BUS 317 - Final Project

Mary Ardoin

2023-04-14

Load Packages

```
library(tidyverse)
library(robotstxt)
library(kableExtra)
library(ggplot2)
library(ggmap)
library(tidygeocoder)
library(maps)
library(lubridate)
library(gridExtra)
library(knitr)
library(RMySQL)
library(ggrepel)
options(ggrepel.max.overlaps = Inf)
```

Scraping test

```
# Scraping setup
# Checks to see if we can scrape data from ratebeer
paths_allowed("https://www.ratebeer.com/")
```

```
##
 www.ratebeer.com
```

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

Question 1

Use Octoparse to scrape the North Carolina breweries page for active breweries, meaderies, cideries, and sake producers.

```
# R script file to web scrape www.ratebeer.com

# Load libraries
library(tidyverse)

# North Carolina Breweries Scraping Using Octoparse

# Create a dataframe for each NC Octoparse file with properly formatted variable names
nc_breweries <- read_csv("data/nc_breweries.csv")
nc_meaderies <- read_csv("data/nc_meaderies.csv")
nc_cideries <- read_csv("data/nc_cideries.csv")
nc_sake_producers <- read_csv("data/nc_sake_producers.csv")

# Create a single dataframe combining all 4 dataframes for North Carolina
ncbreweries <- bind_rows(nc_breweries,
                         nc_meaderies,
                         nc_cideries,
                         nc_sake_producers)

# Add state column to the combined dataframe
ncbreweries <- ncbreweries %>%
  mutate(state = c("North Carolina"))

# Write csv file to data folder
write_csv(ncbreweries, file = "data/ncbreweries.csv")
```

Question 2

Repeat all the procedures in Question 1, but for the breweries in the state of California.

```
# R script file to web scrape www.ratebeer.com

# We already loaded tidyverse for the North Carolina breweries

# California Breweries Scraping Using Octoparse

# Create a dataframe for each California Octoparse file with properly formatted variable names
ca_breweries <- read_csv("data/ca_breweries.csv")
ca_meaderies <- read_csv("data/ca_meaderies.csv")
ca_cideries <- read_csv("data/ca_cideries.csv")
ca_sake_producers <- read_csv("data/ca_sake_producers.csv")

# Create a single dataframe combining all 4 dataframes for California
cabreweries <- bind_rows(ca_breweries,
                         ca_meaderies,
                         ca_cideries,
                         ca_sake_producers)

# Add state column to the combined dataframe
cabreweries <- cabreweries %>%
  mutate(state = c("California"))

# Write csv file to data folder
write_csv(cabreweries, file = "data/cabreweries.csv")
```

Load Data

```
# Read in the two state dataframes created in the chunks above
ncbreweries <- read_csv("data/ncbreweries.csv")
cabreweries <- read_csv("data/cabreweries.csv")
```

Question 3

Create a single dataframe, named breweries, from the two state dataframes. Be sure to save this dataframe as a csv file in your data folder, include the code in an appropriate code chunk.

1. You are to clean up any data as necessary, see Lab 07, for one example.
2. All variable names are to be in the proper format.
3. You are to factor the type variable with all beer producer types appearing first, followed by meaderies, cideries, and sake producers. However, contract and commercial breweries are to appear last. With the beer producers you are to determine the ranking within this group that makes the most sense for data visualizations.
4. Be sure the beverage count is numeric.

You are to inspect the breweries dataframe and display at least 20 rows of data, but present no more that 10 rows to the user at any one time. You are to create data visualizations of the distribution of the data within each variable.

```
# Create a single dataframe, named breweries, from the two state dataframes
breweries <- bind_rows(ncbreweries, cabreweries)

# Save the breweries dataframe as a csv in your data folder
write_csv(breweries, file = "data/breweries.csv")
```

```
# Load data
breweries <- read_csv("data/breweries.csv")

# Clean up data
glimpse(breweries)
```

```
## Rows: 1,357
## Columns: 7
## $ name           <chr> "12 Bones Brewing", "1718 Brewing Ocracoke", "3rd Rock …
## $ url            <chr> "https://www.ratebeer.com/brewers/12-bones-brewing/4010…
## $ city           <chr> "Arden", "Ocracoke", "Trenton", "Winston-Salem", "Chero…
## $ type           <chr> "Brewpub", "Brewpub", "Microbrewery", "Client Brewer", …
## $ established    <dbl> 2019, 2018, 2016, 2019, 2018, 2018, 2020, 2018, 2014, 2…
## $ beverage_count <chr> " 10", " 34", " 14", " 4", " 7", " 12", " 7", " 18", " …
## $ state          <chr> "North Carolina", "North Carolina", "North Carolina", "…
```

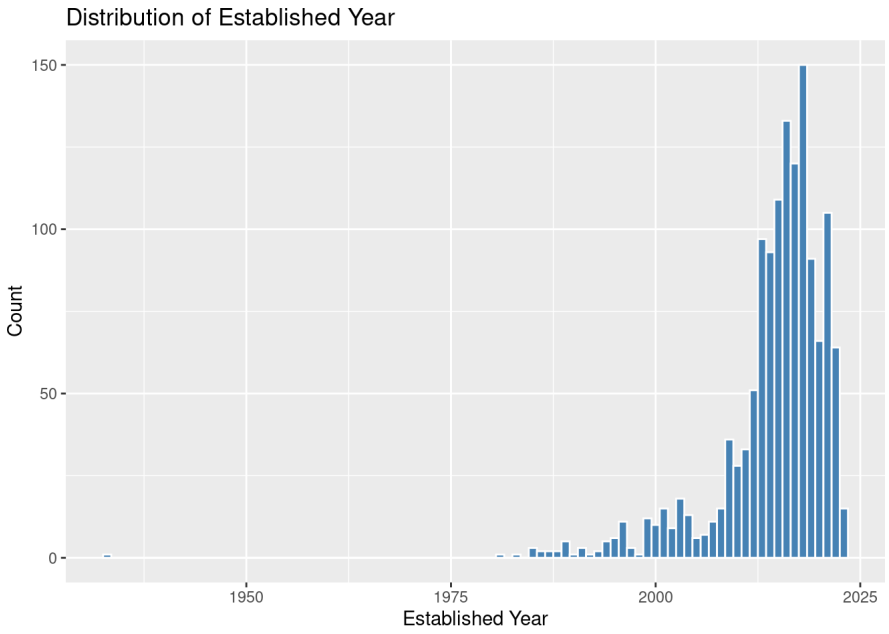
```
breweries <- breweries %>%
    mutate(established = ifelse(
      name == "Edenton Brewing Company", 2003, established),
      state = factor(state),
      type = factor(type, levels = c(
        "Microbrewery",
        "Brewpub",
        "Brewpub/Brewery",
        "Client Brewer",
        "Commissioner",
        "Meadery",
        "Cidery",
        "Sake Producer",
        "Commercial Brewery",
        "Contract Brewery")),
      beverage_count = as.numeric(str_trim(beverage_count)))

# Inspect breweries dataframe
glimpse(breweries)
```

```
## Rows: 1,357
## Columns: 7
## $ name           <chr> "12 Bones Brewing", "1718 Brewing Ocracoke", "3rd Rock …
## $ url            <chr> "https://www.ratebeer.com/brewers/12-bones-brewing/4010…
## $ city           <chr> "Arden", "Ocracoke", "Trenton", "Winston-Salem", "Chero…
## $ type           <fct> Brewpub, Brewpub, Microbrewery, Client Brewer, Client B…
## $ established    <dbl> 2019, 2018, 2016, 2019, 2018, 2018, 2020, 2018, 2014, 2…
## $ beverage_count <dbl> 10, 34, 14, 4, 7, 12, 7, 18, 24, 17, 7, 148, 45, 12, 10…
## $ state          <fct> North Carolina, North Carolina, North Carolina, North C…
```

