Lab 6

Alyssa Andrichik

Math 241, Week 9

```
# Put all necessary libraries here
library(tidyverse)
library(lubridate)
library(httr)
library(glue)
library(stringr)
library(tidytext)
```

Due: Thursday, April 1st at 8:30am

Goals of this lab

1. Practice iteration with for loops and `purrr:map()`.
2. Practice converting lists to data frames.
3. Practice working with dates using `lubridate`.
4. Discover the issues with found data.
5. Practice wrangling factors with `forcats`.
6. Practice locating and acting on patterns in strings using regular expressions with `stringr`.
7. Practice tidying and analyzing text data with `tidytext`.

Problem 1: Loop De Loop

I bet several of us have watched more movies over the past year than we usually would so let's practice reducing duplication with a movies example. For this problem we are going to be pulling data from the OMDB API. Go here to set-up your free API key.

a. Insert your API key.

```
# Insert key
omdb_key <- "4cc164a0"
```

b. Below I am using the `httr` package to pull and parse the data for *Knives Out*. Modify the code to pull and parse the data for *Office Space*. Notice: I am also using `glue` to add my key to the url.

```
office_space_pull <- GET(glue("http://www.omdbapi.com/?i=tt3896198&apikey={omdb_key}"),
                  query = list(t = "Office Space",
                               y = 1999,
                               plot = "short",
                               r = "json"))

office_space <- content(office_space_pull, as = "parsed", type = "application/json")
glimpse(office_space)
```

```
## List of 25
##  $ Title     : chr "Office Space"
```

```
##  $ Year      : chr "1999"
##  $ Rated     : chr "R"
##  $ Released  : chr "19 Feb 1999"
##  $ Runtime   : chr "89 min"
##  $ Genre     : chr "Comedy"
##  $ Director  : chr "Mike Judge"
##  $ Writer    : chr "Mike Judge (Milton animated shorts), Mike Judge (screenplay)"
##  $ Actors    : chr "Ron Livingston, Jennifer Aniston, David Herman, Ajay Naidu"
##  $ Plot      : chr "Three company workers who hate their jobs decide to rebel against their greedy bo
##  $ Language  : chr "English"
##  $ Country   : chr "USA"
##  $ Awards    : chr "1 win & 2 nominations."
##  $ Poster    : chr "https://m.media-amazon.com/images/M/MV5BOTA5MzQ3MzI1NV5BMl5BanBnXkFtZTgwNTcxNTYxM
##  $ Ratings   :List of 3
##   ..$ :List of 2
##   .. ..$ Source: chr "Internet Movie Database"
##   .. ..$ Value : chr "7.7/10"
##   ..$ :List of 2
##   .. ..$ Source: chr "Rotten Tomatoes"
##   .. ..$ Value : chr "80%"
##   ..$ :List of 2
##   .. ..$ Source: chr "Metacritic"
##   .. ..$ Value : chr "68/100"
##  $ Metascore : chr "68"
##  $ imdbRating: chr "7.7"
##  $ imdbVotes : chr "243,369"
##  $ imdbID    : chr "tt0151804"
##  $ Type      : chr "movie"
##  $ DVD       : chr "25 Nov 2015"
##  $ BoxOffice : chr "$10,827,810"
##  $ Production: chr "Cubicle, Inc."
##  $ Website   : chr "N/A"
##  $ Response  : chr "True"
```

c. Write a function, called `grab_movie()`, to pull and parse the data on a given movie. Include `title`, `year`, and `omdb_key` as arguments. Test your function on one movie. (Don't worry about putting in error or warning messages for faulty arguments.)

```
grab_movie <- function(title, year, omdb_key) {
  movie_pull <- GET(glue("http://www.omdbapi.com/?i=tt3896198&apikey={omdb_key}"),
                query = list(t = title,
                             y = year,
                             plot = "short",
                             r = "json"))
  movie <- content(movie_pull, as = "parsed", type = "application/json")
  movie
}

glimpse(grab_movie(title = "Ocean's Eight", year = 2018, omdb_key = omdb_key))
```

```
## List of 25
##  $ Title     : chr "Ocean's Eight"
##  $ Year      : chr "2018"
##  $ Rated     : chr "PG-13"
##  $ Released  : chr "08 Jun 2018"
```

```
##  $ Runtime   : chr "110 min"
##  $ Genre     : chr "Action, Comedy, Crime, Thriller"
##  $ Director  : chr "Gary Ross"
##  $ Writer    : chr "Gary Ross (screenplay by), Olivia Milch (screenplay by), Gary Ross (story by), G
##  $ Actors    : chr "Sandra Bullock, Griffin Dunne, Deidre Goodwin, Daniella Rabbani"
##  $ Plot      : chr "Debbie Ocean gathers an all-female crew to attempt an impossible heist at New Yo
##  $ Language  : chr "English, Hindi, French, German, Mandarin"
##  $ Country   : chr "USA"
##  $ Awards    : chr "3 wins & 16 nominations."
##  $ Poster    : chr "https://m.media-amazon.com/images/M/MV5BMjAyNDEyMzc4Ml5BMl5BanBnXkFtZTgwMjEzNjM0
##  $ Ratings   :List of 3
##   ..$ :List of 2
##   .. ..$ Source: chr "Internet Movie Database"
##   .. ..$ Value : chr "6.9/10"
##   ..$ :List of 2
##   .. ..$ Source: chr "Rotten Tomatoes"
##   .. ..$ Value : chr "69%"
##   ..$ :List of 2
##   .. ..$ Source: chr "Metacritic"
##   .. ..$ Value : chr "61/100"
##  $ Metascore : chr "61"
##  $ imdbRating: chr "6.9"
##  $ imdbVotes : chr "190,620"
##  $ imdbID    : chr "tt5164214"
##  $ Type      : chr "movie"
##  $ DVD       : chr "21 Aug 2018"
##  $ BoxOffice : chr "$140,218,711"
##  $ Production: chr "Warner Bros. Pictures, Smoke House, Village Roadshow Pictures"
##  $ Website   : chr "N/A"
##  $ Response  : chr "True"
```

d. Pick three movies that came out in 2020 that you want to see. Use the `map()` function or a for loop to pull and parse the data on those three movies, storing the output in a list.

```
# 2020 Movies you want to see
titles_2020 <- c("Promising Young Woman", "I Care a Lot", "Kiss the Ground")

movies_2020 <-
  map(.x = titles_2020,
      .f = function(.x) grab_movie(title = .x,
                                                  year = 2020,
                                                  omdb_key = omdb_key))
```

e. Now pick 10 movies you want to see (or rewatch) but this time they can't all have come out in 2020. Use the `map2()` function or for loops to pull and parse the data on these movies, storing the output in a list called `movies`.

