Homework4_JKGWAS

Pabitra Joshi and Lindsey Kornowske

23 March 2021

Contents

Data and Functions

The GAPIT demo datasets were used for this assignment. They are available from "http://www.zzlab.net/GAPIT/index.html" All credit for these data belong to the Zhiwu Zhang Laboratory.

```r
library(compiler); #need for R version 4.0.4
source("http://www.zzlab.net/StaGen/2021/R/G2P.R");
```

Figure 1: JKGWAS Package Logo

```r
source("http://www.zzlab.net/StaGen/2021/R/GWASbyCor.R");
source("./../functions/HW4_JKGWAS_functions.R");

# Genotype Data
X = read.csv(file = "./../datasets/mdp_numeric.txt", header = TRUE, sep ="");
# Phenotype Data
y = read.csv(file = "./../datasets/CROPS545_Phenotype.txt", header = TRUE, sep = "");
# Covariates Data
CV = read.csv(file = "./../datasets/CROPS545_Covariates.txt", header = TRUE, sep = "");
# SNP information data
SNP = read.csv(file = "./../datasets/mdp_SNP_information.txt", header = TRUE, sep = "");
```

Question 1 and 2

(1) The package should contain at least three input: y, X , and C that are R objects of numeric data frame. Their dimensions are n by 1, n by m, and n by t corresponding to phenotype, genotype and covariate data, where n is number of individuals, m is number of markers, and t is number of covariates. The function should return probability values with dimension of 1 by m for the association tests between phenotype and markers. Markers are tested one at a time with covariates in C included as covariates (15 points). (2) The package should perform PCA and incorporate PCs as cofactors for GWAS. Your package should also automatically exclude the PCs that are in linear dependent to the covariates provided by users. (25 points).

```r
#library(devtools)
#install_github("lindseymaek/HORT545/JKGWAS")
library(JKGWAS); #JKGWAS package
```

JKGWAS Summary

The JKGWAS Package contains four functions that are summarized briefly as follows, more information is located in the JKGWAS Package documentation:

• JKPCA takes genotype (X) data and covariate data (CV), computes the PCA on X, then automatically removes PCs that are linearly dependent to the CVs by method of comparing matrix rank. PCs are removed from the matrix in succesion and those that do not change the rank by removal are determined to be linearly independent because they do not provide additional information.

• JKGLM takes phenotype (y), genotype (X), covariate (CV), and principal component (PC) inputs (ideally provided from JKPCA) and returns p-values calculated for the association tests between the phenotype and

SNPs

• JKQQ takes the pvalues from JKGLM and visualizes them by QQ plot. Expected p-values of length m are simulated from the continuous distribution.

• JKManhattan visualizes the pvalues from JKGLM by Manhattan plot. User input QTNs can also be visualized. The significance threshold can be set, or it will default to Bonferoni correction for alpha = 0.05

Question 3

(3) Develop a user manual and tutorials. Name your package and create a logo. (20 points).

The JKGWAS package is named for Pabitra Joshi and Lindsey Kornowske, the label is displayed in Figure 1.

The JKGWAS package documentation is provided in a separate file, "JKGWAS_0.1.0_PackageDocumentation.pdf" and further information about its use are provided in the user tutorial, "JKGWAS_UserTutorial.pdf"

Question 4

(4) Perform GWAS on the data provided or your own data which must contain cofactors (15 points).

