Assignment 3:

Problem 1:

```
#Step 1: Load packages you will use in serer script, including shiny package.
library(tm)
library(wordcloud)
library(quantmod)
library(RMySQL)
library(hunspell)
library(e1071)

# Result preparation - Method 1: outside server part.
# Use the same text data from Lecture 9.
### You can choose to do your analysis outside "server" part and use the result in the server part

# Load libraries


#````````````````````````````````````````````````````
# Problem 1
#````````````````````````````````````````````````````
# load driver for SQL
drv <- dbDriver("MySQL")

con <- dbConnect(drv, user = "gang", password = "gang",
                 host = "localhost", dbname = "fe513_twitter")

#checking table
```

```
dbListTables(con)

# Step 1 - creating a data frame for unique user_ids
SQLtext1 <- "SELECT DISTINCT(user_id) FROM twitter_message"

unique_userid <- dbSendQuery(con, SQLtext1)

unique_users <- dbFetch(unique_userid)
unique_users

# Step 2 - creating a data frame for random user_ids
random_users <- data.frame(unique_users[sample(nrow(unique_users), 3, replace = FALSE, prob = NULL),])

random_users
names(random_users)[1] <- paste("user_id")
random_users

random_users[1,1]
random_users[2,1]
random_users[3,1]

dbWriteTable(con, "random_users", random_users, overwrite = TRUE)

dbReadTable(con, "random_users")

# step 3 - create query to extract all of those user tweets
SQLtext2 <- "SELECT * FROM twitter_message where user_id IN (SELECT user_id FROM random_users)"
```

```
r_user_query <- dbSendQuery(con, SQLtext2)

r_user_tweets <- dbFetch(r_user_query)

# step 4 - combine tweets from the same user into one variable. As a result, you will have 3 variables(long strings) in R.
user1 <- r_user_tweets[which(r_user_tweets$user_id == random_users[1,1]), ]
user2 <- r_user_tweets[which(r_user_tweets$user_id == random_users[2,1]), ]
user3 <- r_user_tweets[which(r_user_tweets$user_id == random_users[3,1]), ]



combined_user1_tweets <- paste(user1$tweets, collapse = " ")
combined_user2_tweets <- paste(user2$tweets, collapse = " ")
combined_user3_tweets <- paste(user3$tweets, collapse = " ")

# step 5 - Creating list of all non-english words from each set and removing non-ASCII characters
combined_user1_tweets <- iconv(combined_user1_tweets, "latin1", "ASCII", sub="")
combined_user2_tweets <- iconv(combined_user2_tweets, "latin1", "ASCII", sub="")
combined_user3_tweets <- iconv(combined_user3_tweets, "latin1", "ASCII", sub="")

u1_nonen <- unlist(hunspell_find(combined_user1_tweets))
u2_nonen <- unlist(hunspell_find(combined_user2_tweets))
u3_nonen <- unlist(hunspell_find(combined_user3_tweets))

nonenglishset <- c(u1_nonen, u2_nonen, u3_nonen)

# step 5.1 - remove all punctuations
ut <- Corpus(VectorSource(c(combined_user1_tweets, combined_user2_tweets, combined_user3_tweets)))
```

```
utnopunc <- tm_map(ut, removePunctuation)
uttolower <- tm_map(utnopunc, content_transformer(tolower))

# step 5.2 - remove all numbers
utnonum <- tm_map(uttolower, removeNumbers)

# step 5.3 - remove stop words
utnostop <- tm_map(utnonum, removeWords, stopwords("en"))

# step 5.4 - remove non-english words
utnoeng <- tm_map(utnostop, removeWords, nonenglishset)

# step 6 - Create the term-document matrxi and plot the wordclouds
TM1 <- TermDocumentMatrix(utnoeng)
M1 <- as.matrix(TM1)

final <- data.frame(word = rownames(M1),freq1=M1[,1], freq2 = M1[,2], freq3 = M1[,3])
par(mfrow = c(1, 3))
wordcloud(words = final$word, freq = final$freq1, min.freq = 2, colors=brewer.pal(8, "Dark2"))
wordcloud(words = final$word, freq = final$freq2, min.freq = 2, colors=brewer.pal(8, "Dark2"))
wordcloud(words = final$word, freq = final$freq3, min.freq = 2, colors=brewer.pal(8, "Dark2"))
```

Problem 2:

ps2_nonen <- unlist(hunspell_find(r_user_tweets$tweets))

step 4 - make a label(index) vector for those tweets/documents.