We need `map2()` because now we want to iterate over 2 arguments: `title` and `year`. Here is an example of `map2()` in action.

```
x <- 1:3
y <- 4:6
map2(x, y, sum)
```

```
## [[1]]
## [1] 5
##
```

```
## [[2]]
## [1] 7
##
## [[3]]
## [1] 9
```

```
# Fill in
title <- c("Promising Young Woman", "I Care a Lot", "Red Sparrow",
           "The Fundamentals of Caring", "Thor: Ragnarok", "Little Women",
           "The Spy Who Dumped Me", "Ocean's Eight", "Ghostbusters", "Captain Marvel")
# Fill in corresponding years for each movie
year <- c(2020, 2020, 2018, 2016, 2017, 2019, 2018, 2018, 2016, 2019)

movies <- map2(.x = title,
               .y = year,
               .f = function(.x, .y) grab_movie(title = .x,
                                                year = .y,
                                                omdb_key = omdb_key))
```

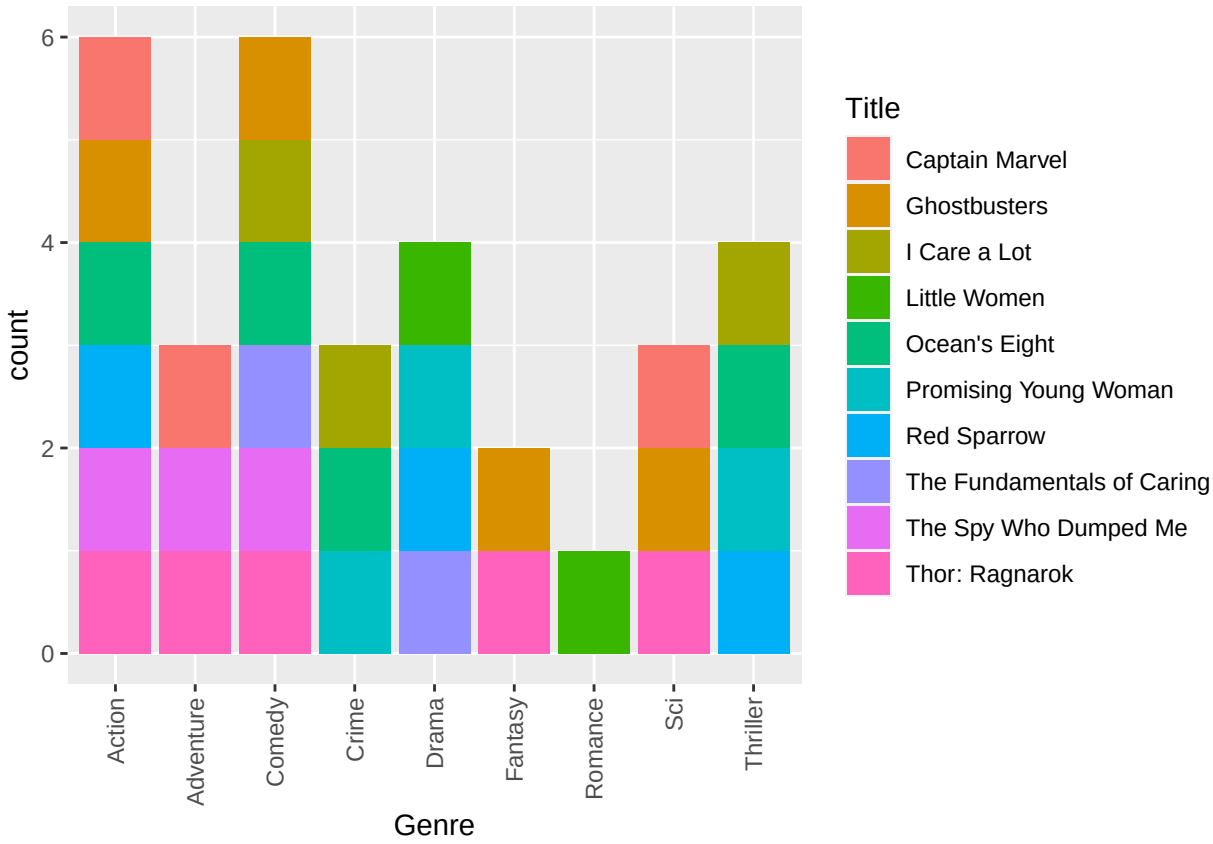
f. Now let's convert some of the data stored in `movies` into a nice `data.frame`. Using `map_dfr()` or for loops, create a data frame that contains a row for each movie and a column for the following variables: `"Title"`, `"Year"`, `"Rated"`, `"Runtime"`, `"Genre"`, `"imdbRating"`, `"imdbVotes"`.

```
movie_df <- map_dfr(movies,`[`, c("Title", "Year", "Rated", "Runtime", "Genre", "imdbRating",
                                  "imdbVotes"))
movie_df
```

```
## # A tibble: 10 x 7
##    Title           Year  Rated Runtime Genre                imdbRating imdbVotes
##    <chr>           <chr> <chr> <chr>   <chr>                <chr>      <chr>
##  1 Promising Youn~ 2020  R     113 min Crime, Drama, Thril~ 7.5        47,359
##  2 I Care a Lot    2020  R     118 min Comedy, Crime, Thri~ 6.2        52,195
##  3 Red Sparrow     2018  R     140 min Action, Drama, Thri~ 6.6        168,326
##  4 The Fundamenta~ 2016  TV-MA 97 min  Comedy, Drama        7.3        61,777
##  5 Thor: Ragnarok  2017  PG-13 130 min Action, Adventure, ~ 7.9        596,684
##  6 Little Women    2019  PG    135 min Drama, Romance       7.8        148,920
##  7 The Spy Who Du~ 2018  R     117 min Action, Adventure, ~ 6.0        69,813
##  8 Ocean's Eight   2018  PG-13 110 min Action, Comedy, Cri~ 6.9        190,620
##  9 Ghostbusters    2016  PG-13 117 min Action, Comedy, Fan~ 6.6        205,653
## 10 Captain Marvel  2019  PG-13 123 min Action, Adventure, ~ 6.9        455,501
```