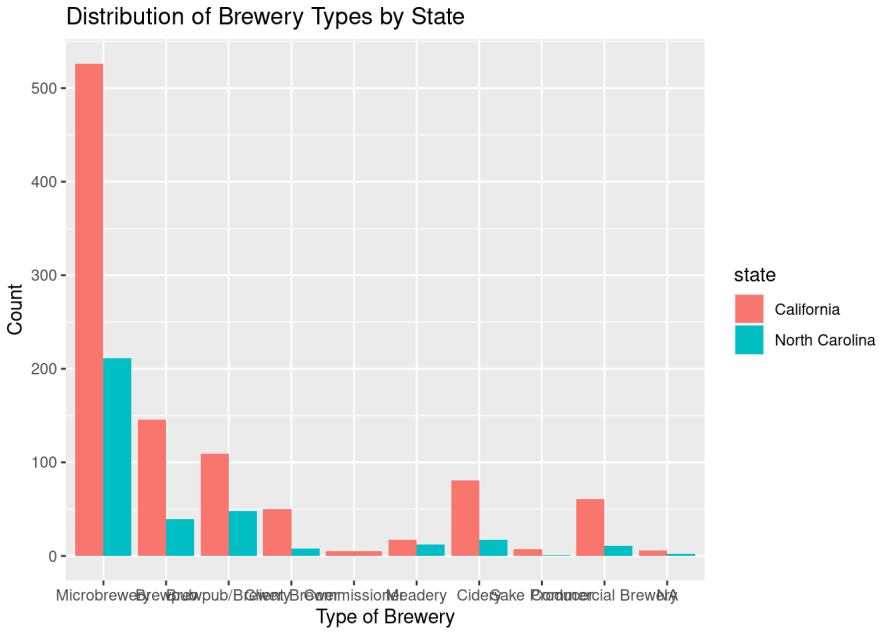
```
# Display data
breweries %>%
  head(20) %>%
  kable() %>%
  kable_styling() %>%
  scroll_box(width = "75%", height = "700px")
```

name	url	city	type	establish
12 Bones Brewing	https://www.ratebeer.com/brewers/12-bones-brewing/40107/ (https://www.ratebeer.com/brewers/12-bones-brewing/40107/)	Arden	Brewpub	20
1718 Brewing Ocracoke	https://www.ratebeer.com/brewers/1718-brewing-ocracoke/35300/ (https://www.ratebeer.com/brewers/1718-brewing-ocracoke/35300/)	Ocracoke	Brewpub	20
3rd Rock Brewing Company	https://www.ratebeer.com/brewers/3rd-rock-brewing-company/27344/ (https://www.ratebeer.com/brewers/3rd-rock-brewing-company/27344/)	Trenton	Microbrewery	20
638 Brewing Company	https://www.ratebeer.com/brewers/638-brewing-company/38962/ (https://www.ratebeer.com/brewers/638-brewing-company/38962/)	Winston-Salem	Client Brewer	20
7 Clans Brewing	https://www.ratebeer.com/brewers/7-clans-brewing/35021/ (https://www.ratebeer.com/brewers/7-clans-brewing/35021/)	Cherokee	Client Brewer	20
All Sevens Brewing	https://www.ratebeer.com/brewers/all-sevens-brewing/36948/ (https://www.ratebeer.com/brewers/all-sevens-brewing/36948/)	Asheville	Microbrewery	20
Altered State Brewing Company	https://www.ratebeer.com/brewers/altered-state-brewing-company/44848/ (https://www.ratebeer.com/brewers/altered-state-brewing-company/44848/)	Raleigh	Microbrewery	20

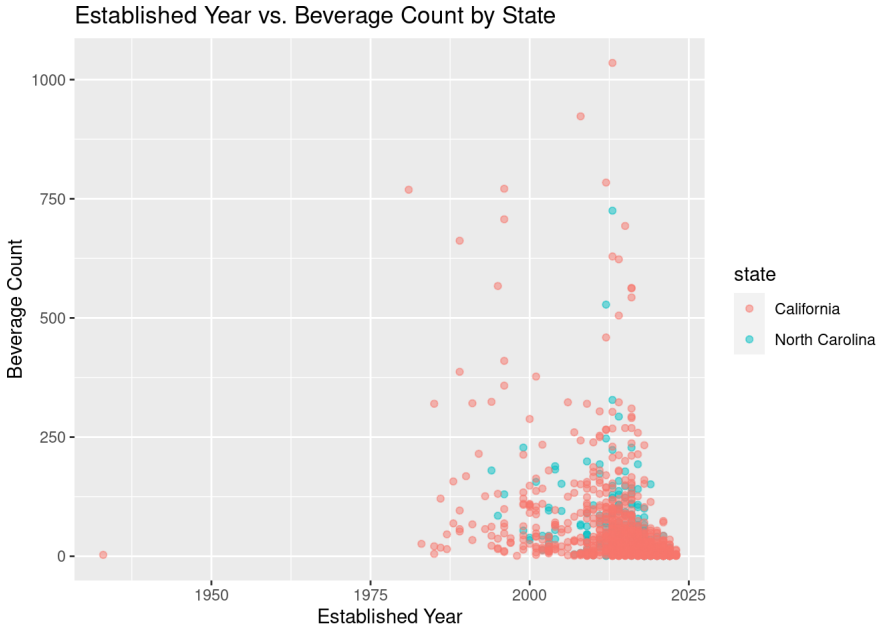
```
# create a histogram of the established year variable
breweries %>%
  ggplot(aes(x = established)) +
  geom_histogram(binwidth = 1, fill = "steelblue", color = "white") +
  labs(x = "Established Year",
       y = "Count",
       title = "Distribution of Established Year")
```




```
# create a bar chart of the type variable
breweries %>%
  ggplot(aes(x = type, fill = state)) +
  geom_bar(position = "dodge") +
  labs(x = "Type of Brewery",
       y = "Count",
       title = "Distribution of Brewery Types by State")
```




```
# create a scatterplot of the beverage count and established year variables
breweries %>%
  ggplot(aes(x = established, y = beverage_count, color = state)) +
  geom_point(alpha = 0.5) +
  labs(x = "Established Year",
       y = "Beverage Count",
       title = "Established Year vs. Beverage Count by State")
```



Mapping Setup

```
# Mapping Setup
register_google(key = "AIzaSyB14h_szuys3SaTv1WTaJ2WEDfGDN0sl-A")
# Checks to see if the API key as been registered
has_google_key()
```

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

Question 4

Update the breweries dataframe with the long and lat data for the cities in the dataframe. You are to do this in the most efficient way possible. Specifically, you are to look up the GPS location of a city once and only once by following the procedures discussed and demonstrated in class. Note: use the osm method for this question.

Inspect this dataframe and display at least 20 rows of data.

Once this step has been successfully completed, you are to save the resulting dataframe as breweries_long_lat.csv in your data folder. After the .csv is created, you are to do the following:

1. Comment out all the code in the code chunk.
2. You are to set the code chunk not be executed when knitted. However, the contents of the code chunk are to be displayed.
3. If you need to reload this data you are to use the .csv file to load the data.

```
## Create coordinates and then comment out the code

## Use osm method to update breweries dataframe
# breweries_coords <- breweries %>%
#   distinct(state, city)%>%
#   mutate(location = paste(city, state, sep = ", ")) %>%
#   tidygeocoder::geocode(location, method = "arcgis")
#
# write_csv(breweries_coords, file = "data/breweries_coords.csv")

## Create new datadrame by joining the coordinates with breweries
# breweries_long_lat <- breweries %>%
#   inner_join(breweries_coords)
#
## Write file to data folder
# write_csv(breweries_long_lat, file = "data/breweries_long_lat")
```

```
# Load data
breweries_coords <- read_csv("data/breweries_coords.csv")
breweries_long_lat <- read_csv("data/breweries_long_lat")

# Display resulting dataframe
breweries_long_lat %>%
  head(20) %>%
  kable() %>%
  kable_styling() %>%
  scroll_box(width = "100%", height = "700px")
```

