Class Notes 9

Tobias Boggess

4/6/2022

Section 19.1 Exercises

Exercise 1: paste() combines two (or more) character strings together into one character string. Try the following (taken from the paste() help page) and report the results:

```
paste("Today's date is", date())
```

a) paste("Today's date is", date())

```
## [1] "Today's date is Mon Apr 11 19:23:59 2022"
```

```
paste("X", 1:5, sep = "")
```

b) paste("X", 1:5, sep = " ")

```
## [1] "X1" "X2" "X3" "X4" "X5"
```

Exercise 2: This exercise concerns paste() and it's opposite, strsplit().

a) paste() combines character strings. Consider the following three character strings.Write a command involving paste() that returns the single character string.
Code:

```
first <- "Louis"
middle <- "Daniel"
last <- "Armstrong"

paste(first, middle, last, sep = " ")
```

```
## [1] "Louis Daniel Armstrong"
```

b) paste() is vectorized. Consider the following three "character" vectors. Write a command involving paste() that returns the single character string
Code:

```
first <- c("Louis", "John", "Miles", "Ella")
middle <- c("Daniel", "William", "Dewey", "Jane")
last <- c("Armstrong", "Coltrane", "Davis", "Fitzgerald")

paste(first, middle, last, sep = " ")
```

```
## [1] "Louis Daniel Armstrong" "John William Coltrane"  "Miles Dewey Davis"
## [4] "Ella Jane Fitzgerald"
```

c) strsplit() does the opposite of paste() – it splits a character string. Consider the following character string. Write a command involving strsplit() that returns the three character strings
Code:

```
full <- "Sarah Lois Vaughan"

strsplit(full, split = " ")
```

```
## [[1]]
## [1] "Sarah"   "Lois"    "Vaughan"
```

d) strsplit() is vectorized. Consider the following "character" vector. Write a command involving strsplit() that returns the three vectors of character strings, Report your R command.
Code:

```
full <-
  c(
    "Sarah Lois Vaughan",
    "Thelonious Sphere Monk",
    "Chet Henry Baker",
    "Wynton Learson Marsalis"
  )

strsplit(full, split = " ")
```

```
## [[1]]
## [1] "Sarah"   "Lois"    "Vaughan"
##
## [[2]]
## [1] "Thelonious" "Sphere"     "Monk"
##
## [[3]]
## [1] "Chet"  "Henry" "Baker"
##
## [[4]]
## [1] "Wynton"   "Learson"  "Marsalis"
```

Exercise 3: Do the following.

Setup:

```
quote <- "Aunt Petunia was horse-faced and bony; Dudley was blond, pink, and
porky. Harry, on the other hand, was small and skinny, with brilliant green
eyes and jet-black hair that was always untidy. He wore round glasses, and on
his forehead was a thin, lightning-shaped scar."


quote <- gsub(pattern = "\n",
              replacement = " ",
              x = quote)
quote <- gsub(pattern = "[\\.,;]",
              replacement = "",
              x = quote)
```

a) gregexpr() searches a character string for a pattern. Search quote for hyphens (-) by running the following command. What do the three values returned by gregexpr(), 23, 156, and 256, represent?
Code:

```
gregexpr(pattern = "-", text = quote)
```

```
## [[1]]
## [1]  23 149 246
## attr(,"match.length")
## [1] 1 1 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

The numbers represent where in the text the character '-' we are looking for is located.

b) regexpr() is another way to search a character string for a pattern. Run the following command.What does the one value returned by gregexpr(), 23, represent?
Code:

```
regexpr(pattern  = "-", text = quote)
```

```
## [1] 23
## attr(,"match.length")
## [1] 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

The 23 represents the first time in the string object quote that has the character '-'.

Exercise 4: Do the following.

