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)

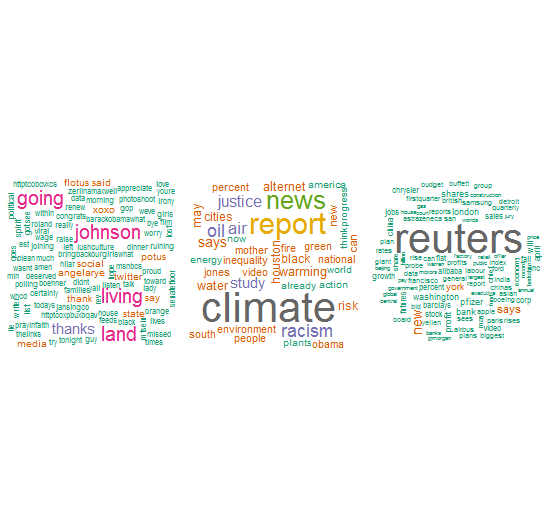
final <- data.frame(word = rownames(M1),freq1=M1[,1], freq2 = M1[,2], freq3 = M1[,3])

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")

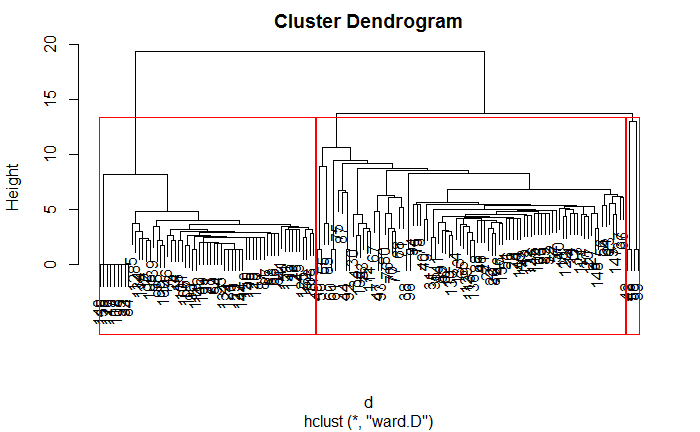
HclustResult <- hclust(d, method="ward.D")