name	url	city	type	established	beverage_count	state	l
12 Bones Brewing	https://www.ratebeer.com/brewers/12-bones-brewing/40107/ (https://www.ratebeer.com/brewers/12-bones-brewing/40107/)	Arden	Brewpub	2019	10	North Carolina	A N C
1718 Brewing Ocracoke	https://www.ratebeer.com/brewers/1718-brewing-ocracoke/35300/ (https://www.ratebeer.com/brewers/1718-brewing-ocracoke/35300/)	Ocracoke	Brewpub	2018	34	North Carolina	O N C
3rd Rock Brewing Company	https://www.ratebeer.com/brewers/3rd-rock-brewing-company/27344/ (https://www.ratebeer.com/brewers/3rd-rock-brewing-company/27344/)	Trenton	Microbrewery	2016	14	North Carolina	T N C
638 Brewing Company	https://www.ratebeer.com/brewers/638-brewing-company/38962/ (https://www.ratebeer.com/brewers/638-brewing-company/38962/)	Winston-Salem	Client Brewer	2019	4	North Carolina	W S N C
7 Clans Brewing	https://www.ratebeer.com/brewers/7-clans-brewing/35021/ (https://www.ratebeer.com/brewers/7-clans-brewing/35021/)	Cherokee	Client Brewer	2018	7	North Carolina	C N C
All Sevens Brewing	https://www.ratebeer.com/brewers/all-sevens-brewing/36948/ (https://www.ratebeer.com/brewers/all-sevens-brewing/36948/)	Asheville	Microbrewery	2018	12	North Carolina	A N C
Altered State Brewing Company	https://www.ratebeer.com/brewers/altered-state-brewing-company/44848/ (https://www.ratebeer.com/brewers/altered-state-brewing-company/44848/)	Raleigh	Microbrewery	2020	7	North Carolina	R N C

Question 5

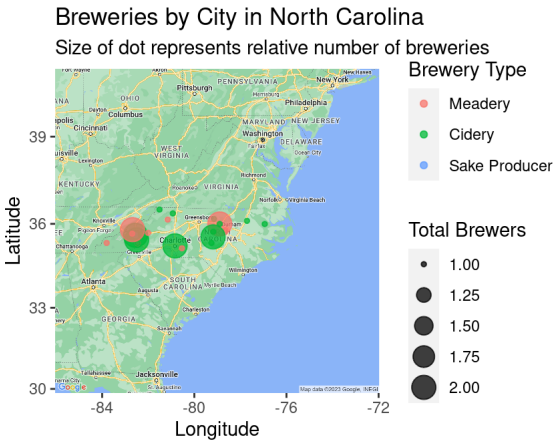
You are to create a set of side by side maps of the North Carolina and California using an appropriate map type that maps the total number of meaderies, cideries, and sake producers by city where the size of the dot is a relative indicator of the total number of brewers in each city by type of brewer. You are not to include any of the beer brewery types. The dot size scale should be the same on both maps.

```
# Filter for brewery types except beer
brewery_types <- c("Meadery", "Cidery", "Sake Producer")

# Create new data frame with the number of each brewery type in each city
breweries_by_city <- breweries %>%
  filter(type %in% brewery_types) %>%
  group_by(city, type) %>%
  summarize(num_breweries = n())

# Join with map data to get coordinates
brewery_map_data <- breweries_by_city %>%
  inner_join(breweries_coords, by = "city")
```

```
## Warning in inner_join(., breweries_coords, by = "city"): Detected an unexpected many-to-many relationship betw
een `x` and `y`.
## ℹ Row 58 of `x` matches multiple rows in `y`.
## ℹ Row 6 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Create data frame for North Carolina
nc_brewery_map_data <- brewery_map_data %>%
  filter(state == "North Carolina")

# Create data frame for California
ca_brewery_map_data <- brewery_map_data %>%
  filter(state == "California")

# Plot map for North Carolina and specify details
nc_plot <- get_map(
  location = 'North Carolina', zoom = 6,
  source = "google", maptype = "roadmap") %>%
  ggmap(base_layer = ggplot(
    nc_brewery_map_data, aes(
      x = long, y = lat,
      size = num_breweries, color = type))) +
  geom_point(alpha = .75) +
  labs(title = "Breweries by City in North Carolina",
       subtitle = "Size of dot represents relative number of breweries",
       x = "Longitude",
       y = "Latitude",
       size = "Total Brewers",
       color = "Brewery Type")

# Plot map for California and specify details
ca_plot <- get_map(
  location = 'California', zoom = 6,
  source = "google", maptype = "roadmap") %>%
  ggmap(base_layer = ggplot(
    ca_brewery_map_data, aes(
      x = long, y = lat,
      size = num_breweries, color = type))) +
  geom_point(alpha = .75) +
  labs(title = "Breweries by City in California",
       subtitle = "Size of dot represents relative number of breweries",
       x = "Longitude",
       y = "Latitude",
       size = "Total Brewers",
       color = "Brewery Type")

# display side-by-side maps
grid.arrange(nc_plot, ca_plot, ncol = 2)
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



Question 6

You are to create a set of side by side zoomed maps of the Los Angels and San Francisco metro areas (see Wikipedia) using an appropriate map type where you map the total number of beer brewery types, exclude contract and commercial breweries, by city in each metro area. The dots are to be color coded by brewery type. The size of the dot is be relative to the total number of brewers by type in each city. The size scale should be the same on all maps.

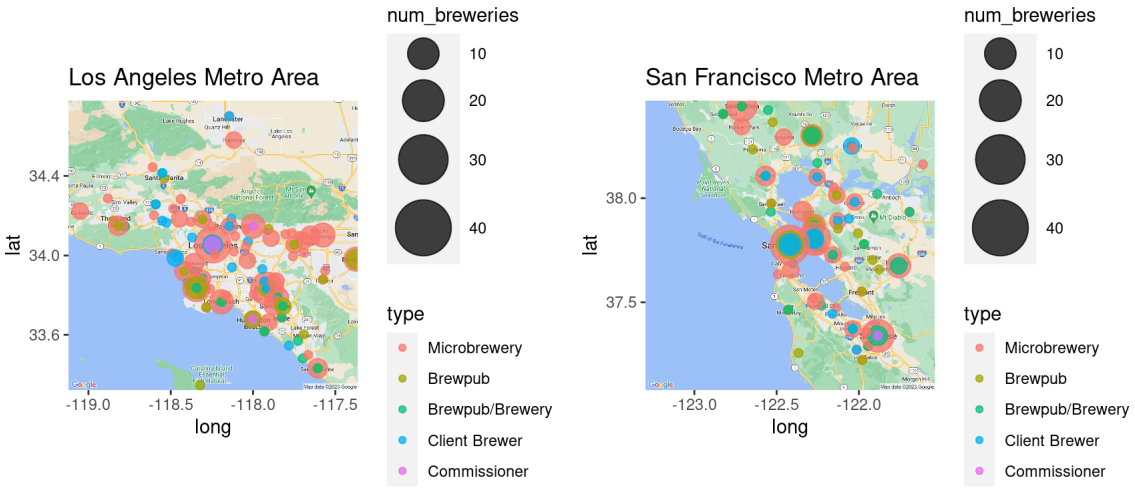
```
# Filter for beer brewery types - contract & commercial
beer_breweries <- c("Microbrewery",
                    "Brewpub",
                    "Brewpub/Brewery",
                    "Client Brewer",
                    "Commissioner")

# Create new data frame with the number of each brewery type in each city
california_beer_breweries <- breweries %>%
 filter(type %in% beer_breweries,
        state == "California") %>%
  group_by(city, type) %>%
  summarize(num_breweries = n())

# Join with map data to get coordinates
beer_brewery_map_data <- california_beer_breweries %>%
  inner_join(breweries_coords, by = "city")
```

```
## Warning in inner_join(., breweries_coords, by = "city"): Detected an unexpected many-to-many relationship betw
een `x` and `y`.
## ℹ Row 76 of `x` matches multiple rows in `y`.
## ℹ Row 368 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
# Plot map of Los Angeles and define specifics
la_beer_map <-
  get_map(location = "Los Angeles, California", zoom = 9,
          source = "google", maptype = "roadmap") %>%
 ggmap(base_layer = ggplot(
    beer_brewery_map_data, aes(
      x =long, y = lat, size = num_breweries, color = type))) +
  geom_point(alpha = .75) +
  scale_size(range = c(2, 15)) +
  labs(title = "Los Angeles Metro Area")

# Plot map of San Francisco and define specifics
sf_beer_map <-
  get_map(location = "San Francisco, California", zoom = 9,
          source = "google", maptype = "roadmap") %>%
  ggmap(base_layer = ggplot(
    beer_brewery_map_data, aes(
      x =long, y = lat, size = num_breweries, color = type))) +
  geom_point(alpha = .75) +
  scale_size(range = c(2, 15)) +
  labs(title = "San Francisco Metro Area")

