```r
#Imported the VAERSDATA, VAERSSYMPTOMS,VAERSVAX
View(VAERSDATA)
View(VAERSSYMPTOMS)
View(VAERSVAX)

summary(VAERSSYMPTOMS)

#Dimensions of each dataset
dim(VAERSDATA)
dim(VAERSSYMPTOMS)
dim(VAERSVAX)

#creating a duplicate of symptoms dataset
VAERSSYMPTOMS2 <- VAERSSYMPTOMS

#Creating a new empty column 'SYMPTOMS' to copy all symptoms into one cell for each VAERS ID
VAERSSYMPTOMS2[ , 'SYMPTOMS'] <- NA

#concatenating symptoms variables horizontally under the new column 'SYMPTOMS'
VAERSSYMPTOMS2$SYMPTOMS <-ifelse(VAERSSYMPTOMS2$SYMPTOM2 != "", paste(VAERSSYMPTOMS2$SYMPTOM1, ",", VAERSSYMPTOMS2$SYMPTOM2),VAERSSYMPTOMS2$SYMPTOM1)
VAERSSYMPTOMS2$SYMPTOMS <-ifelse(VAERSSYMPTOMS2$SYMPTOM3 != "", paste(VAERSSYMPTOMS2$SYMPTOMS, ",", VAERSSYMPTOMS2$SYMPTOM3),VAERSSYMPTOMS2$SYMPTOMS)
VAERSSYMPTOMS2$SYMPTOMS <-ifelse(VAERSSYMPTOMS2$SYMPTOM4 != "", paste(VAERSSYMPTOMS2$SYMPTOMS, ",", VAERSSYMPTOMS2$SYMPTOM4),VAERSSYMPTOMS2$SYMPTOMS)
VAERSSYMPTOMS2$SYMPTOMS <-ifelse(VAERSSYMPTOMS2$SYMPTOMS != "", paste(VAERSSYMPTOMS2$SYMPTOMS, ",", VAERSSYMPTOMS2$SYMPTOM5),VAERSSYMPTOMS2$SYMPTOMS)

#Concatenating symptoms belonging to the same VAERS ID
VAERSSYMPTOMS2 <- within(VAERSSYMPTOMS2,
                         {SYMPTOMS <- as.character(SYMPTOMS);
                         SYMPTOMS <- ave(SYMPTOMS, VAERS_ID, FUN = toString)})

#Dropping SYMPTOMVERSIONS and individual SYMPTOMS from the VAERSSYMPTOMS dataset
VAERSSYMPTOMS2 <- subset(VAERSSYMPTOMS2, select = -c(SYMPTOM1,SYMPTOM2,SYMPTOM3,SYMPTOM4,SYMPTOM5,SYMPTOMVERSION1,SYMPTOMVERSION2,SYMPTOMVERSION3,SYMPTOMVERSION4,SYMPTOMVERSION5))

#Dropping duplicate rows in the VAERSSYMPTOMS2 based on VAERS_ID
VAERSSYMPTOMS2 <- VAERSSYMPTOMS2[!duplicated(VAERSSYMPTOMS2$VAERS_ID), ]
View(VAERSSYMPTOMS2)

#Checking for duplicates in VAERSDATA  for VAERS_ID
dim(VAERSDATA[duplicated(VAERSDATA$VAERS_ID),])[1]

#count for duplicates in VAERSVAX  for VAERS_ID
dim(VAERSVAX[duplicated(VAERSVAX$VAERS_ID),])[1]

#left joining VAERSDATA and VAERSVAX
DATAVAX<-merge(x=VAERSDATA,y=VAERSVAX,by="VAERS_ID",all.x=TRUE)
View(DATAVAX)

#left joining DATASYMP and VAERSSYMPTOMS2
MASTER<-merge(x=DATAVAX,y=VAERSSYMPTOMS2,by="VAERS_ID",all.x=TRUE)
View(MASTER)

#creating a duplicate version for cleaning
MASTER.CLEAN <- MASTER

#formatting variable SEX
MASTER.CLEAN$SEX <- as.factor(MASTER.CLEAN$SEX)

#Filling empty cells in state column with NA
MASTER.CLEAN$STATE[MASTER.CLEAN$STATE == ""] <- NA
View(MASTER.CLEAN)

#Correcting blank cells of age
MASTER.CLEAN$AGE_YRS[MASTER.CLEAN$AGE_YRS == ""] <- NA
MASTER.CLEAN$CAGE_YR[MASTER.CLEAN$CAGE_YR == ""] <- NA
MASTER.CLEAN$AGE_YRS <-ifelse(is.na(MASTER.CLEAN$AGE_YRS) & !is.na(MASTER.CLEAN$CAGE_YR), MASTER.CLEAN$CAGE_YR, MASTER.CLEAN$AGE_YRS)
MASTER.CLEAN$CAGE_YR <-ifelse(is.na(MASTER.CLEAN$CAGE_YR) & !is.na(MASTER.CLEAN$AGE_YRS), MASTER.CLEAN$AGE_YRS, MASTER.CLEAN$CAGE_YR)

#Filling blank cells with U in RECOVD column
