STAT 4410/8416 Homework 3

Danuwar Ramesh

Due on Oct 20, 2020

```
library(ggplot2)
library(data.table)
library(readxl)
library(stringr)
library(reshape2)
library(dplyr)
library(tm)
```

1. Text Data analysis: Download "lincoln-last-speech.txt" from Canvas which contains Lincoln's last public address. Now answer the following questions and include your codes.

a) Read the text and store the text in `lAddress`. Show the first 70 characters from the first element of the text.

```
lAddress <- readLines("lincoln-last-speech.txt", warn = FALSE)
substr(lAddress[1],1,70)
```

```
## [1] "We meet this evening, not in sorrow, but in gladness of heart. The eva"
```

b) Now we are interested in the words used in his speech. Extract all the words from `lAddress`, convert all of them to lower case and store the result in `vWord`. Display first few words.

```
wPattern <- '\\b\\w+\\b'
wList <- str_extract_all(lAddress, wPattern)
vWord <- tolower(unlist(wList))
head(vWord)
```

```
## [1] "we"      "meet"    "this"    "evening" "not"     "in"
```

c) The words like `am`, `is`, `my` or `through` are not much of our interest and these types of words are called stop-words. The package `tm` has a function called `stopwords()`. Get all the English stop words and store them in `sWord`. Display few stop words in your report.

```
sWord <- stopwords()
head(sWord)
```

```
## [1] "i"      "me"     "my"     "myself" "we"     "our"
```

d) Remove all the `sWord` from `vWord` and store the result in `cleanWord`. Display first few clean words.

```
cleanWord <- vWord[!(vWord %in% sWord)]
head(cleanWord)
```

```
## [1] "meet"       "evening"    "sorrow"     "gladness"   "heart"
## [6] "evacuation"
```

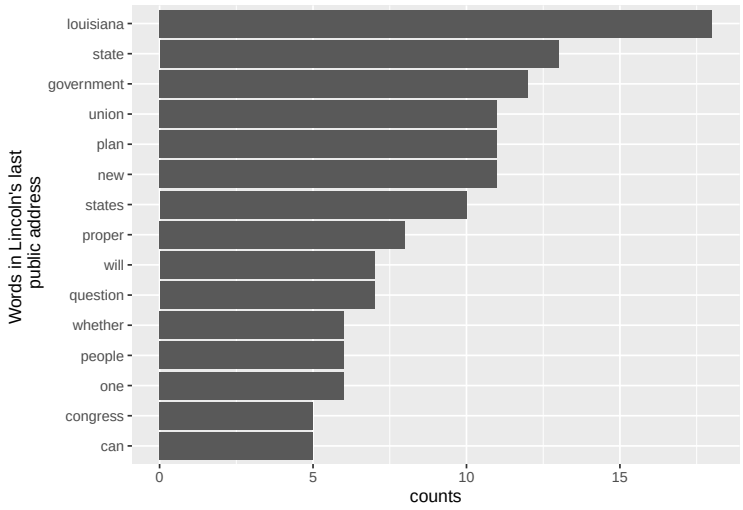
e) `cleanWord` contains all the cleaned words used in Lincoln's address. We would like to see which words are more frequently used. Find 15 most frequently used clean words and store the result in `fWord`. Display first 5 words from `fWord` along with their frequencies.

```
fWord <- sort(table(cleanWord),decreasing=TRUE)[1:15]
fWord[1:5]
```

```
## cleanWord
##  louisiana      state government        new       plan
##         18         13         12         11         11
```

f) Construct a bar chart showing the count of each words for the 15 most frequently used words. Add a layer `+coord_flip()` with your plot.

