Midterm

Lin Pin Tzu (Ruby)

2022-07-15

1) U <- c("Maine" , "Texas", "Delaware", "Oregon", "Utah", "Vermont", "Ohio")

```
U <- c("Maine" , "Texas", "Delaware", "Oregon", "Utah", "Vermont", "Ohio")
str(U)
```

```
##  chr [1:7] "Maine" "Texas" "Delaware" "Oregon" "Utah" "Vermont" "Ohio"
```

a) Is U an atomic vector or a list ?

U is an atomic vector since they are all character vectors.

b) Use and show R code that will extract the elements "Maine" and "Vermont".

```
U[c(1,6)]
```

```
## [1] "Maine"   "Vermont"
```

c) Use and show R code that will extract all elements except "Texas".

```
U[-2]
```

```
## [1] "Maine"    "Delaware" "Oregon"   "Utah"     "Vermont"  "Ohio"
```

d) Use and show R code that will produce length of U.

```
length(U)
```

```
## [1] 7
```

2) V = list("Chicago", k = list(2, 6, 18, 24), FALSE, 13, 1.3, y = 1:10)

```
V = list("Chicago", k = list( 2, 6, 18, 24), FALSE, 13, 1.3,  y = 1:10)
str(V)
```

```
## List of 6
##  $  : chr "Chicago"
##  $ k:List of 4
##   ..$ : num 2
```

```
##   ..$ : num 6
##   ..$ : num 18
##   ..$ : num 24
##  $  : logi FALSE
##  $  : num 13
##  $  : num 1.3
##  $ y: int [1:10] 1 2 3 4 5 6 7 8 9 10
```

a) Is V an atomic vector or a list ?

V is list since it has more than one type of vectors.

b) Use and show R code that will extract the 5th element of V.

```
V[5]
```

```
## [[1]]
## [1] 1.3
```

c) If the vector V is a list, use and show R code to identify the type of each object in V.

```
str(V)
```

```
## List of 6
##  $  : chr "Chicago"
##  $ k:List of 4
##   ..$ : num 2
##   ..$ : num 6
##   ..$ : num 18
##   ..$ : num 24
##  $  : logi FALSE
##  $  : num 13
##  $  : num 1.3
##  $ y: int [1:10] 1 2 3 4 5 6 7 8 9 10
```

3) Copy paste and run the tribble given below.

```
tribble( ~John, ~Raymond, ~Martha,~Alice, ~Juan,
          86,    77,    81,       88,    90,
              79,           78,                  85,             81,          78,
              76,           75,                  88,             94,          81,
              84,           90,                  71,              84,          89,
              100,          80,                 93,             85,          84,
              90,           73,                  70,             88,          93,
) -> TestScores
TestScores
```

```
## # A tibble: 6 x 5
##    John Raymond Martha Alice  Juan
##   <dbl>   <dbl>  <dbl> <dbl> <dbl>
## 1    86      77     81    88    90
## 2    79      78     85    81    78
```

```
## 3    76      75     88    94    81
## 4    84      90     71    84    89
## 5   100      80     93    85    84
## 6    90      73     70    88    93
```

a) Use and show R code (a map function) to find the median for each column.

```
map_dbl(TestScores,median)
```

```
##    John Raymond  Martha   Alice    Juan
##    85.0    77.5    83.0    86.5    86.5
```

b) Use and show R code (a map function) to find the cube root of each column element.

```
TestScores %>%
  map(~.^(1/3)) -> T
T
```

```
## $John
## [1] 4.414005 4.290840 4.235824 4.379519 4.641589 4.481405
##
## $Raymond
## [1] 4.254321 4.272659 4.217163 4.481405 4.308869 4.179339
##
## $Martha
## [1] 4.326749 4.396830 4.447960 4.140818 4.530655 4.121285
##
## $Alice
## [1] 4.447960 4.326749 4.546836 4.379519 4.396830 4.447960
##
## $Juan
## [1] 4.481405 4.272659 4.326749 4.464745 4.379519 4.530655
```

c) Use and show R code (a map function) to convert each column value to 0.

```
zero <-TestScores %>%
    map(~. * 0)
zero
```

```
## $John
## [1] 0 0 0 0 0 0
##
## $Raymond
## [1] 0 0 0 0 0 0
##
## $Martha
## [1] 0 0 0 0 0 0
##
## $Alice
## [1] 0 0 0 0 0 0
##
## $Juan
## [1] 0 0 0 0 0 0
```