The label is the user id you have. Thus, all documents posted

by one user should have the same label, and you will have 3 unique

```
# labels in total.

cleaned_user1_tweets <- iconv(r_user_tweets$tweets, "latin1", "ASCII", sub="")

PS2_cleaned_table <- cbind(r_user_tweets$user_id, cleaned_user1_tweets)

PS2 <- as.data.frame(PS2_cleaned_table)
print.sum <- summary(PS2)

# step 5.1 - remove all punctuations
utps2 <- Corpus(VectorSource(c(cleaned_user1_tweets)))
utnopuncps2 <- tm_map(utps2, removePunctuation)
uttolowerps2 <- tm_map(utnopuncps2, content_transformer(tolower))

# step 5.2 - remove all numbers
utnonumps2 <- tm_map(uttolowerps2, removeNumbers)

# step 5.3 - remove stop words
utnostopps2 <- tm_map(uttolowerps2, removeWords, stopwords("en"))

# step 5.4 - remove non-english words
utnoengps2 <- tm_map(utnostopps2, removeWords, ps2_nonen)

# step 6 - Create the term-document matrix
TMPS2 <- DocumentTermMatrix(utnoengps2)
M2 <- as.matrix(TMPS2)
ncol(M2)

# DFM2 <- as.data.frame((M2))
```

