Statistical Learning and Data Analysis 2021 - 52525

Lab 2 - Elections and RNA-sequencing

Abigail Gutman and Shahar Shalom

18/5/2021

# **1. Simulation Study:**

### Q1:

In this question we generate the first 10 coordinates of each µj vector j = 1, ..,3 :

ob <- c(rep(1,20),rep(2,30),rep(3,50))   
sampling <- NULL  
  
for (i in 1:10) {  
first\_10 <- rnorm(3,0,1) #mu  
sampling <- rbind(sampling,first\_10)   
colnames(sampling) <- c(1:3)  
}

Here we creates an auxiliary functions that will help us in the following questions:

calculate.accuracy <- function(data, cluster\_data){  
 tab <- table(data,cluster\_data)  
 s <- 0  
 b <- NULL  
 for(i in 1:3){  
 w<- tab[,i]  
 m <- max(w)  
 wm <- which.max(w)  
 while(wm %in% b){  
 w <- w[-wm,]  
 m <- max(w)  
 wm <- colnames(w)[which.max(w)]  
 }  
 s<- s+m  
 }  
 s <- s/100  
 return(s)  
 }  
  
multi\_fun <- function(s){  
 start.time <- Sys.time()  
 model <- kmeans(s,3,1,algorithm = "Lloyd")  
 end.time <- Sys.time()  
 time.taken <- end.time - start.time  
 accurancy <- model$cluster  
 accurancy <- calculate.accuracy(accurancy,ob)  
 return(list(accurancy,time.taken))  
}

### Q2:

Write a function that outputs a simulated dataset of dimension (100 × p)

create\_data <- function(sampling, p, sigma\_e){  
 x\_data <- NULL  
 if(p >10){  
 mu\_num <- rep(0,(p - 10))  
 mu <- cbind(mu\_num, mu\_num, mu\_num)  
 mu <- rbind(sampling,mu)  
 mu <- as.data.frame(mu)   
 } else {  
 mu <- as.data.frame(sampling)  
 }  
 d <- diag(sqrt(rep(sigma\_e,p)))  
 dat\_a = mvrnorm(20,mu = mu[,1],Sigma= d)  
 dat\_b = mvrnorm(30,mu = mu[,2],Sigma= d)  
 dat\_c = mvrnorm(50,mu = mu[,3],Sigma= d)  
 x\_data = rbind(dat\_a, dat\_b,dat\_c)  
 return(x\_data)  
}

### Q3

Choose 4 levels of σ^2 and use p = 10, 20, 50. choosing sigma : 1,2,6,9

sigma\_list <- c(1,2,6,9)

### Q4

For each combination of σ^2 and p we would like to generate multiple datasets (say B = 50). We will write the solution for this question in the code of question 5 below.

### Q5

For each data-set, run K-means once on the p dimensional data, and save accuracy and run-time.

The following function performs fifty simulations for each combination of sigma and p.  In which after the data is generated it calculates for it the accuracy of the K-MEAN algorithm and its runtime and keeps the information relevant to us for further reference only.

p <- c(10,20,50)  
simulation <- function(sampling,p,sigma\_list,B){  
 s\_total <- NULL  
 for (i in p) {  
 for (j in sigma\_list) {  
 s <- NULL  
 s <- replicate(n = B,sampling,simplify = F)  
 s <- lapply(s,create\_data,p = i,sigma\_e = j)  
 s <- lapply(s, multi\_fun)  
 s\_total <- cbind(s\_total,s)  
 }  
 }  
 return(s\_total)  
}  
  
gene <- simulation(sampling,p,sigma\_list,50)  
l <- c(paste0("p = ",10, ",sigma = ",1),paste0("p = ",10, ",sigma = ",2),  
 paste0("p = ",10, ",sigma = ",6),paste0("p = ",10, ",sigma = ",9),  
 paste0("p = ",20, ",sigma = ",1),paste0("p = ",20, ",sigma = ",2),  
 paste0("p = ",20, ",sigma = ",6),paste0("p = ",20, ",sigma = ",9),  
 paste0("p = ",50, ",sigma = ",1),paste0("p = ",50, ",sigma = ",2),  
 paste0("p = ",50, ",sigma = ",6),paste0("p = ",50, ",sigma = ",9))  
  
k\_sh\_p <- c(rep(10,4),rep(20,4),rep(50,4))  
k\_col\_sig <- rep(sigma\_list,3)