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

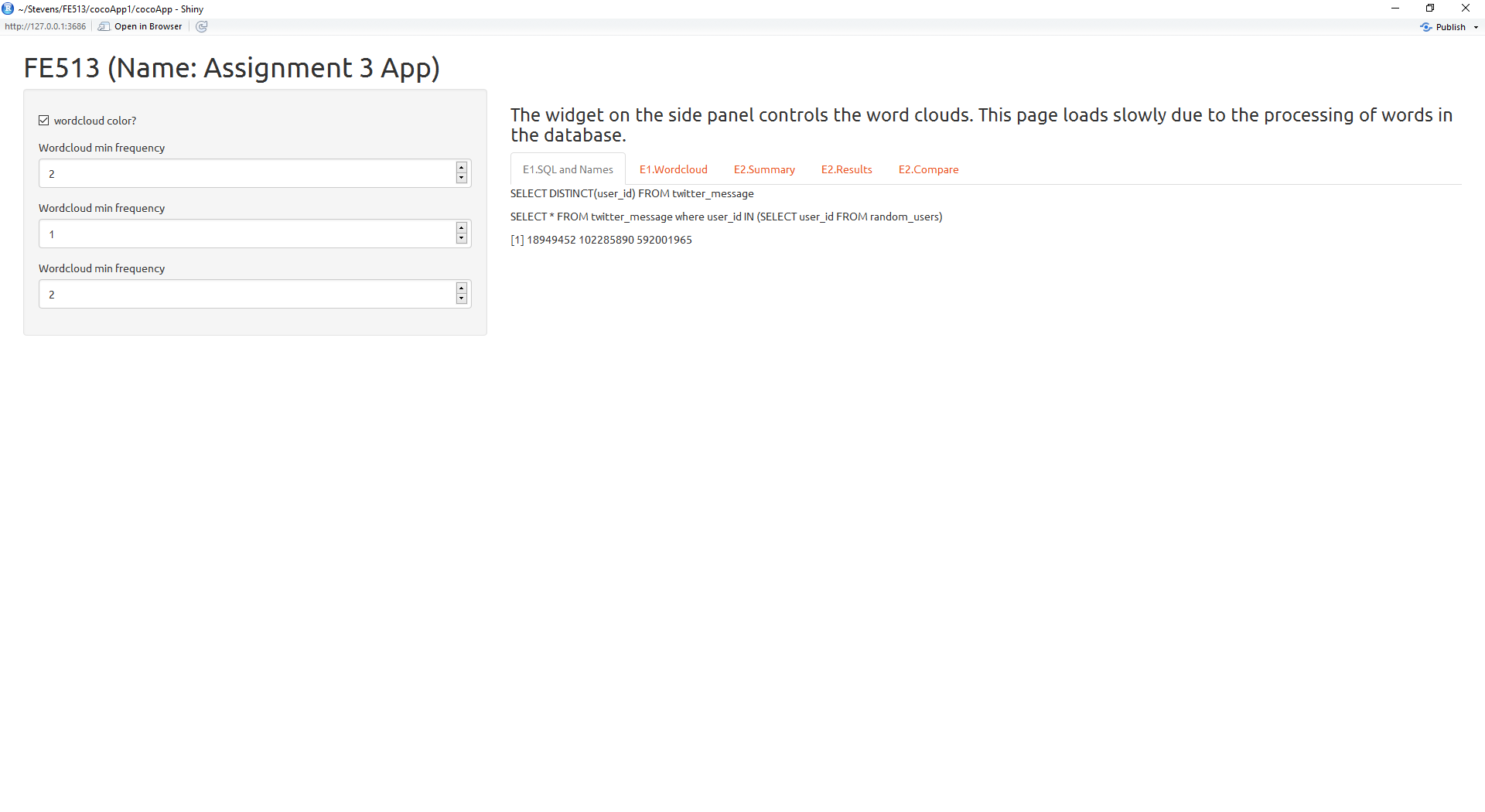