4) Use and show R code, as demonstrated in class to produce the following matrix

```
z <- matrix( nrow = 3, ncol = 4)
for (m in 1:3) {
  for (n in 1:4) {
    z[m, n] <- (-(m+n)^2)
  }
}
print(z)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   -4   -9  -16  -25
## [2,]   -9  -16  -25  -36
## [3,]  -16  -25  -36  -49
```

5)

a) Show and use a census API key that gives you access to the ACS data. Do not use my API key, use and show your own key.

```
# census_api_key("4009f73e21670e9fb8801c8067991ecb855c1632", install=TRUE)
census_api_key("4009f73e21670e9fb8801c8067991ecb855c1632",overwrite=TRUE)
```

```
## To install your API key for use in future sessions, run this function with `install = TRUE`.
```

```
# For line 102 I can not knit that is why I use #
```

b) Using ACS census data from 2015, show and use R code to do the following to produce a tibble that shows the median income estimates and the margin of errors for white males ages 35 - 44 in the counties of California. The required variable code starts with the characters B01OO1. Use the table to find the other characters.

```
c3544 <- load_variables(2015,"acs5",cache = TRUE)

ca <- get_acs(geography = "county",
              variables = c(medincome = "B01001A_011"),
              state = "06",
              year = 2015)
```

```
## Getting data from the 2011-2015 5-year ACS
```

```
ca
```

```
## # A tibble: 58 x 5
##    GEOID NAME                          variable  estimate   moe
##    <chr> <chr>                         <chr>        <dbl> <dbl>
##  1 06001 Alameda County, California    medincome    51644   667
##  2 06003 Alpine County, California     medincome       50    26
##  3 06005 Amador County, California     medincome     1809    72
```

```
##  4 06007 Butte County, California        medincome     9962   128
##  5 06009 Calaveras County, California    medincome     1927    74
##  6 06011 Colusa County, California       medincome     1147    79
##  7 06013 Contra Costa County, California medincome    42756   605
##  8 06015 Del Norte County, California    medincome     1629    90
##  9 06017 El Dorado County, California    medincome     8609   141
## 10 06019 Fresno County, California       medincome    34979   714
## # ... with 48 more rows
```

```
head(ca,5)
```

```
## # A tibble: 5 x 5
##   GEOID NAME                          variable  estimate   moe
##   <chr> <chr>                         <chr>        <dbl> <dbl>
## 1 06001 Alameda County, California    medincome    51644   667
## 2 06003 Alpine County, California     medincome       50    26
## 3 06005 Amador County, California     medincome     1809    72
## 4 06007 Butte County, California      medincome     9962   128
## 5 06009 Calaveras County, California medincome      1927    74
```

c) Use dplyr functions to change your table of part a so that it reflects estimates that are greater than $30,000 dollars and list the estimates in descending order.

