Homework4_JKGWAS

Pabitra Joshi and Lindsey Kornowske

20 March 2021

Contents

Data and Functions

The class maize datasets were used for this assignment.

```
library(JKGWAS)

# Genotype Data
```

```r
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).

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.

Question 4

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

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

Figure 1: JKGWAS Package Logo

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

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

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

Question 5

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

```
# compareGWASnTimes = function(n = 100, X = X, qtn = 10) {
#    X = dplyr::select_if(X, is.numeric);
#    # create array to store average for each iteration
#   store.means = array();
#   store.sig = array();
#   store.total = array();
#
#   for(i in 1:n){
# # first, simulate phenotype
#    # same parameters as Q2-4
# G2P.sim = G2P(X= X,
#              h2= 0.75,
#              alpha=1,
#              NQTN=qtn,
#              distribution="norm");
#
# # get QTN positions in GD data
# G2P.sim.qtn = G2P.sim$QTN.position;
#
# #get vector of pvals
#
# #by correlation
# p.list.cor = GWASbyCor(X, G2P.sim$y);
#
# #by glm
# p.list.glm = JKGLM(X, y, CV, PC);
#
# p.df = as.matrix(cbind(p.list.cor, p.list.glm));
# names(p.df) = c("COR", "GLM");
#
# for(j in 1:ncol(p.df)){
#   p.index = p.df[,i]
# }
#
# #sort vector of pvals as increasing
# p.index.cor =sort(p.list.cor, decreasing = FALSE);
# p.sig.cor = p.index.cor[p.index.cor < 0.05/length(p.index.cor)];
#
# #identify qtn pvals
# qtn.p.cor = p.list.cor[ G2P.sim.qtn]
#
# #find all qtn pvals in total list
# truePos.cor=intersect(p.sig.cor,qtn.p.cor);
# #find all significant pvals not in qtn list
# falsePos.cor=setdiff(p.sig.cor, qtn.p.cor);
# #save the number of true positives for this iteration
# store.means[i] = length(truePos.cor);
#
# p.list.glm = JKGLM(X, y, CV, PC);
#
# }
#
#   }
#   #get numeric count data
#   store.means = as.numeric(as.character(store.means));
#   #combine into single df for summary stats
#   res = as.data.frame(cbind(store.means, store.sig, store.total));
#   #compute summary statistics
#   means.result = res %>% summarize_if(is.numeric, .funs = c("mean", "sd"));
#   means.result;
# }
```

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).