# display side-by-side maps
grid.arrange(la_beer_map, sf_beer_map, ncol = 2, nrow = 1)
```

```
## Warning: Removed 336 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 358 rows containing missing values (`geom_point()`).
```



Question 7

Note: The below cannot be accomplished in a single pipe. To do this successfully, you will need multiple pipes and multiple code chunks. You are to write your code as succinctly as possible.

The following analysis is only to be done only for cities in California that have 10 or more breweries, excluding contract and commercial breweries.

For breweries in these cities, you are to use Octoparse to extract each of the brewery's address information from the brewery's detailed page. You can get to this page manually by clicking on the name of the brewery on the list of breweries page. For instance the address of 13 Point Brewing in California is 8035 Broadway, Lemon Grove, California, 91945, United States.

This process can be automated in Octoparse by using the Batch URL Input method. Follow the instructions carefully in this tutorial, You should use the first method in the tutorial, Import URLs from a file. You should export a list URLs as a .csv file of breweries for which you need addresses. All your work in R should be documented and run in a code chunk. When you create the file of scraped addresses in Octoparse be sure you also output the name of the brewery for each address, as you will need a way to join the file to existing dataframes.

Import this file into your data folder and create a dataframe, ca_addresses, containing this data. Update this dataframe with the long and lat data for each address in the dataframe. Use the census method instead of osm for this question It is considerably faster. With the census method you only need to use the address argument. The address argument is the full address, e.g. 2350 Hendersonville Rd, Arden, North Carolina, 28704, United States.

Note some address may have errors that return an NA when the GPS coordinates are looked up. It may be due to a misspelling of a word such suite. An address that appends a unit location to the address number, such as 141-b will return an NA. To clean up remove the “-b”. A few addresses may just not be found. You are to clean up the address data the best you can. You need to document these efforts. Once you have a clean ca_addresses dataframe, output all the data in so that the user can scroll or click through the data while displaying only 10 to 15 rows at a time.

Once the above has been successfully completed, you are to save the resulting dataframe as ca_addresses.csv in your data folder. After the .csv is created, you are to do the following:

1. Comment out all the code in the code chunk.
2. You are to set the code chunk not be executed when knitted. However, the contents of the code chunk are to be displayed.
3. If you need to reload this data you are to use the .csv file to load the data.

Now for the reason we are doing the above. In many cities, craft breweries of all types seem to be located in the same geographic area. Our goal is to determine the city with the highest concentration of breweries relative to one another and then generate a map of the city indicating the location of each brewery and its type.

In a new code chunk you will need to determine the distance of each brewery to the other breweries in the city. You should not use an anti join to accomplish this as the problem is different from a similar problem we worked on in the past. An inner join would be a more efficient choice. Next you need to calculate the mean distance between all breweries in each city. Do not include breweries that are joined to themselves in the calculation. You are to display all your results. The city with the highest concentration of breweries is to be at the top of the list.

Finally, you are to produce a zoomed map of the city that shows the location of each brewery included in mean distance calculation and its type. Extra points if you also display the name of the brewery on the map. The map should be as large as possible.

```
# Filter breweries to those in California with 10 or more non-commercial/non-contract breweries
ca_breweries <- breweries %>%
  filter(state == "California",
         type != "Commercial",
         type != "Contract") %>%
  group_by(city) %>%
  summarize(num_breweries = n()) %>%
  filter(num_breweries >= 10)

# Extract URLs for each brewery in the filtered cities
ca_brewery_urls <- breweries %>%
  filter(city %in% ca_breweries$city) %>%
  select(url)

# Export URLs as a CSV file
write_csv(ca_brewery_urls, file = "data/ca_brewery_urls.csv")
```

```
# ca_addresses <- read_csv("data/ca_addresses.csv")
#
# ca_addresses <- ca_addresses %>%
#   rename(
#     name = Text,
#     address = Text1) %>%
#   mutate(address = str_replace(address, "-b", ""))
#
# ca_addresses_coords <- ca_addresses %>%
#   geocode(address = address, method = "census")
#
# write_csv(ca_addresses_coords, file = "data/ca_addresses.csv")
```

```
# Load dataframe
ca_addresses_coords <- read_csv("data/ca_addresses.csv")

# Calculate pairwise distances between breweries in each city
brewery_distances <- ca_addresses_coords %>%
  group_by(city) %>%
  inner_join(ca_addresses_coords, by = "city") %>%
  filter(address.x != address.y) %>%
  mutate(distance = distm(c(lon.x, lat.x), c(lon.y, lat.y), fun = distHaversine)/1000) %>%
  select(city, name.x, name.y, distance)
```

Database Connection

```
db = dbConnect(MySQL(),
     user = 'ofx_user',
     password = 'TestyTester#2021',
     dbname = 'ofx',
     host = 'ballenger.wlu.edu')

knitr::opts_knit$set(sql.max.print = -1)
```

Load Data

```
# load product table
rs_product <- dbSendQuery(db, "SELECT * FROM product")
# fetch product results
product <- fetch(rs_product, n = -1)

# load category table
rs_category <- dbSendQuery(db, "SELECT * FROM category")
# fetch category results
category <- fetch(rs_category, n = -1)

# load buyer table
rs_buyer <- dbSendQuery(db, "SELECT * FROM buyer")
# fetch buyer results
buyer <- fetch(rs_buyer, n = -1)

# load orders table
rs_orders <- dbSendQuery(db, "SELECT * FROM orders")
# fetch orders results
orders <- fetch(rs_orders, n = -1)

# load location table
rs_location <- dbSendQuery(db, "SELECT * FROM location")
# fetch location results
location <- fetch(rs_location, n = -1)

# load order_product table
rs_order_product <- dbSendQuery(db, "SELECT * FROM order_product")
# fetch order_product results
order_product <- fetch(rs_order_product, n = -1)
```

Combine SQL data innto one dataframw

```
#Start by creating athe first element of the dataframe, buyer + orders
order_buyer <- orders %>%
  inner_join(buyer)
```

```
## Joining with `by = join_by(Buyer_ID)`
```

```
# Add the other tables loaded to the dataframme
ofx <- order_buyer %>%
  inner_join(location) %>%
  inner_join(order_product) %>%
  inner_join(product) %>%
  inner_join(category)
```

```
## Joining with `by = join_by(Postal_Code)`
## Joining with `by = join_by(Order_ID)`
## Joining with `by = join_by(Product_ID)`
## Joining with `by = join_by(Sub_Category)`
```

```
# view the resulting dataframe
ofx %>%
  head(20) %>%
  kable() %>%
  kable_styling() %>%
  scroll_box(width = "100%", height = "700px")
```

Order_ID	Order_Date	Ship_Date	Ship_Mode	Buyer_ID	Postal_Code	Last_Name	First_Name	Type	City	Stat
CA-2011-100006	2011-09-07	2011-09-13	Standard Class	DK-13375	10024	Kane	Dennis	Consumer	New York City	New York
CA-2011-100090	2011-07-08	2011-07-12	Standard Class	EB-13705	94122	Braxton	Ed	Corporate	San Francisco	Cali
CA-2011-100090	2011-07-08	2011-07-12	Standard Class	EB-13705	94122	Braxton	Ed	Corporate	San Francisco	Cali
CA-2011-100293	2011-03-14	2011-03-18	Standard Class	NF-18475	32216	Französisch	Neil	Home Office	Jacksonville	Flor
CA-2011-100328	2011-01-29	2011-02-04	Standard Class	JC-15340	10024	Cacioppo	Jasper	Consumer	New York City	New York
CA-2011-100363	2011-04-08	2011-04-15	Standard Class	JM-15655	85301	Mitchum	Jim	Corporate	Glendale	Ariz
CA-2011-100363	2011-04-08	2011-04-15	Standard Class	JM-15655	85301	Mitchum	Jim	Corporate	Glendale	Ariz
CA-2011-	2011-05-25	2011-05-29	Standard Class	BW-11065	10035	Weirich	Barry	Consumer	New York City	New York

Geocoding Location Data

You are to manage your use of geocoding location data. You are to gather the geocoding data for each state & city combination and save it as a csv file in your data folder in your RStudio Cloud project. You should only need to do this once. Subsequent reloading of your geocoding & location data is to be done using the saved csv file.

