*Final Doc P1*

Gabrielle Salamanca

August 14, 2023

This document is Part 1 of the final doc research. This will explain the prep work and jump into the SKAT and plot work with only Y. Part 2 will jump into work with Y1.

### Dataset Reading

C19vj is the dataset holding all the vj genes and its values of covid patients, both recovered and active.

vj is the dataset holding all the vj genes and its values of healthy patients.

patients is the dataset that documents certain patients, both covid and healthy; symptom timing, both in days and weeks; and extra commentary.

### New dataset

With some transforming, standardization, and cleanup, all three datasets were combined to make a new dataset called gene.

C19vj and vj were first merged by the column vjGene, the dataset being named combine. Because of the multiple NAs in the value column, we replaced them with . The rows were then renamed to the entries in the vjGene column, which was then made NULL. After that, we natural logged the entire dataset before transposing it. This allowed the genes to be the columns and the patients as rows. Then, the log dataset was standardized.

We did have to update the rownames, because we want the patient ID to look like this: 1\_1, 10\_3, and such. Now, we do need to add in a Sample.ID column, because without a common column, we can’t merge the last dataset: patients.

It is because, we want the dataset to look like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Patient | VJ1 | VJ2 | … | Y | Y1 |
| Pat 1 | … | … | … | Healthy | Healthy |
| Pat 2 | … | … | … | Disease | Active |
| .  .  . | .  .  . | .  .  . | .  .  . | .  .  . | .  .  . |
| Pat n-1 | … | … | … | Disease | Recovered |
| Pat n | .. | … | … | Healthy | Healthy |

The dataset had to become a dataframe first before the new column could be added. Once added, we can rearrange the columns and delete some that we don’t need. Then, we can make a new Y column to have entries “disease” and “healthy” based on the Y1 column.

To note, there are NAs in terms of patients. For example, the healthy patients doesn’t extend all the way to 39, it ended at 23. But they were kept in instead of removed, with the idea that it’ll be done later down the line or the NAs will be replaced accordingly.

Let’s do a quick dimension check:

## Dimensions of C19vj:

## [1] 699 71

##   
## Dimensions of vj:

## [1] 687 40

##   
## Dimensions of patients:

## [1] 92 8

##   
## Dimensions of genes:

## [1] 109 632

### 

### SKAT

Now that the gene dataset has been prepped, we can start prepping for the work with the SKAT package. Now, we’re going to separating the genes by v and j genes, so we need to make partial strings. Then, we use these partial strings to make columns. We will then use these columns to make subsets of gene, and do note that we are using as.matrix(). The reason is because SKAT only uses matrices for the object so we have to turn dataframes into them.

Now, we need to fix up the NAs in Y. As mentioned before, the patients dataset does not hold every single possible patient, but we can easily insert the Y’s due to knowing what dataset they originate from.

Now, we can start doing SKAT for loops for both v and j genes. We will need to make the

null model and p-value vectors to store them.

We’ll start for the v gene:

## vgene.idx pvalue  
## 1 1 0.668651045  
## 2 2 1.000000000  
## 3 3 0.420950423  
## 4 4 0.262211024  
## 5 5 0.555894154  
## 6 6 1.000000000  
## 7 7 1.000000000  
## 8 8 0.499602847  
## 9 9 0.079478338  
## 10 10 1.000000000  
## 11 11 0.378742469  
## 12 12 0.546002782  
## 13 13 0.497412541  
## 14 14 1.000000000  
## 15 15 0.661848393  
## 16 16 0.025982621  
## 17 17 0.137833835  
## 18 18 0.586074879  
## 19 19 1.000000000  
## 20 20 0.553002489  
## 21 21 1.000000000  
## 22 22 0.100939543  
## 23 23 0.110306121  
## 24 24 0.555211809  
## 25 25 1.000000000  
## 26 26 0.030314578  
## 27 27 0.001152038  
## 28 28 0.632268120  
## 29 29 0.586814812  
## 30 30 0.165333038  
## 31 31 0.134271002  
## 32 32 1.000000000  
## 33 33 0.598178476  
## 34 34 0.860537592  
## 35 35 0.362517231  
## 36 36 1.000000000  
## 37 37 0.784106886  
## 38 38 1.000000000  
## 39 39 1.000000000  
## 40 40 0.594078754  
## 41 41 0.288679374  
## 42 42 1.000000000  
## 43 43 1.000000000  
## 44 44 1.000000000  
## 45 45 0.174043297  
## 46 46 1.000000000  
## 47 47 0.329845475  
## 48 48 0.898799640  
## 49 49 0.283366840  
## 50 50 0.571729323