g. Now use your nice clean data frame to create an interesting graph about the movies you want to see (or rewatch). Draw some conclusions about your movie interests from the graph. Before you can create the graph, you will likely still need to do a little data wrangling (e.g., maybe a `parse_number()` in a `mutate()`).

```
movie_df <- movie_df %>%
  mutate(parse_number(imdbRating), parse_number(imdbVotes)) %>%
  separate_rows(Genre, convert = TRUE) %>%
  filter(Genre != "Fi")

ggplot(movie_df, aes(x = Genre, fill = Title)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



It looks like I am mostly looking forward to watch movies that are considered action or comedy movies. 6/10 of the movies on my watch list are comedy or action movies, and 4/10 of the selected movies are considered both. That leads me to believe that action and comedy movies are my favorite genre.

Problem 2: Are Volcanic Eruptions Increasing?

The Smithsonian's Global Volcanism Program (GVP) keeps up-to-date information on the volcanoes of the world and their eruptions. GVP also maintains two databases of all documented volcanoes and eruptions from the last 10,000 years. In this problem and the next, we will explore the GVP dataset of all documented eruptions. Note: I downloaded the data in Spring of 2016.

```
Eruptions <- read_csv("/home/courses/math241s21/Data/GVP_Eruption_Results.csv")
```

a. Subset the data frame to only include confirmed eruptions (`EruptionCategory`). Now how many observations do we have? What does each row of the dataset represent? (Use this subset for the rest of the problem.)