```
# geocode the data, then comment out
ofx_coords <- ofx %>%
  distinct(City, State) %>%
  geocode(city = City, state = State)

#write to csv, then comment out
write_csv(ofx_coords, file = "data/ofxcoords.csv")

# IT took away my access? It repeadelty said my access was blocked and I was forbidden
```

```
# Load ofx coordinate data
ofx_coords <- read_csv("data/ofx_coords.csv")

# Load location data into df
ofx <- ofx %>%
  inner_join(ofx_coords)

# Display ofx df with location data
ofx %>%
  head(20) %>%
  kable() %>%
  kable_styling() %>%
  scroll_box(width = "100%", height = "700px")
```

The Assignment

You have been hired as a data analytics consultant by OFX. Your engagement letter states your are to probe into the provided OFX data to see what business intelligence you can uncover. You are to clean up the data and column names as necessary. You are to inspect the data and conduct graphical exploratory data analysis (EDA). You are to document these efforts and write a brief description (2 to 4 sentences) of each pertinent finding.

Next you are to explore and analyze the data contained in the database and provide at least 6 key findings which may be summarized data tables and/or data visualizations. A single finding might consist of multiple tables and data visualizations. Your are to write a brief description of your findings. Be sure each finding is clearly labeled.

Inspecting the dataframe

```
# General functions to get a look at the dataframe and see if there's anything that stounds out or any issues tha
t need to be fixed
ofx %>%
  glimpse() %>%
  summary()
```

```
## Rows: 9,986
## Columns: 20
## $ Order_ID              <chr> "CA-2011-100006", "CA-2011-100090", "CA-2011-100…
## $ Order_Date            <chr> "2011-09-07", "2011-07-08", "2011-07-08", "2011-…
## $ Ship_Date             <chr> "2011-09-13", "2011-07-12", "2011-07-12", "2011-…
## $ Ship_Mode             <chr> "Standard Class", "Standard Class", "Standard Cl…
## $ Buyer_ID              <chr> "DK-13375", "EB-13705", "EB-13705", "NF-18475", …
## $ Postal_Code           <chr> "10024", "94122", "94122", "32216", "10024", "85…
## $ Last_Name             <chr> "Kane", "Braxton", "Braxton", "Französisch", "Ca…
## $ First_Name            <chr> "Dennis", "Ed", "Ed", "Neil", "Jasper", "Jim", "…
## $ Type                  <chr> "Consumer", "Corporate", "Corporate", "Home Offi…
## $ City                  <chr> "New York City", "San Francisco", "San Francisco…
## $ State                 <chr> "New York", "California", "California", "Florida…
## $ Region                <chr> "East", "West", "West", "South", "East", "West",…
## $ Product_ID            <chr> "TEC-PH-10002075", "FUR-TA-10003715", "OFF-BI-10…
## $ Quantity              <int> 3, 3, 6, 6, 1, 2, 3, 2, 3, 2, 3, 3, 6, 2, 4, 2, …
## $ Unit_Price            <dbl> 125.990, 167.496, 32.784, 15.176, 3.928, 1.184, …
## $ Discount              <dbl> 0.0, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.0, 0.3, 0.2…
## $ Gross_Profit_Per_Unit <dbl> 36.5371, -29.3118, 11.4744, 5.3116, 1.3257, 0.41…
## $ Product_Name          <chr> "AT&T EL51110 DECT", "Hon 2111 Invitation Series…
## $ Sub_Category          <chr> "Phones", "Tables", "Binders", "Paper", "Binders…
## $ Category              <chr> "Technology", "Furniture", "Office Supplies", "O…
```

```
##    Order_ID          Order_Date         Ship_Date          Ship_Mode
##  Length:9986        Length:9986        Length:9986        Length:9986
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##    Buyer_ID         Postal_Code         Last_Name          First_Name
##  Length:9986        Length:9986        Length:9986        Length:9986
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##      Type               City              State              Region
##  Length:9986        Length:9986        Length:9986        Length:9986
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##   Product_ID           Quantity       Unit_Price          Discount
##  Length:9986        Min.   : 1.00   Min.   :   0.336   Min.   :0.0000
##  Class :character   1st Qu.: 2.00   1st Qu.:   5.448   1st Qu.:0.0000
##  Mode  :character   Median : 3.00   Median :  16.267   Median :0.2000
##                     Mean   : 3.79   Mean   :  60.914   Mean   :0.1563
##                     3rd Qu.: 5.00   3rd Qu.:  63.926   3rd Qu.:0.2000
##                     Max.   :14.00   Max.   :3773.080   Max.   :0.8000
##  Gross_Profit_Per_Unit Product_Name       Sub_Category         Category
##  Min.   :-1320.0000    Length:9986        Length:9986        Length:9986
##  1st Qu.:    0.7202    Class :character   Class :character   Class :character
##  Median :    2.7626    Mode  :character   Mode  :character   Mode  :character
##  Mean   :    7.7946
##  3rd Qu.:    8.6900
##  Max.   : 1680.0000
```

Cleaning the data

```
# Make all variable names lowercase
ofx <- ofx %>%
  rename_with(tolower) %>%
  # convert the date variables to date format
  mutate(order_date = ymd(order_date),
         ship_date = ymd(ship_date),
         # convert postal code to a numeric value
         postal_code = as.numeric(postal_code),
         # factor categorical data
         # THIS IS DEBATABLE
         ship_mode = factor(ship_mode),
         type = factor(type),
         region = factor(region),
         sub_category = factor(sub_category),
         category = factor(category))

# Display changes made to the dataframe
ofx %>%
  glimpse()
```

```
## Rows: 9,986
## Columns: 20
## $ order_id              <chr> "CA-2011-100006", "CA-2011-100090", "CA-2011-100…
## $ order_date            <date> 2011-09-07, 2011-07-08, 2011-07-08, 2011-03-14,…
## $ ship_date             <date> 2011-09-13, 2011-07-12, 2011-07-12, 2011-03-18,…
## $ ship_mode             <fct> Standard Class, Standard Class, Standard Class, …
## $ buyer_id              <chr> "DK-13375", "EB-13705", "EB-13705", "NF-18475", …
## $ postal_code           <dbl> 10024, 94122, 94122, 32216, 10024, 85301, 85301,…
## $ last_name             <chr> "Kane", "Braxton", "Braxton", "Französisch", "Ca…
## $ first_name            <chr> "Dennis", "Ed", "Ed", "Neil", "Jasper", "Jim", "…
## $ type                  <fct> Consumer, Corporate, Corporate, Home Office, Con…
## $ city                  <chr> "New York City", "San Francisco", "San Francisco…
## $ state                 <chr> "New York", "California", "California", "Florida…
## $ region                <fct> East, West, West, South, East, West, West, East,…
## $ product_id            <chr> "TEC-PH-10002075", "FUR-TA-10003715", "OFF-BI-10…
## $ quantity              <int> 3, 3, 6, 6, 1, 2, 3, 2, 3, 2, 3, 3, 6, 2, 4, 2, …
## $ unit_price            <dbl> 125.990, 167.496, 32.784, 15.176, 3.928, 1.184, …
## $ discount              <dbl> 0.0, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.0, 0.3, 0.2…
## $ gross_profit_per_unit <dbl> 36.5371, -29.3118, 11.4744, 5.3116, 1.3257, 0.41…
## $ product_name          <chr> "AT&T EL51110 DECT", "Hon 2111 Invitation Series…
## $ sub_category          <fct> Phones, Tables, Binders, Paper, Binders, Fastene…
## $ category              <fct> Technology, Furniture, Office Supplies, Office S…
```

Checking for NAs