Next is the j gene:

## jgene.idx p-value  
## 1 1 0.626006017  
## 2 2 0.496538969  
## 3 3 0.660185045  
## 4 4 0.849857480  
## 5 5 0.330521399  
## 6 6 0.256266217  
## 7 7 0.036341153  
## 8 8 0.237610902  
## 9 9 0.009703486  
## 10 10 0.033938129  
## 11 11 0.282042372  
## 12 12 0.317750321  
## 13 13 0.255148224

Now with these, we can see that the notable v strings are TRBV2 (v16), TRBV30 (v26), and TRBV4-1 (v27); while the notable j strings are TRBJ2-1 (j7), TRBJ2-3 (j9), and TRBJ2-4 (j10). Let’s do a p-value adjustment to see if they still stand, especially because these are small samples.

Here’s for the v genes:

## V1 p.pv  
## 1 1 1.0000000  
## 2 2 1.0000000  
## 3 3 1.0000000  
## 4 4 1.0000000  
## 5 5 1.0000000  
## 6 6 1.0000000  
## 7 7 1.0000000  
## 8 8 1.0000000  
## 9 9 1.0000000  
## 10 10 1.0000000  
## 11 11 1.0000000  
## 12 12 1.0000000  
## 13 13 1.0000000  
## 14 14 1.0000000  
## 15 15 1.0000000  
## 16 16 1.0000000  
## 17 17 1.0000000  
## 18 18 1.0000000  
## 19 19 1.0000000  
## 20 20 1.0000000  
## 21 21 1.0000000  
## 22 22 1.0000000  
## 23 23 1.0000000  
## 24 24 1.0000000  
## 25 25 1.0000000  
## 26 26 1.0000000  
## 27 27 0.0576019  
## 28 28 1.0000000  
## 29 29 1.0000000  
## 30 30 1.0000000  
## 31 31 1.0000000  
## 32 32 1.0000000  
## 33 33 1.0000000  
## 34 34 1.0000000  
## 35 35 1.0000000  
## 36 36 1.0000000  
## 37 37 1.0000000  
## 38 38 1.0000000  
## 39 39 1.0000000  
## 40 40 1.0000000  
## 41 41 1.0000000  
## 42 42 1.0000000  
## 43 43 1.0000000  
## 44 44 1.0000000  
## 45 45 1.0000000  
## 46 46 1.0000000  
## 47 47 1.0000000  
## 48 48 1.0000000  
## 49 49 1.0000000  
## 50 50 1.0000000

And here’s for the j genes:

## V1 p.pj  
## 1 1 1.0000000  
## 2 2 1.0000000  
## 3 3 1.0000000  
## 4 4 1.0000000  
## 5 5 1.0000000  
## 6 6 1.0000000  
## 7 7 0.4072576  
## 8 8 1.0000000  
## 9 9 0.1261453  
## 10 10 0.4072576  
## 11 11 1.0000000  
## 12 12 1.0000000  
## 13 13 1.0000000

None of them now seem significant, all too large to pass an .