### Q6

Compute the average accuracy and the standard-error for each (p, σ2). Display these in a figure and a table.

temp <- NULL  
temp2 <- NULL  
  
for(i in 1:12){  
 df <- gene[,i]  
 df <- rbindlist(lapply(df, as.data.frame.list))  
 colnames(df) <- c("accurancy", "run.time")  
 temp2 <- rbind(temp2,list(df))  
 df <- round(c(avg = mean(df$accurancy),sd\_accurancy = sd(df$accurancy)/sqrt(50)),3)  
 temp <- rbind(temp,df)  
}

## Column 1 ['X0.74'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.7'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.59'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.67'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.75'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.65'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.59'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.6'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.66'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.5'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

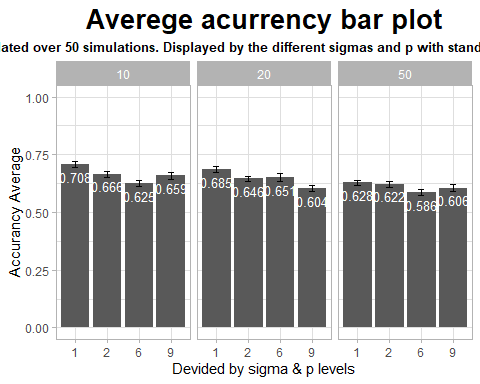
## Column 1 ['X0.75'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

## Column 1 ['X0.55'] of item 2 is missing in item 1. Use fill=TRUE to fill with NA (NULL for list columns), or use.names=FALSE to ignore column names. use.names='check' (default from v1.12.2) emits this message and proceeds as if use.names=FALSE for backwards compatibility. See news item 5 in v1.12.2 for options to control this message.

names(temp2) <- l  
row.names(temp) <- l

temp <- as.data.frame(temp)  
temp <- cbind(temp,k\_col\_sig,k\_sh\_p)

temp <- as.data.frame(temp)  
ggplot(temp,aes( y = avg,x=factor(k\_col\_sig))) +  
 geom\_bar(stat="identity",position="dodge") +  
 facet\_grid(~factor(k\_sh\_p))+  
 geom\_errorbar(data = temp,aes(ymin=avg-sd\_accurancy,  
 ymax=avg + sd\_accurancy), width=.2,position=position\_dodge(.9)) +   
 geom\_text(aes(label=avg), vjust=1.6, color="white", size=3.5)+  
 facet\_grid(~factor(k\_sh\_p))+ theme\_light() +  
 xlab("Devided by sigma & p levels")+  
 ylab("Accurancy Average") +  
 ggtitle(label = "Averege acurrency bar plot", subtitle = "Caculated over 50 simulations. Displayed by the different sigmas and p with standard error bar") +  
 theme(plot.title = element\_text(hjust = 0.5,size = 20, face = "bold"),  
 plot.subtitle = element\_text(hjust = 0.5,size = 10, face = "bold")) +  
 theme(legend.title = element\_text(color = "black", size = 10),  
 legend.text = element\_text(color = "black")) +  
 labs(shape="Vectors Variables \n Number", colour="SE of \n the Vectors") + ylim(c(0,1))



The graph shows the average accuracy for fifty simulations for each combination of sigma and p compared to perfect accuracy. For each column you can see the standard deviation of the accuracy for the simulations. It can be understood from the low standard error that the estimate for the average accuracy compliance by the number of simulations is relatively reliable. The graph shows the difference in the ability to achieve high accuracy in classification by KMEAN as the standard deviation of the distribution from it is sampled, and the more variables there are.