Quote:

```
badges <- "Badges? We ain't got no badges. We don't need no badges. I don't have to show you any stinki
```

a) Use tolower() to convert the quote to all lower case, overwriting the previous version of badges.
Code:

```
badges <- tolower(badges)
badges
```

```
## [1] "badges? we ain't got no badges. we don't need no badges. i don't have to show you any stinking l
```

b) We want to "clean up" the quote a bit by removing the punctuation marks (periods, question mark, and exclamation marks). Use gsub(), with pattern = "!" and replacement = "", to remove the exclamation mark from badges, overwriting the previous version of badges. Report your R command.
Code:

```
badges <- gsub(x = badges, pattern = "!", replacement = "")
badges
```

```
## [1] "badges? we ain't got no badges. we don't need no badges. i don't have to show you any stinking l
```

c) Remove the question mark and periods, overwriting the previous version of badges.
Code:

```
badges <- gsub(pattern = "\\?", replacement = "", x = badges)
badges
```

```
## [1] "badges we ain't got no badges. we don't need no badges. i don't have to show you any stinking ba
```

```
badges <- gsub(pattern = "\\.", replacement = "", x = badges)
badges
```

```
## [1] "badges we ain't got no badges we don't need no badges i don't have to show you any stinking badg
```

d) Now use strsplit(), with split = " ", to split the badges quote into individual words, overwriting the previous version of badges.
Code:

```
badges <- strsplit(x = badges, split = " ")
badges
```

```
## [[1]]
##  [1] "badges"   "we"       "ain't"    "got"      "no"       "badges"
##  [7] "we"       "don't"    "need"     "no"       "badges"   "i"
## [13] "don't"    "have"     "to"       "show"     "you"      "any"
## [19] "stinking" "badges"
```

e) Note that strsplit() returned a list with one element, which is a "character" vector of words from the quote. Now use grep(), with pattern = "badges", to find the instances of the word "badges" in the quote. Then do the same thing, but using grepl(). Report your two R commands.
Code:

```
badges <- badges[[1]]
badges
```

```
##  [1] "badges"   "we"       "ain't"    "got"      "no"       "badges"
##  [7] "we"       "don't"    "need"     "no"       "badges"   "i"
## [13] "don't"    "have"     "to"       "show"     "you"      "any"
## [19] "stinking" "badges"
```

```
grep(pattern = "badges", x = badges)
```

```
## [1]  1  6 11 20
```

```
grepl(pattern = "badges", x = badges)
```

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

f) Now use nchar() to count the number of letters (characters actually) in each word. Report your R command.
Code:

```
nchar(badges)
```

```
##  [1] 6 2 5 3 2 6 2 5 4 2 6 1 5 4 2 4 3 3 8 6
```

Exercise 5: Do the following.

Setup:

```
quote <- "I have a dream that one day this nation will rise up and live out
the true meaning of its creed, 'We hold these truths to be self-evident,
that all men are created equal.' I have a dream that one day on the
red hills of Georgia, sons of former slaves and the sons of former
slave owners will be able to sit down together at the table of
brotherhood. I have a dream that one day even the state of Mississippi,
a state sweltering with the heat of injustice, sweltering with the
heat of oppression, will be transformed into an oasis of freedom and
justice. I have a dream that my four little children will one day live
in a nation where they will not be judged by the color of their skin
but by the content of their character."

quote <- gsub(pattern = "\n", replacement = " ", x = quote)
quote <- gsub(pattern = "[\\.,']", replacement = "", x = quote)
```