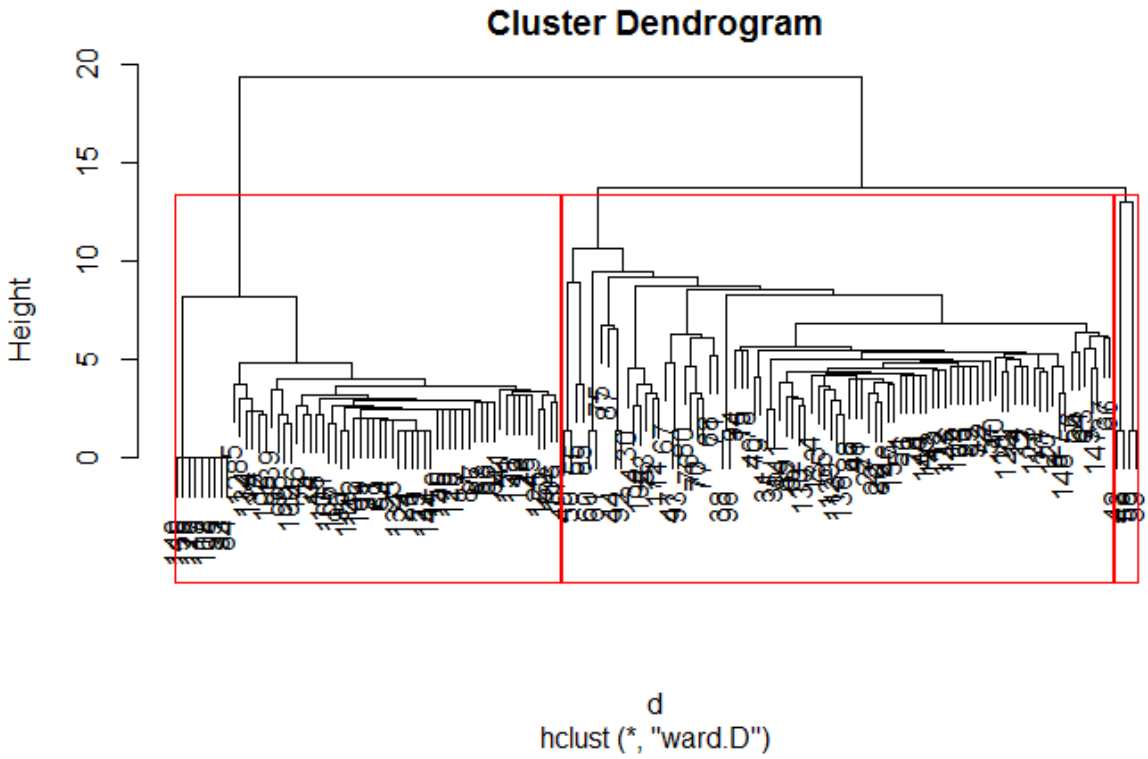
```
# DFM2[,1][DFM2[,1] == "1"] <- random_users[1,1]

# DFM2[,2][DFM2[,2] == "1"] <- random_users[2,1]

# DFM2[,3][DFM2[,3] == "1"] <- random_users[3,1]


# Step 7 - run kmeans clustering

kmRes <- kmeans(M2, 3, nstart = 20)


# Step 8 - use table() function in R showing the difference between cluster results and user id label.

ktable <- table(res =  kmRes$cluster, real = PS2$V1)

  real
res 20659892 241205643 39221596
  1       84         42       20
  2        0          4        0
  3        0          2        0


#returns cluster label

cmRes <- cmeans(M2, centers = 3, iter.max = 100)

head(cmRes$cluster)

1 2 3 4 5 6
2 2 2 2 2 2


#return membership (the probability of one data point belongs to one group. )

#the clutser label is based on max(membership)

head(cmRes$membership)

          1         2         3
1 0.3333251 0.3333495 0.3333254
2 0.3333333 0.3333334 0.3333333
3 0.3333321 0.3333358 0.3333321
4 0.3333330 0.3333339 0.3333330
5 0.3333333 0.3333335 0.3333333
6 0.3333329 0.3333342 0.3333329


# Step 7 - run hierarchical cluster

d <- dist(M2, method = "euclidean")

HclustResult <- hclust(d, method="ward.D")
```

plot(HclustResult)

groups <- cutree(HclustResult, k=3) # cut tree into n clusters

draw dendogram with red borders around the n clusters

rect.hclust(HclustResult, k=3, border="red")



Step 8 - use table() function in R showing the difference between cluster results and user id label.

htable <- table(res = groups, real = PS2$V1)

```
   real
res 20659892 241205643 39221596
  1       53         6        2
  2       31        38       18
  3        0         4        0
```

Problem 3 Screenshots and Code:

~/Stevens/FE513/cocoApp1/cocoApp - Shiny

http://127.0.0.1:3686 | Open in Browser | Publish

FE513 (Name: Assignment 3 App)

wordcloud color?

Wordcloud min frequency

2

Wordcloud min frequency

1

Wordcloud min frequency

2

The widget on the side panel controls the word clouds. This page loads slowly due to the processing of words in the database.

E1.SQL and Names | E1.Wordcloud | E2.Summary | E2.Results | E2.Compare

```
SELECT DISTINCT(user_id) FROM twitter_message

SELECT * FROM twitter_message where user_id IN (SELECT user_id FROM random_users)

[1] 18949452 102285890 592001965
```

~/Stevens/FE513/cocoApp1/cocoApp - Shiny

http://127.0.0.1:3686 | Open in Browser | Publish

FE513 (Name: Assignment 3 App)

wordcloud color?

Wordcloud min frequency

2

Wordcloud min frequency

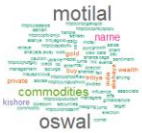
1

Wordcloud min frequency

2

The widget on the side panel controls the word clouds. This page loads slowly due to the processing of words in the database.