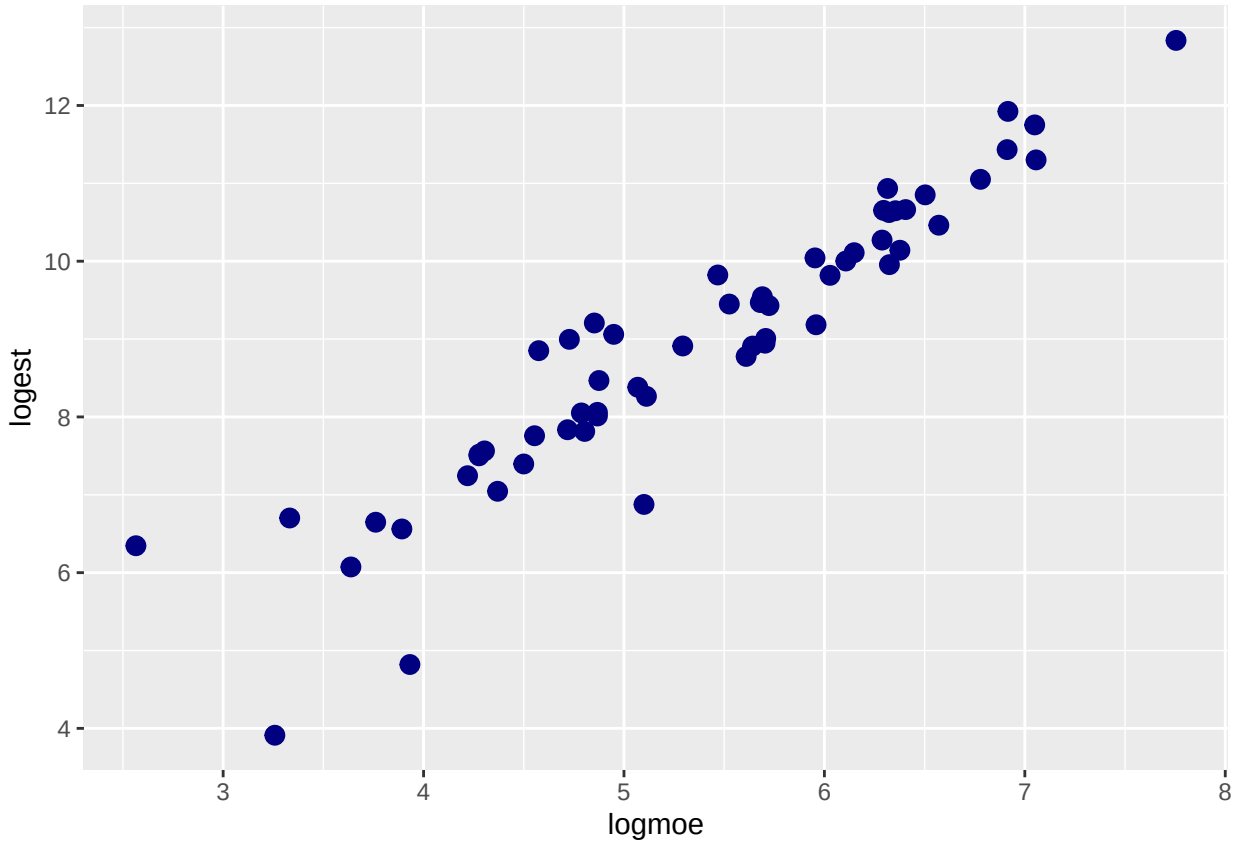
```
ca %>%
  filter(estimate>= 30000) %>%
  arrange(desc(estimate)) ->ca30
ca30
```

```
## # A tibble: 13 x 5
##    GEOID NAME                                variable  estimate   moe
##    <chr> <chr>                               <chr>        <dbl> <dbl>
##  1 06037 Los Angeles County, California      medincome   375435  2332
##  2 06073 San Diego County, California        medincome   150891  1008
##  3 06059 Orange County, California           medincome   126819  1152
##  4 06065 Riverside County, California        medincome    92346  1004
##  5 06071 San Bernardino County, California medincome      80925  1160
##  6 06085 Santa Clara County, California      medincome    63036   879
##  7 06067 Sacramento County, California       medincome    56066   553
##  8 06001 Alameda County, California          medincome    51644   667
##  9 06013 Contra Costa County, California     medincome    42756   605
## 10 06075 San Francisco County, California    medincome    42307   542
## 11 06029 Kern County, California             medincome    42121   575
## 12 06111 Ventura County, California          medincome    41155   557
## 13 06019 Fresno County, California           medincome    34979   714
```

d) Use and show ggplot R coding to produce a scatter plot that features x = natural log of moe plotted against y = natural log of estimate. Does your plot suggest a linear relationship between the varibles ? If so, what general trend can be inferred? (Use the full data table that you generated for part b)

```
log(ca$moe) ->logmoe
log(ca$estimate)-> logest
```

```
ggplot(ca,mapping =aes(x =logmoe,y=logest))+
         geom_point(color= "navy blue",size=3,aplha=0.5)
```