a) Now use strsplit(), with split = ” “, to split the quote into individual words, overwriting the previous version of quote.
Code:

```
quote <- strsplit(quote, split = " ")
quote
```

```
## [[1]]
##   [1] "I"           "have"        "a"           "dream"       "that"
##   [6] "one"         "day"         "this"        "nation"      "will"
##  [11] "rise"        "up"          "and"         "live"        "out"
##  [16] "the"         "true"        "meaning"     "of"          "its"
##  [21] "creed"       "We"          "hold"        "these"       "truths"
##  [26] "to"          "be"          "self-evident" "that"       "all"
##  [31] "men"         "are"         "created"     "equal"       "I"
##  [36] "have"        "a"           "dream"       "that"        "one"
##  [41] "day"         "on"          "the"         "red"         "hills"
##  [46] "of"          "Georgia"     "sons"        "of"          "former"
##  [51] "slaves"      "and"         "the"         "sons"        "of"
##  [56] "former"      "slave"       "owners"      "will"        "be"
##  [61] "able"        "to"          "sit"         "down"        "together"
##  [66] "at"          "the"         "table"       "of"          "brotherhood"
##  [71] "I"           "have"        "a"           "dream"       "that"
##  [76] "one"         "day"         "even"        "the"         "state"
##  [81] "of"          "Mississippi" "a"           "state"       "sweltering"
##  [86] "with"        "the"         "heat"        "of"          "injustice"
##  [91] "sweltering"  "with"        "the"         "heat"        "of"
##  [96] "oppression"  "will"        "be"          "transformed" "into"
## [101] "an"          "oasis"       "of"          "freedom"     "and"
## [106] "justice"     "I"           "have"        "a"           "dream"
## [111] "that"        "my"          "four"        "little"      "children"
## [116] "will"        "one"         "day"         "live"        "in"
## [121] "a"           "nation"      "where"       "they"        "will"
## [126] "not"         "be"          "judged"      "by"          "the"
## [131] "color"       "of"          "their"       "skin"        "but"
## [136] "by"          "the"         "content"     "of"          "their"
## [141] "character"
```

b) Note that strsplit() returned a list with one element, which is a "character" vector of words from the quote.Now use nchar() to obtain a vector, my.nchars, say, containing character counts for in the words of quote. Report your R command(s).
Code:

```
quote <- quote[[1]]

head(quote)
```

```
## [1] "I"     "have"  "a"     "dream" "that"  "one"
```

```
length(quote)
```

```
## [1] 141
```

```
my.nchars <- nchar(quote)
my.nchars
```

```
##   [1]  1  4  1  5  4  3  3  4  6  4  4  2  3  4  3  3  4  7  2  3  5  2  4  5  6
##  [26]  2  2 12  4  3  3  3  7  5  1  4  1  5  4  3  3  2  3  3  5  2  7  4  2  6
##  [51]  6  3  3  4  2  6  5  6  4  2  4  2  3  4  8  2  3  5  2 11  1  4  1  5  4
##  [76]  3  3  4  3  5  2 11  1  5 10  4  3  4  2  9 10  4  3  4  2 10  4  2 11  4
## [101]  2  5  2  7  3  7  1  4  1  5  4  2  4  6  8  4  3  3  4  2  1  6  5  4  4
## [126]  3  2  6  2  3  5  2  5  4  3  2  3  7  2  5  9
```