E1.SQL and Names | E1.Wordcloud | E2.Summary | E2.Results | E2.Compare



~/Stevens/FE513/cocoApp1/cocoApp - Shiny

http://127.0.0.1:3686 | Open in Browser | Publish

FE513 (Name: Assignment 3 App)

wordcloud color?

Wordcloud min frequency: 2

Wordcloud min frequency: 1

Wordcloud min frequency: 2

The widget on the side panel controls the word clouds. This page loads slowly due to the processing of words in the database.

E1.SQL and Names | E1.Wordcloud | E2.Summary | E2.Results | E2.Compare

[1] "102285890: 10 "

[1] "18949452 :371 "

[1] "592001965:119 "

V1	cleaned_user1_tweets
18949452	Asian economies now need more debt to generate growth: http://t.co/CToQpYXSH0 http://t.co/de6LhZVOLF
18949452	A Swift way to curb Putin's ambitions http://t.co/RNzH1H5tcx http://t.co/feQFrEPtWP
18949452	Ouch. "Google has helped give the world attention deficit disorder while destroying privacy," writes @LukeJohnsonRCP http://t.co/6NqtXdWoOJ
18949452	Free to read: What the Google privacy ruling means for the internet? Can it even be enforced? http://t.co/f3OLsjBf1F
18949452	Bob Diamond's Africa vehicle raises money for second time in less than 6 months: http://t.co/Hir7snASDK
18949452	Can Asia overcome its addiction to debt? http://t.co/2o4625O4yj http://t.co/X7XXvvbCqp
18949452	Sneaky rhetoric: The art of saying something without appearing to say it http://t.co/yV0KdnSSRj
592001965	Phytoceramides supplements have been called your "secret to cheat your age" by Dr. Oz http://t.co/34sX9jeQje --> https://t.co/VxNagGP1ku
592001965	$205,000 - Miami, FL Condo For Sale - 133 NE 2nd Ave --> http://t.co/kDgCNKV9dm
592001965	$390,000 - Aventura, FL Condo For Sale - 3500 MYSTIC POINTE DR --> http://t.co/5yeajpRlhJ
592001965	Body Wraps --> https://t.co/oubLfF8xY2
592001965	find best electric percolators, choose the best electric perclotors --> http://t.co/QMTOQwFI9U
592001965	Vemma - The Real Truth --> http://t.co/M5IUDFyNs5
592001965	Professional social media services buy facebook likes --> http://t.co/OUq4YoV7w6
592001965	Check out this exclusive review of the Article Factory Pro software by Josh Zamora --> https://t.co/PeSMit8bxz
18949452	Turkish mine death toll at 157, says mayor http://t.co/09fmTLDeAS
18949452	UN Syria envoy Brahimi to step down http://t.co/LCHQDRyJ9q
18949452	Front page of the Financial Times US for Wednesday, May 14 http://t.co/8RPzDuQpnH
18949452	"For those who find the commercial world draining, I suggest the intellectual succour of the Stoics": http://t.co/bx3QEJK9Er
18949452	Saudi Arabia moves to ease Iran tensions http://t.co/Zmw0IO8ByH
18949452	US steel imports surge on shale boom http://t.co/hqQ918z5Ul
18949452	Goldmans Twitter banker joins hedge fund http://t.co/es13SxG5HS

~/Stevens/FE513/cocoApp1/cocoApp - Shiny

http://127.0.0.1:3686 | Open in Browser | Publish

FE513 (Name: Assignment 3 App)

wordcloud color?

Wordcloud min frequency: 2

Wordcloud min frequency: 1

Wordcloud min frequency: 2

The widget on the side panel controls the word clouds. This page loads slowly due to the processing of words in the database.

E1.SQL and Names | E1.Wordcloud | E2.Summary | E2.Results | E2.Compare

The information below is the size of the clusters.

[1] 494 1 5

The information below indicates variance between clusters, within clusters, and of clusters.