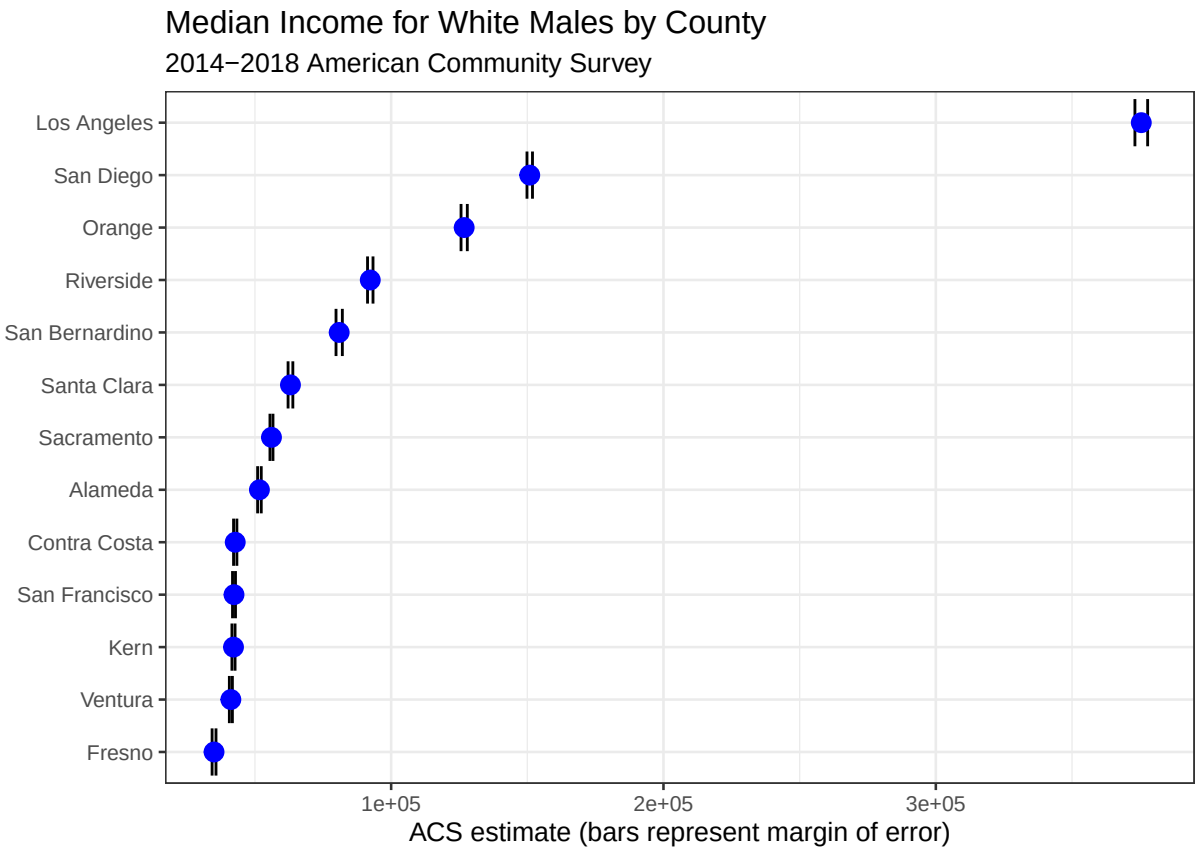



```
calog <- lm(logest~logmoe,ca)
```

I use ggplot to create a scatter plot it shows that natural log of moe and natural log of estimate has a positive linear relationship in here and when logmoe increase log of estimate will also increase so we can say that natural log of moe and natural log of estimate is a linear relationship.