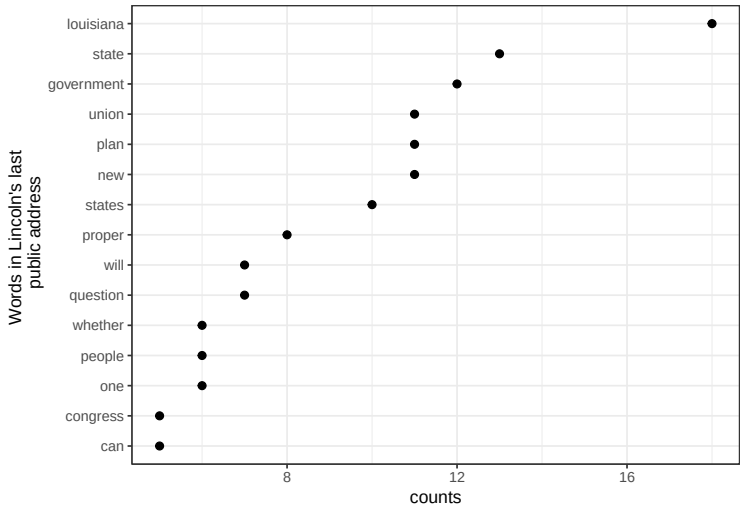
```
myWord <- data.frame(words = names(fWord), counts = as.vector(fWord))
ggplot(myWord, aes(reorder(words,counts) , counts)) + geom_bar(stat="identity") +
  coord_flip() + xlab("Words in Lincoln's last \n public address")
```



g) What is the reason for adding a layer `+coord_flip()` with the plot in question (2f). Explain what would happen if we would not have done that.
The reason for adding a layer +coord_flip() with the plot is so that the horizontal becomes vertical, and vertical, horizontal. Also, if we would not have done that it would be difficult for converting geoms and statistics which display conditional on x, to x conditional on y.

h) The plot in question (2f) uses bar plot to display the data. Can you think of another plot that delivers the same information but looks much simpler? Demonstrate your answer by generating such a plot.

```
ggplot(myWord, aes(reorder(words,counts) , counts)) +
  geom_point(size=2) +
  coord_flip() +
  xlab("Words in Lincoln's last \n public address") +
  theme_bw()
```



i) In the question (2c), you removed words that are called `stop-words`. Now please answer the following:

a) Count the total stop words from `lAddress` and store it in `stopWordsCount`

```
stopWordsCount <- length(sWord)
head(stopWordsCount)
```

```
## [1] 174
```

b) Count the total words (including stop-words) from 'lAddress' and store it in 'lAddressCount'

```
lAddressCount <- length(vWord)
head(lAddressCount)
```

```
## [1] 1827
```

c) Divide 'stopWordsCount' by 'lAddressCount' and report the percentage

```
stopWordsCount/lAddressCount
```

```
## [1] 0.0952381
```

d) Explain in your own words what does the percentage indicate in this context?

In this context, it indicates the total percentage of stop words which is 9.52 %

2. Regular Expressions: Write a regular expression to match patterns in the following strings. Demonstrate that your regular expression indeed matched that pattern by including codes and results. Carefully review how the first problem is solved for you.

a) We have a vector `vText` as follows. Write a regular expression that matches `g, og, go or ogo` in `vText` and replace the matches with '.'.

```
vText <- c('google','logo','dig', 'blog', 'boogie' )
```

Answer:

```
pattern <- 'o?go?'
gsub(pattern, '.', vText)
```

```
## [1] "..le"  "l."    "di."   "bl."   "bo.ie"
```

b) Replace only the 5 or 6 digit numbers with the word "found" in the following vector. Please make sure that 3, 4, or 7 digit numbers do not get changed.

```
vPhone <- c('874','6783','345345', '32120', '468349', '8149674' )
```

```
pattern <- '^\\d{5}$|^\\d{6}$'
gsub(pattern, 'found', vPhone)
```

```
## [1] "874"     "6783"    "found"   "found"   "found"   "8149674"
```

c) Replace all the characters that are not among the 26 English characters or a space. Please replace with an empty spring.

```
myText <- "#y%o$u @g!o*t t9h(e) so#lu!tio$n c%or_r+e%ct"
```

```
pattern <- '[^a-z ]'
gsub(pattern, '', myText)
```

```
## [1] "you got the solution correct"
```

d) In the following text, replace all the words that are exactly 3 or 4 characters long with triple dots '...'

```
myText <- "Each of the three and four character words will be gone now"
```

```
pattern <- '\\b\\w{3,4}\\b'
gsub(pattern, '...', myText)
```

```
## [1] "... of ... three ... ... character words ... be ... ..."
```

