log10(obs), main = "GWAS QQ plot, qtl", xlab="Expected $-\log_{10}(p-value)$", ylab="Observed $-\log_{10}(p-value)$")

abline(0, 1, col = "red")
abline(h = 8, lty = 2)

# Manhattan plot
plot(test.qt, col = "black")

```
Figure 2: The QQ and Manhattan plots

Notice that the expected p-values under the null follow a uniform distribution and they are plotted vs. the observed p-values, and both are converted into $-\log_{10} P$ in the QQ plot. Please examine the plots to make sure you understand them.

Adding potential confounders to the analysis

You can add predictors to \texttt{scan.glm()} or \texttt{qtscore()} functions. Covariates could be sex, age, and importantly, principal components that corrects for population structure. For example:

\begin{verbatim}
# example
# test qt-score for sex

qtscore <- qtscore(disease ~ sex, data = g.dat, trait = "gaussian")
row.names(results(qtscore))[results(qtscore)[p1df < alpha]

# evaluate the significance of sex

summary(lm(disease ~ sex, data = g.dat))
\end{verbatim}

Working with binary traits

Change the argument \texttt{gaussian} into \texttt{binomial} in \texttt{scan.glm()} and \texttt{qtscore()} functions when you analyze case-control data. For binary trait, the 1 df test is an additive test and is equivalent to the Armitage trend test when no covariates are included in the analysis.
Q3: Please use the R codes below (or at the end of Code.R.txt document) to work with binary trait. For this part you could check the result with your PLINK exercise. Which SNPs are significantly associated with the trait?

```r
# load files to GenABEL
cvtColorped(tped = "gwa_gabel.tped", tfam = "gwa_gabel.tfam", out = "gwa_gabel.raw", strand = "u")
b.dat <- load.gwaa.data(phen = "gwa_gabel.praw", gen = "gwa_gabel.raw", force = T)

slotNames(b.dat)
slotNames(b.dat@gtdata)
columnNames(b.dat@phdata)

# sample size
b.dat@gtdata@nids

# number of cases and controls
case.size <- length(which(b.dat@phdata$disease == 1))
control.size <- length(which(b.dat@phdata$disease == 0))

case.size
control.size

# number of SNPs
snpsb.total <- b.dat@gtdata@nsnps

# GLM test
testb.snp <- scan.glm('disease ~ CRSNP', family = binomial(), data = b.dat)
names(testb.snp)
alpha <- 5e-8

testb.snp$snpnames[testb.snp$P1df < alpha]
testb.snp$P1df[testb.snp$P1df < alpha]

# Score test
testb.qt <- qtscore(disease, data = b.dat, trait = "binomial")
slotNames(testb.qt)
descriptives.scan(testb.qt)

row.names(results(testb.qt))[results(testb.qt)$P1df < alpha]
results(testb.qt)$P1df[results(testb.qt)$P1df < alpha]
results(testb.qt)$Pc1df[results(testb.qt)$Pc1df < alpha]
```

2. MDS Analysis for Global Ancestry Inference

Now let us perform Multidimensional Scaling analysis with GenABEL to detect potential population structure. Note that MDS is closely related to PC analysis, and that for PC analysis GenABEL uses the EIGENSTRAT algorithm. Firstly we compute a genome-wide IBS matrix:

```r
gkin <- ibs(g.dat, weight = "freq")
gkin[1:10,1:10]
```
Look at the Identity-by-state (IBS) matrix for the first 10 people. This matrix gives average IBS (kinship) values between a pair below the diagonal, and number of genotype successfully typed for both members of the pair above the diagonal. The diagonal is $0.5 + \text{homozygosity}$.

Now we convert it into a distance matrix and pass it into our MDS analysis function, the `cmdscale()`:

```r
cps.full <- cmdscale(as.dist(.5 - gkin), eig = T, k = 10)
names(cps.full)
cps <- cps.full$points
```

See `?as.dist` if you are interested in the distance.

The distance matrix is a measure of dissimilarities between individuals; $k = 10$ means that the output will contain the first 10 components. Let us plot out the first and the second components:

```r
plot(cps[,1], cps[,2], pch = g.dat@phdata$popn)
legend("topright", c("TSI", "MEX", "CEU"), pch = c(1,2,3))
```

**Figure 3: The first two components plot**

Please read the legend on your plot. If you recall from your PLINK exercise, the dataset is comprised of individuals from three groups from Hapmap CEU (Mormons from Utah, originally from Northern Europe, TSI (Tuscans - Northern Italy) and MEX (Mexican Americans). In practice you should never knowingly analyze data from different population ethnic groups together, even if you adjust for population structure.
Q4: Do you see population stratification in our data set? Which component allows you to distinguish between CEU and MEX? What about TSI and CEU?

Answer:

3. Association Tests with Correction for Population Structure

We will introduce three methods below, for illustration purpose. They do not uniformly work very well with our dataset but we want you to learn the commands and concepts. You may look at the GenABEL tutorial Chapter 3 where there is a discussion on which method to use in your analysis.

Incorporating PCs as predictors

As we have computed components scores with MDS analysis, we could correct for population structure by adding the components as predictors. For example:

| colnames(cps)<-c('C1','C2','C3','C4','C5','C6','C7','C8','C9','C10') colnames(cps)<-c('C1','C2','C3','C4','C5','C6','C7','C8','C9','C10') |
| gpc.dat <- g.dat gpc.dat<-cbind(g.dat@phdata, cps) |
| test.pc.a <- scan.glm('disease ~ CRSNP + C1 + C2 + C3 + C4 + C5', family=gaussian(), data = gpc.dat) test.pc.a$snpnames[test.pc.a$P1df < alpha] test.pc.a$P1df[test.pc.a$P1df < alpha] |
| test.pc.b <- qt(score(disease ~ C1 + C2 + C3 + C4 + C5, data = gpc.dat, trait = "gaussian") test.pc.b@lambda) |

Please check the result on your own. You can produce new QQ and Manhattan plots to compare p-value with previous analysis, and compare test.qt@lambda and test.pc.b@lambda to see if genomic inflation factor λ has decreased.

Note

For PCA and MDS analysis, markers should be independent, which is the case for the data for this tutorial. For analysis of real-world GWAS data we need to select a subset of markers that are not in LD with each other.

If you include more components than it is necessary you would introduce noise to your analysis. Therefore you should examine the eigenvalues. Knowing that the sum of the first $k$ eigenvalues divided by the sum of all the eigenvalues represents the proportion of total variation explained by the first $k$ components, we can visually evaluate the contribution of each component via the following plots:
# scree plot
plot(cps.full$eig[1:10]/sum(cps.full$eig), axes = F, type = "b", xlab = "Components", ylab = "Proportion of Variations", main = "MDS analysis scree plot")
axis(1, 1:10)
axis(2)

# cumulative plot
plot(cumsum(cps.full$eig[1:10])/sum(cps.full$eig), axes = F, type = "b", ylim = c(0,0.2), xlab = "Components", ylab = "Proportion of Variations", main = "MDS analysis cumulative plot")
axis(1, 1:10)
axis(2)

Figure 4: Scree plot and cumulative plot for the first 10 eigenvalues of dissimilarity matrix

From the plots we see that more than 10% of the information can be explained by the first 10 components. The first two components could explain more than 5% of the total variation. We could start building the association model projecting the total variation onto two dimensions by including the first two components, and evaluate the change of $\lambda$ as has been demonstrated in the PLINK exercise. Although the GenABEL tutorial states that $\lambda > 1.01$ is considered as small, $\lambda < 1.11$ is acceptable and $\lambda > 1.11$ is problematic, in many cases a $\lambda$ of 1.1 is considered too large particularly when you are trying to detect small effect sizes. There is, however, no general guideline on $\lambda$ since it describes how much a specific test is inflated – it depends on the test statistic.

**Tip**

Alternatively you may look into EIGENSTRAT package, which applies other methods for deciding how many components to include in analysis.

**Task 2:** You may change the number of components you include in your analysis and evaluate how $\lambda$ changes.

**Answer:** 

9
Genomic Control

An alternative method for correcting population structure is to use Genomic Control. If you make QQ plot of the chi2.1df (cf. Task 1) vs. \( \chi^2_{\text{PCH}} \) under the null you will see many points off the diagonal, indicating the test statistic is inflated, as shown in the plot below (slope of the dashed line is \( \lambda \)). For genomic control method, the observed test statistic will be divided by \( \lambda \) to correct for inflation of test statistic with corrected p-value \( P_{\text{c1df}} \). Note that although in most literature \( \lambda \) is defined as the median of observed \( \chi^2 \) test statistic divided by 0.456 (see Devlin and Roeder 1999 3; Devlin et al. 2001 4), GenABEL uses a different definition, the regression coefficient (slope) of observed \( \chi^2 \) w.r.t. expected \( \chi^2 \), which is usually a bit more conservative. GenABEL also uses only the lower 90% of the distribution to calculate \( \lambda \).

Genomic control method many not adequately correct for population substructure, and it is not recommended to use for admixed populations.

\[
\begin{align*}
\text{# corrected p-value} \\
\text{row.names(results(test.qt))}[\text{results(test.qt)}$P_{\text{c1df}} < \alpha] \\
\text{results(test.qt)}$P_{\text{c1df}}[\text{results(test.qt)}$P_{\text{c1df}} < \alpha]
\end{align*}
\]

# Uncorrected GIF calculated by GenABEL
test.qt@lambda

# Check for inflation of statistic
obs <- sort(results(test.qt)$chi2.1df)
ept <- sort(qchisq(ppoints(obs), df = 1))

plot(ept, obs, main = "Genomic control (lambda = slope of the dashed line)", xlab="Expected chisq, 1df", ylab="Observed chisq, 1df")
abline(0, 1, col = "red")
abline(0, test.qt@lambda[1], lty = 2)

# Definition of GIF
# Conventional definition
median(results(test.qt)$chi2.1df)/0.456

# QQ plot, corrected p-value
obs <- sort(results(test.qt)$P_{\text{c1df}})
ept <- ppoints(obs)
plot(-log10(ept), -log10(obs), main = "GWAS QQ plot adj. via Genomic Control", xlab="Expected - log10(pvalue)", ylab="Observed -log10(pvalue)")
abline(0, 1, col = "red")
abline(h = 8, lty = 2)

Figure 5: The QQ plots. Left: Genomic inflation Right: p-value adjusted via genomic control
Q5: Compare the corrected QQ plot for p-value with the uncorrected plot. What is the value for corrected lambda? Does it suggest that genomic control works better than PCA?

Answer: 

Correction with PCA

We recommend doing correction for population structure using PCA method, as described in Price et al. 2006. GenABEL incorporates EIGENSTRAT method to test for association while correcting for population structure. This is implemented in egscore() function. The following code prepares input for the function and then carries out association testing correcting for the first 2 components:

```r
adj.gkin = gkin
diag(adj.gkin) = hom(g.dat)$Var
test.eg <- egscore(disease, data = g.dat, kin = adj.gkin, naxes = 2)
descriptives.scan(test.eg)
snp.eg <- row.names(results(test.eg))[results(test.eg)$P1df < alpha]
pvalue.eg <- results(test.eg)$P1df[results(test.eg)$P1df < alpha]
lambda.eg <- test.eg@lambda
```

Q6: Is the test still significant? What is the p-value for ‘rs4571722’? What about GIF?

Answer: 

```r
```
Note
You may notice that the \texttt{egscore()} function gives a different $\lambda$ value than using \texttt{qtscore} adjusted for the same number of components, with a smaller $\lambda$ value for the \texttt{qtscore} where the components from MDS have been used in the analysis. The two types of analysis are different: the \texttt{egscore} test (1 df) adjusts both genotype and phenotype onto PCs and computes their correlation after adjustment, while the \texttt{qtscore} test adjusted only for the genotype.

If you want to determine how many PCs to include based on GIF, you may use the following command, which calculates $\lambda$ for the first 1 to 10 PCs:

\begin{verbatim}
for (k in 1:10){
  test.tmp <- egscore(disease, data = g.dat, kin = adj.gkin, naxes = k)
  print(test.tmp@lambda$estimate)
}
\end{verbatim}

You may update your analysis based on number of PCs that you want to include.

Finally, you can make a new QQ plot after adjusting for population substructure. You can also compare your Manhattan plot adjusting for population substructure with the one that did not adjust for population structure by plotting them onto one graph and using different symbols for the two types of analyses:

\begin{verbatim}
# QQ plot
obs <- sort(results(test.eg)$Pc1df)
ept <- ppoints(obs)
plot(-log10(ept), -log10(obs), main = "GWAS QQ plot adj. w/ EIGENSTRAT",
     xlab="Expected -log10(pvalue)", ylab="Observed -log10(pvalue)"
) 
abline(0, 1, col = "red")
abline(h = 8, lty = 2)

# Manhattan plot comparison
plot(test.qt, col = "black")
add.plot(test.eg, col = "gray", pch = 3)
legend("topright", c("Original plot","After correction w/ EIGENSTRAT"), pch = c(1,3))
\end{verbatim}
Please note the change of p-value on your new Manhattan plot: it is now less significant for ‘rs4571722’.

**Final task:** You can carry out analysis on binary data as in 1.3 by yourself, if time permits. As you have done Q3 at the end of 1.3, data for binary trait is now loaded as the ‘b.dat’ object. You need to change the argument ‘gaussian’ into ‘binomial’ in ‘scan.glm’ and ‘qtscore’ function, and replace ‘g.dat’ with ‘b.dat’ for all commands in 1.3 and run the new commands to carry out analysis.

**Answers to Problems**

**A1:** In gwa.gabel.tfam file sex is coded as ‘1’ and ‘2’, so is disease status. But GenABEL requires both to be coded as ‘0’ and ‘1’, which you may find in gwa.gabel.praw file. You need to keep in mind these difference when you use a script language to convert *.tfam file to GenABEL phenotype input file.

**A2:** We have 244 samples and 6,388 SNPs. We will be working with quantitative trait (See Figure 1).

**A3:** rs4571722 and rs10008252.

**A4:** Component 1 distinguishes between CEU and MEX (as well as TSI and MEX). Component 2 distinguishes between TSI and CEU.

**A5:** Genomic control adjusts for inflation via dividing the observed $\chi^2$ statistic by $\lambda$ and compute corrected p-values associated with the adjusted $\chi^2$; hence the corrected $\lambda=1$. If we only consider $\lambda$ when we evaluate performance of techniques for adjusting population structure, genomic control, by definition, will naturally outperform all other methods.

**A6:** rs4571722 is found significant (p-value 1.84E-12). The inflation for the test is $\lambda = 1.06$.

**About**

Please contact Dr. Suzanne Leal (sleal@bcm.edu) for questions and comments.
Appendix: Analysis Commands with Output

# Load files
library(GenABEL)