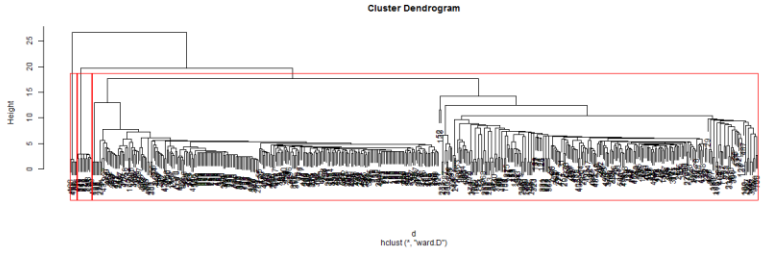
[1] 170.6798

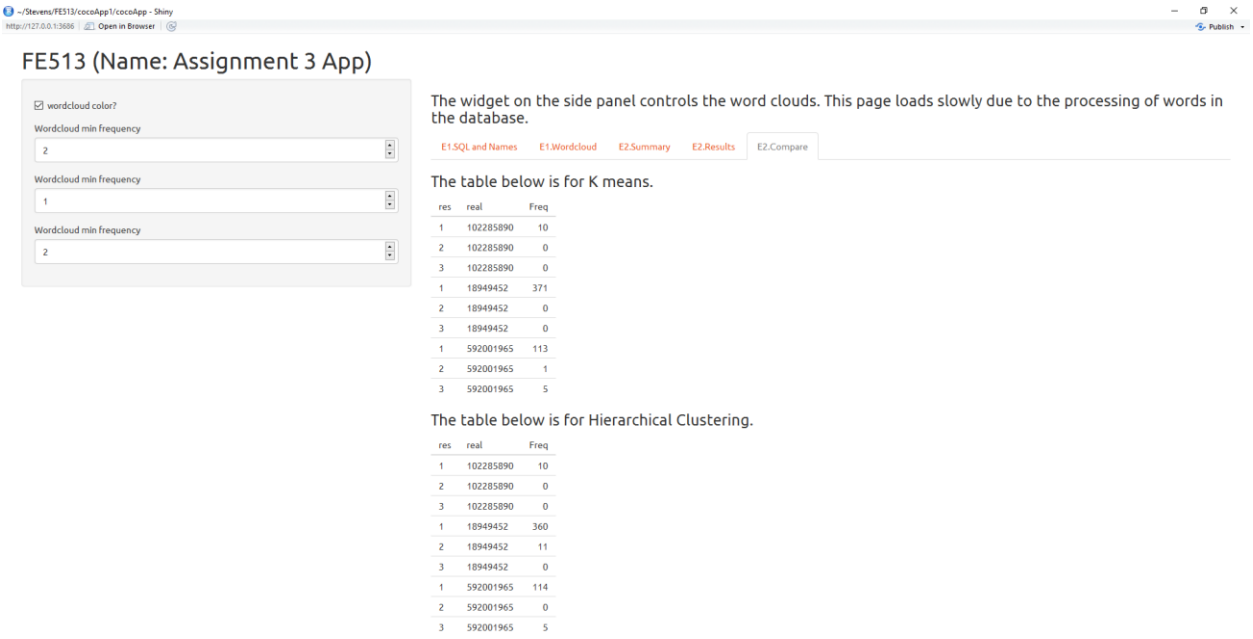
[1] 4381.666

[1] 4206.186 0.000 4.800

[1] 4210.986

Call: hclust(d = d, method = "ward.D") Cluster method : ward.D Distance : euclidean Number of objects: 500





Server File:

#Step 1: Load packages you will use in serer script, including shiny package.

library(tm)

library(wordcloud)

library(quantmod)

library(RMySQL)

library(hunspell)

library(e1071)

Result preparation - Method 1: outside server part.

Use the same text data from Lecture 9.