```
confirmed_eruptions <- Eruptions %>%
  filter(EruptionCategory == "Confirmed Eruption")
```

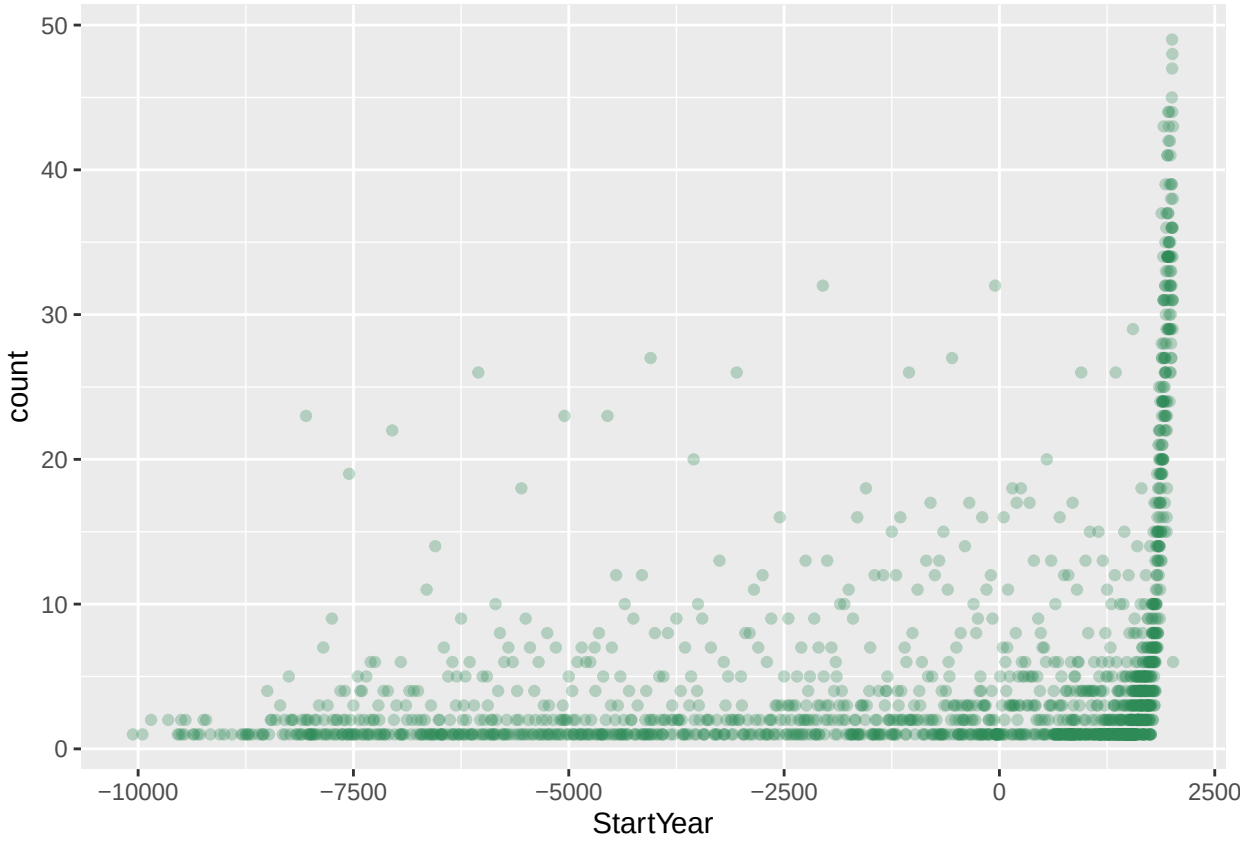
Out of the original 11019 observations in the original dataset, only 99756 of those were confirmed eruptions. Each row represents each time there was a confirmed volcano eruption.

b. If we want to plot year versus the number of eruptions that started that year, what do we want each row to represent? What variables would you want to include in that dataset?

We want each row to represent a year. We would include the StartYear variable as well as a count variable

c. Create the dataset you described above and a graph of start year versus the number of eruptions. Discuss any trends you see in the plot.

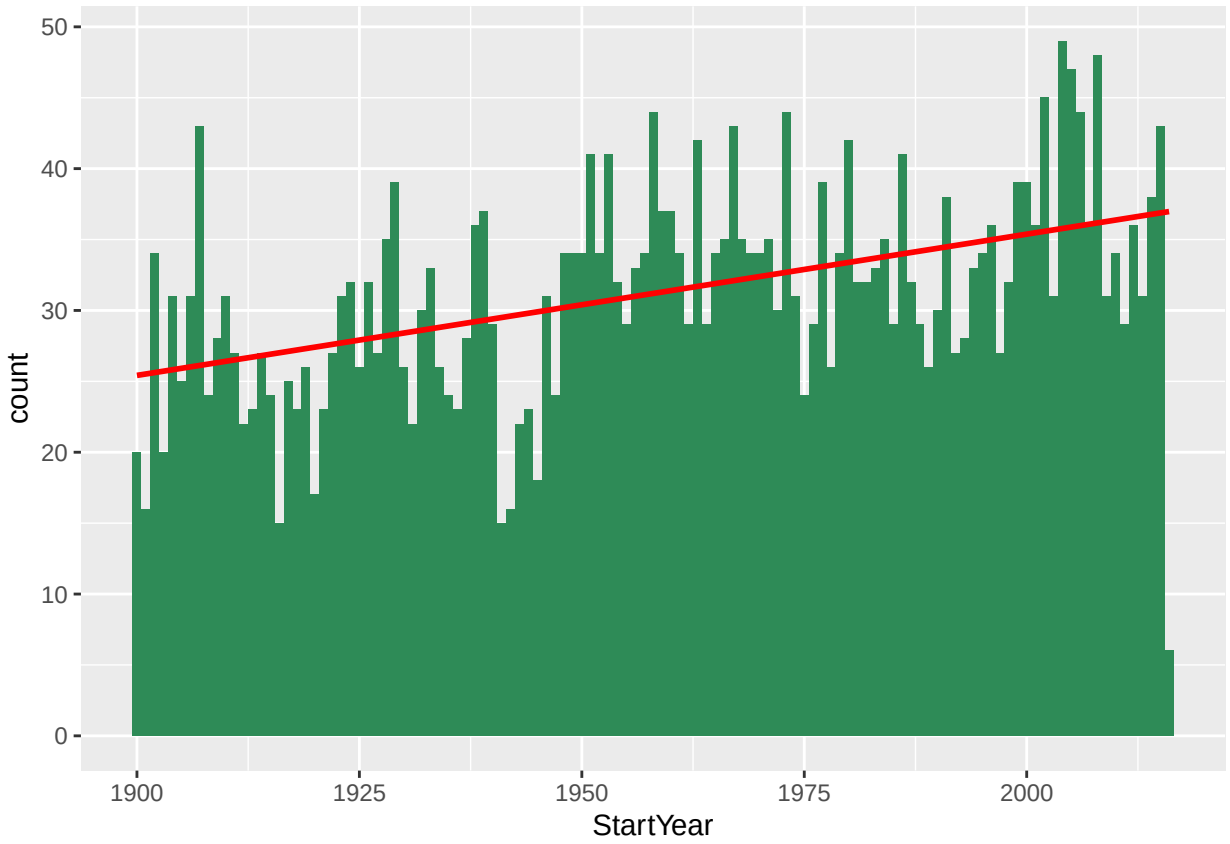
```r
confirmed_eruptions <- confirmed_eruptions %>%
  select(StartYear) %>%
  group_by(StartYear) %>%
  summarise(count = n())

ggplot(confirmed_eruptions, aes(x = StartYear, y = count)) +
  geom_point(alpha = 0.3, color = 'seagreen')
```



There seems to be many years before year 1250 approximately where there we no volcanic eruptions, but if a year had an eruption, there were many. Starting after year 1250, there seems to be more volcanic eruptions, at least there are more documented for each year. This plot makes it seem that only in the last 750 years or so have volcanic eruptions be so common.

d. Let's focus on eruptions that started in 1900 onward. From 1900 onward, produce a graph of start year versus the number of eruptions and include the line of best fit (i.e., linear regression line). Address the question: "Are volcanic eruptions increasing?"

```r
confirmed_eruptions <- confirmed_eruptions %>%
  filter(StartYear >= 1900)