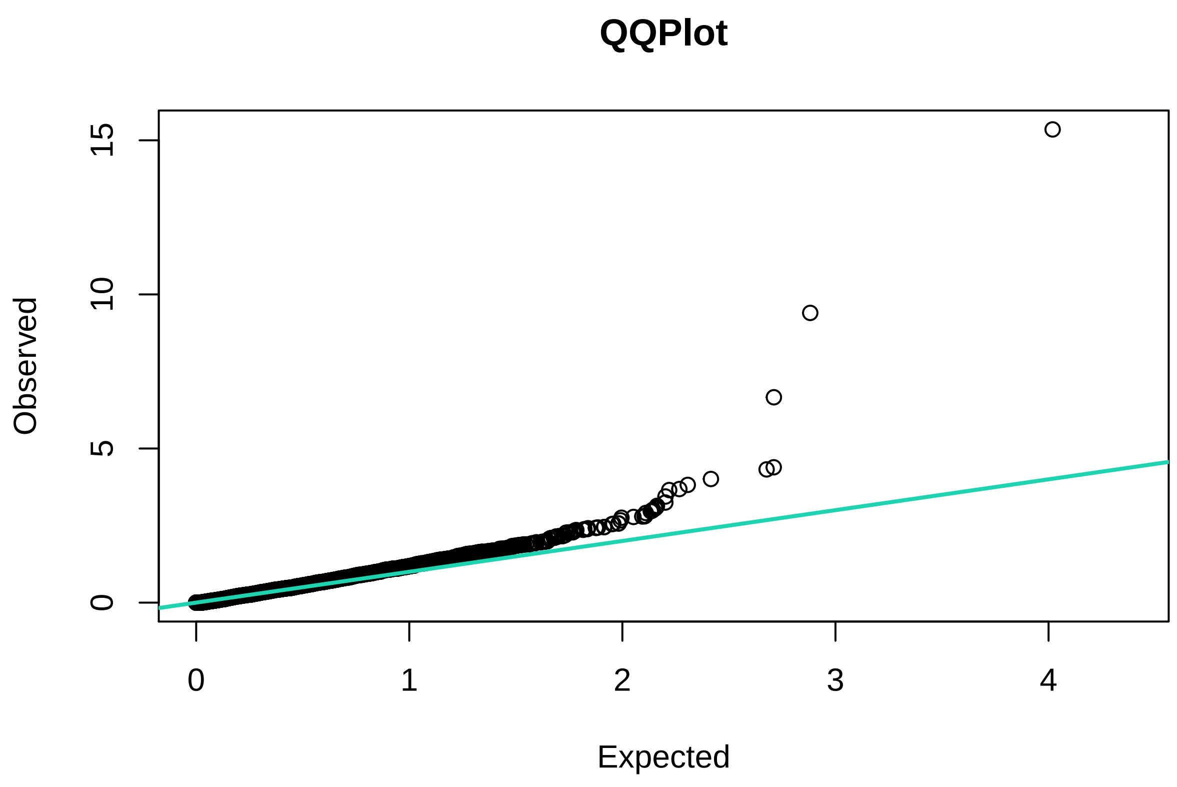
First, GWAS was performed with the phenotype data provided. In the Manhattan plot below, we can see that 4 SNP were detected, but because we do not have information about the QTNs, we do not know whether these significant observations represent true positives or not. Next, we use a simulated phenotype to better assess the performance of the JKGWAS approach.

GWAS with provided phenotype data

```r
## Get Principal Components with JKPCA()
PC = JKPCA(X, CV, npc = 10);

## Perform GWAS by GLM with JKGLM()
Pvals = JKGLM(X = X, y = y, CV, PC);
```

QQ Plot

```r
## Visualize GWAS by QQ Plot with JKQQ()
JKQQ(Pvals);
```



Genome-wide Threshold

With the Bonferroni correction, our threshold is as follows:

```
sig.threshold = 0.05/length(Pvals);
sig.threshold;
```

```
## [1] 1.616554e-05
```

List of Associated SNP

Because the genomic data and SNP data are in the same order, we can index the significant p-values in the SNP data.

```
SNP[Pvals<sig.threshold,];
```

```
##                SNP Chromosome Position
## 937   PZA00615.3          3  1882847
## 1013  PZA02699.1          3 63847564
## 2715 PZA03058.17          9 19292988
## 2717 PZA03058.21          9 19293212
```

