Lab 5

Math 241, Week 6

```
# Put all necessary libraries here
library(tidyverse)
library(rvest)
library(httr)
#library(rnoaa)
#library(httr)
library(jsonlite)
# all the stuff with rnoaa won't knit cause I only got it working on my laptop and now it doesn't even
```

Due: Friday, March 1st at 8:30am

Goals of this lab

1. Practice grabbing data from the internet.
2. Learn to navigate new R packages.
3. Grab data from an API (either directly or using an API wrapper).
4. Scrape data from the web.

Potential API Wrapper Packages

Problem 1: Predicting the ~~Un~~predictable: Portland Weather

In this problem let's get comfortable with extracting data from the National Oceanic and Atmospheric Administration's (NOAA) API via the R API wrapper package `rnoaa`.

You can find more information about the datasets and variables here.

a. First things first, go to this NOAA website to get a key emailed to you. Then insert your key below:

```
#options(noaakey = "OyJVNkFdchuXhDIHjsofsJGGpyKmqStd")
```

```
#precip_se_pdx <- ncdc(datasetid = "GHCND", datatypeid = "PRCP",
                       #startdate = "2024-01-01",
                       #enddate = "2024-01-31",
                       #stationid = "GHCND:US1ORMT0006",
                       #limit = 1000)

#oogabooga <- precip_se_pdx$data
```

b. From the National Climate Data Center (NCDC) data, use the following code to grab the stations in Multnomah County. How many stations are in Multnomah County?

```
#stations <- ncdc_stations(datasetid = "GHCND",
                           #locationid = "FIPS:41051")

#mult_stations <- stations$data
```

#there are 25 stations

c. January was not so rainy this year, was it? Let's grab the precipitation data for site `GHCND:US1ORMT0006` for this past January.

```
# First fill-in and run to following to determine the
# datatypeid
#blah <- ncdc_datatypes(datasetid = "GHCND",
               #stationid = "GHCND:US1ORMT0006")

# Now grab the data using ncdc()
#precip_se_pdx <- ncdc(datasetid = "GHCND",
               #stationid = "GHCND:US1ORMT0006", token = "OyJVNkFdchuXhDIHjsofsJGGpyKmqStd")
```

d. What is the class of `precip_se_pdx`? Grab the data frame nested in `precip_se_pdx` and call it `precip_se_pdx_data`.

```
#it's empty tho; using NE data instead, also empty; using SW data
#se_pdx_df <- precip_se_pdx[["tbl_df"]]

# look up examples of this packages function
```

it's a list

e. Use `ymd_hms()` in the package `lubridate` to wrangle the date column into the correct format.

```
#ymd_hms(
  #oogabooga,
  #quiet = FALSE,
  #tz = "UTC",
  #locale = Sys.getlocale("LC_TIME"),
  #truncated = 0
#)
```

f. Plot the precipitation data for this site in Portland over time. Rumor has it that we had only one day where it didn't rain. Is that true?

```
#ggplot(data = oogabooga,
 # aes(x = date, y = value)
 # ) + geom_point()
```