ggplot(confirmed_eruptions, aes(x = StartYear, y = count)) +
  geom_bar(stat= "identity", fill = 'seagreen')+
  geom_smooth(aes(x = StartYear, y = count),
              method = "lm", se= FALSE, color = "red")
```



As the line of best fit has a positive slope, it implies that there has been an increase in volcanic eruptions since the year 1900. So yes, volcanic eruptions are increasing.

e. Let's explore how sampling bias might have impacted these data. We want to answer the following questions:

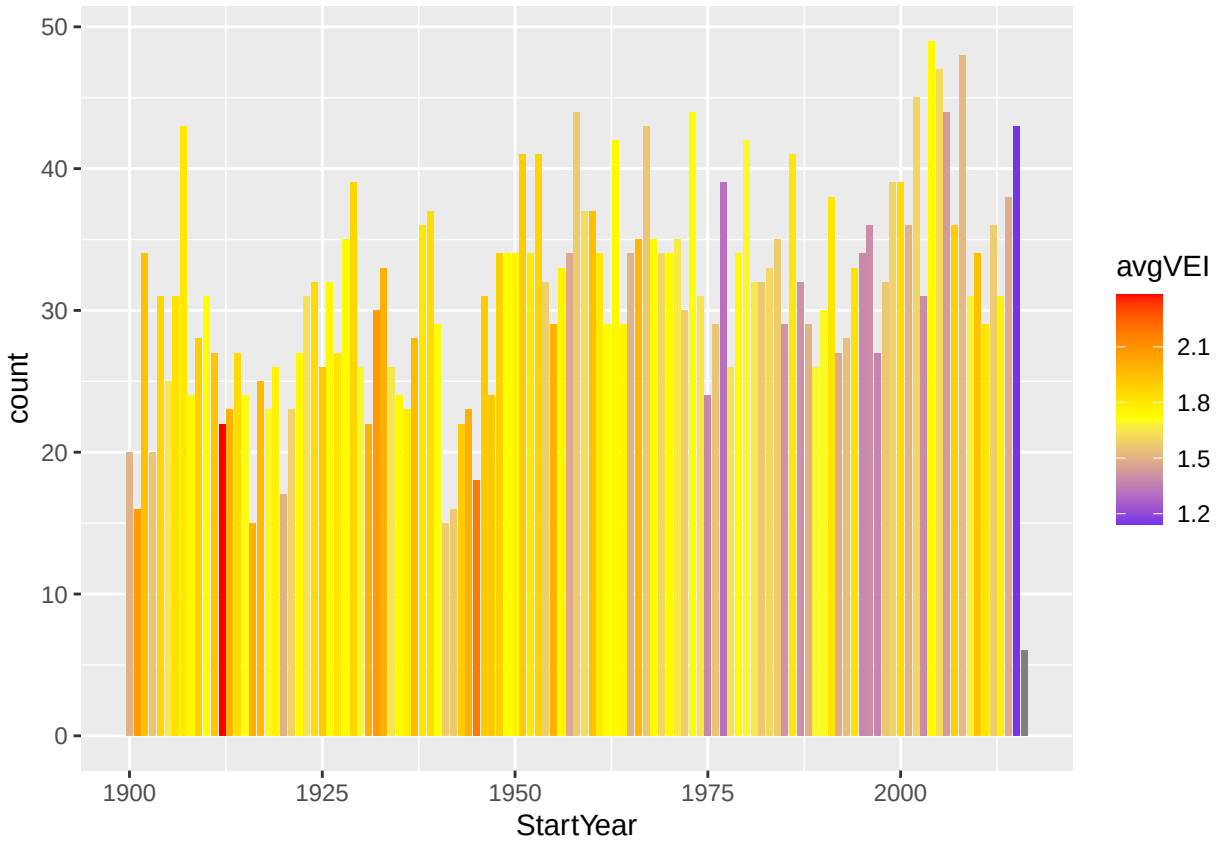
- Why are there dips in number of eruptions for two time periods?
- How might size of eruption relate to probability of detection over time?

We can investigate the second bullet by re-wrangling the data and including the average size of the explosions in a given year. We will measure size using the Volcanic Explosivity Index (VEI), a measure of how explosive an eruption is. Create a dataset with starting year, frequency of eruptions, and average VEI per year for 1900 onward. Then produce a graph of starting year versus count with average VEI mapped to color.

```
new_confirmed_eruptions <- Eruptions %>%
  filter(EruptionCategory == "Confirmed Eruption", StartYear >= 1900) %>%
  group_by(StartYear) %>%
  summarise(count = n(),
            avgVEI = mean(VEI, na.rm = TRUE))

mid <- mean(new_confirmed_eruptions$avgVEI, na.rm = TRUE)
mid
```

```
## [1] 1.716838
```

```
ggplot(new_confirmed_eruptions, aes(x = StartYear, y = count, fill = avgVEI)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient2(midpoint = mid, low = "blue", mid = "yellow",
                       high = "red", space ="Lab")
```



f. Address the following questions and how they might impact the quality of the data (as a reflection of all eruptions from 1900 onward):

- Why are there dips in number of eruptions for two time periods?
- How might size of eruption relate to probability of detection over time?

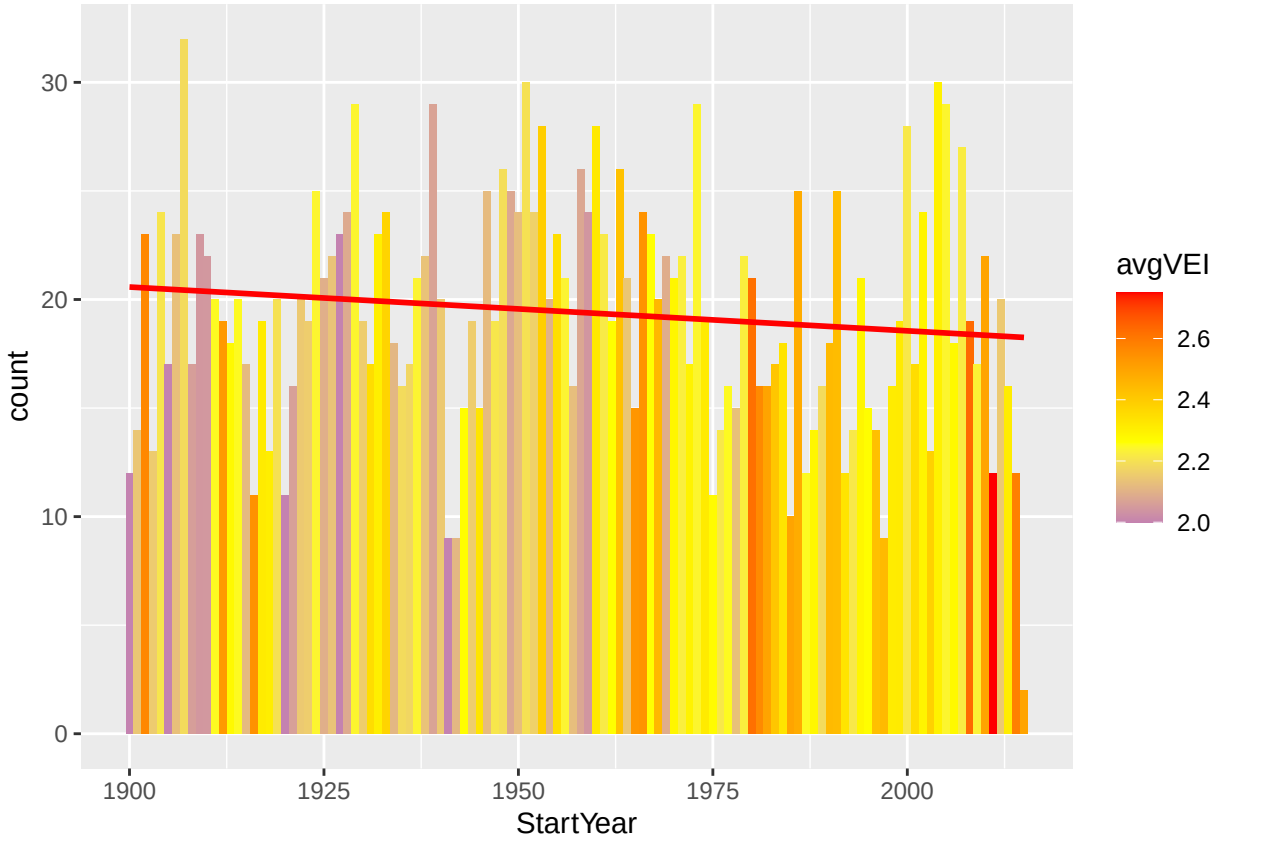
Those two relatively dramatic dips occur when the volcanic eruptions are more explosive than usual (red/orange bars). There seems to be an inverse correlation between average explosivity and total count of eruptions. So, in years where volcanic eruptions are more explosive, we are likely to see less total number of eruptions. However, acknowledging how much technology has advanced in the past 100ish years, it is likely that in the early 1900s only the more explosive volcanic eruptions were recorded and other, less explosive eruptions did not catch enough attention and were not documented. This created bias in the data based on scientist's technological ability to measure the true total number of eruptions per year. This is most likely why the second half of the dataset has more counts of volcanic eruptions, but a lower VEI average.

g. Subset the data to only larger confirmed eruptions (VEI >=2) from 1900 onward and recreate the graph of starting year versus count with average VEI mapped to color. Based on this graph do eruptions appear to be increasing over time?