MAF of Associated SNP

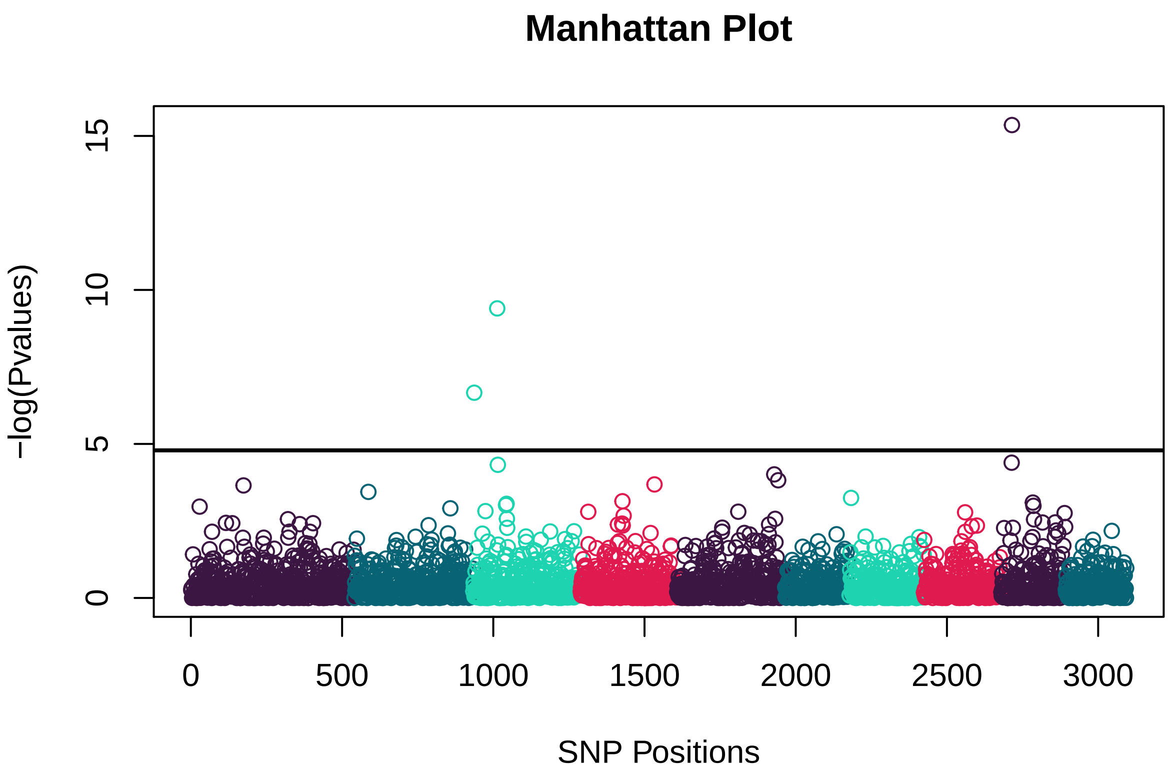
```
X.num = dplyr::select_if(X, is.numeric);

#save associated SNP genomic data to dataframe
SNP.GD = X.num[,Pvals<sig.threshold];

#calculate MAF and store in array
MAF = apply(SNP.GD, 2, function(x)
  {
  allele.freq1 = (sum(x==0)*2 + sum(x==1))/(sum(!is.na(x))*2);
  allele.freq2 = (sum(x==2)*2 + sum(x==1))/(sum(!is.na(x))*2);

  return(min(allele.freq1, allele.freq2));
  })

MAF;
```

```
##  PZA00615.3  PZA02699.1 PZA03058.17 PZA03058.21
##   0.1103203   0.3451957   0.1120996   0.1032028
```

The MAFs for the significant QTNs are shown above. In this case, the MAF is fairly low for 3/4 associated SNP, which tells us that these variants are comparably rare to the major alleles.

Manhattan Plot

```
## Visualize GWAS by Manhattan Plot with JKManhattan()
JKManhattan(Pvals = Pvals, SNP = SNP,sigcutoff = NULL );
```