e) Extract all the three numbers embedded in the following text.

```
bigText <- 'There are four 20@14 numbers hid989den in the 500 texts'
```

```
pattern <- '\\d+'
library(stringr)
nExtracted <- str_extract_all(bigText, pattern)
unlist(nExtracted )
```

```
## [1] "20"  "14"  "989" "500"
```

f) Extract all the words between parenthesis from the following string text and count number of words.

```
myText <- 'The salries are reported (in millions) for every company.'
```

```
sPattern <- '\\(.+\\)'
myString <- unlist(str_extract_all(myText, sPattern))

wPattern <- '\\b\\w+\\b'
myWords <- unlist(str_extract_all(myString, wPattern))
myWords
```

```
## [1] "in"       "millions"
```

g) Extract the texts in between _ and dot(.) in the following vector. Your output should be 'bill', 'pay', 'fine-book'.

```
myText <- c("H_bill.xls", "Big_H_pay.xls", "Use_case_fine-book.pdf")
```

```
pattern <- '[^_]+\\.'
myString <- unlist(str_extract(myText, pattern))
sub("\\.","",myString)
```

```
## [1] "bill"      "pay"       "fine-book"
```

h) Extract the numbers (return only integers) that are followed by the units 'ml' or 'lb' in the following text.

```
myText <- 'Received 10 apples with 200ml water at 8pm with 15 lb meat and 2lb salt'
```

```
pattern <- '\\d+(ml| *lb)'
myString <- unlist(str_extract_all(myText, pattern))
myNumbers <- str_extract(myString, "\\d+")
myNumbers
```

```
## [1] "200" "15"  "2"
```

i) Extract only the word in between pair of symbols $. Count number of words you have found between pairs of dollar sign $.

```
myText <- 'Math symbols are $written$ in $between$ dollar $signs$'
```

```
pattern <- '\\$[^$]*\\$'
myString <- unlist(str_extract_all(myText, pattern))
myString <- gsub("\\$","",myString)
myString
```

```
## [1] "written" "between" "signs"
```

```
Count <- length(myString)
Count
```

```
## [1] 3
```

j) Extract all the valid equations in the following text.

```
myText <- 'equation1: 2+3=5, equation2 is: 2*3=6, do not extract 2w3=6'
```

```
pattern <- '\\d[+*]\\d=\\d'
myEquation <- unlist(str_extract_all(myText, pattern))
myEquation
```

```
## [1] "2+3=5" "2*3=6"
```

k) Extract all the letters of the following sentence and check if it contains all 26 letters in the alphabet. If not, produce code that will return the total number of unique letters that are included and list the letters that are not present as unique elements in a single vector.

```
myText <- 'there are five wizard boxing matches to be judged'
```

```
pattern <- '[a-z]'
cExtracted <- str_extract_all(myText, pattern)
myLetters <- unlist(cExtracted)
letters %in% myLetters
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [13]  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [25] FALSE  TRUE
```

```
sum(letters %in% myLetters)
```

```
## [1] 21
```

```
letters[!letters %in% myLetters]
```

```
## [1] "k" "l" "p" "q" "y"
```

3. Extracting data from the web: Our plan is to extract data from web sources. This includes email addresses, phone numbers or other useful data. The function `readLines()` is very useful for this purpose.

a) Why are text data `ubiquitous`? and why is it important? Text data is 'Ubiquitous' because it affects our everyday lives. It is important because it is used in our everyday life like mobile maps, gps,online shopping,planning,social medias etc

b) Explain in your own words by giving 2 examples why is it challenging to work with `Text Data`? It is challenging to work with 'Text Data' because there are lot of text data which are uncleaned and meaningless.

c) List any 3 meta characters of a Regular Expression? For each of the meta character you listed, explain what they do? They are: ^ to match from the beginning of the text $ to match from the end of the text