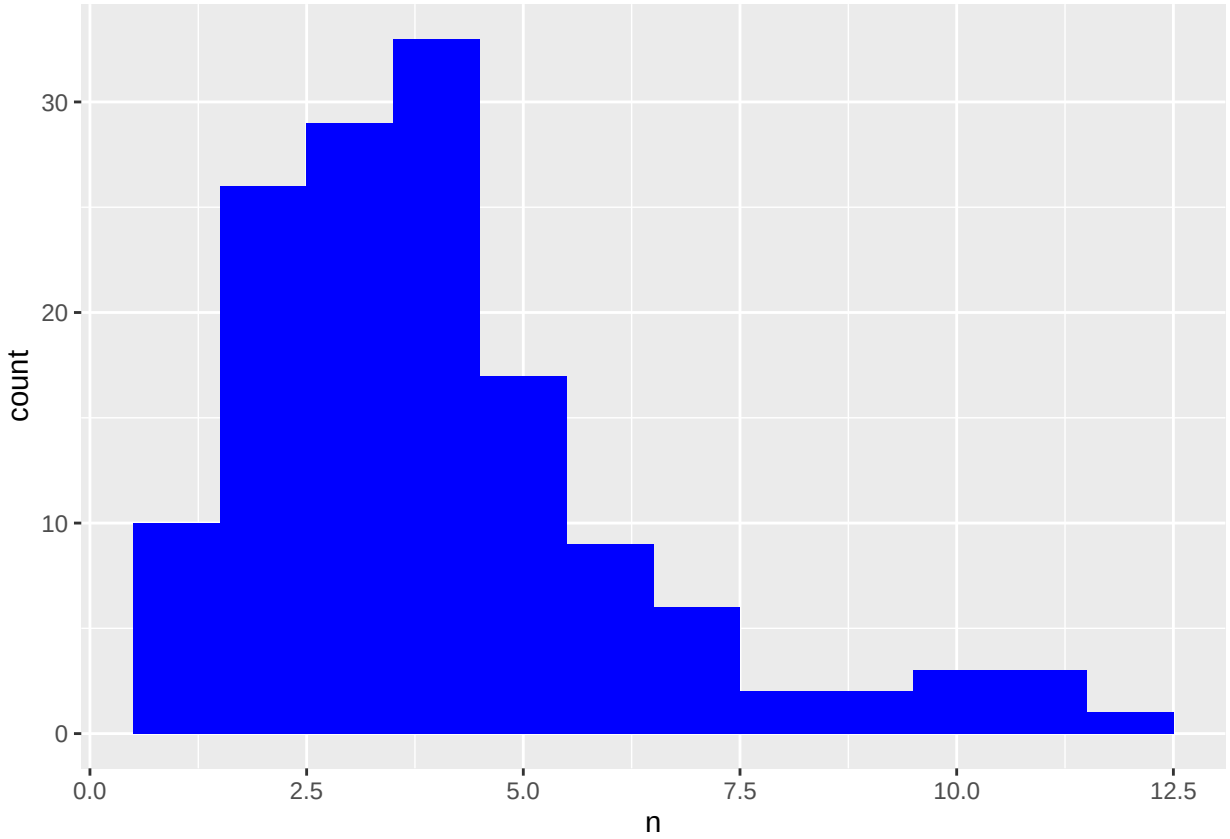
c) Use mean() to determine the mean number of characters of the words in quote. Report the value of the mean.
Code:

```
mean.nchars <- mean(my.nchars)
mean.nchars
```

```
## [1] 4.049645
```

d) Make a histogram of the numbers of characters of the words in quote. Describe the shape of the histogram (right skewed, left skewed, or symmetrical and bell-shaped).
Code:

```
library(ggplot2)
ggplot(data = data.frame(n = my.nchars)) +
  geom_histogram(mapping = aes(x = n),
                 fill = "blue",
                 binwidth = 1)
```



The shape is a right skewed bell-shaped histogram.

Section 19.2 Exercises

Exercise 6: Do the following.

Setup:

```
quote <- "I have a dream that one day this nation will rise up and live out
the true meaning of its creed, 'We hold these truths to be self-evident,
that all men are created equal.' I have a dream that one day on the
red hills of Georgia, sons of former slaves and the sons of former
slave owners will be able to sit down together at the table of
brotherhood. I have a dream that one day even the state of Mississippi,
a state sweltering with the heat of injustice, sweltering with the
heat of oppression, will be transformed into an oasis of freedom and
justice. I have a dream that my four little children will one day live
in a nation where they will not be judged by the color of their skin
but by the content of their character."

quote <- gsub(pattern = "\n", replacement = " ", x = quote)

quote <- gsub(pattern = "[\\.,']", replacement = "", x = quote)

quote
```

```
## [1] "I have a dream that one day this nation will rise up and live out the true meaning of its creed
```

a) Use gregexpr() to determine the starting character position(s) of the word "freedom" in the quote. Report your R command(s).
Code:

```
gregexpr(pattern = "freedom", text = quote)
```

```
## [[1]]
## [1] 524
## attr(,"match.length")
## [1] 7
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

b) strsplit() splits the elements of a "character" string into substrings according to matches of a character pattern (or regular expression) specified via the argument split, and returns a list with one element, a "character" vector of the substrings. Run the following commands, and report the results
Code:

```
quote.list <- strsplit(quote, split = " ")
quote.vec <-
  unlist(quote.list) # Could also use quote.vec <- quote.list[[1]]
quote.vec
```