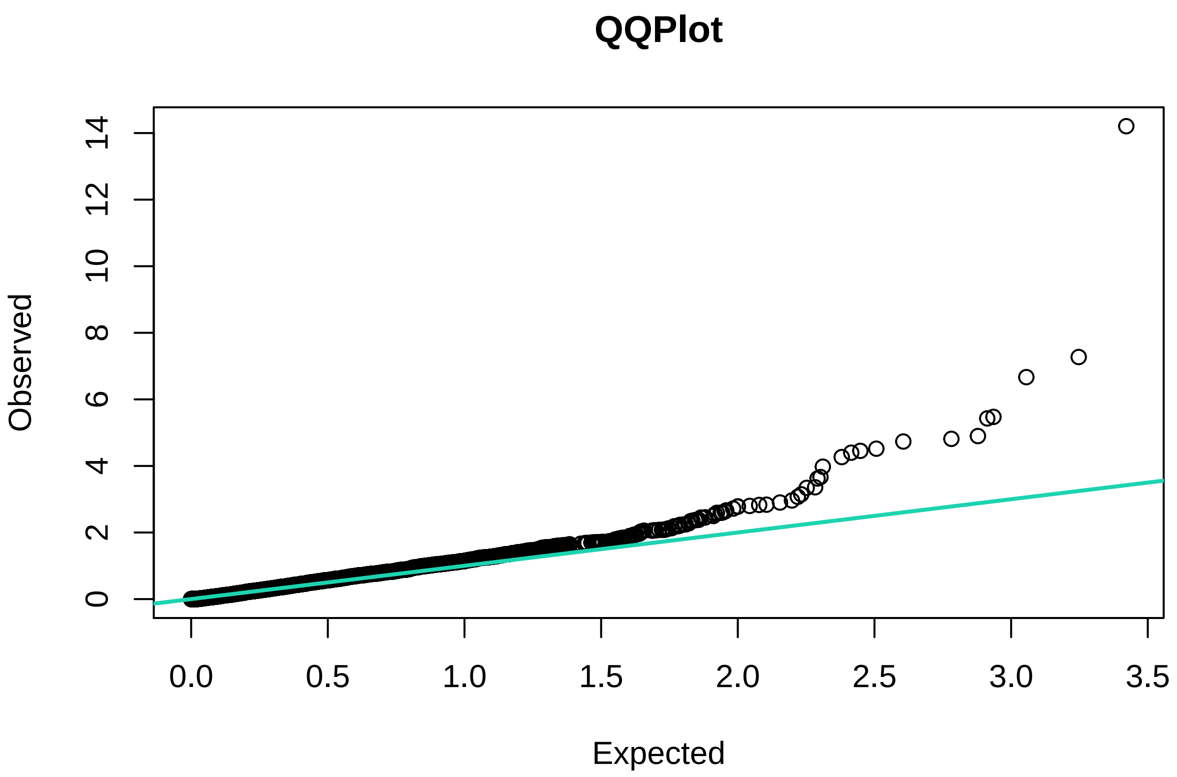
GWAS with simulated phenotype

```
set.seed(12);
#simulate phenotype with heritability value 0.75
G2P.sim = G2P(X= X.num,
              h2= 0.75,
              alpha=1,
              NQTN=10,
              distribution="norm");
```

```r
#save the phenotype to variable for the JKGLM() function
G2P.y = as.data.frame(G2P.sim$y);

G2P.qtn = G2P.sim$QTN.position;

#perform GWAS by GLM with JKGLM
Pvals.sim = JKGLM(X = X, y = G2P.y, CV, PC);
```

QQ Plot

```r
## Visualize GWAS by QQ Plot with JKQQ()
JKQQ(Pvals.sim);
```



Unlike the provided data, the simulated data shows a more dramatic departure from the expected values. We can anticipate more associated SNPs but also more false positives.

List of Associated SNP

```r
SNP[Pvals.sim<sig.threshold,];
```

```
##              SNP Chromosome  Position
## 336   PZB00011.1          1 233401031
## 339   PZB00116.2          1 233401241
## 450   PZA03188.2          1 280719532
## 1093  PZB01683.2          3 156252241
## 1160 PZA00482.10          3 180294773
## 2907  PZB01301.7         10   9748233
## 2909  PZB01301.5         10   9748559
```