- to match none or any number of time

d) Read all the text in http://mamajumder.github.io/index.html and store your texts in `myText`. Show first few rows of `myText` and examine the structure of the data.

```
myText = readLines('http://mamajumder.github.io/index.html',
                   warn = FALSE)
print(head(myText))
```

```
## [1] "<!DOCTYPE html>"
## [2] "<head>"
## [3] "<link rel=\"stylesheet\" href=\"script/style.css\">"
## [4] "<link href=\"http://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css\" rel=\
## [5] ""
## [6] "<title>Mahbubul Majumder</title>"
```

e) Write a regular expression that would extract all the http web links addresses from `myText`. Include your codes and display the results that show only the http web link addresses and nothing else.

```
wPattern <- 'http:[^"]*'
wList <- str_extract_all(myText, wPattern)
unlist(wList)
```

```
## [1] "http://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css"
## [2] "http://www.unomaha.edu/math/"
```

f) Now write a regular expression that would extract all the emails from `myText`. Include your codes and display the results that show only the email addresses and nothing else.

```
ePattern <- '[_a-z0-9-]+(\\.[_a-z0-9-]+)*\\@[_a-z0-9-]+\\.[_a-z0-9-]+'
eList <- str_extract_all(myText, ePattern)
unlist(eList)
```

```
## [1] "mmajumder@unomaha.edu" "mmajumder@unomaha.edu"
```

g) Now we want to extract all the phone/fax numbers in `myText`. Write a regular expression that would do this. Demonstrate your codes showing the results.

```
phPattern <- '\\(*\\d{3}\\)*( |-)*\\d{3}\\.*( |-)*\\d{4}'
phList <- str_extract_all(myText, phPattern)
unlist(phList)
```

```
## [1] "(402) 554-2734" "(402) 554-2975"
```

h) The link of ggplot2 documentation is http://docs.ggplot2.org/current/ and we would like to get the list of ggplot2 geoms from there. Write a regular expression that would extract all the geoms names (geom_bar is one of them) from this link and display the unique geoms. How many unique geoms does it have?

4. Big data problem: Download the sample of big data from canvas. Note that the data is in csv format and compressed for easy handling. Now answer the following questions.

a) Read the data and select only the columns that contains the word 'human'. Store the data in an object `dat`. Report first few rows of your data.

```
dat <- read.csv("bigDataSample.csv", header=TRUE) %>%
  select(contains('human'))
head(dat)
```

```
##   var_human_1_g var_human_1_p var_human_1_b var_human_1_e var_human_1_n
## 1      18.99545            21             1    21.6321136      26.03268
## 2      15.02303            34             3     0.3838458      26.92529
## 3      37.44410            28             2    33.4801022      39.30039
## 4      36.33714            26             2     2.8761174      33.75177
## 5      21.06330            25             1     3.1657313      26.19248
## 6      16.52637            35             2     5.3108922      25.07192
```

b) The data frame `dat` should have 5 columns. Rename the column names keeping only the last character of the column names. So each column name will have only one character. Report first few rows of your data now.

```
names(dat) <- str_extract(names(dat),'.$')
head(dat)
```

```
##          g  p b          e        n
## 1 18.99545 21 1 21.6321136 26.03268
## 2 15.02303 34 3  0.3838458 26.92529
## 3 37.44410 28 2 33.4801022 39.30039
## 4 36.33714 26 2  2.8761174 33.75177
## 5 21.06330 25 1  3.1657313 26.19248
## 6 16.52637 35 2  5.3108922 25.07192
```

c) Compute and report the means of each columns group by column b in a nice table.

```
sdat <- dat %>%
  group_by(b) %>%
  summarise_all(funs(mean))
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
kable(sdat, digits=2)
```

b	g	p	e	n
0	28.75	23.76	12.21	29.44
1	22.48	25.28	10.42	29.34
2	23.85	24.95	9.62	30.63
3	23.81	25.41	10.48	30.25