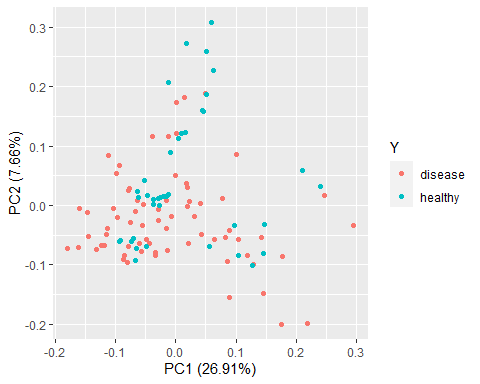
### PCA Plots

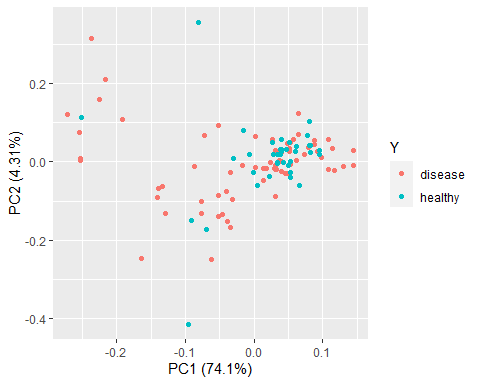
Now, let’s try running some Principal Component Analysis plots. These are used to visualize and explore the variability in high-dimensional data through dimensionality reduction. They show the distribution of data points in a lower-dimensional space defined by the principal components.

First, we need to do some prep such as make dataframes and doing the actual pca on them. We’ll do it on the three notable strings for each gene (before p-value adjustment) and the entire dataset.

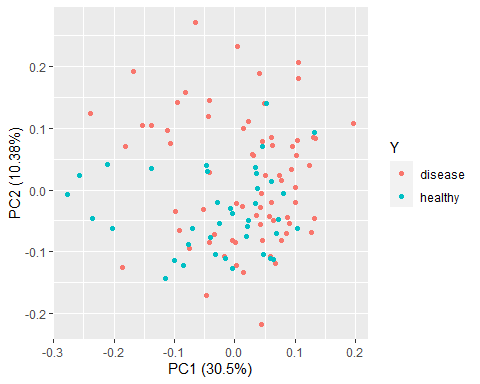
Finally, we can do the plots:

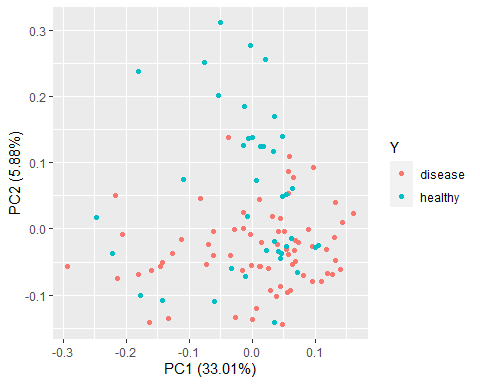
**PCA: TRBV2 (v16)**

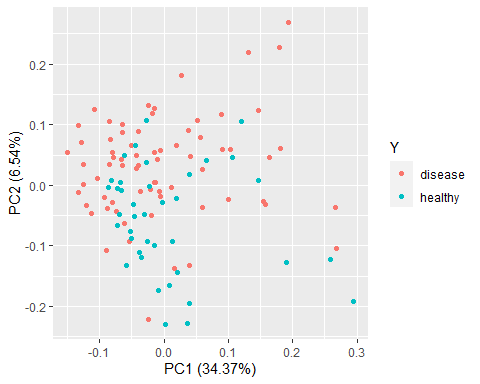


**PCA: TRBV30 (v26)**

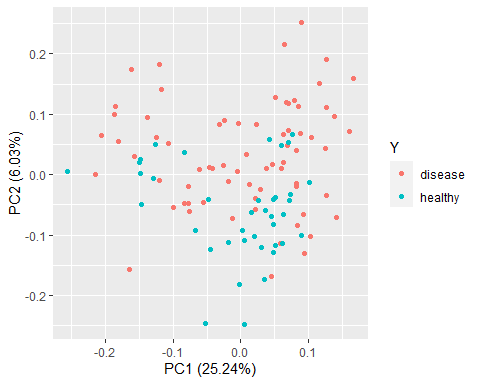
**PCA: TRBV4-1 (v27)**



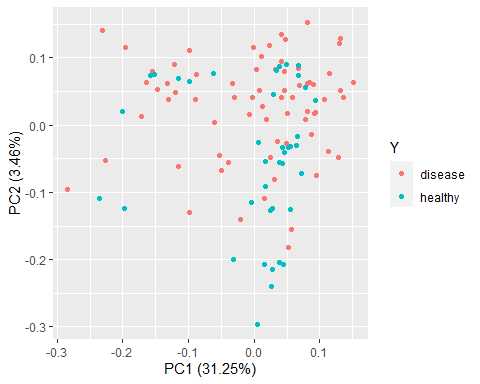
**PCA: TRBJ2-1 (j7)**

**PCA: TRBJ2-3 (j9)**

**PCA: TRBJ2-4 (j10)**



**PCA: Full Dataset**



**Code Appendix**

knitr::opts\_chunk$set(echo = FALSE)  
library(dplyr)  
library(factoextra)  
library(ggfortify)  
library(ggplot2)  
library(psych)  
library(readr)  
library(readxl)  
library(SKAT)  
library(tidyr)  
C19vj <- read\_csv("D:/Coding/R Storage/Summer TCR Project/TCR Datasets/dt.COVID\_TCR.vjGene.p.csv",  
 show\_col\_types = FALSE)  
vj <- read\_csv("D:/Coding/R Storage/Summer TCR Project/TCR Datasets/dt.HD\_TCR.vjGene.p.csv",  
 show\_col\_types = FALSE)  
patients <- read\_excel("D:/Coding/R Storage/Summer TCR Project/TCR Datasets/dt.info\_edited.xlsx")  
gene <- read\_excel("D:/Coding/R Storage/Summer TCR Project/TCR Datasets/fullgenes.xlsx")  
attach(gene)  
cat("Dimensions of C19vj: \n")  
dim(C19vj)  
cat("\nDimensions of vj: \n")  
dim(vj)  
cat("\nDimensions of patients: \n")  
dim(patients)  
cat("\nDimensions of genes: \n")  
dim(gene)  
# v gene  
stringv1 <- "TRBV10-1"  
stringv2 <- "TRBV10-2"  
stringv3 <- "TRBV10-3"  
stringv4 <- "TRBV11-1"  
stringv5 <- "TRBV11-2"  
stringv6 <- "TRBV11-3"  
stringv7 <- "TRBV12-1"  
stringv8 <- "TRBV12-3"  
stringv9 <- "TRBV12-4"  
stringv10 <- "TRBV12-5"  
  
stringv11 <- "TRBV13"  
stringv12 <- "TRBV14"  
stringv13 <- "TRBV15"  
stringv14 <- "TRBV18"  
stringv15 <- "TRBV19"  
stringv16 <- "TRBV2"  
stringv17 <- "TRBV20-1"  
stringv18 <- "TRBV21-1"  
stringv19 <- "TRBV23-1"  
stringv20 <- "TRBV24-1"  
  
stringv21 <- "TRBV25-1"  
stringv22 <- "TRBV27"  
stringv23 <- "TRBV28"  
stringv24 <- "TRBV29-1"  
stringv25 <- "TRBV3-2"  
stringv26 <- "TRBV30"  
stringv27 <- "TRBV4-1"  
stringv28 <- "TRBV4-2"  
stringv29 <- "TRBV4-3"  
stringv30 <- "TRBV5-1"  
  
stringv31 <- "TRBV5-3"  
stringv32 <- "TRBV5-4"  
stringv33 <- "TRBV5-5"  
stringv34 <- "TRBV5-6"  
stringv35 <- "TRBV5-7"  
stringv36 <- "TRBV5-8"  
stringv37 <- "TRBV6-1"  
stringv38 <- "TRBV6-2"  
stringv39 <- "TRBV6-3"  
stringv40 <- "TRBV6-4"  
  
stringv41 <- "TRBV6-5"  
stringv42 <- "TRBV6-6"  
stringv43 <- "TRBV6-7"  
stringv44 <- "TRBV6-8"  
stringv45 <- "TRBV6-9"  
stringv46 <- "TRBV7-2"  
stringv47 <- "TRBV7-3"  
stringv48 <- "TRBV7-4"  
stringv49 <- "TRBV7-5"  
stringv50 <- "TRBV7-6"  
  