MASTER.CLEAN$RECOVD[MASTER.CLEAN$RECOVD == ""] <- "U"
View(MASTER.CLEAN)

#formatting variable RECOVD
MASTER.CLEAN$RECOVD <- as.factor(MASTER.CLEAN$RECOVD)

#Filling empty cells in VAX_DATE, ONSET_DATE, NUMDAYS, TODAYS_DATE, RPT_DATE, VAX_LOT column with NA
MASTER.CLEAN$VAX_DATE[MASTER.CLEAN$VAX_DATE == ""] <- NA
MASTER.CLEAN$ONSET_DATE[MASTER.CLEAN$ONSET_DATE == ""] <- NA
MASTER.CLEAN$NUMDAYS[MASTER.CLEAN$NUMDAYS == ""] <- NA
MASTER.CLEAN$RPT_DATE[MASTER.CLEAN$RPT_DATE == ""] <- NA
MASTER.CLEAN$TODAYS_DATE[MASTER.CLEAN$TODAYS_DATE == ""] <- NA
MASTER.CLEAN$VAX_LOT[MASTER.CLEAN$VAX_LOT == ""] <- NA
MASTER.CLEAN$VAX_ROUTE[MASTER.CLEAN$VAX_ROUTE == ""] <- "UN"
MASTER.CLEAN$VAX_SITE[MASTER.CLEAN$VAX_SITE == ""] <- NA
#Filling empty cells in V_FUNDBY with UNK
MASTER.CLEAN$V_FUNDBY[MASTER.CLEAN$V_FUNDBY == ""] <- "UNK"

#formatting VAX and ONSET dates
library(lubridate)
MASTER.CLEAN$VAX_DATE <- mdy(MASTER.CLEAN$VAX_DATE)
MASTER.CLEAN$ONSET_DATE <- mdy(MASTER.CLEAN$ONSET_DATE)

#filling numdays wherever possible using vax date and onset date
MASTER.CLEAN$NUMDAYS <-ifelse(is.na(MASTER.CLEAN$NUMDAYS) & !is.na(MASTER.CLEAN$VAX_DATE) & !is.na(MASTER.CLEAN$ONSET_DATE), (MASTER.CLEAN$ONSET_DATE-MASTER.CLEAN$VAX_DATE), MASTER.CLEAN$NUMDAYS)

#filling TODAYS_DATE(form2) blank cells from RPT_DATE(form1)
MASTER.CLEAN$TODAYS_DATE <-ifelse(is.na(MASTER.CLEAN$TODAYS_DATE) & !is.na(MASTER.CLEAN$RPT_DATE), MASTER.CLEAN$RPT_DATE, MASTER.CLEAN$TODAYS_DATE)
View(MASTER.CLEAN)

#Dropping RPT_DATE as we have updated version TODAYS_DATE
MASTER.CLEAN <- subset(MASTER.CLEAN, select = -c(RPT_DATE))

#formatting RECVD DATE and TODAYS DATE
MASTER.CLEAN$RECVDATE <- mdy(MASTER.CLEAN$RECVDATE)
MASTER.CLEAN$TODAYS_DATE <- mdy(MASTER.CLEAN$TODAYS_DATE)
View(MASTER.CLEAN)

#range of variables in the dataset HOSPDAYS  ,NUMDAYS
summary(MASTER.CLEAN)
View(MASTER.CLEAN)

#count of negative values in NUMDAYS
sum(MASTER.CLEAN$NUMDAYS < 0, na.rm=TRUE)

#Converting negative NUMDAYS and corresponding VAX_DATE to NA
MASTER.CLEAN$VAX_DATE[MASTER.CLEAN$NUMDAYS<0] <- NA
MASTER.CLEAN$ONSET_DATE[MASTER.CLEAN$NUMDAYS<0] <- NA
MASTER.CLEAN$NUMDAYS[MASTER.CLEAN$NUMDAYS<0] <- NA
sum(MASTER.CLEAN$NUMDAYS < 0,na.rm=TRUE)
View(MASTER.CLEAN)

#Box-plots for HOSPDAYS, NUMDAYS - Outliers
library(ggplot2)
qplot(x="",y= MASTER.CLEAN$HOSPDAYS, geom="boxplot", col = I("darkblue"), fill = I("lightblue"), ylab = "HOSPDAYS", xlab = "", main = "HOSPDAYS box plot")
qplot(x="",y= MASTER.CLEAN$NUMDAYS, geom="boxplot", col = I("darkblue"), fill = I("lightblue"), ylab = "NUMDAYS", xlab = "", main = "NUMDAYS box plot")
```

```r
#checking count of observations, if any, where age is less than onset days
sum(MASTER.CLEAN$AGE_YRS < (MASTER.CLEAN$NUMDAYS/365),na.rm=TRUE)

#checking count of observations, if any, where age is less than HOSPDAYS days
sum(MASTER.CLEAN$AGE_YRS < (MASTER.CLEAN$HOSPDAYS/365),na.rm=TRUE)

#Replacing NUMDAYS with NA where age is less than onset days