e) Use and show R code that will produce the following graph ing graph for the data generated in part c

```
ca30 %>%
  mutate(NAME = gsub(" County, California", "", NAME)) %>%
  ggplot(aes(x = estimate, y = reorder(NAME, estimate))) +
  geom_errorbarh(aes(xmin = estimate - moe, xmax = estimate + moe)) +
  geom_point(color = "Blue", size = 3) +
  labs(title = "Median Income for White Males by County",
       subtitle = "2014-2018 American Community Survey",
       y = "",
       x = "ACS estimate (bars represent margin of error)")+
  theme_bw(base_size = 10)
```



6) Provided below is the famous poem " Stopping by the Wood On a Snowy Evening" by Robert Frost.Use the text mining sequence of steps and the R code modeled in class to

a) create a tibble

b) find line locations of words

c) produce a word frequency table, and

d) and create a bar graph data visualization plot that will also display word frequency trends.

Hint(do not forget to process a single spaced body of text ; be careful about commas and double quotation marks. Use the examples demonstrated in class.)

```
text <- c("Whose woods these are I think I know.",
          "His house is in the village though;",
          "He will not see me stopping here",
          "To watch his woods fill up with snow.",
          "My little horse must think it queer",
          "To stop without a farmhouse near",
          "Between the woods and frozen lake",
          "The darkest evening of the year.",
          "He gives his harness bells a shake",
          "To ask if there is some mistake.",
```

```
          "The only other sound's the sweep",
          "Of easy wind and downy flake.",
          "The woods are lovely, dark and deep,",
          "But I have promises to keep,",
          "And miles to go before I sleep,",
          "And miles to go before I sleep.")
text
```

```
##  [1] "Whose woods these are I think I know."
##  [2] "His house is in the village though;"
##  [3] "He will not see me stopping here"
##  [4] "To watch his woods fill up with snow."
##  [5] "My little horse must think it queer"
##  [6] "To stop without a farmhouse near"
##  [7] "Between the woods and frozen lake"
##  [8] "The darkest evening of the year."
##  [9] "He gives his harness bells a shake"
## [10] "To ask if there is some mistake."
## [11] "The only other sound's the sweep"
## [12] "Of easy wind and downy flake."
## [13] "The woods are lovely, dark and deep,"
## [14] "But I have promises to keep,"
## [15] "And miles to go before I sleep,"
## [16] "And miles to go before I sleep."
```

```
text_tibble <- tibble(line = 167:182, text = text)

text_tibble
```

```
## # A tibble: 16 x 2
##     line text
##    <int> <chr>
##  1   167 Whose woods these are I think I know.
##  2   168 His house is in the village though;
##  3   169 He will not see me stopping here
##  4   170 To watch his woods fill up with snow.
##  5   171 My little horse must think it queer
##  6   172 To stop without a farmhouse near
##  7   173 Between the woods and frozen lake
##  8   174 The darkest evening of the year.
##  9   175 He gives his harness bells a shake
## 10   176 To ask if there is some mistake.
## 11   177 The only other sound's the sweep
## 12   178 Of easy wind and downy flake.
## 13   179 The woods are lovely, dark and deep,
## 14   180 But I have promises to keep,
## 15   181 And miles to go before I sleep,
## 16   182 And miles to go before I sleep.
```

```
text_tibble %>%
  unnest_tokens(word, text) -> tibble1
tibble1
```

