Homework 9

Tobias Boggess

2022-04-23

Chapter 13 Problems

Problem 6: Sally and Joan plan to meet to study in their college campus center. They are both impatient people who will only wait 10 minutes for the other before leaving. Rather than pick a specific time to meet, they agree to head over to the campus center sometime between 7:00 and 8:00 pm. Let both arrival times be normally distributed with mean 30 minutes past and a standard deviation of 10 minutes. Assume that they are independent of each other. What is the probability that they actually meet? Estimate the answer using simulation techniques introduced in this chapter, with at least 10,000 simulations.

Code:

```
set.seed(23)
sim.cllg_cent <- tibble(
  Sally = rnorm(n = 10000, mean = 30, sd = 10),
  Joan = rnorm(n = 10000, mean = 30, sd = 10),
  results = ifelse(
    abs(Sally - Joan) <= 10, "Meeting", "No meeting"
  )
)

tally(~ results, format = "percent", data = sim.cllg_cent)
```

```
## results
##    Meeting No meeting
##      52.29      47.71
```

Sally and Joan will only meet approximately 52.29% of the time and will not meet within 47.71% of the time.

Problem 9: Generate n = 5,000 observations from a logistic regression model with parameters intercept $\beta_0 = -1$, slope $\beta_1 = 0.5$, and distribution of the predictor being normal with mean 1 and standard deviation 1. Calculate and interpret the resulting parameter estimates and confidence intervals.

Code:

```
set.seed(54)

x <- rnorm(n = 5000, mean = 1, sd = 1)

true_probs <- exp(-1 + 0.5 * x) / (1 + exp(-1 + 0.5 * x))

y <- rbinom(n = 5000, size = 1, prob = true_probs)

sim.data <- data.frame(X = x, Y = y)

sim.data.reg <- glm(Y ~ X, data = sim.data)

sim.data.reg
```

```
##
## Call:  glm(formula = Y ~ X, data = sim.data)
##
## Coefficients:
## (Intercept)            X
##      0.2838       0.1086
##
## Degrees of Freedom: 4999 Total (i.e. Null);  4998 Residual
## Null Deviance:       1193
## Residual Deviance: 1133  AIC: 6771
```

```
summary(sim.data.reg)
```

```
##
## Call:
## glm(formula = Y ~ X, data = sim.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7575  -0.3969  -0.2753   0.5391   0.9192
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.283848   0.009497   29.89   <2e-16 ***
## X           0.108618   0.006678   16.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2266083)
##
##     Null deviance: 1192.5  on 4999  degrees of freedom
## Residual deviance: 1132.6  on 4998  degrees of freedom
## AIC: 6770.7
##
## Number of Fisher Scoring iterations: 2
```

The interval for the parameters is:
$CI = (0.283848 \pm 2 * 0.009497) + X * (0.108618 \pm 2 * 0.006678)$ based on the equation given from the summary function above.

Chapter 19 Problems

Problem 3: Given the vector of words below, determine the output of the following regular expressions without running the R code.

str_subset(x, pattern = "pop") #1

My Guess: popular, popularity, popularize, popularise, population, repopulate
This will look for the substring "pop" in all the words and returns the unique words regardless of capitalization.

str_detect(x, pattern = "^pop") #2

My Guess: TRUE TRUE TRUE TRUE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
This command will search for words that contain the substring "pop" exactly as its spelled and true or false.

str_detect(x, pattern = "populari[sz]e") #3

My Guess: FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
This command will find words that have the pattern as long as they are spelled with an s or z and return true or false if it meets the pattern.

str_detect(x, pattern = "pop.*e") #4

My Guess: FALSE FALSE TRUE TRUE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
This command looks for the substring "pop" with any other characters as long as it contains an 'e' after the "pop".

str_detect(x, pattern = "p[a-z]*e") #5

My Guess: FALSE FALSE TRUE TRUE FALSE FALSE TRUE TRUE FALSE TRUE FALSE FALSE
This command returns true or false depending on if the word in the string vector contains the beginning letter of p followed by any characters between a and z and ending in e. The string can be of any length.

str_detect(x, pattern = "[1][a-z]+.*n") #6

My Guess: FALSE FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
This command looks for either a capital P or lower case p then characters in the alphabet with any repetition and any length and finally ending in n.

str_subset(x, pattern = "[2]") #7

My Guess: repopulate, reproduce, happy family, happier\tfamily, happy family
This will search for and return any word that doesn't contain P or p at the beginning of the word.

str_detect(x, pattern = "[3]") #8

My Guess: TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE TRUE
This will return true of false depending on if the word starts with a capital letter or lowercase letter between a and p, respectively.