load.gwaa.data(phen = "gwa_gabel_qtl.praw", gen = "gwa_gabel_qtl.raw", force = T)

# Data Summary
slotNames(g.dat)

# GLM test
test.snp <- scan.glm(phen ~ CESNP, family = gaussian(), data = g.dat)
	names(test.snp)

# Score test
test.qt <- qtscore(disease, data = g.dat, trait = "gaussian")

 analysis Log
[1] 0.003559959

> descriptives.scan(test qt)
Summary for top 10 results, sorted by Pldf
Chromosome Position Strand A1 A2 N effB se_effB chi2.1df
rs4571722 8 60326734 u C T 242 -14.943693 2.048587 55.57329
rs10008252 4 179853616 u A G 244 -9.823880 2.085360 22.19239
rs12648775 19 61010979 u C T 244 12.171225 3.300709 13.99734
rs2911297 12 27298622 u C T 244 -7.757537 2.129015 13.34119
rs11865295 16 7471881 u G A 243 6.972167 1.994012 12.25587
rs10520770 18 4485662 u T C 244 7.132231 2.045209 12.16117
rs10459671 17 79655068 u G C 244 10.597100 3.136921 11.41213
rs10509521 11 19856370 u G C 244 -18.127917 4.595076 11.06070
rs9950584 18 72793301 u T C 243 22.001829 6.710260 10.70575
rs10495538 2 6707489 u C T 244 -9.197564 2.843220 10.46465

> rownames(results(test.qt))$Pldf < alpha]
"rs4571722"
> results(test.qt)$Pldf[results(test.qt)$Pldf < alpha]

> results(test.qt)$Pcldf[results(test.qt)$Pcldf < alpha]

1.595956e-12

# --- Plots----#

# QQ plot
obs <- sort(results(test.qt)$Pldf)
plot(log10(p), log10(u), main = "GOF QQ plot, pqt", xlab="Expected -log10(pvalue)", ylab="Observed -log10(pvalue)")
abline(0, 1, col = "red")
abline(h = 8, lty = 2)

# Manhattan plot
plot(test.qt, col = "black")

# ---Adding confounders-----#

test.qt.sex <- qtscore(disease ~ sex, data = g.dat, trait = "gaussian")
rownames(results(test.qt.sex))$Pldf < alpha]
"rs4571722"

> summary(lm(disease ~ sex, data = g.dat)) Call:

lm(formula = disease ~ sex, data = g.dat)

Residuals:
   Min     1Q Median     3Q    Max
-46.355 -17.045   -1.707  18.993  51.125

Coefficients:
                Estimate Std. Error t value Pr(>|t|) (Intercept)
127.975 2.048 62.474 2.0e-16 *** sex
2.802 2.909 -0.963 0.336

Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

Residual standard error: 22.72 on 242 degrees of freedom
Multiple R-squared: 0.00382,  Adjusted R-squared: -0.00296
F-statistic: 0.9281 on 1 and 242 DF,  p-value: 0.3363

# ---WBS analysis----#

gkin <- lbg(g.dat, weight = "freq"
gkin(lld: lld, 10)
NA06989 NA11891 NA11843 NA06984 NA12275
NA06989 0.5045911748 6.281000e+03 6.359000e+03 6.328000e+03 6.342000e+03
NA11891 -0.0034037618 5.033373e-01 6.294000e+03 6.263000e+03 6.280000e+03
NA11843 0.0063830574 4.235451e-03 5.225284e-01 6.341000e+03 6.355000e+03
NA06984 -0.0066330304 7.436543e-05 8.753082e-03 5.200900e-01 6.323000e+03
NA12275 0.0033332693 1.052177e-02 3.398087e-03 1.217706e-02 9.056554e-01
NA06989 -0.0036655267 1.307613e-05 5.198255e-05 3.558385e-04 7.699489e-03
NA12272 -0.0032323012 8.638754e-03 3.031671e-01 7.818281e-04 9.429297e-03
NA07051 0.006402150 1.662206e-02 5.109245e-03 7.993169e-03 3.715566e-03
NA12400 0.0086247762 2.812382e-03 9.973034e-03 1.016512e-02 9.235554e-04
NA12777 -0.0062599088 7.882974e-03 6.707316e-03 1.855176e-04 2.917005e-03
NA06986 NA12272 NA07051 NA12400 NA12777
NA06989 6.357000e+03 6.363000e+03 6.359000e+03 6.356000e+03 6.357000e+03
NA11891 6.291000e+03 6.297000e+03 6.294000e+03 6.290000e+03 6.292000e+03
NA11843 6.369000e+03 6.375000e+03 6.370000e+03 6.367000e+03 6.370000e+03
NA06984 6.330000e+03 6.340000e+03 6.340000e+03 6.335000e+03 6.338000e+03
NA12275 6.352000e+03 6.358000e+03 6.355000e+03 6.353000e+03 6.353000e+03
NA06986 5.428161e-01 6.372000e+03 6.369000e+03 6.364000e+03 6.3680000000

> cps.full <- cmdscale(as.dist(5 - gkin), eig = T, k = 10)
> names(cps.full)
[1] "points" "eig" "g" "ac" "GOF"
> cps <- cps.full@points
> plot(cps[,1], cps[,2], pch = g.dat@phdata@pspn)
> legend(-0.16, 0.06, c("TSI", "MEX", "CEU"), pch = c(1,2,3))

15
### GLM and score test adjusted by including components

#### Incorporating PCs as predictors