it is not true

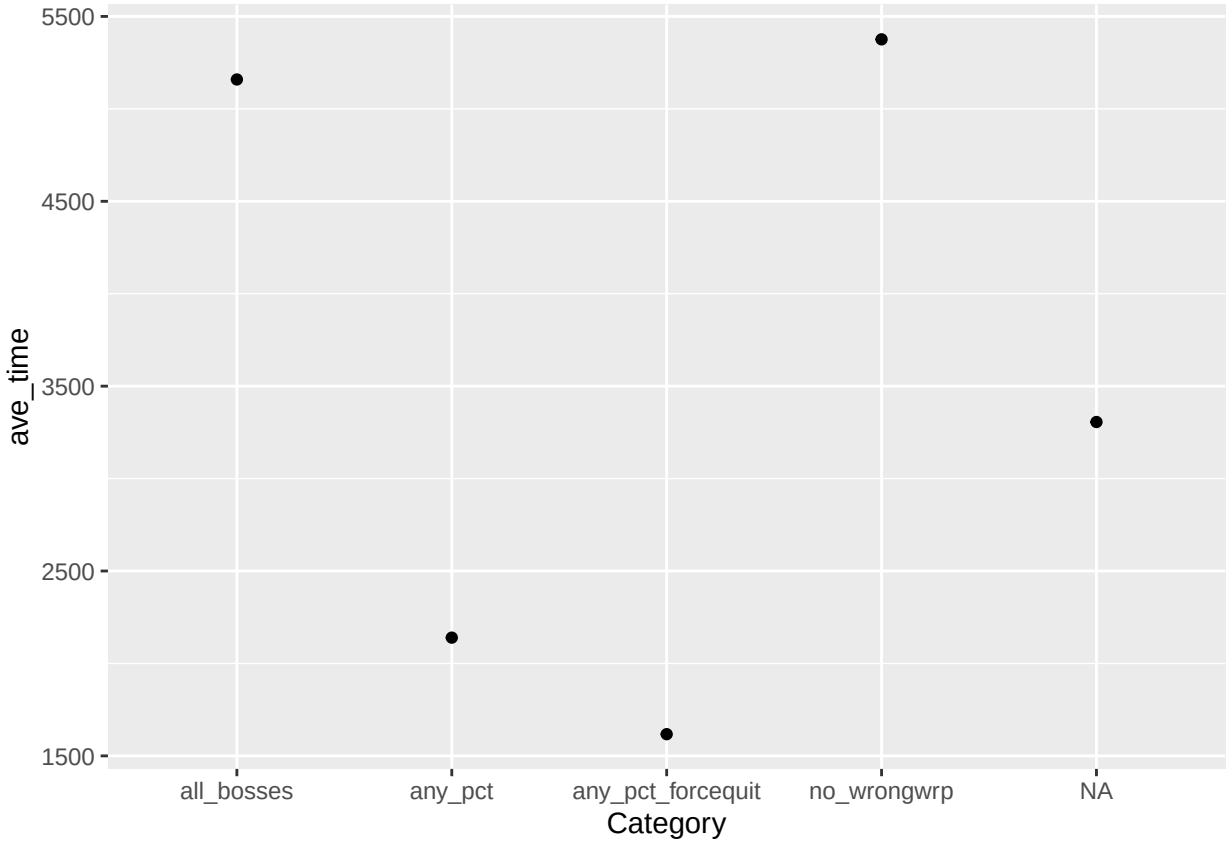
g. (Bonus) Adapt the code to create a visualization that compares the precipitation data for January over the the last four years. Do you notice any trend over time?

Problem 2: From API to R

For this problem I want you to grab web data by either talking to an API directly with `httr` or using an API wrapper. It must be an API that we have NOT used in class or in Problem 1.

Once you have grabbed the data, do any necessary wrangling to graph it and/or produce some summary statistics. Draw some conclusions from your graph and summary statistics.

```
json_souls <- fromJSON(txt = "https://www.speedrun.com/api/v1/games?name=darksouls")

#grabbing data for darksouls remastered
darksouls_r <- fromJSON(txt = "https://www.speedrun.com/api/v1/games/lde3woe6")

#grabbing  "runs"
darksouls_r_runs <- fromJSON(txt = "https://www.speedrun.com/api/v1/runs?game=lde3woe6")

#speedrun_categories <- fromJSON(txt = "https://www.speedrun.com/api/v1/categories?game=lde3woe6")

#grabing category codes (this one is "All Bosses")
vdo3qoyd_all_bosses <- fromJSON(txt = "https://www.speedrun.com/api/v1/categories/vdo3qoyd")

#grabing category codes (this one is Any%")
ndx1pm52_any_percent <- fromJSON(txt = "https://www.speedrun.com/api/v1/categories/ndx1pm52")

#grabing category codes (this one is Any% Force Quit")
xd173pzd_anypct_forcequit <- fromJSON(txt = "https://www.speedrun.com/api/v1/categories/xd173pzd")

#grabing category codes (this one is "No Wrong Warp")
wdm84w52_no_wrongwrp <- fromJSON(txt = "https://www.speedrun.com/api/v1/categories/wdm84w52")


dsr_runs_data <- darksouls_r_runs[["data"]]

dsr_runs_data <- dsr_runs_data %>%
  mutate(Category = factor(
    category, levels = c("vdo3qoyd", "ndx1pm52", "xd173pzd", "wdm84w52"), labels = c("all_bosses", "any
    time_ingame = times$ingame_t
    ) %>%
  select(time_ingame, Category, date, comment) %>%
  group_by(Category) %>%
  summarize(
    ave_time = mean(time_ingame)
  )

  ggplot(data = dsr_runs_data,
    aes(x = Category, y = ave_time)
  ) +
  geom_point()
```