The data shown in the table:

kable(temp[,1:2]) %>% kable\_styling()

avg

sd\_accurancy

p = 10,sigma = 1

0.708

0.013

p = 10,sigma = 2

0.666

0.013

p = 10,sigma = 6

0.625

0.013

p = 10,sigma = 9

0.659

0.016

p = 20,sigma = 1

0.685

0.013

p = 20,sigma = 2

0.646

0.012

p = 20,sigma = 6

0.651

0.016

p = 20,sigma = 9

0.604

0.014

p = 50,sigma = 1

0.628

0.012

p = 50,sigma = 2

0.622

0.012

p = 50,sigma = 6

0.586

0.013

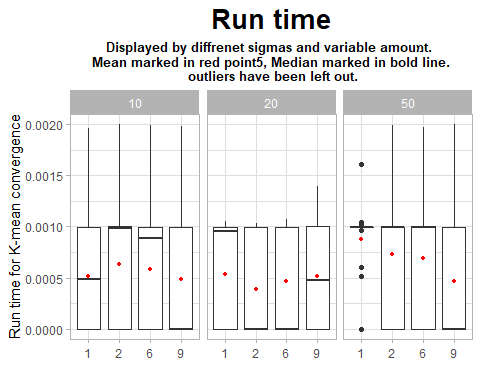
p = 50,sigma = 9

0.606

0.016

`

temp2 <-bind\_rows(temp2, .id = "column\_label")  
k\_sh\_p <- c(rep(10,4\*50),rep(20,4\*50),rep(50,4\*50))  
k\_col\_sig <- rep(rep(sigma\_list, each = 50),3)  
temp2 <- cbind(temp2,k\_col\_sig,k\_sh\_p)  
  
ggplot(temp2, aes(y = run.time, x =factor(k\_col\_sig))) + facet\_grid(~factor(k\_sh\_p))+  
 geom\_boxplot()+  
 stat\_summary(geom = "point",  
 fun = "mean", col = "red",size = 1, fill = "red") +  
 theme\_light() + ylim(c(0,0.002))+  
 xlab("")+ ylab("Run time for K-mean convergence") +  
 ggtitle(label = "Run time", subtitle = "Displayed by diffrenet sigmas and variable amouמt. \n Mean marked in red point5, Median marked in bold line. \n outliers have been left out.")+  
 theme(plot.title = element\_text(hjust = 0.5,size = 20, face = "bold"),  
 plot.subtitle = element\_text(hjust = 0.5,size = 10, face = "bold")) +  
 theme(axis.text.x = element\_text(angle = 00, vjust = 0.5))



### Q8

We have seen that the SE we obtained on the average accuracy of the simulations allows us to treat the average accuracy as reliable. On this basis it can be seen that the larger the standard deviation of the distribution from which we have drawn in advance the greater we encounter the lower ability of KMEAN to perform a correct classification, regardless of the length of the vector and the amount of variables in it. In addition it can be seen that the more variables we have in each vector, here too the algorithm has difficulty performing a correct classification, regardless of the standard deviation. It can also be seen in the graph of the runtime required for the algorithm convergence that there is an increase in time for large standard deviations and multivariate vectors, except for outliers, it should be noted that in this case these are relatively low dimensions, but for Big Data the effect may be significant.

# **2. Comparing demographic and election data:**

#load the data frames  
knesset\_df <- read.csv("C:/Users/Shahar/Dropbox/zugi/lemida/knesset\_24.csv", encoding = "UTF-8")  
CBS\_df <- read\_excel("C:/Users/Shahar/Dropbox/zugi/lemida/t01.xls")

### Q1

In this question we sampled 20 different locations that we would like to investigate. We used the socio-economic data from 2013 from the ISB website for the socio-economic data frame and the 24 election results data frame.

