Assignment 2

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

```
library(ggpomological)
```

Please do all your work for this assignment within this R Markdown file. Unless I specifically ask for a written response, your code is sufficient as an answer to a question. I've left code chunks and comments to indicate where to write your code, and typically the size of the code chunk suggests how much code you'll need to write (so a chunk with one empty line could be solved with one line of code). Of course, take more space if you like. And if you have favorite packages you would like to use, feel free, but it is not necessary.

Part 1: Exploring online data

Find and download a tabular dataset that interests you from a publicly available database or repository (see the Lab 2 powerpoint for suggestions). Find one that has at least one numeric column (such as age, score, etc.), and at least one categorical column (such as gender, race, etc.). A .csv file will likely be easiest to work with, but choose any dataset you feel comfortable you can import. Read the dataframe into R below:

```
library(readr)
avocado <- read_csv("avocado.csv")
```

```
## New names:
## * `` -> ...1

## Rows: 18249 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (2): type, region
## dbl  (11): ...1, AveragePrice, Total Volume, 4046, 4225, 4770, Total Bags, S...
## date  (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

How many rows and columns are in the dataset? What does each row represent (i.e. what is the dataset about)?

```
dim(avocado)
```

```
## [1] 18249    14
```

Each row represents a weeks scan data of avocadoes. It is scan data for National retail unites and prices sold. The data comes directly from retailers' cash registered and was provided by Hass avocados "Avocado Board"

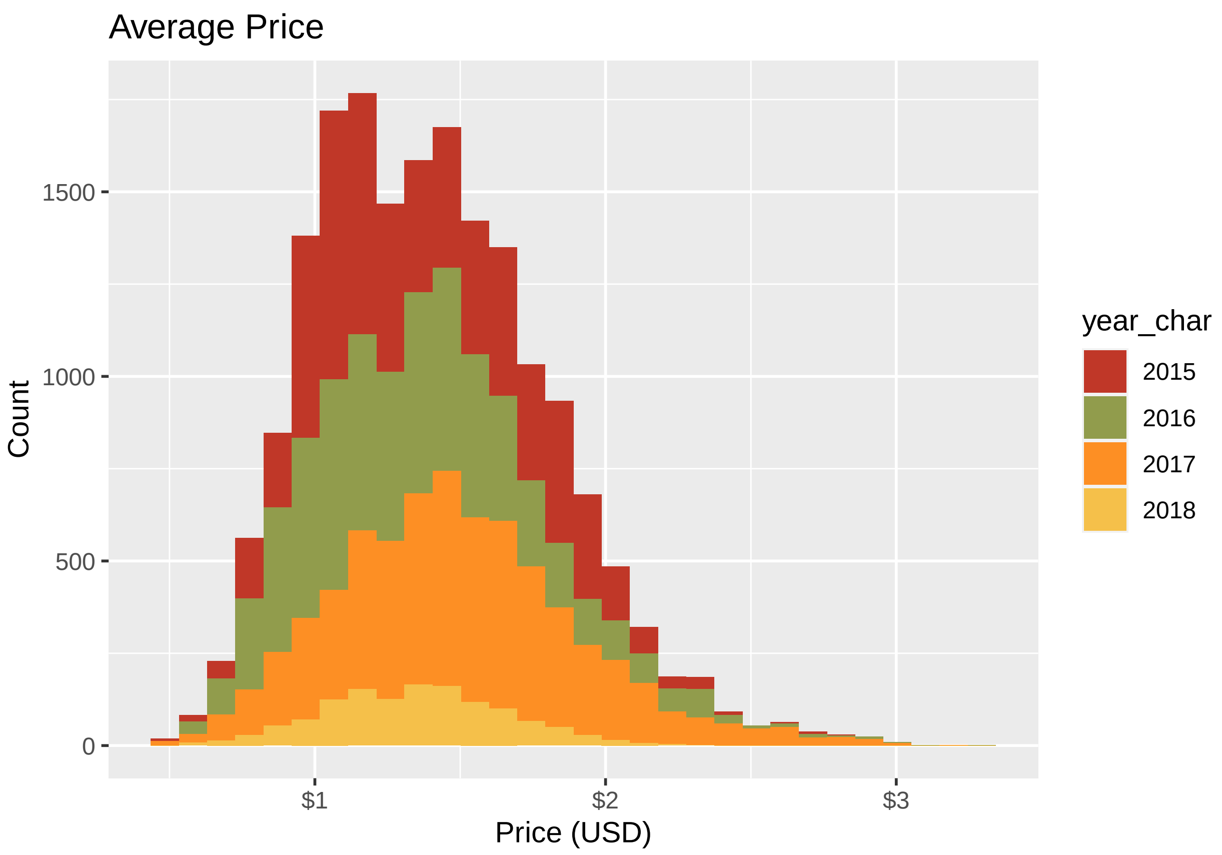
Pick three variables (columns) that interest you, and describe what they measure.

Average price is the average price across all selling locations that was paid for avocados. The number columns represent individual SKU's of avocados (different types of Hass avocadoes like large or extra large). Type was fun that it was included, shows conventional or organic.

Pick a single column with numeric data (numbers, not text), plot a histogram of that column, and describe what the figure tells you about the data.