MAF of Associated SNP

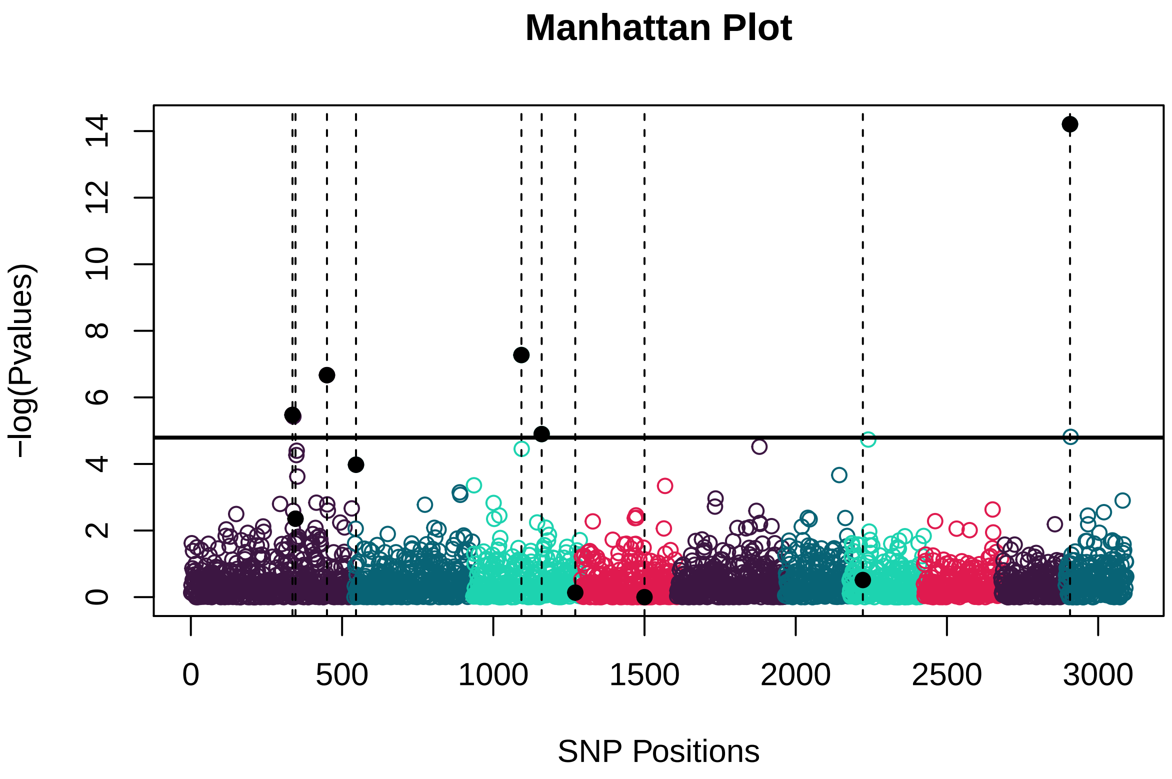
```r
#save associated SNP genomic data to dataframe
SNP.GD = X.num[,Pvals.sim<sig.threshold];

#calculate MAF and store in array
MAF = apply(SNP.GD, 2, function(x)
  {
  allele.freq1 = (sum(x==0)*2 + sum(x==1))/(sum(!is.na(x))*2);
  allele.freq2 = (sum(x==2)*2 + sum(x==1))/(sum(!is.na(x))*2);

  return(min(allele.freq1, allele.freq2));
  })

MAF;
```

```
##  PZB00011.1  PZB00116.2  PZA03188.2  PZB01683.2 PZA00482.10  PZB01301.7
##   0.4448399   0.4430605   0.4056940   0.1494662   0.4412811   0.3647687
##  PZB01301.5
##   0.3523132
```

In the case of the simulated data, the MAF is much closer to 0.5, for at least half of the associated SNPs. These alleles do not represent rare variants as compared with the provided phenotype data.

Manhattan Plot

```r
## Visualize GWAS by Manhattan Plot with JKManhattan()
JKManhattan(Pvals = Pvals.sim, SNP = SNP, sigcutoff = NULL, QTN = G2P.qtn);
```



If we use a simulated phenotype instead of the provided phenotype data, 50% of our 10 QTN are detected correctly. meaning that there is a 50% false negative rate. With 5 QTN detected out of 7 associated SNPs total, that is a True Positive rate of 71.4%.

Question 5

(5) Demonstrate that your method is superior to the competing method (GWASbyCor) through simulation with at least 30 replicates (25 points).

See file HW4_JKGWAS_functions.R for the function source code.

In order to compare GWASbyCor and GWASbyGLM, we created a function called compareGWASnTimes. The function arguments are:

- n, the number of times to run the simulation
- X, the numeric genomic data
- qtn, the number of qtns to be simulated
- CV, the covariate matrix to be passed to the JKGLM function. The default value is NULL
- PC, the principal component matrix to be passed to the JKGLM function. The default value is NULL

For each iteration, the G2P function simulates the phenotype for X with a heritability of 0.75. Then, the output phenotype is used to compute the GWAS by cor and the GWAS by GLM. The number of True Positives, as well as the True Positive Rate, which is calculated as the number of QTNs that is correctly identified out of all significant SNPs (p-value is smaller than 0.05/total pvalues, Bonferroni correction is automatic). These two dataframes are output as a list, where the first item is the count of true positive QTNs and the second item is the true positive rate.

```r
# Run the simulation comparison 30 times
set.seed(8345);
test = compareGWASnTimes(n=30, X = X, qtn = 10, CV = CV, PC = PC);
#test;
```

The output of the function compareGWASnTimes() is a list containing lists for the results for GWAS by COR and GWAS by GLM for 4 items; 1) the number of QTNs detected among the significant SNPs 2) the true positive rate 3) the number of non-QTN SNPs detectected among the significant SNPs and 4) the false positive rate.