#first, we merge the data frames to find the matching locations in both.   
merged\_df <- left\_join(CBS\_df, knesset\_df, by = "סמל.ישוב")  
  
#add "שם.ישוב" coloumn to the CBS data frame   
names\_df <- as.data.frame(knesset\_df[,2:3])  
CBS\_df <- left\_join(CBS\_df, names\_df)  
  
#semple 20 locations   
set.seed(20)  
Semp\_20 <- sample(merged\_df$"סמל.ישוב" , 20)  
  
# sum all the votes for each location   
#calculate the persents of votes for each party in every location   
knesset\_df\_norm <- knesset\_df %>%rowwise %>% mutate(total\_votes = sum(c\_across( 8:46))) %>% ungroup() %>% mutate(round(across(8:46, ~ . / total\_votes),3))  
  
#filter each data frame for the sempled values.  
knesset\_semp20 <- filter(knesset\_df\_norm, knesset\_df\_norm$"סמל.ישוב" %in% Semp\_20)  
CBS\_semp20 <- filter(CBS\_df, CBS\_df$"סמל.ישוב" %in% Semp\_20)

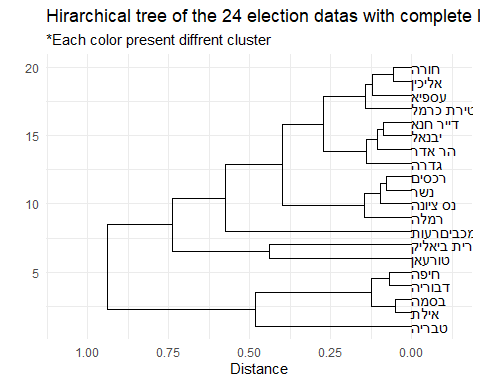
### Q2

In this question we construct a hierarchical tree for the elections data.

for this calculation we decided to take only the scaled votes results from the data frame, the reason is that we want to investigate wich cities behave in a similer way and to see if the algorithem will collapse this cites to similer clusters.

We decided to define the distances between two cities by the complete linked algorithem. we want to see in the tree the conection between clusters of cities that have the maximal distance between them.

knesst\_dist <- knesset\_semp20[,c(8:46)] %>% dist %>% hclust(method = "complete")  
knesst\_dend <- as.dendrogram(knesst\_dist)  
labels(knesst\_dend) <- as.character(knesset\_semp20$'שם.ישוב')  
  
Knesst\_plot <- ggplot(knesst\_dend %>%   
 set('branches\_lwd', 0.7) %>%  
 set('labels\_cex', 0.8),  
 horiz = TRUE, theme = theme\_minimal()) +   
 ggtitle("Hirarchical tree of the 24 election datas with complete linked algorithm", "\*Each color present diffrent cluster") + ylab("Distance") + xlab(" ")  
  
Knesst\_plot

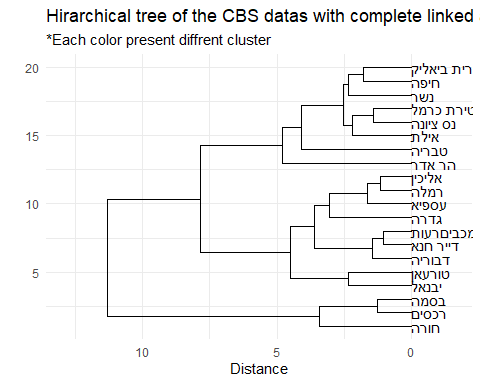


### Q3

In this question we construct a hierarchical tree for the socio-economic 2013 data frame. we used only the values which dascribe the diffrenete in socio-economic status between the cities.

Again we decided to use the complete linked algorithm.