# j gene  
stringj1 <- "TRBJ1-1"  
stringj2 <- "TRBJ1-2"  
stringj3 <- "TRBJ1-3"  
stringj4 <- "TRBJ1-4"  
stringj5 <- "TRBJ1-5"  
stringj6 <- "TRBJ1-6"  
stringj7 <- "TRBJ2-1"  
stringj8 <- "TRBJ2-2"  
stringj9 <- "TRBJ2-3"  
stringj10 <- "TRBJ2-4"  
  
stringj11 <- "TRBJ2-5"  
stringj12 <- "TRBJ2-6"  
stringj13 <- "TRBJ2-7"  
# v gene  
colv1 <- grep(stringv1, names(gene), value = TRUE)  
colv2 <- grep(stringv2, names(gene), value = TRUE)  
colv3 <- grep(stringv3, names(gene), value = TRUE)  
colv4 <- grep(stringv4, names(gene), value = TRUE)  
colv5 <- grep(stringv5, names(gene), value = TRUE)  
colv6 <- grep(stringv6, names(gene), value = TRUE)  
colv7 <- grep(stringv7, names(gene), value = TRUE)  
colv8 <- grep(stringv8, names(gene), value = TRUE)  
colv9 <- grep(stringv9, names(gene), value = TRUE)  
colv10 <- grep(stringv10, names(gene), value = TRUE)  
  
colv11 <- grep(stringv11, names(gene), value = TRUE)  
colv12 <- grep(stringv12, names(gene), value = TRUE)  
colv13 <- grep(stringv13, names(gene), value = TRUE)  
colv14 <- grep(stringv14, names(gene), value = TRUE)  
colv15 <- grep(stringv15, names(gene), value = TRUE)  
colv16 <- grep(stringv16, names(gene), value = TRUE)  
colv17 <- grep(stringv17, names(gene), value = TRUE)  
colv18 <- grep(stringv18, names(gene), value = TRUE)  
colv19 <- grep(stringv19, names(gene), value = TRUE)  
colv20 <- grep(stringv20, names(gene), value = TRUE)  
  
colv21 <- grep(stringv21, names(gene), value = TRUE)  
colv22 <- grep(stringv22, names(gene), value = TRUE)  
colv23 <- grep(stringv23, names(gene), value = TRUE)  
colv24 <- grep(stringv24, names(gene), value = TRUE)  
colv25 <- grep(stringv25, names(gene), value = TRUE)  
colv26 <- grep(stringv26, names(gene), value = TRUE)  
colv27 <- grep(stringv27, names(gene), value = TRUE)  
colv28 <- grep(stringv28, names(gene), value = TRUE)  
colv29 <- grep(stringv29, names(gene), value = TRUE)  
colv30 <- grep(stringv30, names(gene), value = TRUE)  
  
colv31 <- grep(stringv31, names(gene), value = TRUE)  
colv32 <- grep(stringv32, names(gene), value = TRUE)  
colv33 <- grep(stringv33, names(gene), value = TRUE)  
colv34 <- grep(stringv34, names(gene), value = TRUE)  
colv35 <- grep(stringv35, names(gene), value = TRUE)  
colv36 <- grep(stringv36, names(gene), value = TRUE)  
colv37 <- grep(stringv37, names(gene), value = TRUE)  
colv38 <- grep(stringv38, names(gene), value = TRUE)  
colv39 <- grep(stringv39, names(gene), value = TRUE)  
colv40 <- grep(stringv40, names(gene), value = TRUE)  
  
colv41 <- grep(stringv41, names(gene), value = TRUE)  
colv42 <- grep(stringv42, names(gene), value = TRUE)  
colv43 <- grep(stringv43, names(gene), value = TRUE)  
colv44 <- grep(stringv44, names(gene), value = TRUE)  
colv45 <- grep(stringv45, names(gene), value = TRUE)  
colv46 <- grep(stringv46, names(gene), value = TRUE)  
colv47 <- grep(stringv47, names(gene), value = TRUE)  
colv48 <- grep(stringv48, names(gene), value = TRUE)  
colv49 <- grep(stringv49, names(gene), value = TRUE)  
colv50 <- grep(stringv50, names(gene), value = TRUE)  
  
