Stats 701 Assignment #1 Solutions

Josh Errickson

Due March 24, 2017, submitted via Canvas.

Libraries I used in my solutions:

```r
library(rvest)
library(stringr)
library(jsonlite)
library(sqldf)
library(readxl)
```

Instructions:

- Submit **both** your RMarkdown file **and** your output file (pdf or html, your choice). The Rmarkdown should compile and show all work.
- If you use an online resource or collaborate with classmates, provide attribution.
- If you cannot finish a question, show your work and explain/demostrate what is causing your issue for partial credit.
- For context, each exercise here *can* be solved in under 20 lines of code. I say this not to provide a limit or to suggest brevity over clarity/completeness, but to say that if your code is going far over that without an end in sight, its likely you're misinterpreting or otherwise making your life more difficult.

Exercise 1

What Texas county currently has the highest percentage of its citizenry on death row?

- Death row offenders: http://www.tdcj.state.tx.us/death_row/dr_offenders_on_dr.html
- Population estimates: https://www.tsl.texas.gov/ref/abouttx/popcnty2010-11.html (Use 2015 data).

Scrape at least one of these sources manually (without rvest; e.g. starting with `readLines`). Scrape the second however you'd like.

Tips:

- You don't need most of the information in either table so don't waste effort on it.
- We discussed `[:print:]` as matching any printable character. An example of a *non*-printable character is `\t`, a tab. `.` matches *any* single character, printable or not.
- Don't forget `stringsAsFactors = FALSE` if you convert anything to a data.frame.
- `table` might be useful. Specifically, the concept behind this code snippet:

```r
data <- c("a", "c", "a", "b", "a")
table(data)[c('c', 'a')]
```

```
## data
## c a
## 1 3
```

Note: If manually scraping the death row data look out for:

1) There's a random blank row in the middle of the data. `"^[:space:]*$"` may come in handy.
2) There's some extra space following some counties. `str_trim` can be used.

Solution

Extract death row offenders

Manual:

```r
page <- readLines("http://www.tdcj.state.tx.us/death_row/dr_offenders_on_dr.html")
# By searching for "Wells" and examining the page, we need lines 112:2992
page <- page[112:2992]

# Let's remove that blank row
page <- page[!str_detect(page, "^[:space:]*$")]

# Each row of data is 12 lines, and we want the 10th.
page <- page[seq_along(page) %% 12 == 10]
page <- str_replace(page, "^[^>]+>", "")
page <- str_replace(page, "</td>", "")
drcounties <- table(str_trim(page))
```

rvest:

```r
page <- read_html("http://www.tdcj.state.tx.us/death_row/dr_offenders_on_dr.html")
html_nodes(page, "table")
```

```
## {xml_nodeset (1)}
## [1] <table class="os" summary="The following table lists each offender c ...
```

```r
dr <- html_table(html_nodes(page, "table")[1])[[1]]
drcounties2 <- table(dr$County)
identical(drcounties, drcounties2)
```

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

Extract population estimates

Manual:

```r
page <- readLines('https://www.tsl.texas.gov/ref/abouttx/popcnty2010-11.html')
# Explore the html and str_which to determine that we need lines 154-2439
page <- page[154:2439]
# For each entry, there are 9 lines and line 2 is the county and line 8 is the population.
county <- page[seq_along(page) %% 9 == 2]
county <- str_replace(county, "^.+\">", "")
county <- sapply(str_split(county, " "), "[", 1)

pop <- page[seq_along(page) %% 9 == 8]
pop <- str_replace(pop, "^.+\">", "")
pop <- as.numeric(str_replace_all(pop, "</td>|,", ""))
pop <- data.frame(county = county,
                  pop2015 = pop,
                  stringsAsFactors = FALSE)
```

rvest:

```r
page <- read_html('https://www.tsl.texas.gov/ref/abouttx/popcnty2010-11.html')
html_nodes(page, "table")
```

```
## {xml_nodeset (1)}
## [1] <table border="1" cellpadding="0" cellspacing="0" style="width: 572p ...
```

```
pop2 <- html_table(html_nodes(page, "table")[1], header = TRUE)[[1]]
pop2$county <- sapply(str_split(pop2$COUNTY, " "), "[", 1)
pop2$pop2015 <- as.numeric(str_replace_all(pop2$`7/1/2015`, ",", ""))
pop2 <- pop2[, c("county", "pop2015")]
identical(pop, pop2)
```

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

Answer the question

```
pop$num_on_deathrow <- drcounties[pop$county]
pop$perc_on_deathrow <- pop$num_on_deathrow/pop$pop2015
pop[which.max(pop$perc_on_deathrow), ]
```

```
##      county pop2015 num_on_deathrow perc_on_deathrow
## 120 Jackson   14816               2     0.0001349892
```

Exercise 2

Use the Open Movie Database API (https://omdbapi.com/) to create function `getMovieData()` that takes in the title of a movie and returns the title, the year the movie was released (as a numeric) and the total number of minutes (as a numeric).