```R
gpc.dat <- g.dat
gpc.dat@phdata <- cbind(g.dat@phdata, cps)
test.pc.a <- scan.glm(Wdisease ~ CESNP + C1 + C2 + C3 + C4 + C5, family="gaussian", data = gpc.dat)
99.41%
```

#### Estimate

```R
$estimate
[1] 1.082631
```

#### SE

```R
$se
[1] 0.003559959
```

#### Check for inflation of statistic

```R
obs <- sort(results(test.qt)$chisq.1df)
ept <- sort(chisq((1:length(obs)) / (length(obs) + 1), df = 1))
plot(ept, obs, main = "Genomic control (slope is the inflation factor)", xlab="Expected chisq. 1df", ylab="Observed chisq. 1df")
abline(0, 1, col = "red")
```

#### Definition of GIF

```R
# Conventional definition
median(results(test.qt)$chisq.1df)/0.456
```

#### GenABEL definition

```R
lm(obs ~ ept)$coef[2]
ept
```

#### Corrected chisq pvalue QQPLOT

```R
obs <- sort(results(test.qt)$Pc1df)
ept <- c(1:length(obs)) / (length(obs) + 1)
plot(-log10(ept), -log10(obs), main = "GWAS QQ plot adj. via Genomic Control", xlab="Expected -log10(pvalue)", ylab="Observed -log10(pvalue)")
abline(0, 1, col = "red")
```
```R
> abline(h = 8, lty = 2)
> #----EIGENSTRAT adjustment, PCA----#
> adj.gkin = gkin
> diag(adj.gkin) = hom(g.dat)$Var
> # naxes = 3 is default value
> test.eq <- egscore(disease, data = g.dat, kin = adj.gkin, naxes = 2)
> descriptives.scan(test.eq)