You can choose to do your analysis outside "server" part and use the result in the server part

Load libraries

```
#``````````````````````````````````````````````````````

# Problem 1

#``````````````````````````````````````````````````````

# load driver for SQL

drv <- dbDriver("MySQL")


con <- dbConnect(drv, user = "gang", password = "gang",

                 host = "localhost", dbname = "fe513_twitter")


#checking table


dbListTables(con)


# Step 1 - creating a data frame for unique user_ids

SQLtext1 <- "SELECT DISTINCT(user_id) FROM twitter_message"


unique_userid <- dbSendQuery(con, SQLtext1)


unique_users <- dbFetch(unique_userid)

unique_users


# Step 2 - creating a data frame for random user_ids

random_users <- data.frame(unique_users[sample(nrow(unique_users), 3, replace = FALSE, prob =
NULL),])


random_users

names(random_users)[1] <- paste("user_id")

random_users
```

```
random_users[1,1]

random_users[2,1]

random_users[3,1]


dbWriteTable(con, "random_users", random_users, overwrite = TRUE)


dbReadTable(con, "random_users")


# step 3 - create query to extract all of those user tweets

SQLtext2 <- "SELECT * FROM twitter_message where user_id IN (SELECT user_id FROM random_users)"


r_user_query <- dbSendQuery(con, SQLtext2)


r_user_tweets <- dbFetch(r_user_query)


# step 4 - combine tweets from the same user into one variable. As a result, you will have 3
variables(long strings) in R.

user1 <- r_user_tweets[which(r_user_tweets$user_id == random_users[1,1]), ]

user2 <- r_user_tweets[which(r_user_tweets$user_id == random_users[2,1]), ]

user3 <- r_user_tweets[which(r_user_tweets$user_id == random_users[3,1]), ]



combined_user1_tweets <- paste(user1$tweets, collapse = " ")

combined_user2_tweets <- paste(user2$tweets, collapse = " ")

combined_user3_tweets <- paste(user3$tweets, collapse = " ")


# step 5 - Creating list of all non-english words from each set and removing non-ASCII characters

combined_user1_tweets <- iconv(combined_user1_tweets, "latin1", "ASCII", sub="")

combined_user2_tweets <- iconv(combined_user2_tweets, "latin1", "ASCII", sub="")
```

```
combined_user3_tweets <- iconv(combined_user3_tweets, "latin1", "ASCII", sub="")

u1_nonen <- unlist(hunspell_find(combined_user1_tweets))
u2_nonen <- unlist(hunspell_find(combined_user2_tweets))
u3_nonen <- unlist(hunspell_find(combined_user3_tweets))

nonenglishset <- c(u1_nonen, u2_nonen, u3_nonen)

# step 5.1 - remove all punctuations
ut <- Corpus(VectorSource(c(combined_user1_tweets, combined_user2_tweets,
combined_user3_tweets)))
utnopunc <- tm_map(ut, removePunctuation)
uttolower <- tm_map(utnopunc, content_transformer(tolower))

# step 5.2 - remove all numbers
utnonum <- tm_map(uttolower, removeNumbers)

# step 5.3 - remove stop words
utnostop <- tm_map(utnonum, removeWords, stopwords("en"))

# step 5.4 - remove non-english words
utnoeng <- tm_map(utnostop, removeWords, nonenglishset)

# step 6 - Create the term-document matrxi and plot the wordclouds
TM1 <- TermDocumentMatrix(utnoeng)
M1 <- as.matrix(TM1)

final <- data.frame(word = rownames(M1),freq1=M1[,1], freq2 = M1[,2], freq3 = M1[,3])
par(mfrow = c(1, 3))
```

```
wordcloud(words = final$word, freq = final$freq1, min.freq = 2, colors=brewer.pal(8, "Dark2"))
wordcloud(words = final$word, freq = final$freq2, min.freq = 2, colors=brewer.pal(8, "Dark2"))
wordcloud(words = final$word, freq = final$freq3, min.freq = 2, colors=brewer.pal(8, "Dark2"))



#``````````````````````````````````````````````````````
# Problem 2
#``````````````````````````````````````````````````````

ps2_nonen <- unlist(hunspell_find(r_user_tweets$tweets))

# step 4 - make a label(index) vector for those tweets/documents.
# The label is the user id you have. Thus, all documents posted
# by one user should have the same label, and you will have 3 unique
# labels in total.

cleaned_user1_tweets <- iconv(r_user_tweets$tweets, "latin1", "ASCII", sub="")

PS2_cleaned_table <- cbind(r_user_tweets$user_id, cleaned_user1_tweets)

PS2 <- as.data.frame(PS2_cleaned_table)
print.sum <- summary(PS2)

# step 5.1 - remove all punctuations
utps2 <- Corpus(VectorSource(c(cleaned_user1_tweets)))