res 20659892 241205643 39221596

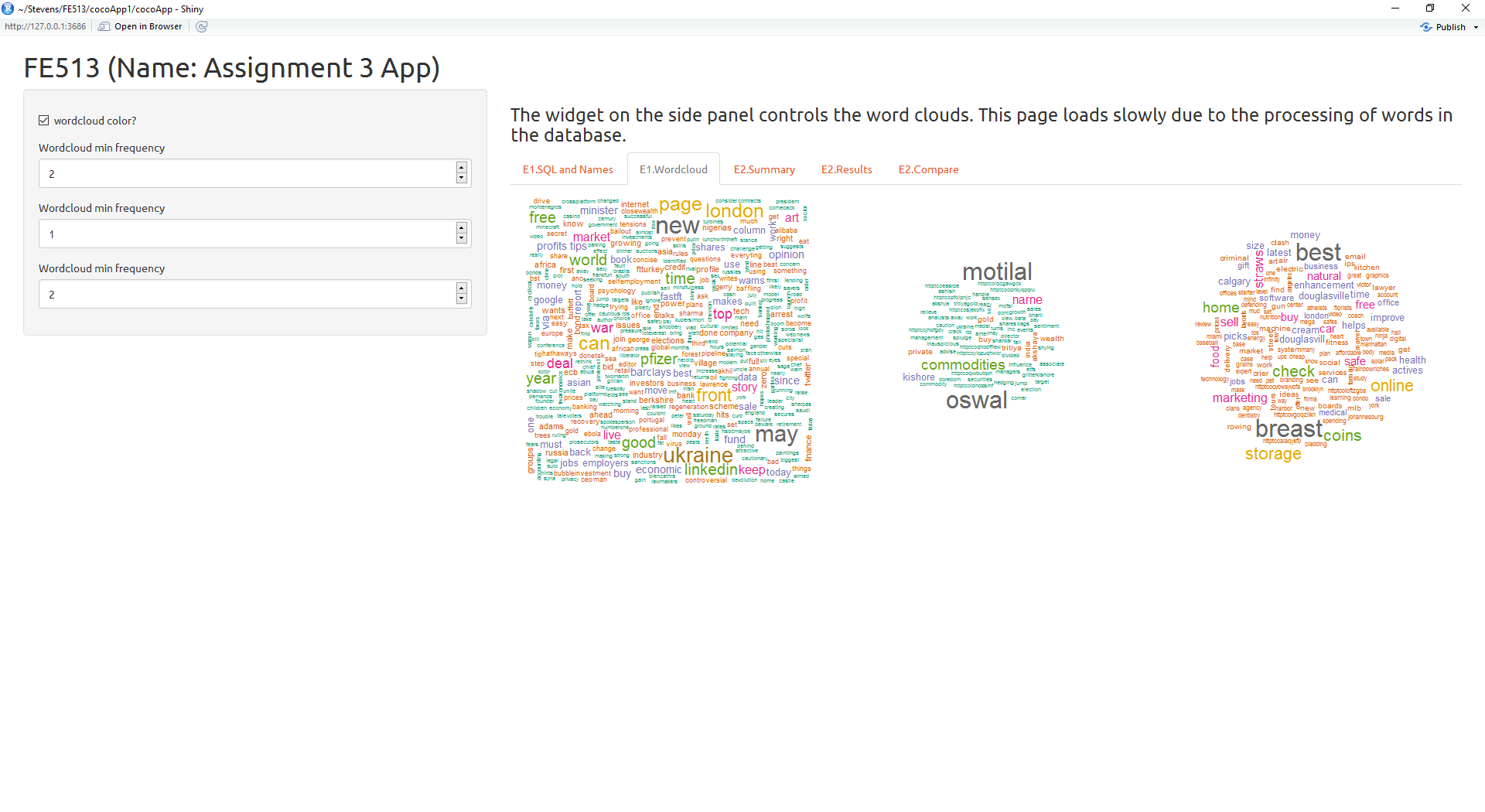
1 53 6 2

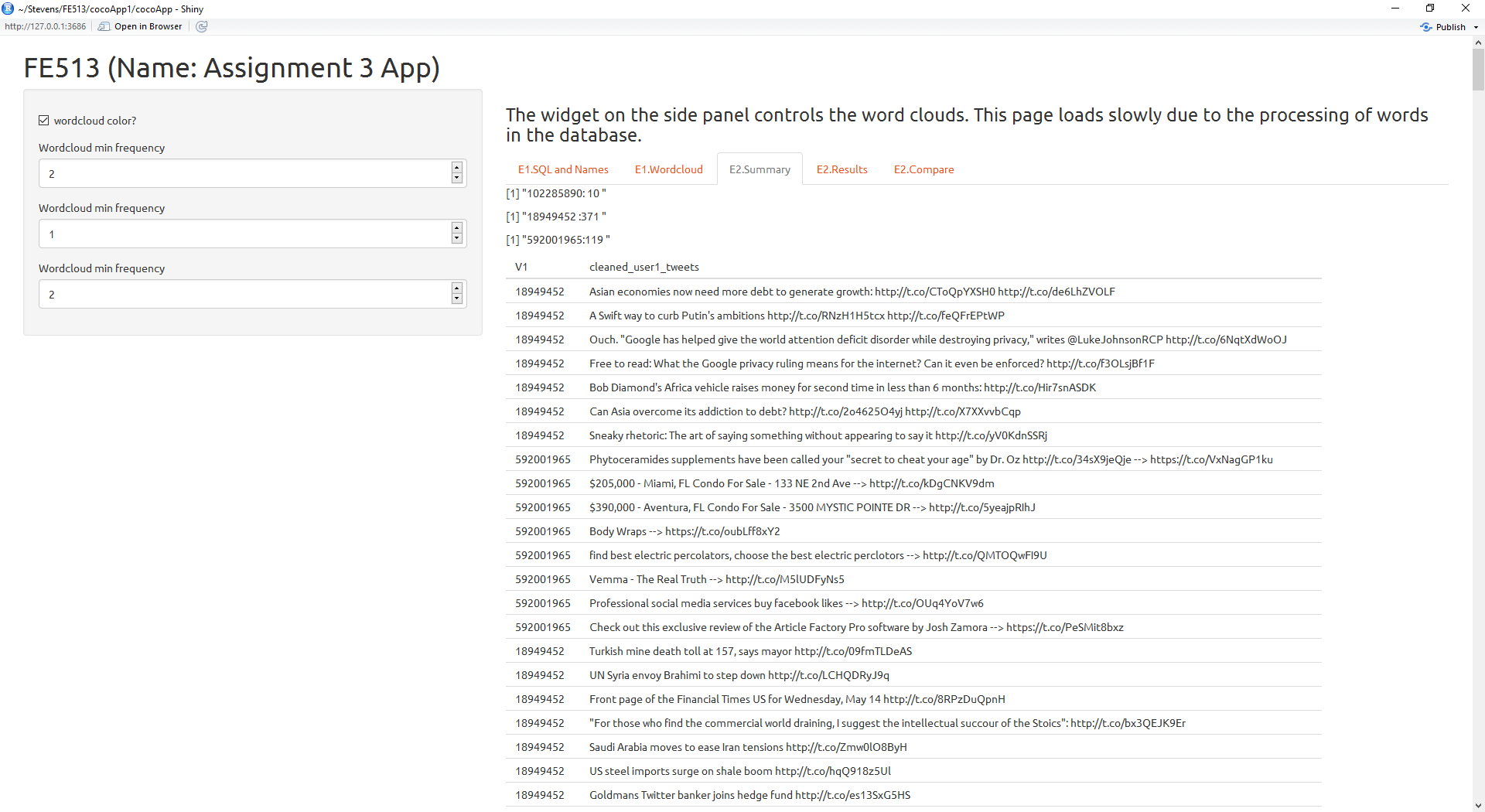