# j gene  
colj1 <- grep(stringj1, names(gene), value = TRUE)  
colj2 <- grep(stringj2, names(gene), value = TRUE)  
colj3 <- grep(stringj3, names(gene), value = TRUE)  
colj4 <- grep(stringj4, names(gene), value = TRUE)  
colj5 <- grep(stringj5, names(gene), value = TRUE)  
colj6 <- grep(stringj6, names(gene), value = TRUE)  
colj7 <- grep(stringj7, names(gene), value = TRUE)  
colj8 <- grep(stringj8, names(gene), value = TRUE)  
colj9 <- grep(stringj9, names(gene), value = TRUE)  
colj10 <- grep(stringj10, names(gene), value = TRUE)  
  
colj11 <- grep(stringj11, names(gene), value = TRUE)  
colj12 <- grep(stringj12, names(gene), value = TRUE)  
colj13 <- grep(stringj13, names(gene), value = TRUE)  
# v gene  
subv1 <- as.matrix(gene[, colv1])   
subv2 <- as.matrix(gene[, colv2])   
subv3 <- as.matrix(gene[, colv3])   
subv4 <- as.matrix(gene[, colv4])   
subv5 <- as.matrix(gene[, colv5])   
subv6 <- as.matrix(gene[, colv6])   
subv7 <- as.matrix(gene[, colv7])   
subv8 <- as.matrix(gene[, colv8])   
subv9 <- as.matrix(gene[, colv9])   
subv10 <- as.matrix(gene[, colv10])   
  
subv11 <- as.matrix(gene[, colv11])   
subv12 <- as.matrix(gene[, colv12])   
subv13 <- as.matrix(gene[, colv13])   
subv14 <- as.matrix(gene[, colv14])   
subv15 <- as.matrix(gene[, colv15])   
subv16 <- as.matrix(gene[, colv16])   
subv17 <- as.matrix(gene[, colv17])   
subv18 <- as.matrix(gene[, colv18])   
subv19 <- as.matrix(gene[, colv19])   
subv20 <- as.matrix(gene[, colv20])   
  
subv21 <- as.matrix(gene[, colv21])   
subv22 <- as.matrix(gene[, colv22])   
subv23 <- as.matrix(gene[, colv23])   
subv24 <- as.matrix(gene[, colv24])   
subv25 <- as.matrix(gene[, colv25])   
subv26 <- as.matrix(gene[, colv26])   
subv27 <- as.matrix(gene[, colv27])   
subv28 <- as.matrix(gene[, colv28])   
subv29 <- as.matrix(gene[, colv29])   
subv30 <- as.matrix(gene[, colv30])   
  
subv31 <- as.matrix(gene[, colv31])   
subv32 <- as.matrix(gene[, colv32])   
subv33 <- as.matrix(gene[, colv33])   
subv34 <- as.matrix(gene[, colv34])   
subv35 <- as.matrix(gene[, colv35])   
subv36 <- as.matrix(gene[, colv36])   
subv37 <- as.matrix(gene[, colv37])   
subv38 <- as.matrix(gene[, colv38])   
subv39 <- as.matrix(gene[, colv39])   
subv40 <- as.matrix(gene[, colv40])   
  
subv41 <- as.matrix(gene[, colv41])   
subv42 <- as.matrix(gene[, colv42])   
subv43 <- as.matrix(gene[, colv43])   
subv44 <- as.matrix(gene[, colv44])   
subv45 <- as.matrix(gene[, colv45])   
subv46 <- as.matrix(gene[, colv46])   
subv47 <- as.matrix(gene[, colv47])   
subv48 <- as.matrix(gene[, colv48])   
subv49 <- as.matrix(gene[, colv49])   
subv50 <- as.matrix(gene[, colv50])   
  