CBS\_dist <- CBS\_semp20[,6:17] %>% scale %>% dist %>% hclust(method = "complete")  
CBS\_dend <- as.dendrogram(CBS\_dist)  
labels(CBS\_dend) <- as.character(CBS\_semp20$'שם.ישוב')  
  
CBS\_plot <- ggplot(CBS\_dend %>%   
 set('branches\_lwd', 0.7) %>%   
 set('labels\_cex', 0.8), horiz = TRUE, theme = theme\_minimal()) +   
 ggtitle("Hirarchical tree of the CBS datas with complete linked algorithm", "\*Each color present diffrent cluster")+ ylab("Distance") + xlab(" ")  
  
CBS\_plot

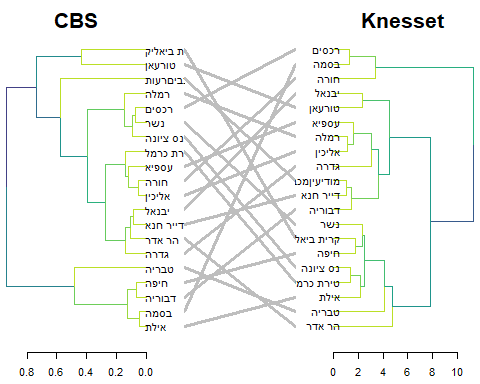


### Q4

In this question we tried to visualized the diffrents between the two dendograms.

Even before we see the visual comparison we tend to think that while there are some cities that have received a similar characterization in the two dendograms, yet most cities get a different height and sometimes completely opposite from the height they received in the other dendogram.

dl <- dendlist(highlight\_branches\_col(knesst\_dend), highlight\_branches\_col(CBS\_dend))  
  
tanglegram(dl, sort = TRUE, highlight\_distinct\_edges = FALSE, highlight\_branches\_lwd = FALSE, main\_left = "CBS", main\_right = "Knesset", dLeaf\_left = 0.01 , dLeaf\_right = -0.6)



After examining the visual comparison between the trees we understand that our hypothesis is Almost right< most of the cities were given a different location in each tree and there appears to be no connection between the two trees

However, this does not mean that one of the dendograms is incorrect, the reason for the large differences between the two is a different effect of socioeconomic status on the population in each city, in addition we should remember that the population in Israel is very far from being a homogeneous population.

### Q5

In this question we were asked to choos score wich calculate the similarity between the two trees. We choose the bakers gamma score, it is defined as the rank correlation between the stages at which pairs of objects combine in each of the two trees.

dend <- knesst\_dend  
dend1 <- CBS\_dend  
  
cor <- cor\_bakers\_gamma(dend,dend1)  
print(paste("The correlation between the two dendograms is: ", round(cor,3)))

## [1] "The correlation between the two dendograms is: -0.042"

In this case we see that we got small correlation between the two dendograms. it means that thr two hirarchical trees almost dosent have conection between them. the conection is a small nagative conection.

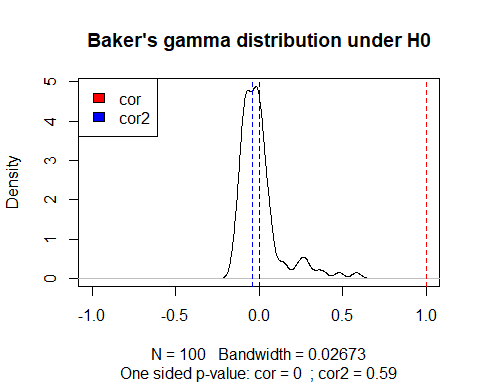
### Q6

In this question we were asked to find a background distribution for this score. To do this we randomly permute the labels of the knesset tree while keeping the labels of thr CBS tree fixed. Our null hypothesis is that the correlation between the two trees is that the dendogram are fully correlated to herself in all of the permutations that we create.

the\_cor <- cor\_bakers\_gamma(dend,dend) #we see that the correlation between the dend to herself is 1  
the\_cor2 <- cor\_bakers\_gamma(dend,dend1) #the correlation between the two dendograms that we created  
  