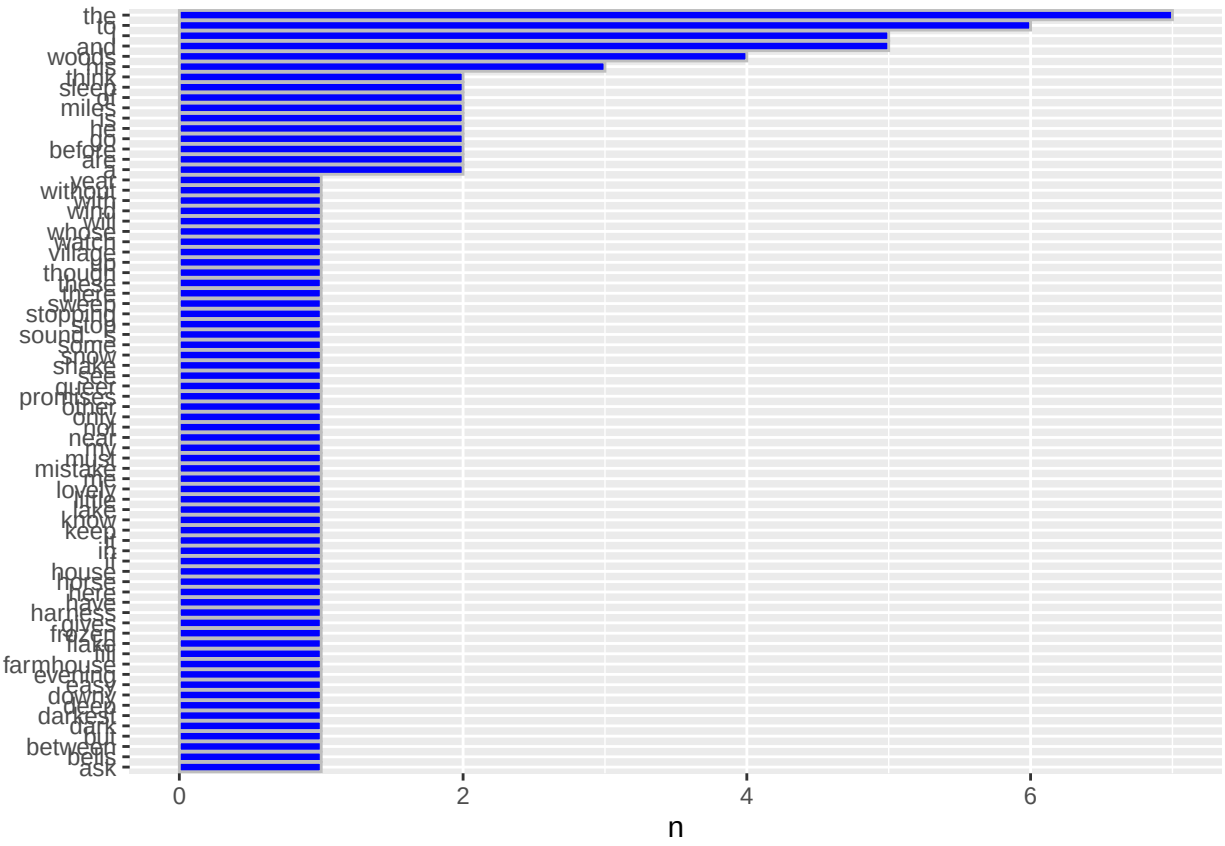
```
## # A tibble: 108 x 2
##     line word
##    <int> <chr>
```

```
##  1   167 whose
##  2   167 woods
##  3   167 these
##  4   167 are
##  5   167 i
##  6   167 think
##  7   167 i
##  8   167 know
##  9   168 his
## 10   168 house
## # ... with 98 more rows
```

```
tibble1%>%
  count(word, sort =TRUE) %>%
 filter(n >= 1)
```

```
## # A tibble: 74 x 2
##    word       n
##    <chr>  <int>
##  1 the        7
##  2 to         6
##  3 and        5
##  4 i          5
##  5 woods      4
##  6 his        3
##  7 a          2
##  8 are        2
##  9 before     2
## 10 go         2
## # ... with 64 more rows
```

```
tibble1%>%
  count(word, sort =TRUE) %>%
  filter(n >= 1) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(n, word)) +
  geom_col(fill = "blue", color = "grey") +
  labs(y = NULL)
```