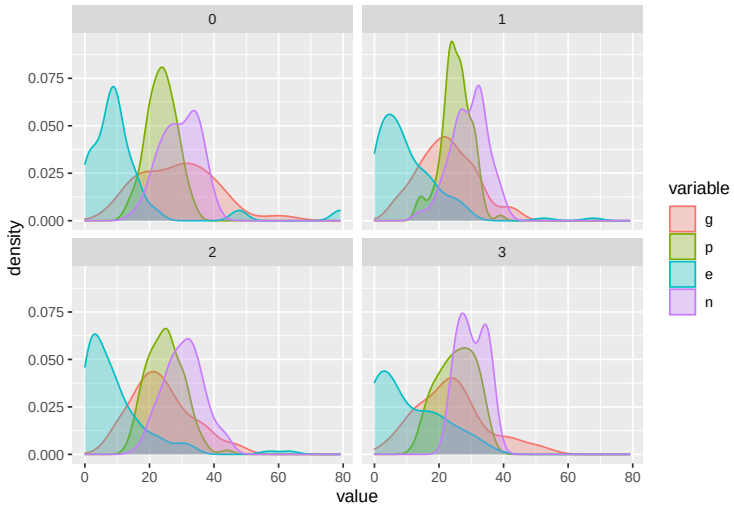
d) Change the data into long form using id='b' and store the data in `mdat`. Report first few rows of data.

```
mdat <- melt(dat, id='b')
head(mdat)
```

```
##   b variable    value
## 1 1        g 18.99545
## 2 3        g 15.02303
## 3 2        g 37.44410
## 4 2        g 36.33714
## 5 1        g 21.06330
## 6 2        g 16.52637
```

e) The data frame `mdat` is now ready for plotting. Generate density plots of value, color and fill by variable and facet by b.

```
ggplot(mdat, aes(value, color=variable)) +
  geom_density(aes(fill=variable), alpha=0.3) +
  facet_wrap(~b)
```



f) The data set `bigDataSample.csv` is a sample of much bigger data set. Here we read the data set and then selected the desired column. Do you think it would be wise do the same thing with the actual larger data set? Explain how you will solve this problem of selecting few columns (as we did in question 6a) without reading the whole data set first. Demonstrate that showing your codes.

```
dat1 <- fread("bigDataSample.csv", nrows = 0, header = T)
myCol <- which(str_detect(colnames(dat1), '.*human.*'))
dat2 <- fread("bigDataSample.csv", select=myCol)
head(dat2)
```

```
##    var_human_1_g var_human_1_p var_human_1_b var_human_1_e var_human_1_n
## 1:      18.99545            21             1    21.6321136      26.03268
## 2:      15.02303            34             3     0.3838458      26.92529
## 3:      37.44410            28             2    33.4801022      39.30039
## 4:      36.33714            26             2     2.8761174      33.75177
## 5:      21.06330            25             1     3.1657313      26.19248
## 6:      16.52637            35             2     5.3108922      25.07192
```

5. Haloween Candy Read the data from the raw Github Link below:

`https://raw.githubusercontent.com/fivethirtyeight/data/master/candy-power-ranking/candy-data.csv`

On the data set, follow these conventions for data attributes/values:

- factor value `1` represents Yes
- factor value `0` represents No
- `sugarpercent` is the percentile of sugar it falls under within the data set
- `pricepercent` is the unit price percentile
- `winpercent` is the overall win percentage
- `competitorname` is the candy name

Answer the following questions below and support your answer showing the codes and a plot (if applicable):

a. Find the candy name that has the 3rd most sugar percentile.

```
haloween_candy <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/candy-power-ra
head(haloween_candy)
```

```
##   competitorname chocolate fruity caramel peanutyalmondy nougat
## 1      100 Grand         1      0       1              0      0
## 2   3 Musketeers         1      0       0              0      1
## 3       One dime         0      0       0              0      0
## 4    One quarter         0      0       0              0      0
## 5      Air Heads         0      1       0              0      0
## 6     Almond Joy         1      0       0              1      0
##   crispedricewafer hard bar pluribus sugarpercent pricepercent winpercent
## 1                1    0   1        0        0.732        0.860   66.97173
## 2                0    0   1        0        0.604        0.511   67.60294
## 3                0    0   0        0        0.011        0.116   32.26109
## 4                0    0   0        0        0.011        0.511   46.11650
## 5                0    0   0        0        0.906        0.511   52.34146
## 6                0    0   1        0        0.465        0.767   50.34755
```

```
haloween_candy_sorted <- haloween_candy %>%
    select("competitorname", "sugarpercent") %>%
    arrange(desc(sugarpercent))
  haloween_candy_sorted[3,1]
```

```
## [1] "Sugar Babies"
```