Summary for top 10 results, sorted by Pldf

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Position</th>
<th>Strand</th>
<th>N</th>
<th>effB</th>
<th>se</th>
<th>chl2.1df</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs451722</td>
<td>8</td>
<td>C</td>
<td>242</td>
<td>-2.765603</td>
<td>0.392406</td>
<td>49.64885</td>
</tr>
<tr>
<td>rs1008252</td>
<td>14</td>
<td>G</td>
<td>244</td>
<td>-1.730304</td>
<td>0.352609</td>
<td>24.37114</td>
</tr>
<tr>
<td>rs3001297</td>
<td>12</td>
<td>C</td>
<td>244</td>
<td>-1.687543</td>
<td>0.437113</td>
<td>14.90466</td>
</tr>
<tr>
<td>rs17071376</td>
<td>8</td>
<td>G</td>
<td>242</td>
<td>-1.574788</td>
<td>0.419575</td>
<td>12.69648</td>
</tr>
<tr>
<td>rs950554</td>
<td>14</td>
<td>T</td>
<td>243</td>
<td>2.038677</td>
<td>2.052900</td>
<td>12.40323</td>
</tr>
<tr>
<td>rs10495538</td>
<td>2</td>
<td>C</td>
<td>244</td>
<td>-2.419074</td>
<td>0.708507</td>
<td>11.35487</td>
</tr>
<tr>
<td>rs0440355</td>
<td>3</td>
<td>C</td>
<td>244</td>
<td>1.885975</td>
<td>0.586865</td>
<td>10.36911</td>
</tr>
<tr>
<td>rs2028374</td>
<td>18</td>
<td>u</td>
<td>244</td>
<td>1.137666</td>
<td>0.659710</td>
<td>10.26615</td>
</tr>
<tr>
<td>rs534288</td>
<td>2</td>
<td>u</td>
<td>244</td>
<td>-1.262303</td>
<td>0.402648</td>
<td>9.828216</td>
</tr>
<tr>
<td>rs7603989</td>
<td>4</td>
<td>u</td>
<td>244</td>
<td>-0.866829</td>
<td>0.279394</td>
<td>9.696651</td>
</tr>
</tbody>
</table>

Pldf  Pe2df  eftAB  efOBG chl2.1df P2df
| rs451722   | 1.838777e-12 | 8.201088e-12 | -2.765603 | 5.531297e+00 | NA | NA |
| rs1008252  | 8.086590e-07 | 1.706511e-06 | -1.730304 | 3.476661e+00 | NA | NA |
| rs2812973  | 1.130859e-04 | 1.804442e-04 | -1.687543 | 3.750059e+00 | NA | NA |
| rs17407744  | 6.634156e-04 | 5.474266e-04 | -1.574788 | 3.149577e+00 | NA | NA |
| rs9486935  | 1.281411e-03 | 1.786589e-03 | 6.740158 | 9.55699e-16  | NA | NA |
| rs2028374  | 1.354930e-03 | 1.883300e-03 | 6.861277 | 9.55699e-16  | NA | NA |
| rs534288 | 1.178550e-03 | 2.357705e-03 | -0.126230 | 5.531297e+00 | NA | NA |
| rs7603989   | 1.936620e-03 | 2.639536e-03 | -0.126230 | 5.531297e+00 | NA | NA |

> snp.eq <- row.names(results(test.eq))
> pvalue.eq <- results(test.eq)$Pldf < alpha
> lambda.eq <- test.eq$lambda
> lambda.eq
[1] 1.838777e-12

$se
[1] 0.003169877

> #-------Inflation factor change-----#
> for (k in 1:10)
+ test.tap <- egscore(disease, data = g.dat, kin = adj.gkin, naxes = k)
+ print(test.tap@lambda$estimate)
+
+ [1] 1.082203
+ [1] 1.062756
+ [1] 1.069796
+ [1] 1.07663
+ [1] 1.087112
+ [1] 1.090747
+ [1] 1.101529
+ [1] 1.110265
+ [1] 1.1159
+ [1] 1.127466