utnopuncps2 <- tm_map(utps2, removePunctuation)
uttolowerps2 <- tm_map(utnopuncps2, content_transformer(tolower))

# step 5.2 - remove all numbers
```

```
utnonumps2 <- tm_map(uttolowerps2, removeNumbers)

# step 5.3 - remove stop words
utnostopps2 <- tm_map(uttolowerps2, removeWords, stopwords("en"))

# step 5.4 - remove non-english words
utnoengps2 <- tm_map(utnostopps2, removeWords, ps2_nonen)

# step 6 - Create the term-document matrix
TMPS2 <- DocumentTermMatrix(utnoengps2)
M2 <- as.matrix(TMPS2)
ncol(M2)

# DFM2 <- as.data.frame((M2))
# DFM2[,1][DFM2[,1] == "1"] <- random_users[1,1]
# DFM2[,2][DFM2[,2] == "1"] <- random_users[2,1]
# DFM2[,3][DFM2[,3] == "1"] <- random_users[3,1]

# Step 7 - run kmeans clustering
kmRes <- kmeans(M2, 3, nstart = 20)

# Step 8 - use table() function in R showing the difference between cluster results and user id label.
ktable <- table(res =  kmRes$cluster, real = PS2$V1)

#returns cluster label
cmRes <- cmeans(M2, centers = 3, iter.max = 100)
head(cmRes$cluster)

#return membership (the probability of one data point belongs to one group. )
```

```
#the clutser label is based on max(membership)
head(cmRes$membership)

# Step 7 - run hierarchical cluster
d <- dist(M2, method = "euclidean")
HclustResult <- hclust(d, method="ward.D")

plot(HclustResult)

groups <- cutree(HclustResult, k=3) # cut tree into n clusters
# draw dendogram with red borders around the n clusters
rect.hclust(HclustResult, k=3, border="red")

# Step 8 - use table() function in R showing the difference between cluster results and user id label.
htable <- table(res = groups, real = PS2$V1)



# Step 2: Make server script.
# Analyze data/model, make output and plots.
# The outputs needs to be defined in render functions.
# If you have selection in UI, we need reactive function.
shinyServer(function(input, output) {

  # Tab Results
  output$kmeans <- renderPrint({
    kmRes$size
  })

  output$ktotss <- renderPrint({
```

```
  kmRes$totss
})

output$kwith <- renderPrint({
  kmRes$withinss
})

output$ktotwith <- renderPrint({
  kmRes$tot.withinss
})

output$kbet <- renderPrint({
  kmRes$betweenss
})

output$Hclust <- renderPrint({
  HclustResult
})

output$Hplot <- renderPlot({
  plot(HclustResult)
  groups <- cutree(HclustResult, k=3)
  rect.hclust(HclustResult, k=3, border="red")
})

# Tab Table Comparisons
output$ktable <- renderTable({
  ktable
})
```

```
output$htable <- renderTable({
  htable
})
# Tab 2 - wordcloud
# We choose the doc and color in UI, and pass them to p_col and freq_column
# p_col defines whether we have color or black/white plot.
# Then freq_column is passed to wordcloud plot.
p_col <- reactive({
  if(input$Colorinp2){
    brewer.pal(8, "Dark2")
  } else {
    NA
  }
})

# wordcloud plot
# It make use of Ninp2 to define the min frequency in the plot
output$pout2 <- renderPlot({
  # plot wordcloud
  par(mfrow = c(1, 3))
  wordcloud(words = final$word, freq = final$freq1, min.freq = input$Ninp2, colors=p_col())
  wordcloud(words = final$word, freq = final$freq2, min.freq = input$Ninp2.1, colors=p_col())
  wordcloud(words = final$word, freq = final$freq3, min.freq = input$Ninp2.2, colors=p_col())
})

# Print out table format with top rows from final data frame
# The number of rows is defined by the user.
# Decide which doc
```

```
output$tout2 <- renderTable({
  head(docContent(), n = input$Ninp3)
})

# Tab Summary
output$tout3 <- renderTable({
  PS2
})

output$summaryu1 <- renderPrint({
  print.sum[1]
})

output$summaryu2 <- renderPrint({
  print.sum[2]
})

output$summaryu3 <- renderPrint({
  print.sum[3]
})

# # Tab 4 - plotly
# output$plotly4 <- renderPlotly({
#    plot_ly(stockdata(), x = ~rownames(stockdata()), y = ~stockdata()[,4], type = "scatter", mode =
"lines", name = "Close")%>%
#      add_trace(y = ~stockdata()[,1], name = 'Open')
#  })
# #
```

```
  # # Tab SQL and Names
  output$printsql1 <- renderText({
    SQLtext1
  })

  output$printsql2 <- renderText({
    SQLtext2
  })

  #Print exactly like the output in R console
  output$randomusers <- renderPrint({
    random_users[,1]
  })
})
```