True Positive Rate

```r
mean(test[[2]]$COR); #mean TPR for Cor is 0.44
```

```
## [1] 0.4387465
```

```r
mean(test[[2]]$GLM); #mean TPR for GLM is 0.71
```

```
## [1] 0.7089453
```

```r
mean(test[[4]]$COR); #mean FPR for Cor is 0.56
```

```
## [1] 0.5592927
```

```r
mean(test[[4]]$GLM); #mean FPR for GLM is 0.29
```

```
## [1] 0.2910547
```

By these calculations, we can see that the standard deviation of the true positive rate for each method is comparable. The True Positive Rate is marginally higher for the GWAS by GLM than the GWAS by Cor.

Statistical Inference

H_o; the mean rates are equal

H_1; the mean rates are not equal

significance threshold: 0.05

True positive rate

p = 1.777e-05

Reject the null hypothesis; there is sufficient evidence to suggest that the mean true positive rates for GWAS by Cor and GWAS by JKGWAS are different. The mean TPR for JKGWAS was 0.71, almost double that for GWAS by Cor (0.44).

```
t.test(x = test[[2]]$COR, y = test[[2]]$GLM)
```

```
##
##  Welch Two Sample t-test
##
## data:  test[[2]]$COR and test[[2]]$GLM
## t = -4.6905, df = 56.355, p-value = 1.777e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3855803 -0.1548172
## sample estimates:
## mean of x mean of y
## 0.4387465 0.7089453
```

False positive rate

p-value = 1.883e-05

Reject the null hypothesis; there is sufficient evidence to suggest that the mean false positive rates for GWAS by Cor and GWAS by JKGWAS are different. The mean FPR for JKGWAS was 0.29, almost half that for GWAS by Cor (0.56).

```
t.test(x = test[[4]]$COR, y = test[[4]]$GLM);
```

```
##
##  Welch Two Sample t-test
##
## data:  test[[4]]$COR and test[[4]]$GLM
## t = 4.6732, df = 56.465, p-value = 1.883e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1532735 0.3832024
## sample estimates:
## mean of x mean of y
## 0.5592927 0.2910547
```

Extra Credit

(6) Demonstrate that your package is better than BLINK C version (http://zzlab.net/blink) on either statistical power or speed (25 points).

See file HW4_JKGWAS_functions.R for the function source code.

We created two functions to test this data. simBLINKnTimes(), where the user can input the number of desired iterations, the genomic data, and the genetic map data, then get the true and false positive rates, as well as the times required for the GWAS by BLINK. simJKGWASnTimes() does the same thing, but for our JKGWAS function.

Three sets of GAPIT demo datasets from ZZlab were used. In each trial, both of the functions mentioned above were run 30 times. Then the run time, true positive rate, and false positive rate were compared by two-sided t-test for each method to evaluate the null hypothesis of equal means. Across all three datasets, the computing time varied within each method, but in each case, the time to complete JKGWAS was significantly faster than BLINK (alpha = 0.05). The average computing times for the datasets are reported in the chunks below.

BLINK, however, appears to be worth the additional time cost, because it was found to have higher power, with a significantly higher True Positive Rate and a significantly lower False Negative Rate than JKGWAS in all trials. The average false and true positive rates are reported in the chunks below.

```
## Loading BLINK packages and source code
source("http://zzlab.net/GAPIT/gapit_functions.txt");
source("http://zzlab.net/GAPIT/GAPIT.library.R")
```

Dataset 1

```
#test1 - Dataset 1 is the same as is used throughout assignment
set.seed(1);
BLINK1 = simBLINKnTimes(30, X = X, SNP = SNP);
set.seed(1);
GLM1 = simJKGWASnTimes(30, X = X, SNP = SNP);
```