```
newer_confirmed_eruptions <- Eruptions %>%
  filter(EruptionCategory == "Confirmed Eruption", StartYear >= 1900, VEI >= 2) %>%
  group_by(StartYear) %>%
  summarise(count = n(),
            avgVEI = mean(VEI, na.rm = TRUE))

mid <- mean(newer_confirmed_eruptions$avgVEI, na.rm = TRUE)
mid
```

```
## [1] 2.261273
```

```
ggplot(newer_confirmed_eruptions, aes(x = StartYear, y = count, fill = avgVEI)) +
  geom_bar(stat = "identity") +
  geom_smooth(aes(x = StartYear, y = count),
              method = "lm", se= FALSE, color = "red") +
  scale_fill_gradient2(midpoint = mid, low = "blue", mid = "yellow",
                       high = "red", space ="Lab")
```



No. Looking at the line of best fit I added to the plot, there seems to be a slight decrease in eruptions (very explosive eruptions) over time. However, it does seem that the 2016 data is incomplete, which should be noted and is likely playing a role in that slight decline. Noting that, and controlling for better technology picking up ALL eruptions, this plot shows that the occurrence of volcanic eruptions seem to be the same over time.

Problem 3: Dates and Eruptions

From what we learned in Problem 2, let's only consider eruptions that **ended** in 1968, the year GVP started documenting eruptions, or later than 1968.

For this problem, I want you to explore the duration of eruptions.

a. Subset the Eruptions dataset to only confirmed eruptions that ended in 1968 or later. Use that dataset for the rest of the problem.

```
end_eruptions <- Eruptions %>%
  filter(EndYear >= 1968)
```

b. Add a `StartDate` column and an `EndDate` column where each incorporates the year, month, and day. Also add a column that contains the date interval. (Use the `lubridate` cheatsheet to figure out how to add the date interval.)

```
end_eruptions$StartDate <- as.Date(with(end_eruptions,
                                        paste(StartYear, StartMonth, StartDay, sep = "-")),
                                   "%Y-%m-%d")
end_eruptions$EndDate <- as.Date(with(end_eruptions,
                                      paste(EndYear, EndMonth, EndDay, sep = "-")),
                                 "%Y-%m-%d")
end_eruptions$DateInterval <- interval(end_eruptions$StartDate, end_eruptions$EndDate)
```

c. Create a dataset of those that failed to parse the `StartDate` (and therefore have an NA entry). Why did those dates fail to parse?

```
failed_parse <- end_eruptions %>%
  filter(is.na(StartDate))
glimpse(failed_parse)
```