```
# Checks for NAs for all columns and totals the number found in each column
ofx %>%
  summarise(across(everything(), ~ sum(is.na(.)))) %>%
  kable() %>%
  kable_styling()
```

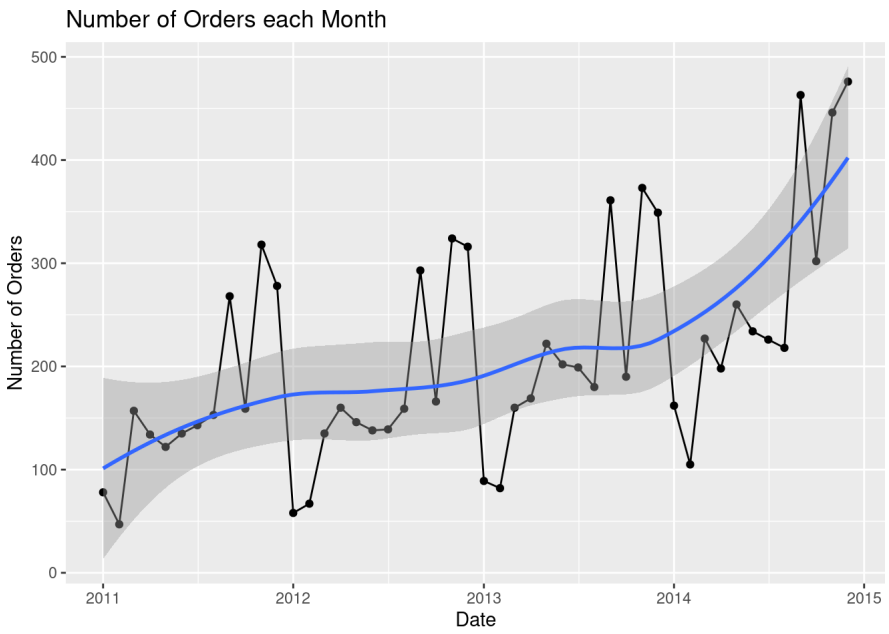
order_id	order_date	ship_date	ship_mode	buyer_id	postal_code	last_name	first_name	type	city	state	region	product_id	quantity
0	0	0	0	0	0	8	0	0	0	0	0	0	0

Graphical Exploratory Data Analysis

Starting out with the average number of orders by month can tell us a lot about the data.

```
ofx %>%
  mutate(order_date_month = round_date(
    order_date, unit = "month")) %>%
  group_by(order_date_month) %>%
  summarise(nbr_orders = n()) %>%
  ggplot(aes(x = order_date_month, y = nbr_orders))+
    geom_point()+
    geom_line()+
  geom_smooth()+
  labs(title = "Number of Orders each Month",
       x = "Date",
       y = "Number of Orders")
```

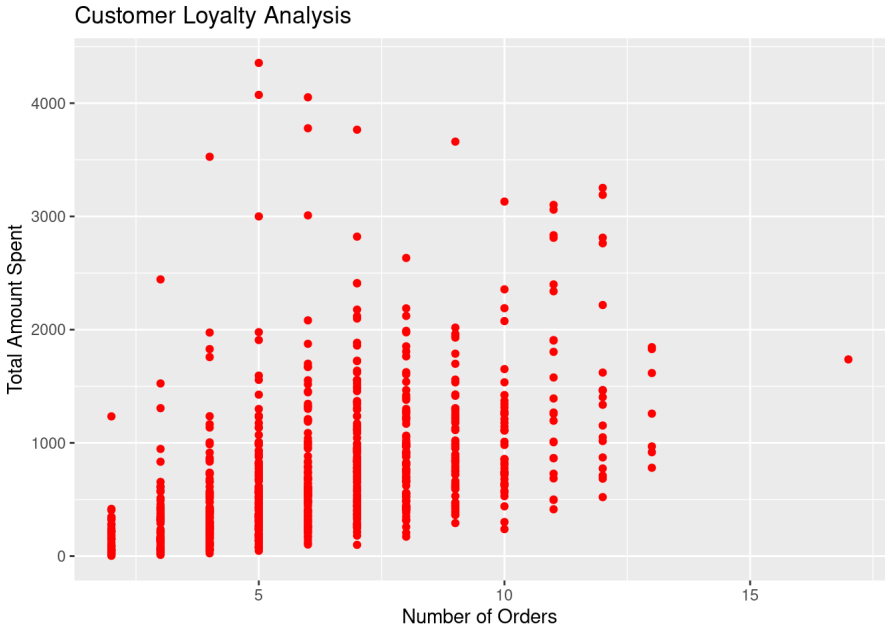
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



As a whole, we're seeing a general increase with each passing month. This shows that their business is doing well.

Another interesting aspect to look at is customer loyalty and how much they spend.

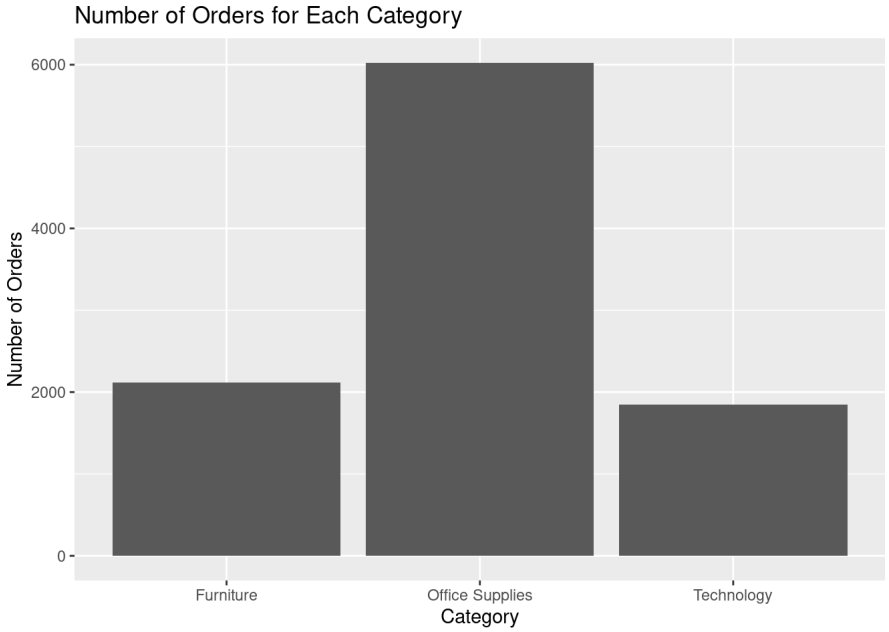
```
ofx %>%
  group_by(buyer_id) %>%
  summarise(nbr_orders = n_distinct(order_id),
            total_spent = sum(unit_price)) %>%
  filter(nbr_orders > 1) %>%
  ggplot(aes(x = nbr_orders, y = total_spent)) +
  geom_point(color = "red") +
  labs(title = "Customer Loyalty Analysis",
       x = "Number of Orders",
       y = "Total Amount Spent")
```



This scatter plot shows us the total amount that customers spend against their number of orders. It looks like the middle ground (maybe small businesses?) creates the highest profit margins.

Finally, I think it'd be interesting to see which category has the most orders.