```
avocado_plot <- avocado %>%
  mutate(year_char = as.character(year))

ggplot(data = avocado_plot) +
  aes(AveragePrice, fill = year_char) +
  geom_histogram() +
  xlab('Price (USD)') +
  ylab('Count') +
  ggtitle("Average Price") +
  scale_x_continuous(labels = scales::dollar_format()) +
  scale_fill_pomological()
```



```
# We'll get fancy with figures later, keep it simple with hist()
```

Certainly not everyone pays the same thing for avocados, all year round. It also appears every year we are eating less avocados!

Let's visualize two variables together. **Pick either two numeric columns, or one categorical and one**

numeric column, and plot them against each other using the `plot()` function. Describe what the figure tells you about the data If you have a categorical variable, often we place these on the X-axis (the first argument to `plot()`). Also, R sometimes requires that categorical variables are treated as `factors` (more on this later). If you have a `character` column that is categorical (like gender, race, etc.), you can convert it to a factor using `as.factor()`.

```
# myData$myVariable = as.factor(myData$myVariable) # Try this template if categorical plot looks weird
```

Optional Challenge: Make one more plot of your choice, using `hist()`, `plot()`, or `pie(table())`, or any function you like. Describe what the figure tells you about the data.

Part 2: Pulling data from the web using R and regular expressions

Let's practice obtaining data from the internet by pulling text information from Wikipedia. *Warning:* Most sites prefer you do not download many pages in a short amount of time (i.e. more/faster than you could by clicking by hand). We'll be respectful of Wikipedia by only accessing a handful of pages, and setting a timer in R to do it slowly.

One great feature of Wikipedia is that articles link to other articles on the site, creating an enormous network of information. We can get data on which topics are related to other topics by analyzing the links present in the article. These links are buried in text information in the site's HTML. We can extract information from HTML elements using R packages and regular expressions. Let's build an R script that reads a Wikipedia page, finds the top 10 links on that article, and stores a short description of each link.

In the code chunk below, install the `rvest` package if you haven't already. Then load `rvest` into R. Create a string variable called `start_URL` that contains the full URL of a Wikipedia article. Let's start with the article for Chicago, because this page presents an interesting challenge. We'll change the start page later to one of your choice. Download the page using the function `read_html()` and assign the output to a variable `start_page`.

```
# Install rvest if you haven't already - you only need to do this once!
# install.packages('rvest') # Comment this line out once you've installed the package to avoid running

# Load the rvest package into R
library(rvest)
```

```
##
## Attaching package: 'rvest'

## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
# Get the full URL of the wikipedia page we want to learn about
start_URL = 'https://en.wikipedia.org/wiki/Chicago'

# Read the page's HTML data into R using read_html()
```

`rvest` contains a few key functions to make parsing HTML easier. If you inspect the HTML of any website, you'll find that all content that appears in your browser is surrounded by tags contained in `<` and `>` brackets. On well-made websites (like Wikipedia), major sections are labeled with a `class =` or an `id =` within the tag. **We want to find all the links (within `<a>` tags) in the main text description of the page**. Using the `html_elements()` function, collect all the `<a>` elements in the main text description into a variable `links`. This can be tricky if you are unfamiliar with HTML! If you get stuck, check the `A2Hints.txt` file on the assignment page in Courseworks for the solution.