[1] Pp
[2] ^Pp
[3] A-Za-p

str_detect(x, pattern = "[ ]") #9

My Guess: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE
This command returns true or false if there is a space in the string. True begin for there is a space and false otherwise.

str_subset(x, pattern = "[]^") #10

My Guess: happier\tfamily
This return the string of characters containing a \t.

str_detect(x, pattern = "[ ]^") #11

My Guess: FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE FALSE
This command return true or false depending if the string contains either a space or a \t.

str_subset(x, pattern = "[4]") #12

My Guess: " happy family"
This command returns the string if it starts with a space as the first character.

[4]

Code:

```
x <- c("popular", "popularity", "popularize", "popularise", "Popular",
"population", "repopulate", "reproduce", "happy family",
"happier\tfamily", " happy family", "P6dn")
x
```

```
##  [1] "popular"         "popularity"      "popularize"      "popularise"
##  [5] "Popular"         "population"      "repopulate"      "reproduce"
##  [9] "happy family"    "happier\tfamily" " happy family"   "P6dn"
```

```
str_subset(x, pattern = "pop")                  #1
```

```
## [1] "popular"    "popularity" "popularize" "popularise" "population"
## [6] "repopulate"
```

```
str_detect(x, pattern = "^pop")                 #2
```

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

```
str_detect(x, pattern = "populari[sz]e")        #3
```

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

```
str_detect(x, pattern = "pop.*e")               #4
```

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

```
str_detect(x, pattern = "p[a-z]*e")             #5
```

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

```
str_detect(x, pattern = "^[Pp][a-z]+.*n")       #6
```

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

```
str_subset(x, pattern = "^[^Pp]")               #7
```

```
## [1] "repopulate"      "reproduce"       "happy family"    "happier\tfamily"
## [5] " happy family"
```

```
str_detect(x, pattern = "^[A-Za-p]")            #8
```

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

```
str_detect(x, pattern = "[ ]")                    #9
```

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

```
str_subset(x, pattern = "[\t]")                   #10
```

```
## [1] "happier\tfamily"
```

```
str_detect(x, pattern = "[ \t]")                  #11
```

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

```
str_subset(x, pattern = "^[ ]")                   #12
```

```
## [1] " happy family"
```

Please note: the string, "Population" in the book is replaced by "population" in the worksheet.

Problem 4: Use the babynames data table from the babynames package to find the 10 most popular:

- a) Boys' names ending in a vowel
- b) Names ending with joe, jo, Joe, or Jo

```
# Part A
boysNames <- filter(.data = babynames, sex == "M")
boysNames <- filter(.data = boysNames,
                    name == grep(x = boysNames$name,
                                 pattern = "[aeiou]$",
                                 value = TRUE))
```

```
## Warning in name == grep(x = boysNames$name, pattern = "[aeiou]$", value = TRUE):
## longer object length is not a multiple of shorter object length
```

```
head(arrange(.data = boysNames, desc(n)), 10)
```

```
## # A tibble: 10 x 5
##     year sex   name            n      prop
##    <dbl> <chr> <chr>       <int>     <dbl>
##  1  1957 M     Reynaldo      262 0.000120
##  2  1893 M     Jessie        214 0.00177
##  3  1949 M     Monroe        167 0.0000927
##  4  2001 M     Freddie       167 0.0000808
##  5  2012 M     Dale          163 0.0000805
##  6  2001 M     Isidro        151 0.0000730
##  7  1961 M     Adolfo        106 0.0000492
##  8  1916 M     Guadalupe      99 0.000107
##  9  1954 M     Dee            98 0.0000474
## 10  1915 M     Constantine    86 0.0000976
```

```r
# Part B
endsnames <- filter(.data = babynames,
                    name == grep(pattern = "[Jj](oe|o)$",
                                 x = babynames$name,
                                 value = TRUE))
```

```
## Warning in name == grep(pattern = "[Jj](oe|o)$", x = babynames$name, value =
## TRUE): longer object length is not a multiple of shorter object length
```

```r
head(arrange(.data = endsnames, desc(n)), 10)
```

```
## # A tibble: 10 x 5
##     year sex   name      n    prop
##    <dbl> <chr> <chr> <int>   <dbl>
##  1  1935 M     Joe    7556 0.00707
##  2  1942 M     Joe    7382 0.00524
##  3  1960 M     Joe    7311 0.00338
##  4  1948 M     Joe    7289 0.00409
##  5  1961 M     Joe    7058 0.00327
##  6  1930 M     Joe    6878 0.00609
##  7  1932 M     Joe    6696 0.00623
##  8  1945 M     Joe    6625 0.00483
##  9  1921 M     Joe    6203 0.00545
## 10  1955 M     Joe    5860 0.00280
```