MASTER.CLEAN$NUMDAYS <-ifelse(MASTER.CLEAN$AGE_YRS<(MASTER.CLEAN$NUMDAYS/365), NA, MASTER.CLEAN$NUMDAYS)

#Replacing HOSPDAYS with NA where age is less than HOSPDAYS days
MASTER.CLEAN$HOSPDAYS <-ifelse(MASTER.CLEAN$AGE_YRS<(MASTER.CLEAN$HOSPDAYS/365), NA, MASTER.CLEAN$HOSPDAYS)

#checking count of observations, if any, where age is less than HOSPDAYS days
sum(MASTER.CLEAN$AGE_YRS < (MASTER.CLEAN$HOSPDAYS/365),na.rm=TRUE)

#checking count of observations, if any, where age is less than onset days
sum(MASTER.CLEAN$AGE_YRS < (MASTER.CLEAN$NUMDAYS/365),na.rm=TRUE)

#Box-plots for HOSPDAYS, NUMDAYS - Outliers
library(ggplot2)
qplot(x="",y= MASTER.CLEAN$HOSPDAYS, geom="boxplot", col = I("darkblue"), fill = I("lightblue"), ylab = "HOSPDAYS", xlab = "", main = "HOSPDAYS box plot")
qplot(x="",y= MASTER.CLEAN$NUMDAYS, geom="boxplot", col = I("darkblue"), fill = I("lightblue"), ylab = "NUMDAYS", xlab = "", main = "NUMDAYS box plot")

#Descriptive Analytics
#bar chart for the count of covid and general vaccines
vac <- c(55348,737587)
label <- c("General","Covid")

# Plot the bar chart
barplot(vac,names.arg=label,xlab="Type of Vaccine",ylab="Count of Reports",col="blue",
        main="General vs Covid Reports",border="red")

#Splitting the dataset into two - covid vs general vaccines
covid <- subset(MASTER.CLEAN,VAX_TYPE %in% c("COVID19"))
general <- MASTER.CLEAN[MASTER.CLEAN$VAX_TYPE != "COVID19", ]
View(covid)
View(general)

#Change Age to Categorical to identify Age as Small, Medium, Large
library(dplyr)
covid$AGE_YRS <- cut(covid$AGE_YRS, breaks = c(0, 18, 40, 59, 120), labels = c("0-18", "19-40", "41-59", "60-120"))
general$AGE_YRS <- cut(general$AGE_YRS, breaks = c(0, 18, 40, 59, 120), labels = c("0-18", "19-40", "41-59", "60-120"))
View(covid)
View(general)

#write the files to local system
library(readr)
write_csv(covid, "/Users/sesh/Desktop/Langara Spring/DANA Quant/covid.csv")
write_csv(general, "/Users/sesh/Desktop/Langara Spring/DANA Quant/general.csv")

#Creating a separate dataframe with SYMPTOM_TEXT for text mining
TEXT_MINE <- subset(MASTER.CLEAN, select = c(VAERS_ID,SYMPTOM_TEXT))
View(TEXT_MINE)

#Text mining on SYMPTOMS_TEXT
#ensuring proper encoding for the column SYMPTOM_TEXT
TEXT_MINE$SYMPTOM_TEXT <- iconv(TEXT_MINE$SYMPTOM_TEXT,"WINDOWS-1252","UTF-8")

#Converting the characters in the SYMPTOM_TEXT column to lower case
TEXT_MINE$SYMPTOM_TEXT = tolower(TEXT_MINE$SYMPTOM_TEXT)

#filling empty cells in the SYMPTOM_TEXT column with NA
TEXT_MINE$SYMPTOM_TEXT[TEXT_MINE$SYMPTOM_TEXT == ""] <- NA

#installing tm package to performing text mining
install.packages('tm')
library(tm)

#Making sure the SYMPTOM_TEXT column is in character format
TEXT_MINE$SYMPTOM_TEXT <- as.character(TEXT_MINE$SYMPTOM_TEXT)

#converting the data into a corpus to perform text mining
corpus <- Corpus(VectorSource(TEXT_MINE$SYMPTOM_TEXT))
inspect(corpus)

#inserting space before and after special characters