> #-------Updated QQ plot with EIGENSTRAT-------#
> obs <- sort(results(test.eq)$P1df)
> ept <- c(1:length(obs)) / (length(obs) + 1)
> qqplot(-log10(ept), -log10(obs), main = "ORAS QQ plot adj. w/ EIGENSTRAT", xlab="Expected -log10(pvalue)", ylab="Observed -log10(pvalue)")
> abline(0, 1, col = "red")
> abline(h = 8, lty = 2)
> # Manhattan plot comparison
> plot(test.eq, col = "black")
> legend("topright", c("Original plot", "After correction w/ EIGENSTRAT"), pch = c(1,3))

> #-------Question 3. binary trait-------#
> # load files to GenABEL
> convert.snp.typed(typed = "gwa_gabel.typed", tfam = "gwa_gabel.tfam", out = "gwa_gabel.raw", strand = "u")
```
Reading individual ids from file gwa_gabel.tfam... 
... done. Read 244 individual ids from file gwa_gabel.tfam#
Reading genotypes from file gwa_gabel.tped#
... done. Read 6388 SNPs from file gwa_gabel.tped#
Writing to file gwa_gabel.raw#... 
... done.  
> b.dat <- load.gwaa.data(phem = "gwa_gabel.phem", gen = "gwa_gabel.raw", force = T)
ids loaded...
marker names loaded...
chromosome data loaded...
map data loaded...
allele coding data loaded...
strand data loaded...
genotype data loaded...
snp, data object created...
assignment of gwa.data object FORCED; X-errors were not checked!
> slotNames(b.dat)
[1] "phdata" "gtdata"
> slotNames(b.dat@gtdata)
[1] "nbytes" "nids" "nsnps" "idnames" "snpnames"
> slotNames(b.dat)
[1] "gtdata" "phdata"
> colnames(b.dat@phdata)
[1] "id" "sex" "disease" "popn"  
> # sample size
> b.dat@gtdata@nids
[1] 244
> # number of cases and controls
> case.size <- length(which(b.dat@phdata@disease == 1))
> control.size <- length(which(b.dat@phdata@disease == 0))
> case.size
[1] 87
> control.size
[1] 157
> # number of SNPs
> snpsb.total <- b.dat@gtdata@nmsnps
> # GLM test
> testb.snp <- scan.glm(disease ~ CSSNP, family = binomial(), data = b.dat)
99.41% Warning message: 
glm.fit: fitted probabilities numerically 0 or 1 occurred
> names(testb.snp)
[1] "P1df" "P2df" "medChilddf" "medCh2df" "snpnames"
[6] "idnames" "formula" "family" "map" "chromosome" 
> alpha <- 5e-8
> testb.snp$snpnames[!testb.snp$P1df < alpha]
[1] "rs10008252" "rs4571722" 
> testb.snpP1df[!testb.snpP1df < alpha]
[1] 1.293455 1.590370
> # Score test
> testb.qt <- qtscore(disease, data = b.dat, trait = "binomial")
> slottedNames(testb.qt)
[1] "results" "dimresults" "annotation" "lambda" "idnames"
[6] "call" "family" 
> descriptives.scan(testb.qt)

Summary for top 10 results, sorted by P1df
<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Position</th>
<th>Strand</th>
<th>A1</th>
<th>A2</th>
<th>N</th>
<th>effB</th>
<th>se_effB</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs4571722</td>
<td>8</td>
<td>C</td>
<td>T</td>
<td>242</td>
<td>0.05946073</td>
<td>0.00639500</td>
<td></td>
</tr>
<tr>
<td>rs10008252</td>
<td>4</td>
<td>C</td>
<td>G</td>
<td>244</td>
<td>0.07409615</td>
<td>0.01029419</td>
<td></td>
</tr>
<tr>
<td>rs2881297</td>
<td>12</td>
<td>C</td>
<td>T</td>
<td>244</td>
<td>0.33987468</td>
<td>0.07372966</td>
<td></td>
</tr>
<tr>
<td>rs2994503</td>
<td>1</td>
<td>G</td>
<td>T</td>
<td>244</td>
<td>2.32325171</td>
<td>0.53814766</td>
