HW: Week 3

36-350 – Statistical Computing

Week 3 – Spring 2021

Name: Jacky Liu

Andrew ID: jackyl1

You must submit **your own** lab as a PDF file on Gradescope.

```
shakespeare.lines = readLines("http://www.andrew.cmu.edu/user/mfarag/shakespeare.txt")
```

Question 1

(10 points)

Display the lines of text in `shakespeare.lines` that contain both of the strings "purse" *and* "gold" (in any order, separated by any amount of text). Do so only using regexp literals.

```
shakespeare.lines[grep("(purse).*(gold)|(gold).*(purse)", shakespeare.lines)]
```

```
## [1] "  HELENA. Take this purse of gold,  "
```

Question 2

(10 points)

Retrieve (but don't display) the lines of text in `shakespeare.lines` that contain "briers". Then break the retrieved lines into individual words (`strsplit(input," ")` to split the character vector `input` into words separated by spaces), and merge those words into a single character vector (`unlist()`!). How many unique words are there? Display the top five most commonly occurring words and how often they occur (combine `sort()` and `table()`!)

```
lines_containing_briers = shakespeare.lines[grep("briers", shakespeare.lines)]
lines_containing_briers = strsplit(lines_containing_briers, " ")
lines_containing_briers = unlist(lines_containing_briers)
sorted = sort(table(lines_containing_briers), decreasing=TRUE)
```

There are 7 unique words.

Top Five most commonly occuring words:

```
sorted[1:5]
```

```
## lines_containing_briers
##            as briers   have leaves
##      4      2      1      1      1
```

Question 3

(10 points)

In Q25 of Lab 3, you coded a regex to match all patterns of the following form: any letter (1 or more), then a punctuation mark, then "ve" or "ll" or "t", then a space or a punctuation mark. You called it `my.pattern`. Use `my.pattern`, along with `regexpr()` and `regmatches()`, to extract and display all the occurrences of the pattern in `shakespeare.lines`. Then repeat the exercise using `gregexpr()` instead of `regexpr`; note that here, you'll want to `unlist()` the output from `regmatches()`. Do you get the same vector of character strings? Why or why not?

```
shakespeare.lines = readLines("http://www.andrew.cmu.edu/user/mfarag/shakespeare.txt")
trial.str.vec =c("we've.", "I'll ", "don't!")
my.pattern = "[A-Za-z]+[[:punct:]](ve|ll|t)( |[[:punct:]])"
grep(my.pattern, trial.str.vec, value=TRUE)
```

```
## [1] "we've." "I'll "  "don't!"
```

```
reg.exp = regexpr(my.pattern, shakespeare.lines)
regmatches(shakespeare.lines, reg.exp)
```

```
##  [1] "on't;"       "Can't "      "is't "       "forswear't;" "in't "
##  [6] "believe't?"  "I'll "       "I'll "       "I'll "       "for't."
## [11] "I'll "       "to't."       "to't,"       "to't."       "there't "
## [16] "call't "     "I'll "       "We'll "      "I'll "       "they'll "
## [21] "I'll "       "I'll "       "I'll "       "I'll "       "in't "
## [26] "I'll "       "I'll "       "I'll "       "you'll "     "you'll "
## [31] "I'll "       "to't;"       "I'll "       "We'll "      "I'll "
## [36] "is't."       "Is't "       "We'll "      "to't;"       "for't."
## [41] "in't,"       "Is't "       "to't,"       "I'll "       "We'll "
## [46] "As't "       "we'll "      "she'll "     "I'll "       "I'll "
## [51] "I'll "       "I'll "       "I'll "       "we'll "      "I'll "
## [56] "I'll "       "I'll "       "I'll "       "I'll "       "think't "
## [61] "in't "       "he'll "      "is't "       "in't,"       "I'll "
## [66] "I'll "       "for't "      "we'll "      "I'll "       "in't,"
## [71] "We'll "      "I'll "       "I'll "       "I'll "       "I'll "
## [76] "We'll "      "they'll "    "I'll "
```

```
greg.exp = gregexpr(my.pattern, shakespeare.lines)
unlist(regmatches(shakespeare.lines, greg.exp))
```

```
##  [1] "on't;"       "Can't "      "is't "       "forswear't;" "in't "
##  [6] "believe't?"  "I'll "       "I'll "       "I'll "       "for't."
## [11] "I'll "       "to't."       "to't,"       "to't."       "there't "
```

```
## [16] "call't "     "I'll "       "We'll "      "I'll "       "they'll "
## [21] "I'll "       "I'll "       "I'll "       "I'll "       "in't "
## [26] "I'll "       "I'll "       "I'll "       "you'll "     "you'll "
## [31] "I'll "       "to't;"       "I'll "       "We'll "      "I'll "
## [36] "is't."       "Is't "       "We'll "      "to't;"       "for't."
## [41] "in't,"       "Is't "       "to't,"       "I'll "       "We'll "
## [46] "As't "       "I'll "       "we'll "      "she'll "     "I'll "
## [51] "I'll "       "I'll "       "I'll "       "I'll "       "we'll "
## [56] "I'll "       "I'll "       "I'll "       "I'll "       "I'll "
## [61] "think't "    "in't "       "he'll "      "is't "       "in't,"
## [66] "I'll "       "I'll "       "for't "      "we'll "      "I'll "
## [71] "in't,"       "We'll "      "I'll "       "I'll "       "I'll "
## [76] "I'll "       "We'll "      "they'll "    "I'll "
```