```
## Rows: 166
## Columns: 27
## $ VolcanoNumber            <dbl> 345040, 252010, 243120, 243130, 263280, 26603~
## $ VolcanoName              <chr> "Poas", "Langila", "Tafu-Maka", "West Mata", ~
## $ EruptionNumber           <dbl> 11193, 15003, 14799, 14800, 15958, 16438, 138~
## $ EruptionCategory         <chr> "Confirmed Eruption", "Confirmed Eruption", "~
## $ AreaofActivity           <chr> NA, "Crater 2", "Maka", "Prometheus and Hades~
## $ VEI                      <dbl> 1, 2, 0, 0, 2, 3, 3, 3, 1, 2, 0, 0, 2, 0, 1, ~
## $ VEIModifier              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ StartYearModifier        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ StartYear                <dbl> 2009, 2009, 2008, 2008, 2007, 2007, 2007, 200~
## $ StartYearUncertainty     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ StartMonth               <dbl> 11, 9, 11, 11, 10, 6, 6, 4, 3, 2, 8, 7, 9, 5,~
## $ StartDayModifier         <chr> NA, "<", "?", "<", "?", "?", NA, "<", NA, "<"~
## $ StartDay                 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ StartDayUncertainty      <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ `EvidenceMethod(dating)` <chr> "Historical Observations", "Historical Observ~
## $ EndYearModifier          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ EndYear                  <dbl> 2014, 2010, 2008, 2009, 2008, 2007, 2010, 200~
## $ EndYearUncertainty       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ EndMonth                 <dbl> 10, 2, 11, 5, 4, 11, 10, 12, 8, 11, 1, 8, 9, ~
## $ EndDayModifier           <chr> NA, "?", "?", ">", "<", "?", "?", "?", "?", "~
## $ EndDay                   <dbl> 31, 0, 0, 0, 0, 9, 9, 29, 9, 0, 0, 0, 0, 22, ~
## $ EndDayUncertainty        <dbl> 5, NA, NA, NA, NA, NA, 1, NA, NA, NA, NA, NA,~
## $ Latitude                 <dbl> 10.200, -5.525, -15.370, -15.100, -7.930, 1.1~
## $ Longitude                <dbl> -84.233, 148.420, -174.230, -173.750, 112.308~
## $ StartDate                <date> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ EndDate                  <date> 2014-10-31, NA, NA, NA, NA, 2007-11-09, 2010~
## $ DateInterval             <Interval> NA--NA, NA--NA, NA--NA, NA--NA, NA--NA, ~
```

Those dates failed to parse because their `StartDay` values were 0, and 0 does not represent an actual day of a month.

d. Using a `lubridate` function, add a column for the length of the eruption intervals in days.

```
end_eruptions$IntervalLength <- time_length(end_eruptions$DateInterval, "day")
```

e. Construct a graph that attempts to answer the question: Is there are relationship between VEI (the size) and the length of an eruption? Draw conclusions from your graph.

```
mid <- mean(end_eruptions$VEI, na.rm = TRUE)
mid #mean
```

```
## [1] 1.548407
```

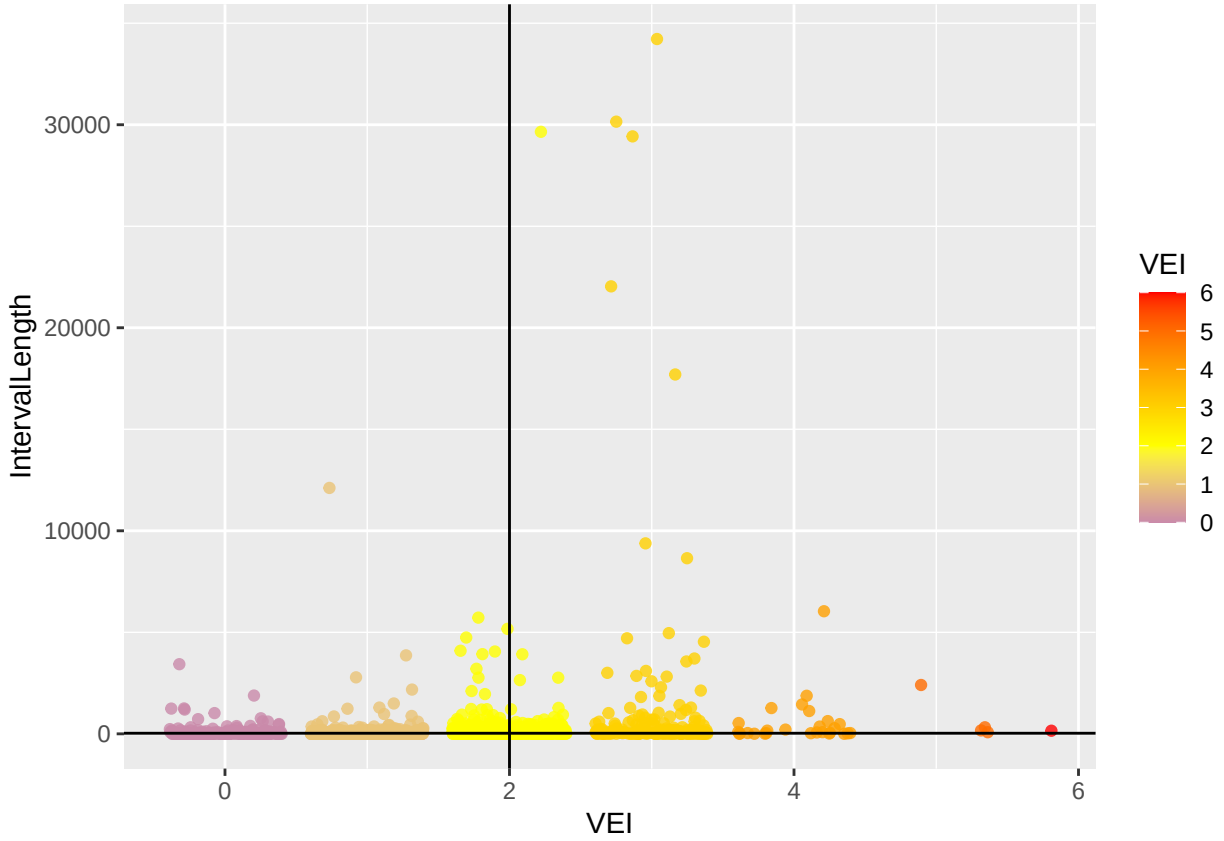
```
mid <- median(end_eruptions$VEI, na.rm = TRUE)
mid #median
```

```
## [1] 2
```

```
midV <- median(end_eruptions$IntervalLength, na.rm = TRUE)
midV
```

```
## [1] 29
```

```
ggplot(end_eruptions, aes(x = VEI, y = IntervalLength, color = VEI)) +
  geom_jitter(alpha = 0.8) +
  geom_vline(xintercept = mid, color = "black") +
  geom_hline(yintercept = midV) +
  scale_color_gradient2(midpoint = mid, low = "blue", mid = "yellow",
                        high = "red", space ="Lab")
```



There is not a clear connection of VEI to the length of the eruptions since the median eruption length is 29 days. However, it looks like that if a volcanic eruption is super long, then it's VEI is a 2 or 3. The mean VEI is 1.5 and the median VEI is 2. This means that the super long volcanic eruptions mostly occur when the volcanic eruption's explosivity is slightly above the average since the plot shows the longest eruptions having a VEI of 3.

f. Create a data frame of the 10 volcanoes with the longest eruptions and display the `VolcanoName`, `total_time`, and `Interval`. This table presents what data quality issue? Recall that I downloaded

these data in the spring of 2016. (Feel free to use the internet to verify the issue.)