tospace <- content_transformer(function(x, pattern){ return (gsub(pattern, " ",x))})
corpus <- tm_map(corpus, tospace, "-")
corpus <- tm_map(corpus, tospace, ":")
corpus <- tm_map(corpus, tospace, "'")
corpus <- tm_map(corpus, tospace, ",")
corpus <- tm_map(corpus, tospace, "/")
corpus <- tm_map(corpus, tospace, "(")
corpus <- tm_map(corpus, tospace, ")")

#remove punctuations
corpus <- tm_map(corpus, removePunctuation)

#remove numbers
corpus <- tm_map(corpus, removeNumbers)

#strip whitespace
corpus <- tm_map(corpus, stripWhitespace)

#removing stopwords using standard stopword list
corpus <- tm_map(corpus, removeWords, stopwords("english"))

#removing stopwords using standard stopword list
corpus <- tm_map(corpus, removeWords, stopwords("SMART"))

#making the corpus a data frame and saving
cp <- data.frame(text = sapply(corpus, as.character), stringsAsFactors = FALSE)

#Placing the corpus data frame in the TEXT_MINE data frame as SYMPTOMS_TEXT_CLEAN
TEXT_MINE$SYMPTOM_TEXT_CLEAN <- cp$text
View(TEXT_MINE)

#Adding the SYMPTOMS column in the TEXT_MINE data frame
TEXT_MINE$SYMPTOMS <- MASTER.CLEAN$SYMPTOMS
View(TEXT_MINE)

#Converting SYMPTOMS column to lowercase
TEXT_MINE$SYMPTOMS<-tolower(TEXT_MINE$SYMPTOMS)

#writing the files to local system
library(readr)
write_csv(TEXT_MINE, "/Users/sesh/Desktop/Langara Spring/DANA Quant/Mining.csv")

###Data set accuracy - idea 1 - checking frequency of words in SYMPTOM_TEXT_CLEAN and comparing with SYMPTOMS
#importing necessary libraries
install.packages("tidytext")
library(tidytext)
library(dplyr)
library(ggplot2)
```

```r
#creating a data frame with only SYMPTOM_TEXT_CLEAN
mined <- subset(TEXT_MINE, select = c(SYMPTOM_TEXT_CLEAN))

#breaking down the data set such that each word is a row
mine <- mined %>%
  unnest_tokens(output = word, input = SYMPTOM_TEXT_CLEAN)

#removing stopping words
mine <- mine %>%
  anti_join(stop_words)

#creating frequency for each word
mine <- mine %>% count(word, sort = TRUE)

#displaying top 50 rows
head(mine, n = 50)

###Data set accuracy - idea 2 - retrieving matching and partially matching keywords between SYMPTOMS and SYMPTOM_TEXT_CLEAN

#Part 1 - retrieving completely matching keywords
#ensuring SYMPTOM_TEXT_CLEAN and SYMPTOMS are in lower case
TEXT_MINE$SYMPTOM_TEXT_CLEAN = tolower(TEXT_MINE$SYMPTOM_TEXT_CLEAN)
TEXT_MINE$SYMPTOMS = tolower(TEXT_MINE$SYMPTOMS)

#retrieving the matching keywords between SYMPTOM_TEXT_CLEAN and SYMPTOMS and saving in a new column match
TEXT_MINE$Match <- matchwords(TEXT_MINE2$SYMPTOM_TEXT_CLEAN, TEXT_MINE2$SYMPTOMS)

#installing required packages
install.packages("stringr")
library(stringr)

#Counting the matched words for each row and inserting the value in total column
TEXT_MINE$total <- sapply(TEXT_MINE$Match, function(x) length(unlist(strsplit(as.character(x), "\\W+"))))

#sum of total column for count of total matched words - 2073516