We do not get the same vector. It is slightly different. This happens because gregexpr gives all matches

Question 4

(10 points)

Come up with a strategy that splits punctuation marks or spaces, except that it keeps intact words like "I've" or "wasn't", that have a punctuation mark in the middle, in between two letters. (Or when the punctuation mark is at the beginning, as in "'em") Apply your strategy to `shakespeare.lines` as defined below such that you display only those words with punctuation marks. (Note that I end up with 704 [not necessarily unique, but total] words when I implement this strategy. Some include'"', which we can easily remove in a subsequent post-processing step if we so choose. Hint: `[[:alnum:]]` is a good thing to use here.)

```
pattern = "([[:punct:]][[:alnum:]]+|[[:alnum:]]+[[:punct:]][[:alnum:]]+)"
reg.exp = regexpr(pattern, shakespeare.lines)
regmatches(shakespeare.lines, reg.exp)
```

```
##   [1] "nature's"       "nature's"       "love's"         "on't"
##   [5] "'a"             "What's"         "'Tis"           "ne'er"
##   [9] "mother's"       "God's"          "What's"         "many-colour"
##  [13] "Can't"          "daughter-in"    "'daughter"      "catch'd"
##  [17] "myst'ry"        "'tis"           "asham'd"        "'tis"
##  [21] "is't"           "forswear't"     "in't"           "appeach'd"
##  [25] "so's"           "lov'd"          "Indian-like"    "riddle-like"
##  [29] "intent-speak"   "prov'd"         "will'd"         "heedfull'st"
##  [33] "approv'd"       "render'd"       "Embowell'd"     "There's"
##  [37] "father's"       "I'd"            "well-lost"      "believe't"
##  [41] "I'll"           "God's"          "to-morrow"      "<THIS"
##  [45] "1990-1993"      "(1"             "(2"             "KING'S"
##  [49] "war-like"       "'tis"           "'Tis"           "well-ent"
##  [53] "monarchy-see"   "[To"            "'Tis"           "'tis"
##  [57] "'Too"           "'t"             "I'll"           "There's"
##  [61] "tortur'd"       "entrench'd"     "Re-enter"       "restrain'd"
##  [65] "receiv'd"       "sword-men"      "[Kneeling"      "I'll"
##  [69] "here's"         "kneel'd"        "ask'd"          "'tis"
##  [73] "That's"         "in's"           "love-line"      "there's"
##  [77] "amaz'd"         "wond'ring"      "I'll"           "Re-enter"
##  [81] "Cressid's"      "On's"           "touch'd"        "father's"
##  [85] "estate-I"       "past-cure"      "call'd"         "thought'st"
```

```
##  [89] "know'st"         "'gainst"         "us'd"            "barr'd"
##  [93] "'tis"            "Hop'st"          "quench'd"        "pilot's"
##  [97] "dar'st"          "strumpet's"      "Traduc'd"        "Sear'd"
## [101] "deserv'd"        "observ'd"        "serv'd"          "resolv'd"
## [105] "cam'st"          "Unquestion'd"    "[Flourish"       "COUNT'S"
## [109] "off's"           "that's"          "barber's"        "Tib's"
## [113] "nun's"           "to't"            "-There"          "-Thick"
## [117] "-Nay"            "whipp'd"         "-Spare"          "'O"
## [121] "'O"              "to't"            "ne'er"           "thing's"
## [125] "-Why"            "KING'S"          "'tis"            "'tis"
## [129] "relinquish'd"    "say-both"        "'tis"            "help'd"
## [133] "'twere"          "what-do"         "[Reading"        "That's"
## [137] "'Fore"           "'tis"            "he's"            "recov'ry"
## [141] "I'll"            "he's"            "'Fore"           "patient's"
## [145] "banish'd"        "repeal'd"        "promis'd"        "O'er"
## [149] "I'd"             "restor'd"        "'We"             "We'll"
## [153] "ames-ace"        "threat'ningly"   "I'd"             "I'll"
## [157] "they'll"         "ne'er"           "There's"         "be'st"
## [161] "[To"             "she's"           "Know'st"         "know'st"
## [165] "father's"        "physician's"     "'Tis"            "pour'd"
## [169] "virtuous-save"   "physician's"     "doer's"          "swell's"
## [173] "she's"           "honour's"        "honour's"        "fore-goers"
## [177] "Debauch'd"       "damn'd"          "honour'd"        "'t"
## [181] "wrong'st"        "restor'd"        "honour's"        "'twere"
## [185] "now-born"        "perform'd"       "lov'st"          "love's"
## [189] "count's"         "thou'rt"         "if-Lord"         "deserv'd"
## [193] "ev'ry"           "Ev'n"            "be'st"           "'He"
## [197] "I'll"            "I'll"            "I'll"            "Re-enter"
## [201] "master's"        "that's"          "I'd"             "Methink'st"
## [205] "I'd"             "conceal'd"       "What's"          "I'll"
## [209] "dog-hole"        "man's"           "There's"         "kicky-wicky"
## [213] "Mars's"          "in't"            "I'll"            "I'll"
## [217] "I'll"            "there's"         "that's"          "'tis"
## [221] "KING'S"          "she's"           "she's"           "she's"
## [225] "she's"           "she's"           "she's"           "man's"
## [229] "master's"        "th'art"          "'Before"         "that's"
## [233] "world's"         "to-night"        "compell'd"       "strew'd"
## [237] "o'erflow"        "What's"          "Strength'ned"    "obtain'd"
## [241] "KING'S"          "sinn'd"          "[To"             "who's"
## [245] "'s"              "[Aside"          "to-night"        "you'll"
## [249] "to-night"        "three-thirds"    "lord's"          "'t"
## [253] "you'll"          "at's"            "procur'd"        "Prepar'd"
## [257] "[Giving"         "'Twill"          "fail'd"          "'tis"
## [261] "<THIS"           "1990-1993"       "(1"              "(2"
## [265] "DUKE's"          "Grace's"         "self-unable"     "guess'd"
## [269] "To-morrow"       "COUNT'S"         "happen'd"        "[Opening"
## [273] "Cupid's"         "E'en"            "[Reads"          "\"not"
## [277] "Re-enter"        "-matter"         "kill'd"          "kill'd"
## [281] "'t"              "'t"              "he's"            "here's"
## [285] "[Reads"          "\"then"          "\"never"         "robb'st"
## [289] "'t"              "[Reads"          "'Tis"            "'Tis"
## [293] "There's"         "well-derived"    "Y'are"           "I'll"
## [297] "'Till"           "is't"            "non-sparing"     "still-piecing"
## [301] "to't"            "'twere"          "roar'd"          "'twere"
```

```
## [305] "'t"              "offic'd"        "I'll"           "DUKE's"
## [309] "We'll"           "COUNT'S"        "[Reads"         "over-night"
## [313] "o'er"            "great'st"       "Duke's"         "let's"
## [317] "I'll"            "[A"             "is't"           "lodg'd"
## [321] "some'er"         "He's"           "'tis"           "What's"
## [325] "examin'd"        "'Tis"           "wheresoe'er"    "pleas'd"
## [329] "arm'd"           "Duke's"         "'tis"           "lov'd"
## [333] "Is't"            "'Tis"           "jack-an"        "he's"
## [337] "He's"            "ring-carrier"   "enjoin'd"       "There's"
## [341] "to-night"        "We'll"          "to't"           "he's"
## [345] "promise-breaker" "lordship's"     "for't"          "in't"
## [349] "Drum's"          "'t"             "Is't"           "blam'd"
## [353] "'hic"            "to't"           "I'll"           "damn'd"
## [357] "man's"           "emboss'd"       "lordship's"     "We'll"
## [361] "smok'd"          "As't"           "she's"          "That's"
## [365] "re-send"         "She's"          "WIDOW'S"        "fall'n"
## [369] "show'd"          "Y'are"          "over-pay"       "Resolv'd"
## [373] "we'll"           "she'll"         "Howe'er"        "I'll"
## [377] "pass'd"          "compos'd"       "'t"             "to-night"
## [381] "let's"           "<THIS"          "1990-1993"      "(1"
## [385] "(2"              "hedge-corner"   "linsey-woolsey" "E'en"
## [389] "adversary's"     "o'clock"        "knock'd"        "e'er"
## [393] "'Came"           "what's"         "butterwoman's"  "Bajazet's"
## [397] "'Twould"         "stripp'd"       "leap'd"         "enemy's"
## [401] "recover'd"       "[Alarum"        "enemy's"        "[They"
## [405] "I'll"            "Kerely-bonto"   "hoodwink'd"     "I'll"
## [409] "I'll"            "'A"             "I'll"           "WIDOW'S"
## [413] "o'that"          "compell'd"      "love's"         "'Tis"
## [417] "vow'd"           "High'st"        "Jove's"         "lov'd"
## [421] "unseal'd"        "holy-cruel"     "ne'er"          "we'll"
## [425] "I'll"            "'longing"       "honour's"       "chastity's"
## [429] "I'll"            "I'll"           "conquer'd"      "deliver'd"
## [433] "I'll"            "in's"           "wife's"         "think't"
## [437] "mother's"        "deliv'red"      "in't"           "tun'd"
## [441] "'tis"            "abhorr'd"       "o'erflows"      "to-night"
## [445] "anatomiz'd"      "accomplish'd"   "confirm'd"      "he'll"
## [449] "acquir'd"        "encount'red"    "cherish'd"      "Where's"
## [453] "King's"          "Here's"         "is't"           "to-night"
## [457] "mourn'd"         "entertain'd"    "deceiv'd"       "[Exeunt"
## [461] "deserv'd"        "confess'd"      "confess'd"      "'a"
## [465] "in't"            "'em"            "'First"         "I'll"
## [469] "All's"           "Y'are"          "militarist-that" "that's"
## [473] "'Five"           "I'll"           "He's"           "for't"
## [477] "'Poor"           "that's"         "truth's"        "'Demand"
## [481] "muster-file"     "that's"         "well-weighing"  "inter'gatories"
## [485] "'a"              "shrieve's"      "Florence's"     "we'll"
## [489] "Duke's"          "'tis"           "[Reads"         "Duke's"
## [493] "I'll"            "in't"           "both-sides"     "[Reads"
## [497] "'When"           "ne'er"          "Count's"        "vow'd"
## [501] "in's"            "he's"           "General's"      "We'll"
## [505] "answer'd"        "'em"            "swine-drunk"    "he's"
## [509] "tragedians-to"   "Mile-end"       "files-I"        "out-villain"
## [513] "he's"            "fee-simple"     "What's"         "What's"
## [517] "E'en"            "I'll"           "[Aside"         "discover'd"
```

```
## [521] "[Unmuffling"     "I'd"             "'t"              "crush'd"
## [525] "'Twould"         "fool'd"          "There's"         "I'll"
## [529] "WIDOW'S"         "wrong'd"         "'tis"            "Tartar's"
## [533] "'Thanks"         "We'll"           "daughter's"      "cozen'd"
## [537] "prepar'd"        "All's"           "Whate'er"        "COUNT'S"
## [541] "snipt-taffeta"   "unbak'd"         "daughter-in"     "advanc'd"
## [545] "'Twas"           "sweet-marjoram"  "sallet-herbs"    "thyself-a"
## [549] "woman's"         "Who's"           "'a"              "there's"
## [553] "talk'st"         "in's"            "they'll"         "flow'ry"
## [557] "look'd"          "'em"             "'a"              "'tis"
## [561] "lady's"          "self-gracious"   "promis'd"        "number'd"
## [565] "deceiv'd"        "to-night"        "Re-enter"        "yonder's"
## [569] "on's"            "'tis"            "liv'ry"          "carbonado'd"
## [573] "there's"         "<THIS"           "1990-1993"       "(1"
## [577] "(2"              "Majesty's"       "fall'n"          "What's"
## [581] "King's"          "remov'd"         "All's"           "I'll"
## [585] "thank'd"         "Whate'er"
```