7) Now using the same body of text found in Problem 6, use and show R code to create a word cloud. You can use the coding and methods that were illustrated in class or you can use alternate coding of your choice to create the word cloud.

```
text1 = read_table("midterm.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Whose = col_character(),
##   woods = col_character(),
##   these = col_character(),
##   are = col_character(),
##   I = col_character(),
##   think = col_character(),
##   I_1 = col_character(),
##   know. = col_character()
## )
```

```
docs <- Corpus(VectorSource(text1))
docs
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
```

```
## Content:  documents: 8
```

```
inspect(docs)
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 8
##
## [1] c("His", "He", "To", "My", "To", "Between", "The", "He", "To", "The", "Of", "The", "But", "And",
## [2] c("house", "will", "watch", "little", "stop", "the", "darkest", "gives", "ask", "only", "easy",
## [3] c("is", "not", "his", "horse", "without", "woods", "evening", "his", "if", "other", "wind", "are
## [4] c("in", "see", "woods", "must", "a", "and", "of", "harness", "there", "sound's", "and", "lovely,
## [5] c("the", "me", "fill", "think", "farmhouse", "frozen", "the", "bells", "is", "the", "downy", "da
## [6] c("village", "stopping", "up", "it", "near", "lake", "year.", "a", "some", "sweep", "flake.", "a
## [7] c("though;", "here", "with", "queer", NA, NA, NA, "shake", "mistake.", NA, NA, "deep,", NA, "sle
## [8] c(NA, NA, "snow.", NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA)
```

```
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "/")
docs <- tm_map(docs, toSpace, "@")
docs <- tm_map(docs, toSpace, "\\|")
docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, removeNumbers)

docs <- tm_map(docs, removeWords, stopwords("english"))

docs <- tm_map(docs, removeWords, c("blabla1", "blabla2"))

docs <- tm_map(docs, removePunctuation)

docs <- tm_map(docs, stripWhitespace)
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)
```

```
##            word freq
## woods     woods    3
## miles     miles    2
## sleep     sleep    2
## ask         ask    1
## chouse   chouse    1
## darkest darkest    1
## easy       easy    1
## gives     gives    1
## little   little    1
## stop       stop    1
```

```
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Blues"))
```

8) Go to the link "https://www.imdb.com/list/ls096735829"and use Selector Gadget , as demonstrated in class to collect data specific to the movie names, the directors of the movies, the movie ratings, and the running times of the movies. Your code should produce the final resulting table given below. Hint: (you may have to first create a data frame, and then convert the data frame to the tibble shown below.

```
Link <- "https://www.imdb.com/list/ls096735829"
page = read_html(Link)
Movies2020 = page%>% html_nodes(".lister-item-header a")%>%
  html_text()

page = read_html(Link)
Directors2020 = page%>% html_nodes(".text-muted a:nth-child(1)")%>%
  html_text()

page = read_html(Link)
Ratings2020 = page%>% html_nodes(".ipl-rating-star.small .ipl-rating-star__rating")%>%
  html_text()

page = read_html(Link)
Runtime2020 = page%>% html_nodes(".runtime")%>%
  html_text()
```

```
moviesdataframe = data.frame(Movies2020,Directors2020, Ratings2020,Runtime2020)

as_tibble(moviesdataframe)
```

```
## # A tibble: 47 x 4
##    Movies2020          Directors2020      Ratings2020 Runtime2020
##    <chr>               <chr>              <chr>       <chr>
##  1 Weathering with You Makoto Shinkai     7.5         112 min
##  2 The Empty Man       David Prior        6.2         137 min
##  3 Monsters of Man     Mark Toia          5.4         131 min
##  4 Songbird            Adam Mason         4.7         84 min
##  5 Tesla               Michael Almereyda  5.1         102 min
##  6 Underwater          William Eubank     5.8         95 min
##  7 Greenland           Ric Roman Waugh    6.4         119 min
##  8 Possessor           Brandon Cronenberg 6.5         103 min
##  9 Tenet               Christopher Nolan  7.3         150 min
## 10 The Phenomenon      James Fox          7.4         100 min
## # ... with 37 more rows
```