```
##   [1] "I"           "have"        "a"           "dream"       "that"
##   [6] "one"         "day"         "this"        "nation"      "will"
##  [11] "rise"        "up"          "and"         "live"        "out"
##  [16] "the"         "true"        "meaning"     "of"          "its"
##  [21] "creed"       "We"          "hold"        "these"       "truths"
##  [26] "to"          "be"          "self-evident" "that"       "all"
##  [31] "men"         "are"         "created"     "equal"       "I"
##  [36] "have"        "a"           "dream"       "that"        "one"
##  [41] "day"         "on"          "the"         "red"         "hills"
##  [46] "of"          "Georgia"     "sons"        "of"          "former"
##  [51] "slaves"      "and"         "the"         "sons"        "of"
##  [56] "former"      "slave"       "owners"      "will"        "be"
##  [61] "able"        "to"          "sit"         "down"        "together"
##  [66] "at"          "the"         "table"       "of"          "brotherhood"
##  [71] "I"           "have"        "a"           "dream"       "that"
##  [76] "one"         "day"         "even"        "the"         "state"
##  [81] "of"          "Mississippi" "a"           "state"       "sweltering"
##  [86] "with"        "the"         "heat"        "of"          "injustice"
##  [91] "sweltering"  "with"        "the"         "heat"        "of"
##  [96] "oppression"  "will"        "be"          "transformed" "into"
## [101] "an"          "oasis"       "of"          "freedom"     "and"
## [106] "justice"     "I"           "have"        "a"           "dream"
## [111] "that"        "my"          "four"        "little"      "children"
## [116] "will"        "one"         "day"         "live"        "in"
## [121] "a"           "nation"      "where"       "they"        "will"
## [126] "not"         "be"          "judged"      "by"          "the"
## [131] "color"       "of"          "their"       "skin"        "but"
## [136] "by"          "the"         "content"     "of"          "their"
## [141] "character"
```

c) grep() takes arguments pattern, a character pattern (or regular expression), and x, a "character" vector, and returns the indices of the elements of x that contain the pattern. Explain why the returned values of the following two commands differ:
Code:

```
grep(pattern = "the", x = quote.vec)
```

```
##  [1]  16  24  43  53  65  67  70  79  87  93 124 130 133 137 140
```

```
which(quote.vec == "the")
```

```
## [1]  16  43  53  67  79  87  93 130 137
```

The command *grep(pattern = "the", x = quote.vec)* will find subsets of string that contain the word " **the**" and will return the location of that substring or string in the vector. The *which* command will show only the strings that are exactly the same as " **the**" but it wont find any words containing a subset of the word " **the**".

d) If we specify value = TRUE in grep(), character pattern (or regular expression) for pattern, and a "character" vector x, it returns the actual elements of x (not their indices) that contain the pattern. Explain in words what each of the following commands does
Code:

```
grep(pattern = "ing$", x = quote.vec, value = TRUE)
```

```
## [1] "meaning"    "sweltering" "sweltering"
```

```
grep(pattern = "^th", x = quote.vec, value = TRUE)
```

```
##  [1] "that"  "this"  "the"   "these" "that"  "that"  "the"   "the"   "the"
## [10] "that"  "the"   "the"   "the"   "that"  "they"  "the"   "their" "the"
## [19] "their"
```

The first command will return all the words that contain the ending "-ing". The second command will return all the words that contain words beginning with "th".

Exercise 7: Do the following.

Setup:

```
County1 <-
  c(
    "De Witt County",
    "Lac qui Parle County",
    "Lewis and Clark County",
    "St John the Baptist Parish"
  )
County2 <-
  c(
    "De Witt County",
    "Lac qui Parle County",
    "Lewis and Clark County",
    "St. John the Baptist Parish"
  )
County3 <-
  c("De Witt", "Lac Qui Parle", "Lewis & Clark", "St. John the Baptist")
```

a) Find and eliminate the word "County" from the County1 and County2 vectors. Consider using sub() or gsub(). Report your R command(s).
Code:

```
County1_2 <- gsub(pattern = "Country", replacement = "", x = County1)
County1_2
```