b. Is the candy with the most sugar percentile (1st) is also the most expensive per `pricepercent`? Answer: No, the candy with the most sugar percetile is also not the most expensive per pricepercent because reeses stuffed with pieces has the most sugar percentile but it does not have the most expensive per pricepercent. milky way simply caramel has the most price percent but it which does not have the most sugar percentile.

```
head(haloween_candy %>%
         select("competitorname", "sugarpercent", "pricepercent") %>%
         arrange(desc(sugarpercent)))
```

```
##                competitorname sugarpercent pricepercent
## 1 ReeseÕs stuffed with pieces        0.988        0.651
## 2    Milky Way Simply Caramel        0.965        0.860
## 3                Sugar Babies        0.965        0.767
## 4           Skittles original        0.941        0.220
## 5          Skittles wildberry        0.941        0.220
## 6                   Air Heads        0.906        0.511
```

c. Which category (chocolate, fruity, caramel, peanutyalmondy, nougat, crispedricewafer, hard, bar, pluribus) of candy from the `dataset` is related to most `sugarpercent`? In other words, if a candy name is either on above category name (meaning the value is `1`), which candy has the highest `sugarpercent`? Answer: According to the below data, chocolate category has the highest sugarpercent becasue it has a number 1 in both reeses stuffed with pieces and milky way simply caramel which were the highest and second highest sugarpercent found in the above data plot.

```
head(haloween_candy %>%
        select("competitorname", "chocolate", "fruity", "caramel", "peanutyalmondy",
               "nougat", "crispedricewafer", "hard", "bar", "pluribus", "sugarpercent") %>%
        filter(chocolate == 1 |
                 fruity == 1 |
                 caramel == 1 |
                 peanutyalmondy == 1 |
                 nougat == 1 |
                 crispedricewafer == 1 |
                 hard == 1 |
                 bar == 1 |
                 pluribus == 1) %>%
        arrange(desc(sugarpercent)))
```

```
##                competitorname chocolate fruity caramel peanutyalmondy nougat
## 1 ReeseÕs stuffed with pieces         1      0       0              1      0
## 2    Milky Way Simply Caramel         1      0       1              0      0
## 3                Sugar Babies         0      0       1              0      0
## 4           Skittles original         0      1       0              0      0
## 5          Skittles wildberry         0      1       0              0      0
## 6                  Air Heads         0      1       0              0      0
##   crispedricewafer hard bar pluribus sugarpercent
## 1                0    0   0        0        0.988
## 2                0    0   1        0        0.965
## 3                0    0   0        1        0.965
## 4                0    0   0        1        0.941
## 5                0    0   0        1        0.941
## 6                0    0   0        0        0.906
```

d. Which candy does not belong to either of the 3 categories - `chocolate` or `crispedricewafer` or `caramel` but has the highest `sugarpercent`? Report ONLY the candy name and the value of corresponding `sugarpercent`.

Answer: The candy which does not belong to either of the 3 categories- chocolate`or`crispedricewafer`or`caramel`but has the highest`sugarpercent is peanutyalmondy

```
haloween_candy_5d <- haloween_candy %>%
    select("competitorname", "chocolate", "fruity", "caramel", "peanutyalmondy",
           "nougat", "crispedricewafer", "hard", "bar", "pluribus", "sugarpercent") %>%
    filter(chocolate == 0 &
             caramel == 0 &
             crispedricewafer == 0 ) %>%
    arrange(desc(sugarpercent))

    head(haloween_candy_5d[c(1,11)])
```

```
##       competitorname sugarpercent
## 1  Skittles original        0.941
## 2 Skittles wildberry        0.941
## 3          Air Heads        0.906
## 4         Candy Corn        0.906
## 5         Gobstopper        0.906
## 6         Mike & Ike        0.872
```