There are only 20 observations in the frame that I ended up with despite the fact that there are 57 observations in the table for Darksouls remastered on the website. But honestly, it was so hard just to find out what the alphanumeric codes that made up so many of the categorical variable levels even stood for that im just happy to have any data from this website

API Wrapper Suggestions for Problem 2

Here are some potential API wrapper packages. Feel free to use one not included in this list for Problem 2.

- `gtrendsR`: "An interface for retrieving and displaying the information returned online by Google Trends is provided. Trends (number of hits) over the time as well as geographic representation of the results can be displayed."
- `rfishbase`: For the fish lovers
- `darksky`: For global historical and current weather conditions

Problem 3: Scraping Reedie Data

Let's see what lovely data we can pull from Reed's own website.

a. Go to https://www.reed.edu/ir/success.html and scrape the two tables.

b. Grab and print out the table that is entitled "GRADUATE SCHOOLS MOST FREQUENTLY ATTENDED BY REED ALUMNI". Why is this data frame not in a tidy format?

```
# Load required libraries
library(rvest)

# Specify the URL of the webpage containing the tables
url <- "https://www.reed.edu/ir/success.html"

# Read the HTML content from the webpage
html_content <- read_html(url)

# Scrape the tables using CSS selectors
tables <- html_nodes(html_content, "table")

# If you know the specific table you want to scrape, you can use indexing (e.g., tables[[1]] for the fi

# Convert the scraped tables to data frames
table_data <- lapply(tables, function(x) {
  html_table(x, fill = TRUE)
})

# Access the scraped data
# For example, if you want the first table data
print(table_data[[1]])
```

```
## # A tibble: 10 x 2
##    X1                   X2
##    <chr>                <chr>
##  1 Business & Industry  28%
##  2 Education            25%
##  3 Self-Employed        19%
##  4 Students             7%
##  5 Government Service   5%
##  6 Health Care          5%
##  7 Law                  4%
##  8 Miscellaneous        4%
##  9 Arts & Communication 2%
## 10 Community Service    1%
```

```
ocupational_dist_alumn <- table_data[[1]]
grad_schl_freq <- table_data[[2]]
```