```
## [1] "De Witt County"             "Lac qui Parle County"
## [3] "Lewis and Clark County"     "St John the Baptist Parish"
```

```
County2_2 <- gsub(pattern = "County", replacement = "", x = County2)
County2_2
```

```
## [1] "De Witt "                    "Lac qui Parle "
## [3] "Lewis and Clark "            "St. John the Baptist Parish"
```

b) Now find and eliminate the word "Parish" from the County1 and County2 vectors. Use sub() or gsub(). Report your R command(s).
Code:

```
County1_2 <- gsub(pattern = "Parish", replacement = "", x = County1_2)
County1_2
```

```
## [1] "De Witt County"         "Lac qui Parle County"   "Lewis and Clark County"
## [4] "St John the Baptist "
```

```
County2_2 <- gsub(pattern = "Parish", replacement = "", x = County2_2)
County2_2
```

```
## [1] "De Witt "              "Lac qui Parle "        "Lewis and Clark "
## [4] "St. John the Baptist "
```

c) The words "County" and "Parish" both have six letters. So another way to eliminate those words from County1 and County2 would be to use nchar() to find the length of each character string and then substr() to keep all but the last 6 characters (or 7?). Verify that the following commands are another way to accomplish the tasks of parts a and b
Code:

```
substr(x = County1, start = 1, stop = nchar(County1) - 7)
```

```
## [1] "De Witt"             "Lac qui Parle"       "Lewis and Clark"
## [4] "St John the Baptist"
```

```
substr(x = County2, start = 1, stop = nchar(County2) - 7)
```

```
## [1] "De Witt"              "Lac qui Parle"        "Lewis and Clark"
## [4] "St. John the Baptist"
```

d) Another way to eliminate "County" and "Parish" words from County1 and County2 would be to use the "or" operator | in a regular expression.Verify that the following commands are another way to accomplish the tasks of parts a and b:
Code:

```
gsub(pattern = " County| Parish",
     replacement = "",
     x = County1)
```

```
## [1] "De Witt"             "Lac qui Parle"       "Lewis and Clark"
## [4] "St John the Baptist"
```

```
gsub(pattern = " County| Parish",
     replacement = "",
     x = County2)
```

```
## [1] "De Witt"              "Lac qui Parle"        "Lewis and Clark"
## [4] "St. John the Baptist"
```

e) In part d, there was a blank before County and Parish. Why wouldn't the following be appropriate?
Code:

```
gsub(pattern = " County|Parish", replacement = "", x = County1)
```

```
## [1] "De Witt"              "Lac qui Parle"        "Lewis and Clark"
## [4] "St John the Baptist "
```

```
gsub(pattern = " County|Parish", replacement = "", x = County2)
```

```
## [1] "De Witt"               "Lac qui Parle"         "Lewis and Clark"
## [4] "St. John the Baptist "
```

The above wouldn't be appropriate because there is an extra space still in the quote before Parish and since there are spaces already in the text in the appropriate places, there would be double spaces where Parish would be.

f) Next we want to remove the period symbol in "St." from County2 and from County3. The regular expression "." matches any character. Why wouldn't the following be appropriate? Try it.
Code:

```
gsub(pattern = ".", replacement = "", x = County2)
```

```
## [1] "" "" "" ""
```

```
gsub(pattern = ".", replacement = "", x = County3)
```

```
## [1] "" "" "" ""
```

This isn't appropriate because the '.' isn't getting removed from County 2 or County 3. The period on its own represents any character. So in the code above, we are essentially removing all letters in County2 and County3.

g) Write a command involving a regular expression with the escape character () to remove the period symbol in "St." from County2 and from County3. Report your R command(s).
Code:

```
gsub(pattern = "\\.", replacement = "", x = County2)
```

```
## [1] "De Witt County"             "Lac qui Parle County"
## [3] "Lewis and Clark County"     "St John the Baptist Parish"
```