R <- 100  
cor\_bakers\_gamma\_results <- numeric(R)  
dend\_mixed <- dend  
for(i in 1:R) { #compute the corr between the original dendograme to the permutation dendogram   
 dend\_mixed <- sample.dendrogram(dend\_mixed, replace = FALSE)  
 cor\_bakers\_gamma\_results[i] <- cor\_bakers\_gamma(dend, dend\_mixed)  
}  
  
plot(density(cor\_bakers\_gamma\_results),  
 main = "Baker's gamma distribution under H0",  
 xlim = c(-1,1))  
abline(v = 0, lty = 2)  
abline(v = the\_cor, lty = 2, col = 2)  
abline(v = the\_cor2, lty = 2, col = 4)  
legend("topleft", legend = c("cor", "cor2"), fill = c(2,4))  
round(sum(the\_cor2 < cor\_bakers\_gamma\_results)/ R, 4)

## [1] 0.59

title(sub = paste("One sided p-value:",  
 "cor =", round(sum(the\_cor < cor\_bakers\_gamma\_results)/ R, 4),  
 " ; cor2 =", round(sum(the\_cor2 < cor\_bakers\_gamma\_results)/ R, 4)))



In this graph we see the background distribution for the correlation score. we see that cor = 1, represent the correlation between the knesset dendogram to herself, and test the H0 that all of the premuted dendograms are fully correlated to the original dendogram. we can see in the graph that this isent what happend. Cor2 = -0.04, we check the P-value of this result and see that Pv >0.05, theres for we reject H0 and say that the premuted dendogram is not fully correalted to the original dendogram.

We have learned that althogh we thoght that permutation dendogram will give us similear trees from the same data frames it doesn’t happend and we get total new dendograme in each permutation.

# **3. Exploratory analysis of RNA seq data:**

# k\_means\_shiny <- function(k){  
# data\_prep <- m  
# m\_new <- as.data.frame(data\_prep[sample(1:53,k),])  
# clus <- apply(data\_prep,1 ,FUN = c\_fun, m=m\_new)  
# m\_old <- as.data.frame(data\_prep[sample(1:53,k),])  
# data\_prep <- cbind(data\_prep, clus)  
# iter <- 1  
# while(any(abs(m\_new - m\_old)) > 0.1 & iter < 1){  
# data\_prep <- as.data.frame(data\_prep)  
# m\_old <- m\_new  
# m\_new <- aggregate(. ~ clus, data\_prep, FUN = mean)  
# m\_new <- m\_new[,-1]  
# data\_prep$clus<- data\_prep[,-1] %>% apply(1,FUN = c\_fun,m = m\_new)  
# print(m\_new)  
# print(m\_old)  
# iter <- iter +1  
#   
# }  
# return(data\_prep)  
# }  
#   
# c\_fun <- function(d,m){  
# p <- NULL  
# for (i in 1:length(m[,1])) {  
# x <- rbind(d,m[i,])  
# di <- dist(x, method = 'euclidean')  
# p <- cbind(p,di)  
# }  
# p <- which.min(p)  
# return(p)  
# }  
#   
# g <- k\_means\_shiny(6)

# #med\_dat <- read.delim('C:/Users/abig4/OneDrive/Documents/GitHub/Statistical-Learning\_Lab\_2/gtex\_Kmeans/gtex.gct', skip = 2 ,row.names=c(1) , header = TRUE)  
# med\_dat <- read.delim("C:/Users/Shahar/Documents/GitHub/Statistical-Learning\_Lab\_2/gtex\_Kmeans/gtex.gct", skip = 2 ,row.names=c(1) , header = TRUE)  
# gen\_names <- med\_dat[, 1]  
# med\_dat <- med\_dat[,-1]  
# ti\_name <- colnames(med\_dat)  
# med\_dat <- transpose(med\_dat)  
# med\_dat <-med\_dat[,colMeans(med\_dat) > 0]  
# v <- apply(med\_dat,2, var)  
# med\_dat <-med\_dat[, v != 0]  
# #med\_dat <- as.data.frame(scale(med\_dat,T,T))  
# m <- prcomp(med\_dat,scale. = T,center = T)  
# sv <-m$sdev^2/sum(m$sdev^2)  
# sv <- cumsum(sv)  
# sv <- length(sv[sv < 0.9])  
# m <- m$x[,1:sv]