```
# Get the links within the main content section
```

If you succeeded, you'll have a variable `links` in your environment that is a `Large xml_nodeset` with 1715 elements. View the first 10 elements by running `links[1:10]` below:

The `href=` within each `<a>` tag is the (partial) URL that the link directs to. URLs that start with `/wiki/` are links to other Wikipedia articles, which we'd like to save. For example, the 3rd element of `links` goes to `upload.wikimedia` and is a sound file, and the 7th element links to a citation. We can use **regular expressions** to pull out only the links that contain references to other Wikipedia articles by matching the pattern `href="/wiki/`. Use the output of `grepl()` to get a subset of links (using square brackets `[]` like above). If you get stuck, study the `lab 2 r code.rmd` from Monday, or check the `A2Hints.txt` file.

```
# Get only links that reference other wikipedia articles
## links =
```

Next, some of the links that start with `/wiki/` go to Help or File pages (`href="/wiki/Help..."`). These are not topics related to the article, so let's get rid of these, again with regular expressions. If we use `grepl()` to match these patterns, we'll get a TRUE where they do occur. But we want to keep the rows where they DON'T occur. We can invert a list of logical values using the `!` symbol, which means 'not'. Subset the `links` variable like you did above, keeping the elements that do not (`!`) match `/Help` or `/File`. It may be easiest to do this in two steps. Check `A2Hints.txt` for help if you're unsure.

```
# Remove links with Help or File
```

These steps should have reduced your `links` variable down to 1340 elements. To keep things simple, let's save only the first 10 links. Overwrite the `links` variable by reassigning it to itself, subsetting elements `1:10` in square brackets `[]`, like you did to view them above. Be sure that your new `links` has 10 elements before moving forward.

```
# Save only first 10 links
```

OK, with this bit of clean-up, we're finally ready to start extracting data about the Wikipedia article! We'll start by building a data frame. First, extract the text within the `links` using `html_text()`, and assign the output to a new variable `topic_names`. Then, construct a data frame using `data.frame()`, giving it `topic_names` as an argument, and assigning the output to a variable `wiki_data`. By default, the name of the column will take the name of the variable. Rename the column to `Topic` using the `colnames()` function. Finally, initiate a new column of `wiki_data`, calling it 'Description', and giving it a value of `NA`. (Check the Lab 2 R code to see how to do these steps)

```
# Get name of the topic that the Wikipedia article links to


# Make a data frame to store and organize the list of topics


# Rename the first dataframe column to 'Topic'


# Initiate a new, empty column called 'Description' that contains NA
```

Almost there! The description of each topic is contained within each article's page. So we'll need to use the URL to travel to each of the 10 pages, and extract a description of each topic. First, get the URLs from the `links` variable using `html_attr()`, accessing the 'href' attribute. Store the output into a variable `URLs`. **If you're up for a challenge**, write a `for` loop that iterates through the `URLs` and stores the `.shortdescription` element of each page into the correct row in the `wiki_data` 'Description' column. **Otherwise,** if you don't think you're up for such an endeavor (which is fine!), copy and paste the for loop in the `A2Hints.txt` file on Courseworks. Either way, **make sure you include the line `Sys.sleep(5)` within the loop** so you don't get flagged by Wikipedia and get your access blocked!

```r
URLs =

# Loop begins here {


  # Read the current article HTML into R

  # Extract the contents of '.shortdescription' tag, and store it in the dataframe


  # Be respectful of Wikipedia's servers by waiting for 5 seconds between requests
  # DO NOT REMOVE the Sys.sleep(5) line below!
  # Requesting webpages too fast can result in your computer being blocked from the website
  Sys.sleep(5)
#}
```

You should now have a data frame containing 10 topics related to the article, and a short description of each topic! Run the code below to view the results in a 'prettier' format.

Congratulations, you've built the foundation for a Wikipedia web scraper (also known as a crawler, or spider). Again, be wary of each site's terms of use and be respectful by downloading pages at a reasonable speed. Now, return to the top of this section, change the `start_URL` to a Wikipedia article of your choice, and re-run all your code. You should get back a list of 10 topics related to that article, and a short description of each!

To submit the assignment, use the `Knit` dropdown and select `Knit to PDF` to save a PDF file, and upload to Courseworks. (Knit to HTML is fine if you can't get the PDF to work)

Part 3: Your feedback on this assignment

On a scale of 1-5, with 1 being "too easy", 5 being "too hard", and 3 being "a good level", how difficult was this assignment?

Approximately how much time did you spend working on this assignment? (i.e. actively solving the problem, not exploring the data independently)

Any thoughts on what you found useful, or what you found mundane, or confusing? (anything, big or small)