```
gsub(pattern = "\\.", replacement = "", x = County3)
```

```
## [1] "De Witt"             "Lac Qui Parle"       "Lewis & Clark"
## [4] "St John the Baptist"
```

h) Setting fixed = TRUE in gsub() applies literal pattern matching rather than treating the contents of pattern as metacharacters. Instead of using a regular expression with the escape character () to remove the period symbol in "St." from County1 and from County2, as in part g, rewrite the command from part f using fixed = TRUE in gsub() to accomplish the task. Report your R command.
Code:

```
gsub(pattern = ".", replacement = "", x = County2, fixed = TRUE)
```

```
## [1] "De Witt County"              "Lac qui Parle County"
## [3] "Lewis and Clark County"      "St John the Baptist Parish"
```

```
gsub(pattern = ".", replacement = "", x = County3, fixed = TRUE)
```

```
## [1] "De Witt"              "Lac Qui Parle"       "Lewis & Clark"
## [4] "St John the Baptist"
```

Exercise 8: What does the regular expression "<.*>" match in my.string above?

Setup:

```
my.string <- "<p> This is a short paragraph.</p>"
gregexpr(pattern = "<.*>", text = my.string)
```

```
## [[1]]
## [1] 1
## attr(,"match.length")
## [1] 34
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

The regular expression "<.*>" matches the first character expressed in the my.string object which is '<'. Since it considers the whole string to be a single value, it will detect there are 34 characters in the string.

Section 19.5 Exercises

Exercise 9: Do the following.

Code:

```
my.file <- file.choose()

sotu.wrd.vec <- scan(my.file,
                     what = "",
                     blank.lines.skip = TRUE)

# This creates a single "character" string (one-element vector) containing all speeches.
sotu.string <- paste(sotu.wrd.vec, collapse = " ")

# This splits sotu.string into separate speeches (demarcated by ***):
sotu.list <- strsplit(sotu.string, split = "\\*\\*\\*")

# This converts the one-element sotu.list to a vector:
sotu.spch.vec <- unlist(sotu.list)
# Could also use sotu.spch.vec <- sotu.list[[1]]

# This removes the empty first element:
sotu.spch.vec <- sotu.spch.vec[-1]

# Create a "VectorSource" class object:
sotu.vecsrc <- VectorSource(sotu.spch.vec)

# Convert to a "Corpus" class object:
sotu.corp <- VCorpus(sotu.vecsrc)

# This cleans the speeches:
sotu.corp <- sotu.corp %>%
  tm_map(FUN = stripWhitespace) %>%
  tm_map(FUN = removeNumbers) %>%
  tm_map(FUN = removePunctuation) %>%
  tm_map(FUN = content_transformer(tolower)) %>%
  tm_map(FUN = removeWords, stopwords("english"))

dtm <- DocumentTermMatrix(sotu.corp)
```

a) Which words were used 3,000 or more times in the speeches?
Code:

```
findFreqTerms(dtm, lowfreq = 3000)
```

```
##  [1] "can"        "congress"   "country"    "government" "great"
##  [6] "made"       "may"        "must"       "people"     "states"
## [11] "united"     "upon"       "will"       "year"
```

```
dtm %>% as.matrix() %>% apply(MARGIN = 2, sum) %>% sort(decreasing = TRUE) %>% head(n = 15)
```

```
##       will government     states   congress     united        can     people
##       9056       6505       6229       4808       4611       4179       3759
##       upon       year        may    country      great       must       made
##       3738       3495       3288       3206       3130       3111       3010
##     public
##       2985
```

b) With which five words does the word "peace" tend to appear most often in the speeches (i.e. which five words' frequencies are most correlated with "peace"'s frequencies?). What are the values of their correlations with "peace"?
Code:

```
findAssocs(dtm, terms = "peace", corlimit = 0.5)
```

```
## $peace
##           war      military          army        forces contributions
##          0.59          0.56          0.54          0.54          0.52
##       nations
##          0.50
```

The five most common words correlated with "peace" are "war", "military", "army", "forces", and "contributions". The most correlated word with peace is war at 0.59 while the least of the top five words correlated to peace is contributions with 0.52.