Below, we read in lines of data from the Advanced National Seismic System (ANSS), on earthquakes of magnitude 6+, between 2002 and 2017. (You don't have to do anything yet.)

```
anss.lines = readLines("http://www.stat.cmu.edu/~mfarag/350/anss.htm")
date.pattern = "[0-9]{4}/[0-9]{2}/[0-9]{2}"
date.lines = grep(date.pattern,anss.lines,value=TRUE)
```

Question 5

(10 points)

Check that all the lines in `date.lines` actually start with a date, of the form YYYY/MM/DD. rather than contain a date of this form somewhere in the middle of the text. (Hint: it might help to note that you can look for non-matches, as opposed to matches, by changing one of `grep()`'s logical arguments.)

```
# This function checks each line to see if they start with a date and prints any lines that don't.
# Returns TRUE if all lines start with a proper date, otherwise returns FALSE.
checkdate = function(text){
  for(line in date.lines){
    if(!grepl("^[0-9]{4}/[0-9]{2}/[0-9]{2}", line)){
      print(line)
      return(FALSE)
    }
  }
  return(TRUE)
}

checkdate(date.lines)
```

```
## [1] TRUE
```

Question 6

(10 points)

Which five days witnessed the most earthquakes, and how many were there, these days? Also, what happened on the day with the most earthquakes: can you find any references to this day in the news?

```
date.pattern = "[0-9]{4}/[0-9]{2}/[0-9]{2}"
date.lines = grep(date.pattern, anss.lines, value=TRUE)
reg.exp = regexpr(date.pattern, date.lines)
head(sort(table(regmatches(date.lines, reg.exp)), decreasing=TRUE), n=5)
```

```
##
## 2011/03/11 2010/02/27 2004/12/26 2006/11/15 2013/02/06
##         42         12         11          7          7
```

The five days are 2011/03/11, 2010/02/27, 2004/12/26, 2006/11/15, and 2013/02/06 with 42, 12, 11, 7, and 7 earthquakes respectively. On the day with the most earthquake was the Tohoku Earthquake and Tsunami.

Question 7

(10 points)

Go back to the data in `date.lines`. Following steps similar to the ones you used in the lab to extract the latitude and longitude of earthquakes, extract the depth and magnitude of earthquakes. In the end, you should have one numeric vector of depths, and one numeric vector of magnitudes. Show the first three depths and the first three magnitudes. (Hint: if you use `regexpr()` and `regmatches()`, then the output from the latter will be a vector of strings. Look at this vector. The last four characters always represent the magnitudes. Use a combination of `substr()` and `as.numeric()` to create the numeric vector of magnitudes. Then use the fact that everything but the last four characters represents the depths. There are a myriad of ways to do this exercise, but this suggested way is the most concise.)

```
thepattern = "[0-9]+(\\.[0-9][0-9])  [6-9](\\.[0-9][0-9])"
#date.lines = grep(date.pattern, anss.lines, value=TRUE)
reg.exp = regexpr(thepattern, date.lines)
regmatches(date.lines, reg.exp)[1:3]
```

```
## [1] "10.00  6.00"  "138.10  6.30" "665.80  6.20"
```

First three depths are 10.00, 138.10, and 665.80 as shown above. First three magnitudes are 6.00, 6.30, and 6.20 as shown above. —

Here we read in text containing the fastest men's 100-meter sprint times. We retain only the lines that correspond to the sprint data, for times 9.99 seconds or better.

```
sprint.lines = readLines("http://www.stat.cmu.edu/~mfarag/350/men_100m.html")
data.lines = grep(" +(9|10)\\.",sprint.lines)
sprint.lines = sprint.lines[min(data.lines):max(data.lines)]
```

Question 8

(10 points)