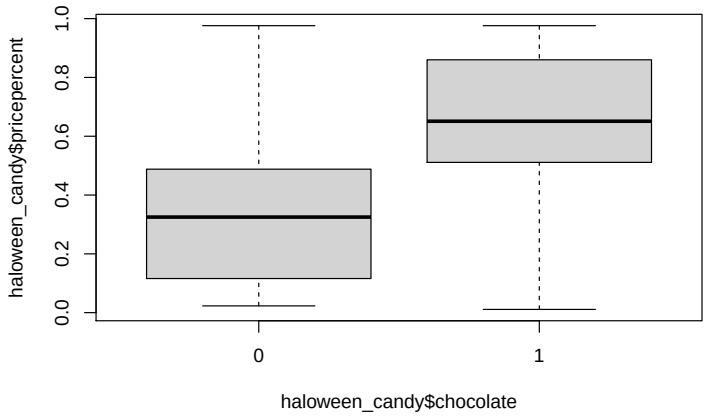
e: Overall, what factor indicates a higher `winpercent`? Basically, what properties or value should a candy name or `competitorname` have to get the highest `winpercent` based on the data set? Answer: Based on the data set , chocolate factor indicates a higher winpercent because its pricepercent is high and its expensive.

```
head(haloween_candy %>%
       arrange(desc(winpercent)))
```

```
##              competitorname chocolate fruity caramel peanutyalmondy nougat
## 1 ReeseÕs Peanut Butter cup         1      0       0              1      0
## 2        ReeseÕs Miniatures         1      0       0              1      0
## 3                      Twix         1      0       1              0      0
## 4                   Kit Kat         1      0       0              0      0
## 5                  Snickers         1      0       1              1      1
## 6            ReeseÕs pieces         1      0       0              1      0
##   crispedricewafer hard bar pluribus sugarpercent pricepercent winpercent
## 1                0    0   0        0        0.720        0.651   84.18029
## 2                0    0   0        0        0.034        0.279   81.86626
## 3                1    0   1        0        0.546        0.906   81.64291
## 4                1    0   1        0        0.313        0.511   76.76860
## 5                0    0   1        0        0.546        0.651   76.67378
## 6                0    0   0        1        0.406        0.651   73.43499
```

f. Generate a side by side box plot of `pricepercent` for the candy name `chocolate`. Explain if `chocolate` category candy are expensive or vice-versa. Answer: Yes, 'chocolate' category are expensive becasue it has the highest pricepercent than other category.

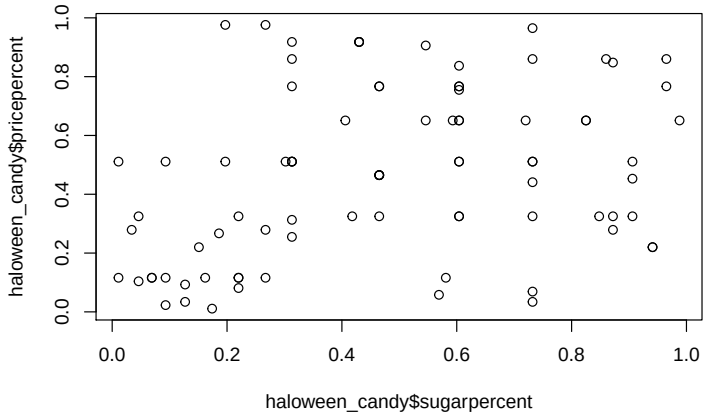
```
boxplot(haloween_candy$pricepercent~haloween_candy$chocolate)
```



g. Do you think that a higher `sugarpercent` indicates a higher `pricepercent`? Generate a plot to support your finding.

Answer: According to the data set, i do not think that a higher sugarpercent indicates a higher pricepercent. For example, reeses stuffed with pieces has the most sugar percentile but it does not have the most expensive per pricepercent. milky way simply caramel has the most price percent but it which does not have the most sugar percentile.

```
plot(haloween_candy$sugarpercent, haloween_candy$pricepercent)
```



6. Optional bonus question (5 points extra) Download the excel file "clean-dat-before.xls" from canvas It contains time series data for many variables. Among the two columns of the data, the first column represents time and the second column represents the measurement. The challenge is that variable names are also included in the time column. Our goal is to clean and reshape the data. First few rows and columns of the desired output is shown below. Notice each time point is converted into an integer time index to make a uniform elapsed time for all the variables.

elapse_time	Area	Bulk.Rotation.	ECG	Endo.MA.Circ..Strain	Endo.MA.Radial.Strain
1	10.924	0.000	0.32157	0.000	0.000
2	10.648	0.070	0.58824	-1.495	0.762
3	10.574	-0.128	0.81176	-1.423	2.619
4	10.487	0.097	0.88627	-0.620	3.591
5	10.342	0.181	0.87451	-1.142	3.472
6	9.995	0.235	0.85882	-3.269	5.812