# j gene  
subj1 <- as.matrix(gene[, colj1])   
subj2 <- as.matrix(gene[, colj2])   
subj3 <- as.matrix(gene[, colj3])   
subj4 <- as.matrix(gene[, colj4])   
subj5 <- as.matrix(gene[, colj5])   
subj6 <- as.matrix(gene[, colj6])   
subj7 <- as.matrix(gene[, colj7])   
subj8 <- as.matrix(gene[, colj8])   
subj9 <- as.matrix(gene[, colj9])   
subj10 <- as.matrix(gene[, colj10])   
  
subj11 <- as.matrix(gene[, colj11])   
subj12 <- as.matrix(gene[, colj12])   
subj13 <- as.matrix(gene[, colj13])  
set.na1 <- c(22)  
set.na2 <- c(94:109)  
Y <- gene$Y  
Y[set.na1] <- "disease"  
Y[set.na2] <- "healthy"  
one.vec <- rep(1,length(Y))  
Y.d <- rep(0, length(Y))  
Y.d[which(Y == "disease")] = 1  
obj.s <- SKAT\_Null\_Model(Y.d ~ 1, out\_type = "D")  
  
# vectors  
pvalue.vec <- rep(0,50)  
pval <- rep(0,13)  
# v gene  
for (i in 1:50) {  
 sub <- get(paste0("subv", i))   
 out <- SKATBinary(sub, obj.s, kernel = "linear.weighted")  
 p <- out$p.value  
 pvalue.vec[i] <- p  
}  
result <- data.frame(cbind(c(1:50), pvalue.vec))  
colnames(result) <- c("vgene.idx", "pvalue")  
result  
for (i in 1:13) {  
 sub <- get(paste0("subj", i))   
 out <- SKATBinary(sub, obj.s, kernel = "linear.weighted")  
 p <- out$p.value  
 pval[i] <- p  
}  
jres <- data.frame(cbind(c(1:13), pval))  
colnames(jres) <- c("jgene.idx", "p-value")  
jres  
# v gene  
pv <- result$pvalue  
p.pv <- p.adjust(pv, method = p.adjust.methods, n = length(pv))  
pv.res <- data.frame(cbind(c(1:50), p.pv))  
pv.res  
# j gene  
pj <- jres$`p-value`  
p.pj <- p.adjust(pj, method = p.adjust.methods, n = length(pj))  
pj.res <- data.frame(cbind(c(1:13), p.pj))  
pj.res  
gene$Y <- Y  
# dataframe  
dfull <- gene[3:630]  
# v gene  
dfv16 <- gene[, colv16]  
dfv26 <- gene[, colv26]  
dfv27 <- gene[, colv27]   
# j gene  
dfj7 <- gene[, colj7]  
dfj9 <- gene[, colj9]  
dfj10 <- gene[, colj10]  
  
# pca res  
pcaFull <- prcomp(dfull, scale. = TRUE)  
# v gene  
pcav16 <- prcomp(dfv16, scale. = TRUE)  
pcav26 <- prcomp(dfv26, scale. = TRUE)  
pcav27 <- prcomp(dfv27, scale. = TRUE)   
# j gene  
pcaj7 <- prcomp(dfj7, scale. = TRUE)  
pcaj9 <- prcomp(dfj9, scale. = TRUE)  
pcaj10 <- prcomp(dfj10, scale. = TRUE)  
# plot  
autoplot(pcav16, data = gene, colour = 'Y')  
autoplot(pcav26, data = gene, colour = 'Y')  
autoplot(pcav27, data = gene, colour = 'Y')  
autoplot(pcaj7, data = gene, colour = 'Y')  
autoplot(pcaj9, data = gene, colour = 'Y')  
autoplot(pcaj10, data = gene, colour = 'Y')  
autoplot(pcaFull, data = gene, colour = 'Y')