```
ofx %>%
  group_by(category) %>%
  summarise(nbr_orders = n()) %>%
  ggplot(aes(x = category, y = nbr_orders)) +
  geom_col()+
  labs(title = "Number of Orders for Each Category",
       x = "Category",
       y = "Number of Orders")
```



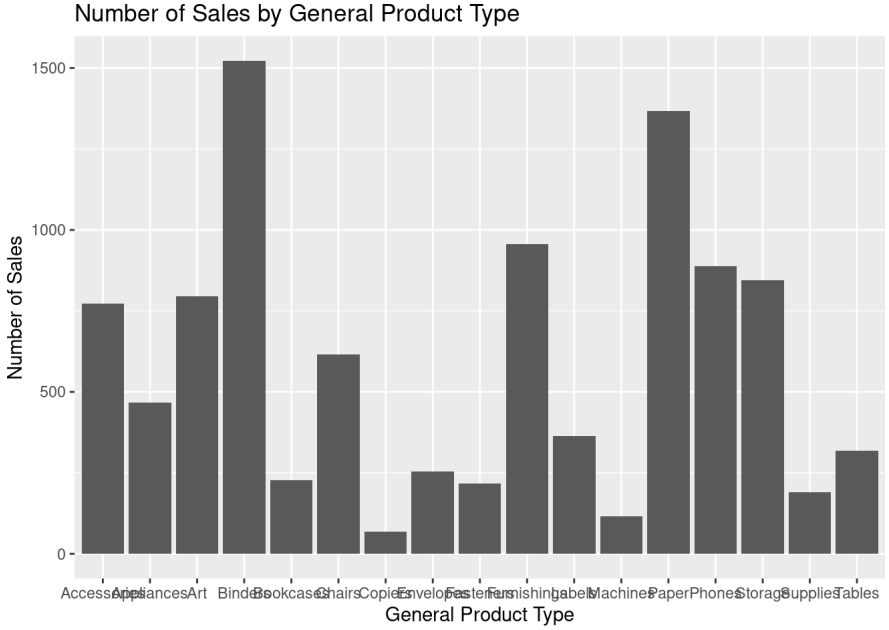
Its clear from the graph that office supplies seem to be their number one seller by far. While they are successful with selling furniture and technology, office supplies is where they thrive.

Key Finding 1

What products or categories of products are under performing?

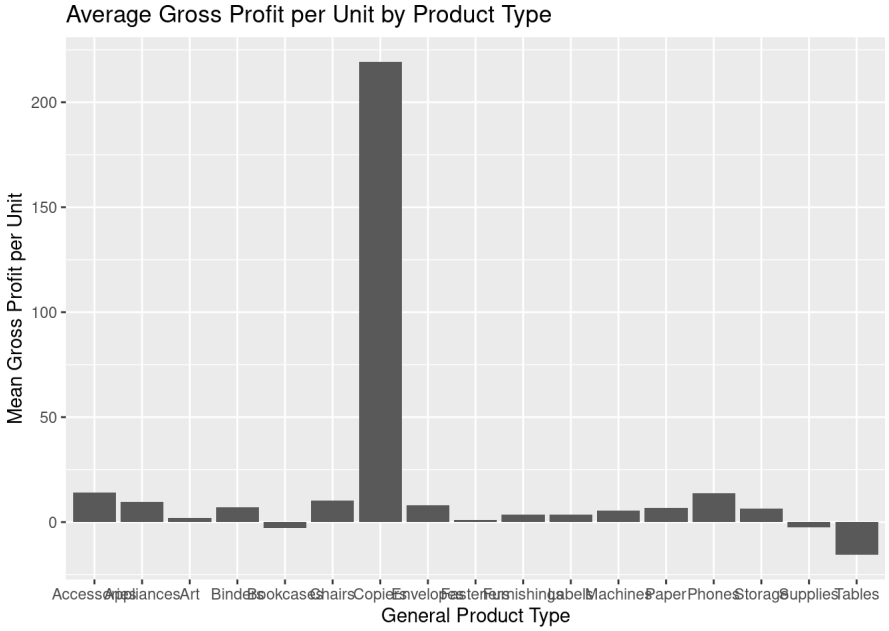
First we want to look at the number of sales for each product type.

```
ofx %>%
  group_by(sub_category) %>%
  ggplot(aes(x = sub_category)) +
  geom_bar()+
  labs(title = "Number of Sales by General Product Type",
       x = "General Product Type",
       y = "Number of Sales")
```



While this graph easily tells us who the biggest sellers are, we need to look at profit to confirm what we see in this graph. Therefore, we need to look at each product's gross profit.

```
ofx %>%
  group_by(sub_category) %>%
  summarise(avg_grs_profit = mean(gross_profit_per_unit)) %>%
  ggplot(aes(x = sub_category, y = avg_grs_profit))+
  geom_col()+
  labs(title = "Average Gross Profit per Unit by Product Type",
       x = "General Product Type",
       y = "Mean Gross Profit per Unit")
```



While it seemed before that the company was loosing out on printers, looking at the profits tells us the opposite. In fact, this graph shows us that they're loosing money on tables and supplies instead.

Key Finding 2

What product correlates with the most profit?

```
ofx %>%
  filter(discount == 0) %>%
  group_by(sub_category) %>%
  summarise(gross_profit_from_product = sum(gross_profit_per_unit*quantity)) %>%
  arrange(desc(gross_profit_from_product)) %>%
  kable() %>%
  kable_styling() %>%
  scroll_box(width = "100%", height = "400px")
```

sub_category	gross_profit_from_product
Binders	39273.0184
Copiers	35556.1774
Accessories	35289.2399
Phones	34365.2006
Machines	27137.8196
Storage	25370.2966
Paper	25296.9117
Appliances	23183.7319
Chairs	21933.0996
Furnishings	16764.9187

According to this table binders are the most profitable item with copiers in a close second.

Key Finding 3

Are there any odd/high spenders.

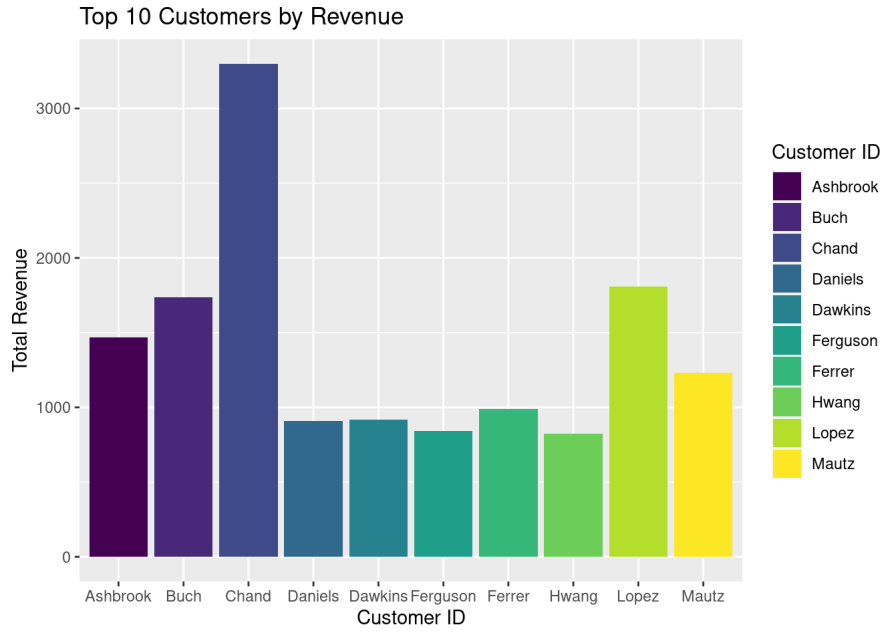
```
ofx %>%
  group_by(state) %>%
  summarise(avg_order_size = mean(unit_price)) %>%
  arrange(desc(avg_order_size))
```

```
## # A tibble: 49 × 2
##    state        avg_order_size
##    <chr>                 <dbl>
##  1 Wyoming               401.
##  2 Vermont               163.
##  3 Nevada                106.
##  4 Montana                96.9
##  5 Rhode Island           94.0
##  6 Minnesota              93.2
##  7 Indiana                88.0
##  8 Missouri               86.8
##  9 Delaware               83.6
## 10 Michigan               82.7
## # … with 39 more rows
```

According to this table, Wyoming has a much higher average order size than any other state by a hefty amount

```
ofx %>%
  group_by(last_name) %>%
  summarise(total_revenue = sum(gross_profit_per_unit)) %>%
  arrange(desc(total_revenue)) %>%
  mutate(percent_total = total_revenue / sum(total_revenue)) %>%
  top_n(10) %>%
  ggplot(aes(x = factor(last_name), y = total_revenue, fill = factor(last_name))) +
  geom_col() +
  scale_fill_viridis_d() +
  labs(title = "Top 10 Customers by Revenue",
       x = "Customer ID",
       y = "Total Revenue",
       fill = "Customer ID")
```