<td></td>
</tr>
<tr>
<td>rs708658</td>
<td>3</td>
<td>A</td>
<td>G</td>
<td>244</td>
<td>2.26263438</td>
<td>0.54620704</td>
<td></td>
</tr>
<tr>
<td>rs10458971</td>
<td>16</td>
<td>C</td>
<td>G</td>
<td>244</td>
<td>3.08341655</td>
<td>0.74526127</td>
<td></td>
</tr>
<tr>
<td>rs7254875</td>
<td>19</td>
<td>C</td>
<td>T</td>
<td>244</td>
<td>3.25630392</td>
<td>0.79173503</td>
<td></td>
</tr>
<tr>
<td>rs8625901</td>
<td>12</td>
<td>A</td>
<td>C</td>
<td>244</td>
<td>2.85015942</td>
<td>0.57490941</td>
<td></td>
</tr>
<tr>
<td>rs10820770</td>
<td>18</td>
<td>T</td>
<td>C</td>
<td>244</td>
<td>1.83905681</td>
<td>0.38859968</td>
<td></td>
</tr>
<tr>
<td>rs755417</td>
<td>2</td>
<td>A</td>
<td>G</td>
<td>244</td>
<td>1.86972960</td>
<td>0.58019239</td>
<td></td>
</tr>
<tr>
<td>rs2994503</td>
<td>1</td>
<td>G</td>
<td>T</td>
<td>244</td>
<td>2.32325171</td>
<td>0.53814766</td>
<td></td>
</tr>
<tr>
<td>rs708658</td>
<td>3</td>
<td>A</td>
<td>G</td>
<td>244</td>
<td>2.26263438</td>
<td>0.54620704</td>
<td></td>
</tr>
<tr>
<td>rs10458971</td>
<td>16</td>
<td>C</td>
<td>G</td>
<td>244</td>
<td>3.08341655</td>
<td>0.74526127</td>
<td></td>
</tr>
<tr>
<td>rs7254875</td>
<td>19</td>
<td>C</td>
<td>T</td>
<td>244</td>
<td>3.25630392</td>
<td>0.79173503</td>
<td></td>
</tr>
<tr>
<td>rs8625901</td>
<td>12</td>
<td>A</td>
<td>C</td>
<td>244</td>
<td>2.85015942</td>
<td>0.57490941</td>
<td></td>
</tr>
<tr>
<td>rs10820770</td>
<td>18</td>
<td>T</td>
<td>C</td>
<td>244</td>
<td>1.83905681</td>
<td>0.38859968</td>
<td></td>
</tr>
<tr>
<td>rs755417</td>
<td>2</td>
<td>A</td>
<td>G</td>
<td>244</td>
<td>1.86972960</td>
<td>0.58019239</td>
<td></td>
</tr>
<tr>
<td>rs2994503</td>
<td>1</td>
<td>G</td>
<td>T</td>
<td>244</td>
<td>2.32325171</td>
<td>0.53814766</td>
<td></td>
</tr>
<tr>
<td>rs708658</td>
<td>3</td>
<td>A</td>
<td>G</td>
<td>244</td>
<td>2.26263438</td>
<td>0.54620704</td>
<td></td>
</tr>
<tr>
<td>rs10458971</td>
<td>16</td>
<td>C</td>
<td>G</td>
<td>244</td>
<td>3.08341655</td>
<td>0.74526127</td>
<td></td>
</tr>
<tr>
<td>rs7254875</td>
<td>19</td>
<td>C</td>
<td>T</td>
<td>244</td>
<td>3.25630392</td>
<td>0.79173503</td>
<td></td>
</tr>
<tr>
<td>rs8625901</td>
<td>12</td>
<td>A</td>
<td>C</td>
<td>244</td>
<td>2.85015942</td>
<td>0.57490941</td>
<td></td>
</tr>
<tr>
<td>rs10820770</td>
<td>18</td>
<td>T</td>
<td>C</td>
<td>244</td>
<td>1.83905681</td>
<td>0.38859968</td>
<td></td>
</tr>
<tr>
<td>rs755417</td>
<td>2</td>
<td>A</td>
<td>G</td>
<td>244</td>
<td>1.86972960</td>
<td>0.58019239</td>
<td></td>
</tr>
</tbody>
</table>
rs10459871  17.1177 3.513198e-05 0.98613522  25.25330 3.283336e-06
rs7254875  16.91608 3.906912e-05 0.86163522  5.0177580 1.763394e-04
rs3825391  12.72594 3.606181e-04 1.1111842  3.81578947 1.329198
rs10520770 11.75523 6.067242 1.4111842  3.283336 1.091129e-03
rs7557417  10.38779 1.268515e-03 2.2916667  3.02500000 4.090650e-03

> row.names(results(testb.qt))
[1] "rs10008252"  "rs4571722"  "rs2881297"  "rs2994553"  "rs798658"  "rs10459871"

References

1. [R]
   www.r-project.org

2. [GenABEL]

3. [Devlin and Roeder 1999]

4. [Devlin et al. 2001]

5. [Price et al. 2006]