Statistical Inference

```
#compare time
t.test(BLINK1$BLINK_Time, GLM1$GLM_Time); # pval <0.05 JKGLM is faster
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK1$BLINK_Time and GLM1$GLM_Time
## t = 14.423, df = 57.187, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  4.107527 5.431890
## sample estimates:
## mean of x mean of y
## -11.26499 -16.03470
```

```
#compare TPR
t.test(BLINK1$TPR, GLM1$TPR); # pval <0.05, Blink is higher
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK1$TPR and GLM1$TPR
## t = 4.9314, df = 45.33, p-value = 1.139e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1649980 0.3927533
## sample estimates:
## mean of x mean of y
## 0.8833333 0.6044577
```

```
#compare FPR
t.test(BLINK1$FPR, GLM1$FPR); #pval <0.05, Blink is lower
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK1$FPR and GLM1$FPR
## t = -3.3676, df = 52.682, p-value = 0.001423
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.25256633 -0.06399452
## sample estimates:
## mean of x mean of y
## 0.1166667 0.2749471
```

Dataset 2

```
#test 2 - Dataset 2 is from zzlab - all files with "2"
# Genotype Data
X2 = read.csv(file = "./../datasets/mdp_numeric2.txt", header = TRUE, sep ="");
# SNP information data
SNP2 = read.csv(file = "./../datasets/mdp_SNP_information2.txt", header = TRUE, sep = "");
```

```
set.seed(2);
BLINK2 = simBLINKnTimes(30, X = X2, SNP = SNP2);
set.seed(2);
GLM2 = simJKGWASnTimes(30, X = X2, SNP = SNP2);
```

Statistical Inference

```
#compare time
t.test(BLINK2$BLINK_Time, GLM2$GLM_Time); # pval <0.05 JKGLM is faster
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK2$BLINK_Time and GLM2$GLM_Time
## t = -21.247, df = 30.377, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.212143 -3.473741
## sample estimates:
## mean of x mean of y
## -8.384335 -4.541393
```

```
#compare TPR
t.test(BLINK2$TPR, GLM2$TPR);# pval <0.05, Blink is higher
```

```
##
##  Welch Two Sample t-test
##
```

```
## data:  BLINK2$TPR and GLM2$TPR
## t = 6.6553, df = 44.485, p-value = 3.489e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2042788 0.3816603
## sample estimates:
## mean of x mean of y
## 0.8966402 0.6036706
```

```r
#compare FPR
t.test(BLINK2$FPR, GLM2$FPR); #pval <0.05, Blink is lower
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK2$FPR and GLM2$FPR
## t = -5.1322, df = 48.475, p-value = 5.023e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2823008 -0.1234002
## sample estimates:
## mean of x mean of y
## 0.1033598 0.3062103
```

Dataset 3

```r
#test 3 - Dataset 3 is from zzlab - all files with "3"
# Genotype Data
X3 = read.csv(file = "./../datasets/mdp_numeric3.txt", header = TRUE, sep ="");
# SNP information data
SNP3 = read.csv(file = "./../datasets/mdp_SNP_information3.txt", header = TRUE, sep = "");
```

```r
set.seed(3);
BLINK3 = simBLINKnTimes(30, X = X3, SNP = SNP3);
set.seed(3);
GLM3 = simJKGWASnTimes(30, X = X3, SNP = SNP3);
```

Statistical Inference

```r
#compare time
t.test(BLINK3$BLINK_Time, GLM3$GLM_Time); # pval <0.05 JKGLM is faster
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK3$BLINK_Time and GLM3$GLM_Time
## t = -22.046, df = 29.862, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.680828 -3.886964
## sample estimates:
## mean of x mean of y
## -8.800219 -4.516323
```

```r
#compare TPR
t.test(BLINK3$TPR, GLM3$TPR); # pval <0.05, Blink is higher
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK3$TPR and GLM3$TPR
## t = 9.0729, df = 50.977, p-value = 3.207e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2693845 0.4224769
## sample estimates:
## mean of x mean of y
## 0.9211111 0.5751804
```

```r
#compare FPR
t.test(BLINK3$FPR, GLM3$FPR); #pval <0.05, Blink is lower
```

```
##
##  Welch Two Sample t-test
##
## data:  BLINK3$FPR and GLM3$FPR
## t = -7.7825, df = 51.584, p-value = 2.96e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3678890 -0.2170412
## sample estimates:
##  mean of x  mean of y
## 0.07888889 0.37135402
```