2 31 38 18

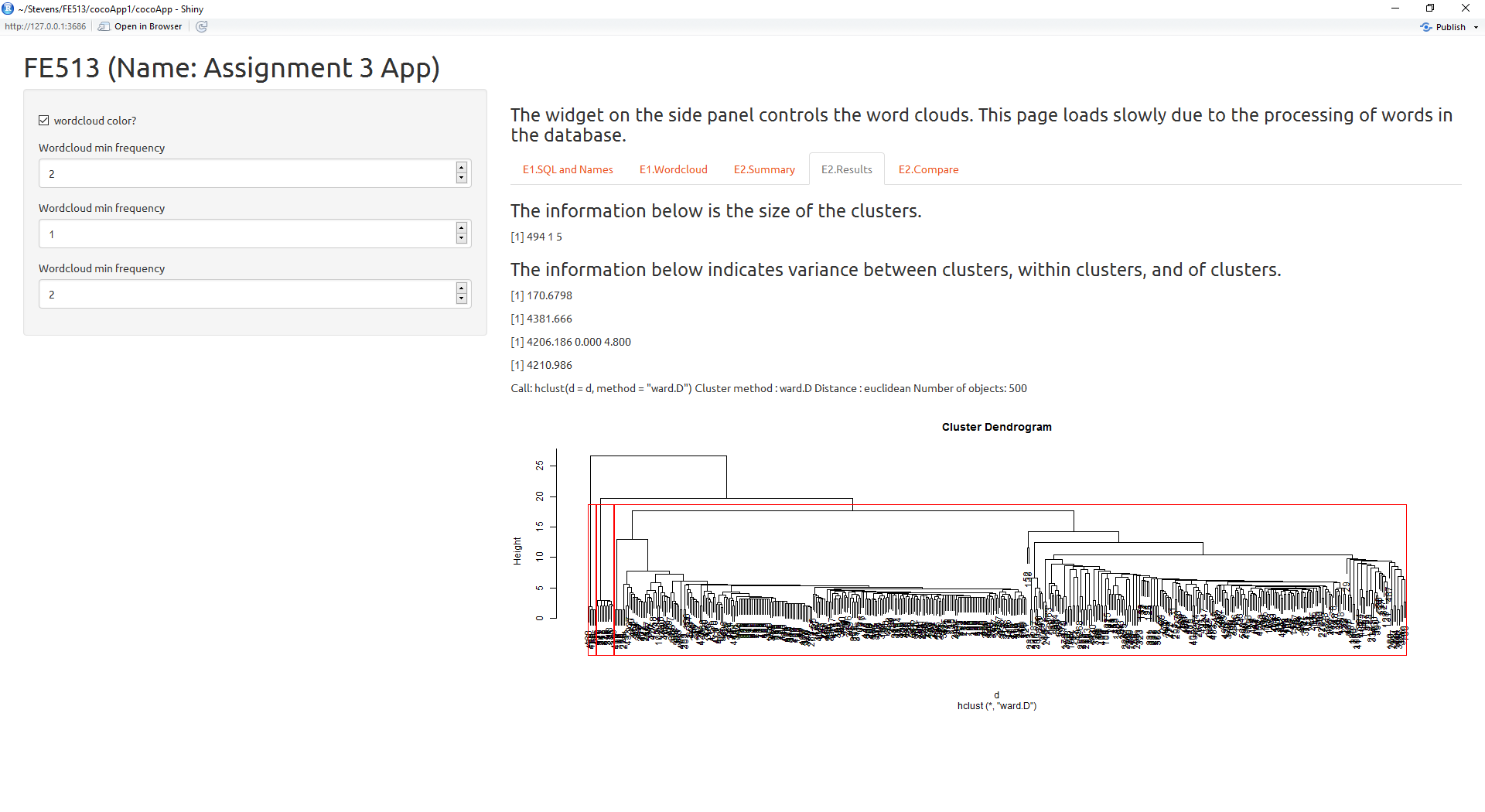
3 0 4 0

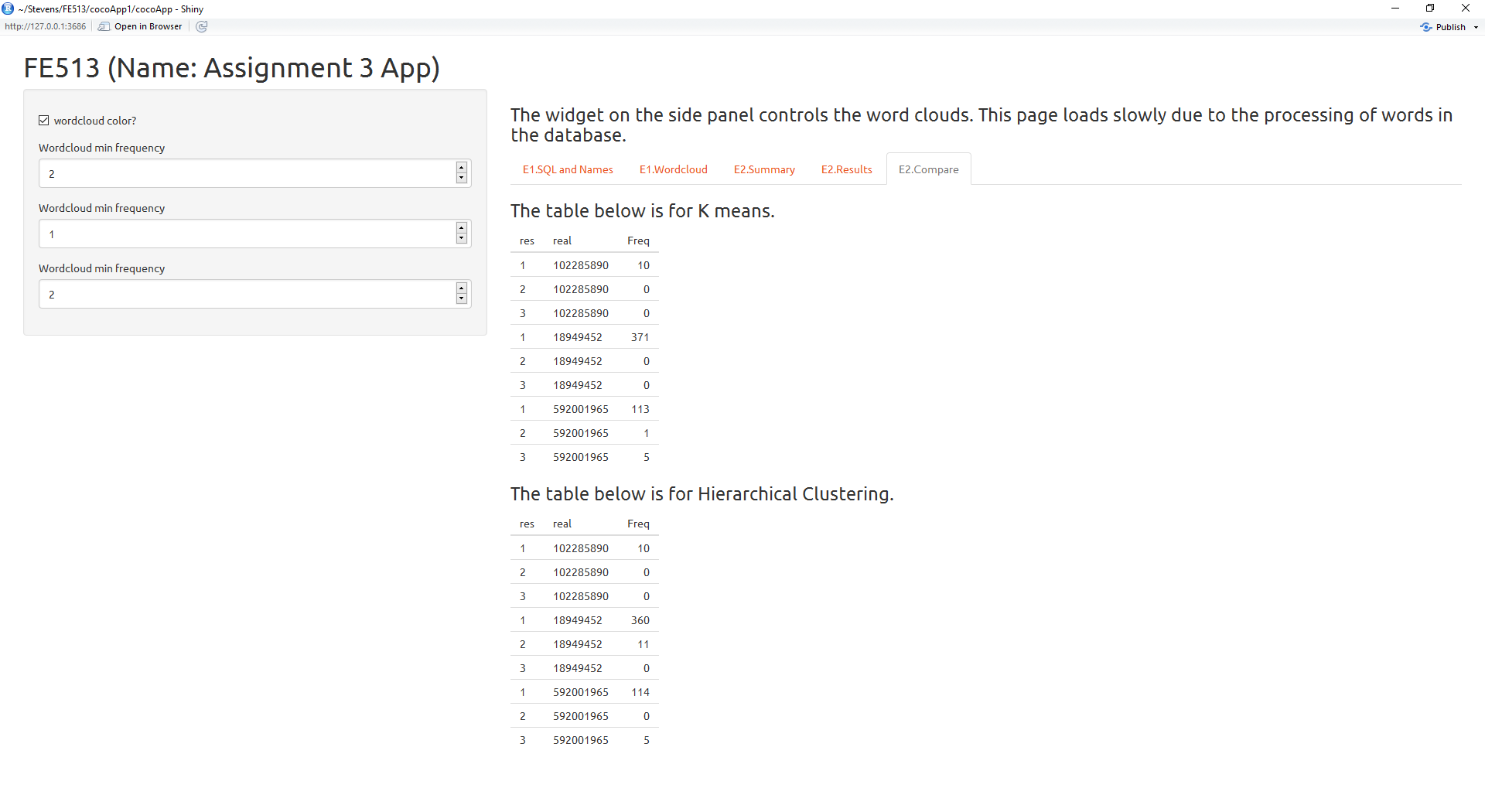
**Problem 3 Screenshots and Code:**











**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"))

)

)

)

)