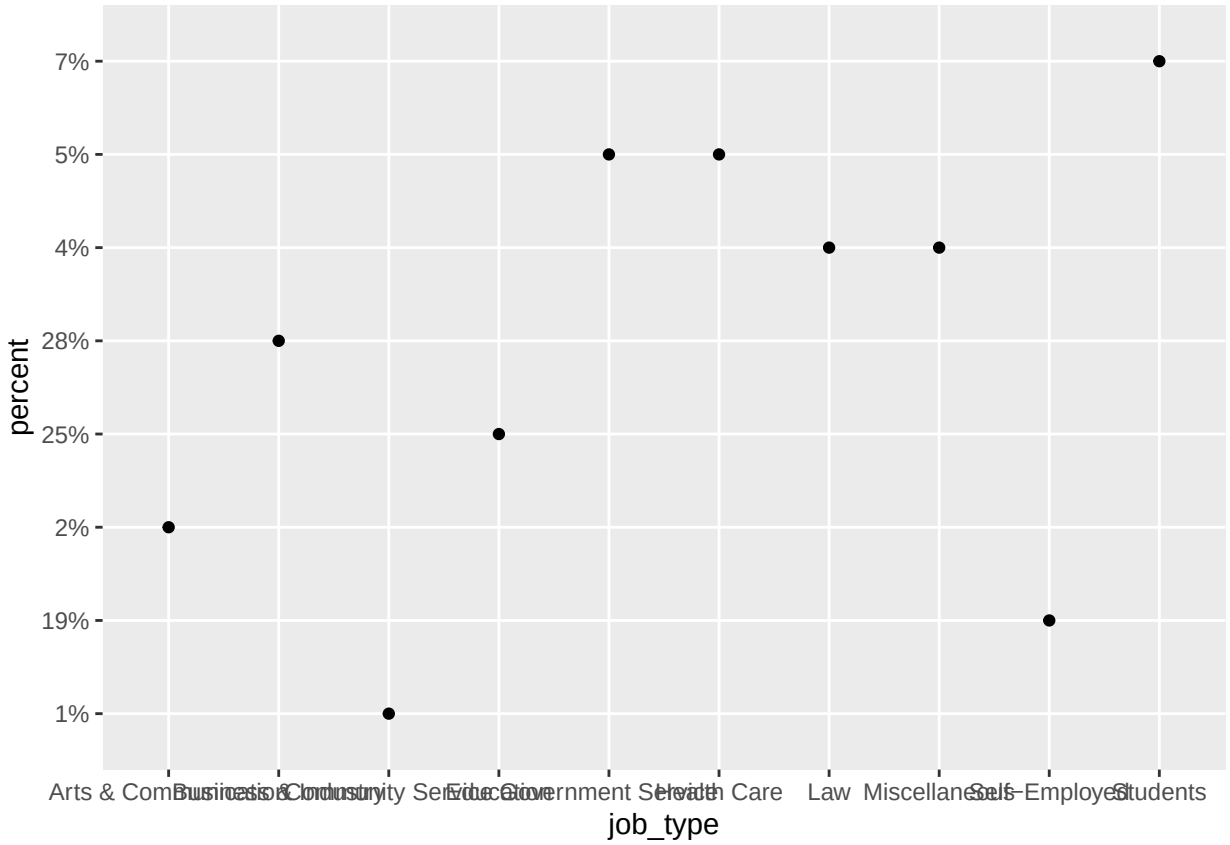
c. Wrangle the data into a tidy format. Glimpse the resulting data frame. This data seems to satisfy the requirements for tidy data already

```
glimpse(grad_schl_freq)
```

```
## Rows: 11
## Columns: 4
## $ MBAs <chr> "U. of Chicago", "Portland State U.", "Harvard U.", "U. of Washin~
## $ JDs  <chr> "Lewis & Clark  Law School", "U.C., Berkeley", "U. of Oregon", "U~
## $ PhDs <chr> "U.C., Berkeley", "U. of Washington", "U. of Chicago", "Stanford ~
## $ MDs  <chr> "Oregon Health & Sci Univ.", "U. of Washington", "Washington U. (~
```

d. Now grab the "OCCUPATIONAL DISTRIBUTION OF ALUMNI" table and turn it into an appropriate graph. What conclusions can we draw from the graph?

```
# Hint: Use `parse_number()` within `mutate()` to fix one of the columns
#it looks fine to me??????


ocupational_dist_alumn <- ocupational_dist_alumn %>%
  mutate(
    job_type = X1,
    percent = X2
         )

ocupational_dist_alumn <- ocupational_dist_alumn %>%
  select(
    job_type,
    percent
         )

# I still don't know how to map bars onto a dot plot, sue me
ggplot(data = ocupational_dist_alumn,
  aes(x = job_type, y = percent)
) + geom_point()
```



```
# I know it looks bad but I'm late and tired
```

e. Let's now grab the Reed graduation rates over time. Grab the data from here.

Do the following to clean up the data:

- Rename the column names.

```
# Hint
colnames(___) <- c("name 1", "name 2", ...)
```

- Remove any extraneous rows.

```
# Hint
filter(row_number() ...)
```

- Reshape the data so that there are columns for
  - Entering class year
  - Cohort size
  - Years to graduation
  - Graduation rate
- Make sure each column has the correct class.

f. Create a graph comparing the graduation rates over time and draw some conclusions.