Extract the years in which the sprint times were recorded. Display them in table format. Do the same for the months. Be sure to extract the month of the sprint, not the birth month of the sprinter! (Hint: the month of the sprint is followed by a four-digit year; other instances of two digits in any given line are not. So you may have to extract more than you need, then apply `strsplit()`.)

```
pattern = "([0-9]{2}\\.[0-9]{2}\\.[2-3][0-9]{3})"
reg.exp = regexpr(pattern, sprint.lines)
recorded_dates = regmatches(sprint.lines, reg.exp)
#recorded_dates
years = vector(mode = "character", length = length(recorded_dates))
for(i in 1:length(recorded_dates)){
  years[i] = unlist(strsplit(recorded_dates[i], '\\.'))[3]
}
table(years)
```

```
## years
## 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014
2015
## 14 16 16 9 33 16 24 18 53 45 36 57 61 59 32 91
## 2016 2017 2018 2019
## 60 48 57 50
```

```
months = vector(mode = "character", length = length(recorded_dates))
for(i in 1:length(recorded_dates)){
  months[i] = unlist(strsplit(recorded_dates[i], '\\.'))[2]
}
table(months)
```

```
## months
##  02  03  04  05  06  07  08  09  10
##   1   6  20  90 207 179 205  85   2
```

Question 9

(10 points)

Extract the countries of origin (for the sprinters). Note that countries of origin are given as a capitalized three-letter abbreviation. Display the table of country origins. Display the proportion of the list that is accounted for by sprinters from the US and Jamaica.

```
pattern = "([A-Z][A-Z][A-Z])"
reg.exp = regexpr(pattern, sprint.lines)
countries = regmatches(sprint.lines, reg.exp)
table(countries)
```

```
## countries
## AHO ANT AUS BAH BAR CAN CAY CHN CIV CUB FRA GBR GHA ITA JAM JPN NAM NED NGR
NOR
```

```
## 5 7 1 3 4 37 1 7 11 2 30 35 4 1 267 4 27 1 25 1
## OMA POR QAT RSA SKN TTO TUR USA ZIM
## 1 4 8 29 10 51 3 406 3
```

Proportion from US and Jamaica:

```
(406+267)/sum(table(countries))
```

```
## [1] 0.6811741
```

Question 10

(10 points)

We conclude with a web scraping exercise. I want you to go to this web site. On it, you see there is a set of 12 bold-faced four-digit numbers: this is the number of submitted astrophysics articles for each month of 2019. I want you to extract these numbers and place them into a single vector, with each vector element having a name: Jan for the first vector element, Feb for the second, etc. You would use `readLines()` to read in the page (pass the URL directly to the function!); this creates a vector of strings. You would then use `regexpr()` and `regmatches()` to extract the numbers (plus potentially some other stuff that you may have to pare off using `substr()`). If necessary, use "view source" to look at the html code for the web page itself to determine how best to extract the 12 numbers and nothing else. You don't want to create a table; you simply want to output the vector of four-digit numbers and add the appropriate names. (Hint: see the documentation for `Constants`. `month.abb` might be helpful here.)

```
astro = readLines("https://arxiv.org/year/astro-ph/19")
pattern = "(<b>[0-9]{4}</b>)"
reg.exp = regexpr(pattern, astro)
counts = regmatches(astro, reg.exp)
counts = substr(counts,4,7)
names(counts) = month.abb
counts
```

```
##    Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov
## "1125" "1038" "1475" "1258" "1159" "1023" "1277" "1225" "1344" "1295" "1100"
##    Dec
## "1100"
```

Question 11

(10 points)

Make a string with "Regards," and then your first name, separated by a newline character. Print it to the console via `print()` and then via `cat()`. Comment on the difference.

```
s = "Regards,\nJacky"
print(s)
```

```
## [1] "Regards,\nJacky"
```

```r
cat(s)
```

```
## Regards,
## Jacky
```

```
print does not treat the \ in \n as an escape and prints \n while cat doesn't.
```

Question 12

(10 points)

Transform the vector so that the words have all uppercase characters.

```r
toupper(s)
```

```
## [1] "REGARDS,\nJACKY"
```

```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.1
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
ggplot(data = mtcars, aes(x = qsec)) + geom_boxplot(fill="red") + ggtitle("Distribution of 1/4 Mile Time
```