sum(TEXT_MINE[, 'total'], na.rm = TRUE)

#Total Symptoms we have is 3162837 and the number of matching words is 2073516.
#So the percentage of accuracy just by checking matching words is 65.58% [(3162837/2073516)*100]

#Part 2 - retrieving partially matching keywords

#before proceeding, the TEXT_MINE data set is downloaded and imported onto Python platform.
#Using python, the keywords that have already matched between SYMPTOM_TEXT_CLEAN and SYMPTOMS are
#removed from both the columns so that these words are not repeated when checking for
#partially matching terms.

#writing the files to local system
write_csv(TEXT_MINE, "/Users/sesh/Desktop/Langara Spring/DANA Quant/Mining.csv")

#importing the new data frame (processed on python) as matching_clean

#creating a duplicate of matching_clean
matching_clean2 <- matching_clean

#removing SYMPTOM_TEXT_CLEAN, SYMPTOMS, Match and X columns as these contain 100% matched keywords
#and unnecessary columns
matching_clean2 <- subset(matching_clean2, select = -c(SYMPTOM_TEXT_CLEAN,SYMPTOMS,Match))
matching_clean2 <- subset(matching_clean2, select = -c(X))

#renaming 'Out' and 'symptoms_not_matching' columns to SYMPTOM_TEXT_CLEAN and SYMPTOMS
#(Initially 'Out' and 'symptoms_not_matching' are columns SYMPTOM_TEXT_CLEAN and SYMPTOMS without
#100% matching keywords)
names(matching_clean2)[names(matching_clean2) == 'Out'] <- "SYMPTOM_TEXT_CLEAN"
names(matching_clean2)[names(matching_clean2) == 'symptoms_not_matching'] <- "SYMPTOMS"

#creating a new column 'leven' for adding partially matching keywords
matching_clean2['leven'] <- NA

#Removing numbers, punctuation and special characters from SYMPTOMS column
matching_clean2$SYMPTOMS <- removeNumbers(matching_clean2$SYMPTOMS)
matching_clean2$SYMPTOMS <- removePunctuation(matching_clean2$SYMPTOMS)
matching_clean2$SYMPTOMS <- gsub(",", " ", matching_clean2$SYMPTOMS, perl=T)
matching_clean2$SYMPTOMS <- gsub("-", " ", matching_clean2$SYMPTOMS, perl=T)

#removing special characters in the 'SYMPTOM_TEXT_CLEAN' column
matching_clean2$SYMPTOM_TEXT_CLEAN <- gsub("×", " ", matching_clean2$SYMPTOM_TEXT_CLEAN, perl=T)
matching_clean2$SYMPTOM_TEXT_CLEAN <- gsub("-", " ", matching_clean2$SYMPTOM_TEXT_CLEAN, perl=T)
matching_clean2$SYMPTOM_TEXT_CLEAN <- gsub("~", " ", matching_clean2$SYMPTOM_TEXT_CLEAN, perl=T)
matching_clean2$SYMPTOM_TEXT_CLEAN <- gsub("_", " ", matching_clean2$SYMPTOM_TEXT_CLEAN, perl=T)
matching_clean2$SYMPTOM_TEXT_CLEAN <- gsub("´", " ", matching_clean2$SYMPTOM_TEXT_CLEAN, perl=T)
matching_clean2$SYMPTOM_TEXT_CLEAN <- gsub("¬", " ", matching_clean2$SYMPTOM_TEXT_CLEAN, perl=T)

#Cleaning SYMPTOM_TEXT_CLEAN and SYMPTOMS of extra spaces
library(stringr)
matching_clean2$SYMPTOM_TEXT_CLEAN <- str_squish(matching_clean2$SYMPTOM_TEXT_CLEAN)
matching_clean2$SYMPTOMS <- str_squish(matching_clean2$SYMPTOMS)

#Algorithm which check every word in SYMPTOMS to every word in SYMPTOM_TEXT_CLEAN and returns