For example:

```
getMovieData("Zootopia")
```

```
##      title year minutes
## 1 Zootopia 2016     108
```

```
str(getMovieData("Star Wars"))
```

```
## 'data.frame':    1 obs. of  3 variables:
##  $ title  : chr "Star Wars: Episode IV - A New Hope"
##  $ year   : num 1977
##  $ minutes: num 121
```

Tips:

- The API is "smart" enough to expand titles; in the example above I did nothing special to let it know that "Star Wars" was really "Star Wars: Episode IV - A New Hope".
- Spaces in titles in the API url will cause an error. Use the "Examples" section on https://omdbapi.com/ to figure out a way around it.
- Don't forget `stringsAsFactors = FALSE` if you convert anything to a data.frame.

Extra credit:

1) Provide a reasonable error message if the title isn't found. (You can pass `error = TRUE` to a knitr R chunk to cause errors to print in your document rather than refusing to compile.)
2) Vectorize the function to accept a vector of movie names and output the results in a data.frame.

```
getMovieData("Statistics the Movie")
```

```
## Warning in FUN(X[[i]], ...): Movie "Statistics the Movie" not found!
##   title year minutes
## 1    NA   NA      NA
```

```r
getMovieData(c("Zootopia", "Star Wars", "Statistics the Movie"))
```

```
## Warning in FUN(X[[i]], ...): Movie "Statistics the Movie" not found!
```

```
##                                 title year minutes
## 1                            Zootopia 2016     108
## 2  Star Wars: Episode IV - A New Hope 1977     121
## 3                                <NA>   NA      NA
```

Solution

```r
getMovieData <- function(title) {
  # Replace spaces with +
  title2 <- str_replace_all(title, "[:space:]+", "+")

  ## Vectorized:
  output <- sapply(title2, function(i) {
    url <- str_c("http://www.omdbapi.com/?t=", i)
    data <- fromJSON(url)

    # Break if a movie title isn't found
    if (data$Response == "False") {
      warning(str_c("Movie \"",
                    str_replace_all(i, "[+]+", " "),
                    "\" not found!"))
      return(c(NA, NA, NA))
    }

    data$Runtime <- str_replace(data$Runtime, " min", "")
    return(c(data$Title, data$Year, data$Runtime))
  })

  output <- data.frame(t(output), stringsAsFactors = FALSE)

  # Clean up names on output.
  colnames(output) <- c("title", "year", "minutes")
  rownames(output) <- NULL

  # Convert to numeric (Note that " min" was stripped earlier.)
  output$year <- as.numeric(output$year)
  output$minutes <- as.numeric(output$minutes)

  return(output)
}
```

Exercise 3

Using only SQL queries, answer the following questions about hydropower potential in the Western U.S.: https://catalog.data.gov/dataset/hydropower-potential-in-the-western-us. Each observation reports the potential benefit of introducing hydroelectric power capabilities at the various sites.

Two easy ways to read the Excel data into R if you aren't familiar with it:

1) Open the file in Excel and save as a .csv file.
2) Use the package readxl (part of Hadley Wickham's tidyverse) and function `read_excel`.

In either case, be sure to check variable types to ensure proper strings/numeric formats.

Aside from reading in the data, **no other function except `sqldf` should be used**.

Tips:

- Variable names with spaces in them can be wrapped in backticks (e.g. `` `variable name` ``) in SQL queries.
- Use `str_c` with argument `sep = " "` to break up longer queries so they display properly (and are easier to read).

1) How many total sites are there in Colorado? (Region "UC" and "LC" represent lower and upper Colorado respectively.) Do this in two ways: Subsetting just these regions and using the `nrow` command, and using SQL exclusively.
2) What would be the total cost of building all sites in Lower Colorado?
3) What 3 sites in the Great Plains (Region "GP") have the lowest Cost Per Installed Capacity, restricted to those whose total cost is under 5 millions dollars?
4) Which Region has the highest median estimated Annual Production?

Solution

```
dt <- read_excel("~/Downloads/ResourceAssessmentSummaryData032011.xlsx")

# 1)
nrow(sqldf("SELECT * FROM dt WHERE Region = 'UC' OR Region = 'LC'"))
```

```
## [1] 70
```

```
sqldf("SELECT COUNT() AS `Sites in CO` FROM dt WHERE Region = 'UC' OR Region = 'LC'")
```

```
##   Sites in CO
## 1          70
```

```
# 2)
sqldf(str_c("SELECT sum(`Total Construction Cost (1,000 $) `) AS",
            "`Total Cost in LC (1,000$)` FROM dt WHERE Region = 'LC'",
             sep = " "))
```

```
##   Total Cost in LC (1,000$)
## 1                   52663.1
```

```
# 3)
sqldf(str_c("SELECT `Site Name/Facility `, `Cost per Installed Capacity ($/kW) `",
            "FROM dt WHERE Region = 'GP' AND `Total Construction Cost (1,000 $) ` < 5000",
            "ORDER BY `Cost per Installed Capacity ($/kW) ` ASC LIMIT 3",
            sep = " "))
```

```
##   Site Name/Facility  Cost per Installed Capacity ($/kW)
## 1      Angostura Dam                                 3358
## 2        Pactola Dam                                 3706
## 3     Glen Elder Dam                                 4229
```

```
# 4)
sqldf(str_c("SELECT Region, median(`Annual Production (MWh) `) AS",
            "`Median Annual Production` FROM dt GROUP BY Region",
            "ORDER BY `Median Annual Production` DESC",
            sep = " "))
```

```
##   Region Median Annual Production
## 1     LC                   5325.0
## 2     PN                   2937.5
## 3     UC                   1844.0
## 4     GP                   1399.0
## 5     MP                    945.5
```