9)

a) Use and show R code that shows both column variables x and y, of the diamonds datatable contain the value 4.93. How many times does the number 4.93 appear in each column ?

```
diamonds %>%
  filter(x== 4.93)-> x4.93
x4.93
```

```
## # A tibble: 42 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.44 Ideal     G     SI2      61.1    55   694  4.93  4.95  3.02
##  2  0.43 Premium   H     SI1      59      58   716  4.93  4.96  2.92
##  3  0.41 Ideal     E     SI2      59.8    55   876  4.93  4.81  2.91
##  4  0.46 Ideal     I     VS1      62.3    56   911  4.93  4.96  3.08
##  5  0.43 Premium   H     SI1      60.2    57   919  4.93  4.91  2.96
##  6  0.45 Premium   F     VS2      61.4    60   945  4.93  4.87  3.01
##  7  0.46 Ideal     J     VVS1     61.8    56   953  4.93  4.97  3.06
##  8  0.43 Premium   D     VS2      59.7    59   963  4.93  4.89  2.93
##  9  0.46 Very Good E     SI1      61.8    57   968  4.93  4.98  3.06
## 10  0.45 Ideal     J     VVS1     61.9    56   978  4.93  4.95  3.06
## # ... with 32 more rows
```

```
diamonds %>%
  filter(y== 4.93)-> y4.93

 y4.93
```

```
## # A tibble: 50 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.44 Ideal     H     SI2      62.3    54   654  4.89  4.93  3.06
##  2  0.41 Ideal     E     SI2      59.8    55   683  4.81  4.93  2.91
```

```
##  3  0.43 Very Good H     SI1      60.2    57   716  4.91  4.93  2.96
##  4  0.44 Premium   H     SI2      59.6    60   759  4.95  4.93  2.96
##  5  0.44 Premium   G     SI1      61      60   772  4.88  4.93  2.99
##  6  0.43 Premium   D     SI1      60.1    58   830  4.89  4.93  2.95
##  7  0.44 Ideal     D     SI1      61.6    56   838  4.9   4.93  3.03
##  8  0.43 Ideal     F     VS2      61.4    54   848  4.9   4.93  3.02
##  9  0.43 Very Good G     VS2      59.1    60   867  4.88  4.93  2.9
## 10  0.43 Premium   D     VS2      59.7    59   901  4.89  4.93  2.93
## # ... with 40 more rows
```

```
nrow(x4.93)
```

```
## [1] 42
```

```
nrow(y4.93)
```

```
## [1] 50
```

b) Use and show R code that shows neither column variable x or y, contain the value 3.62.

```
diamonds %>%
  filter(x!=3.62 | y!= 3.62)-> neither
head(neither)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

```
tibble(neither)
```

```
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # ... with 53,930 more rows
```

c) Now show and use R code to find all values that the column variables x and y have in common.

```
diamonds %>%
  filter(x==y & y==x) ->common
common
```

```
## # A tibble: 17 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.3  Ideal     H     VVS2     62.5    54   567  4.3   4.3   2.7
##  2  0.27 Very Good F     VVS1     62      55   591  4.16  4.16  2.59
##  3  1    Very Good H     VS2      63.3    53  5139  0     0     0
##  4  1.14 Fair      G     VS1      57.5    67  6381  0     0     0
##  5  1    Premium   E     VS2      60      60  6600  6.43  6.43  3.89
##  6  1    Premium   E     VS2      60      60  6720  6.43  6.43  3.89
##  7  1.22 Premium   G     SI2      62.4    61  6969  6.79  6.79  4.23
##  8  1.56 Ideal     G     VS2      62.2    54 12800  0     0     0
##  9  1.2  Premium   D     VVS1     62.1    59 15686  0     0     0
## 10  2.25 Premium   H     SI2      62.8    59 18034  0     0     0
## 11  0.32 Ideal     D     VVS2     62.1    54   858  4.4   4.4   2.74
## 12  0.42 Ideal     H     VVS1     62.8    57  1108  4.79  4.79  3.01
## 13  0.61 Premium   G     SI1      60.8    60  1255  5.42  5.42  3.31
## 14  0.48 Ideal     F     VS2      62.4    54  1279  5.03  5.03  3.15
## 15  0.51 Premium   F     SI1      61.4    59  1421  5.13  5.13  3.16
## 16  0.71 Good      F     SI2      64.1    60  2130  0     0     0
## 17  0.71 Good      F     SI2      64.1    60  2130  0     0     0
```