UI File:

```
#Step 1: load packages
library(plotly)
library(ggvis)
library(shinythemes)

#UI define the layout and format of whole web interface
shinyUI(
  fluidPage(

    # Use the theme in shinythemes package
    theme = shinytheme("united"),
    # import .css file (external file)
    # You need to put the css file into www folder
    #theme = "style.css",
```

```
########### Set header ##########
# Application title
headerPanel("FE513 (Name: Assignment 3 App)"),

########### Set sidebar ##########
# Sidebar with a slider input for number of observations
sidebarPanel(
  # Input option for colors on wordclod
  checkboxInput("Colorinp2", "wordcloud color?", FALSE),

  # Define a input numeric block with default value = 2, each word cloud can have a different frequency
  numericInput("Ninp2", "Wordcloud min frequency", 2),

  numericInput("Ninp2.1", "Wordcloud min frequency", 2),

  numericInput("Ninp2.2", "Wordcloud min frequency", 2)

),

########### Set mainpanel ##########
# Show a plot of the generated distribution
mainPanel(
  h3("The widget on the side panel controls the word clouds. This page loads slowly due to the processing of words in the database."),

  tabsetPanel(
```

```
#       tabPanel("1.Random Line", code("data <- rnorm(N)"),plotOutput("pout11"),
plotOutput("pout12")),

      tabPanel("E1.SQL and Names", textOutput("printsql1"), p("\n"), textOutput("printsql2"), p("\n"),
textOutput("randomusers")),

      tabPanel("E1.Wordcloud", plotOutput("pout2")),

      tabPanel("E2.Summary", textOutput("summaryu1"), p("\n"), textOutput("summaryu2"), p("\n"),
textOutput("summaryu3"), p("\n"), tableOutput("tout3")),

      tabPanel("E2.Results", h3("The information below is the size of the clusters."),
textOutput("kmeans"), p("\n"), h3("The information below indicates variance between clusters, within
clusters, and of clusters."), textOutput("kbet"), p("\n"), textOutput("ktotss"), p("\n"),
textOutput("kwith"), p("\n"), textOutput("ktotwith"), p("\n"), textOutput("Hclust"), p("\n"),
plotOutput("Hplot")),

      tabPanel("E2.Compare", h3("The table below is for K means."), p("\n"), tableOutput("ktable"),
p("\n"), h3("The table below is for Hierarchical Clustering."), p("\n"), tableOutput("htable"))

     )



   )

  )

)
```