#40% matching words.

#loading required packages
library(RecordLinkage)

#code
###########
i <- 0     #initializing dummy variables
j <- 0
k <- 0
s <- ""   #creating a null string

while (i < nrow(matching_clean2)) {     #until all the rows of the data set are checked
  i <- i + 1                            #increment the row count for every iteration
  j <- 0                                #setting the word position in SYMPTOM_TEXT_CLEAN for which the comparison is being done to zero for every iteration
  s<-""                                 #string set to empty for every iteration
  s1 <- matching_clean2$SYMPTOM_TEXT_CLEAN[i]   #saving one row of SYMPTOM_TEXT_CLEAN at a time into s1, for each iteration
  s2 <- matching_clean2$SYMPTOMS[i]             #saving one row of SYMPTOMS at a time into s2, for each iteration
  a <- strsplit(s1, split = " ")                #splitting s1(SYMPTOM_TEXT_CLEAN) based on space and saving it in a
  b <- strsplit(s2, split = " ")                #splitting s2(SYMPTOMS) based on space and saving it in b
  count1 <- str_count(s1, '\\w+')               #saving count of variables in s1(SYMPTOM_TEXT_CLEAN) in count1
  count2 <- str_count(s2, '\\w+')               #saving count of variables in s2(SYMPTOMS) in count2
  while (j<count1) {                            #until all words in s1 are checked
    j <- j+1                                    #increment the word position in SYMPTOM_TEXT_CLEAN after every iteration
    k <- 0                                      #setting the word position in SYMPTOMS for which the comparison is being done to zero for every iteration
    while (k<count2) {                          #until all words in s2 are checked
      k<-k+1                                    #increment the word position in SYMPTOMS after every iteration
      for (x in a[[1]][j]) {                    #each word in a(SYMPTOM_TEXT_CLEAN) at a time
        for (y in b[[1]][k])                    #each word in b(SYMPTOMS) at a time
        {
          c = levenshteinSim(x, y)              #generating levenshtein score for a pair at a time
          if(c > 0.4)                           #checking with the threshold of 0.4 (40%)
            s <- paste(s, y, sep=" ")           #pasting the 40% matched words into a string
        }
```

```
      }
    }
  }
  matching_clean2[[i,4]] <- s                  #copying all the elements pasted in 's'(string) to leven column before proceeding to next row.
}

###########

#writing the files to local system
write_csv(matching_clean2, "/Users/sesh/Desktop/Langara Spring/DANA Quant/matching_clean2.csv")

#The resulting leven column is combined with match(100% matching column) and checked for accuracy on python.

#with 15% estimated error rate in the code, the accuracy percentage was calculated to be 80%, but if checked more accurateley using
#better techniques, the data set can be verified upto 100% accuracy.
```