```
## Selecting by percent_total
```



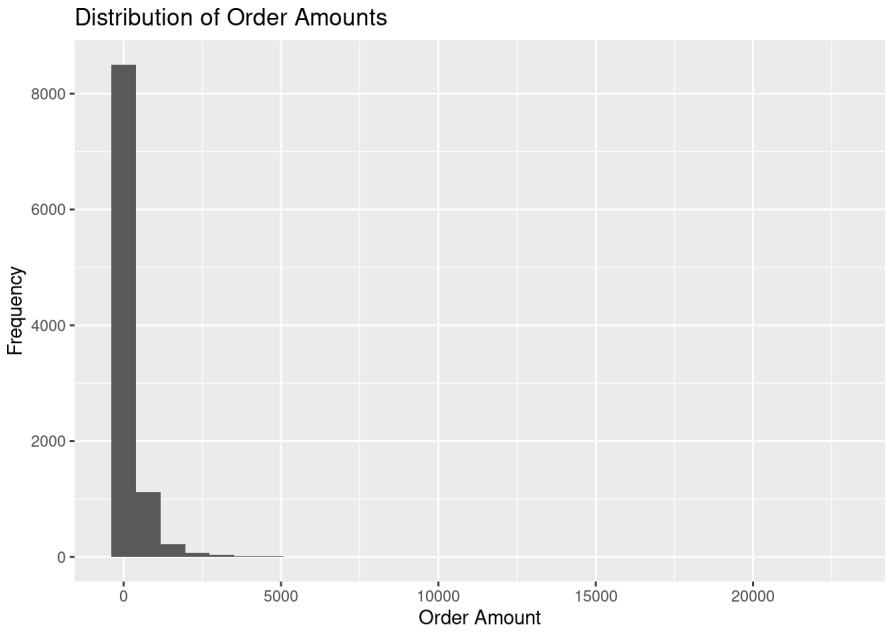
Similar to Wyoming, these 10 customers purchase a lot from the store and contribute greatly to their revenue.

Key Finding 4

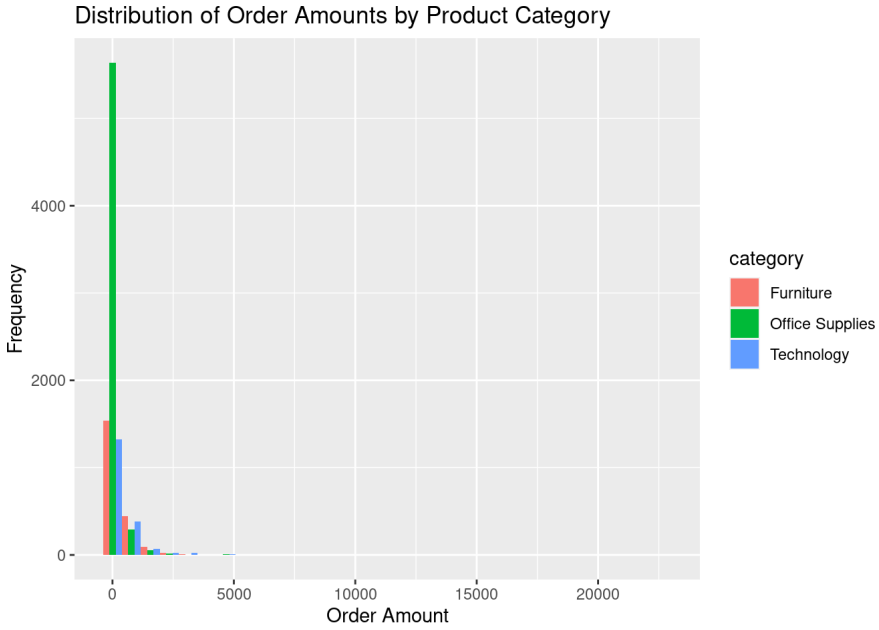
What is the distribution of order amounts for the firm and how does it vary by product category?

```
# create a new variable for order amounts
ofx <- ofx %>%
  mutate(order_amount = quantity * unit_price)

# plot the distribution of order amounts
ggplot(ofx, aes(x = order_amount)) +
  geom_histogram() +
  labs(title = "Distribution of Order Amounts", x = "Order Amount", y = "Frequency")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# compare the distribution of order amounts by product category
ggplot(ofx, aes(x = order_amount, fill = category)) +
  geom_histogram(bins = 30, position = "dodge") +
  labs(title = "Distribution of Order Amounts by Product Category", x = "Order Amount", y = "Frequency")
```



It's clear from both graphs that smaller orders and office supplies are the most frequently distributed.

Key Finding 5

Based on the given data, we can find the top selling products by revenue and the percentage of the total revenue they contribute.

```
ofx %>%
  group_by(product_name) %>%
  summarise(total_revenue = sum(unit_price * quantity)) %>%
  arrange(desc(total_revenue)) %>%
  mutate(revenue_percentage = total_revenue / sum(total_revenue) * 100) %>%
  head(10)
```

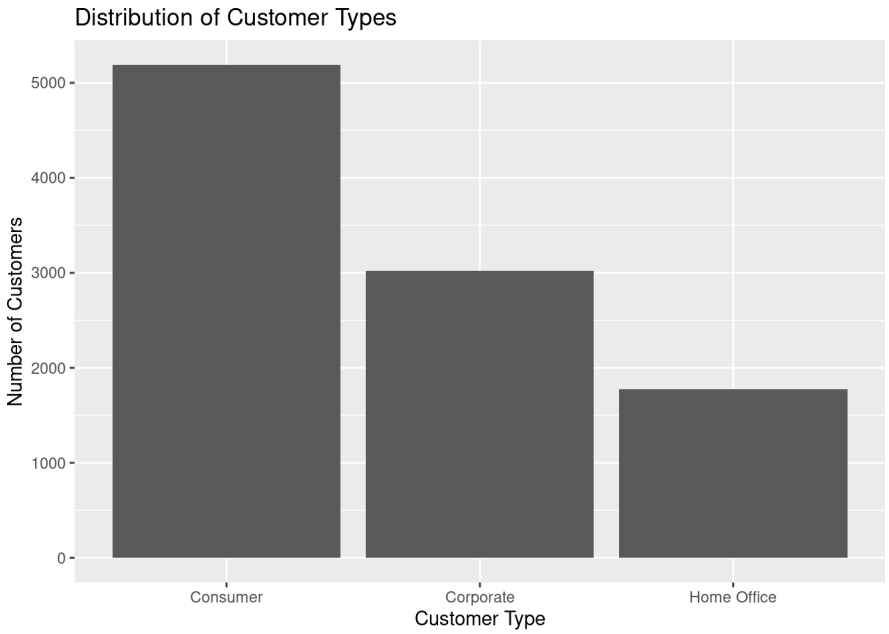
```
## # A tibble: 10 × 3
##    product_name                                              total…¹ reven…²
##    <chr>                                                       <dbl>   <dbl>
##  1 "Canon imageCLASS 2200 Advanced Copier"                     61600.   2.68
##  2 "Fellowes PB500 Electric Punch Plastic Comb Binding Machine …  27453.   1.20
##  3 "Cisco TelePresence System EX90 Videoconferencing Unit"     22638.   0.986
##  4 "HON 5400 Series Task Chairs for Big and Tall"              21871.   0.953
##  5 "GBC DocuBind TL300 Electric Binding System"                19823.   0.864
##  6 "GBC Ibimaster 500 Manual ProClick Binding System"          19024.   0.829
##  7 "Hewlett Packard LaserJet 3310 Copier"                      18840.   0.821
##  8 "HP Designjet T520 Inkjet Large Format Printer - 24\" Color"   18375.   0.800
##  9 "GBC DocuBind P400 Electric Binding System"                 17965.   0.783
## 10 "High Speed Automatic Electric Letter Opener"               17030.   0.742
## # … with abbreviated variable names ¹total_revenue, ²revenue_percentage
```

This code will group the data by the product names and then calculate the total revenue for each product by multiplying the price by the quantity. It will then arrange the data in descending order based on the total revenue and calculate the percentage of revenue each product contributes. Finally, it will output the top 10 products based on their revenue and the percentage of the total revenue they contribute. Here we see that the canon printer alone is a huge contributer to their revenue.

Key Finding 6

What is the most common type of shopper at OFX?

```
ofx %>%
  ggplot(aes(x = type))+
  geom_bar()+
  labs(title = "Distribution of Customer Types",
       x = "Customer Type",
       y = "Number of Customers")
```



It seems as though most of OFX's customers are everyday consumers, meaning it wouldn't be a bad idea to lean heavier into that side of marketing.

Project Log

https://stackoverflow.com/questions/29587881/increase-plot-size-width-in-ggplot2 (https://stackoverflow.com/questions/29587881/increase-plot-size-width-in-ggplot2) : Looked up how to increase the size/output of my plot

The Pledge

On my honor, I have neither given nor recieved any unacknowledged aid on this assignment. Mary Ardoin