```
end_eruptions_top <- end_eruptions %>%
  arrange(desc(IntervalLength)) %>%
  slice(1:10) %>%
  rename(Interval = DateInterval,
         total_time = IntervalLength) %>%
  select(VolcanoName, total_time, Interval)
end_eruptions_top
```

```
## # A tibble: 10 x 3
##    VolcanoName total_time Interval
##    <chr>            <dbl> <Interval>
##  1 Santa Maria      34221 1922-06-22 UTC--2016-03-01 UTC
##  2 Dukono           30151 1933-08-13 UTC--2016-03-01 UTC
##  3 Stromboli        29652 1934-02-02 UTC--2015-04-10 UTC
##  4 Sangay           29427 1934-08-08 UTC--2015-03-03 UTC
##  5 Aira             22039 1955-10-13 UTC--2016-02-14 UTC
##  6 Semeru           17699 1967-08-31 UTC--2016-02-14 UTC
##  7 Kilauea          12111 1983-01-03 UTC--2016-03-01 UTC
##  8 Manam             9381 1974-03-04 UTC--1999-11-09 UTC
##  9 Pacaya            8650 1965-07-04 UTC--1989-03-10 UTC
## 10 Sheveluch         6039 1999-08-15 UTC--2016-02-26 UTC
```

The data was collected during 2016, so we are missing data of volcanoes that stopped erupting after the download data, or maybe volcanoes were given premature eruption end dates? I am not sure exactly which, or if it is both.

Problem 4: Food Consumption Comparisons

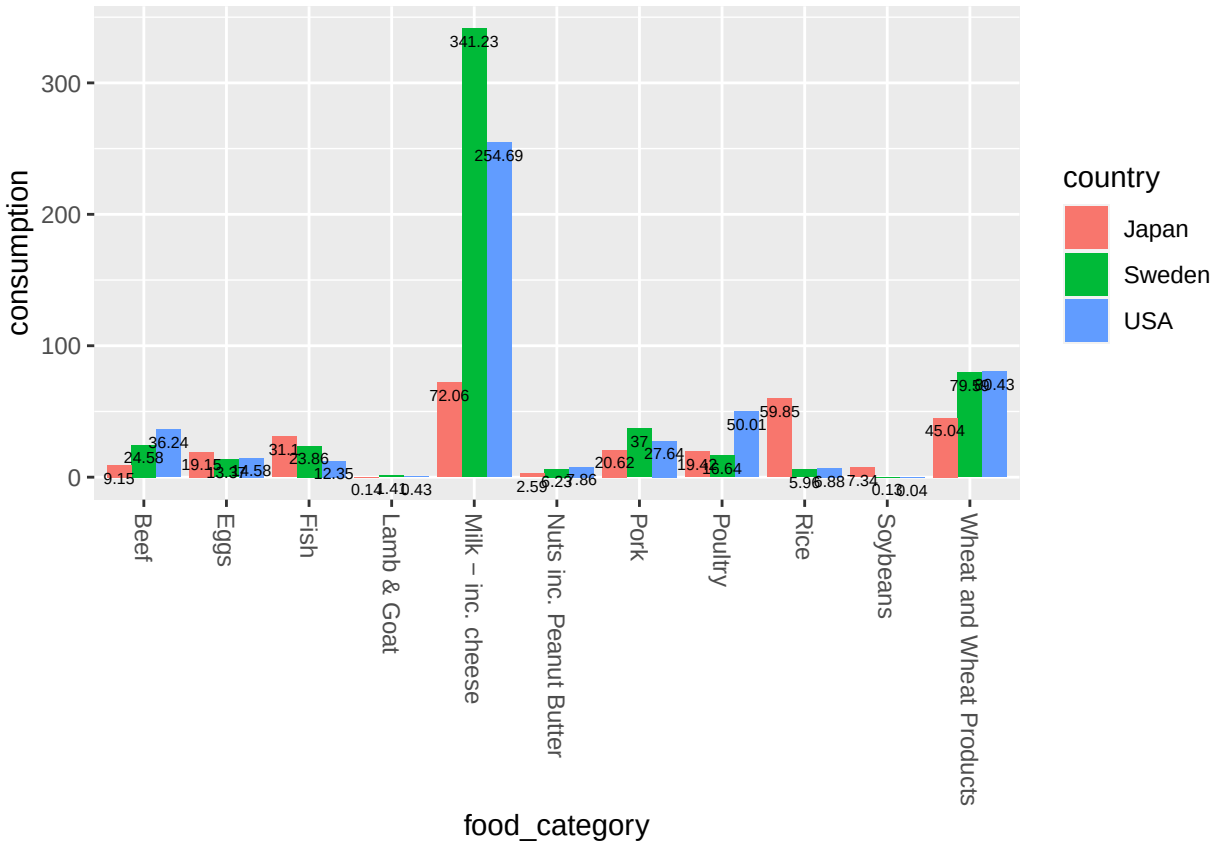
For this problem, let's compare the food consumption patterns of other countries to the USA. The data come from the "R for Data Science's" (R4DS) Tidy Tuesday challenge. You can read more about the data here.

Throughout this problem use `forcats` to wrangle your factors to create easier to read, more compelling graphs.

```
food_consumption <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/maste
```

a. Create a graph that compares the food consumption in the USA to two other countries. Draw some conclusions from the graph.

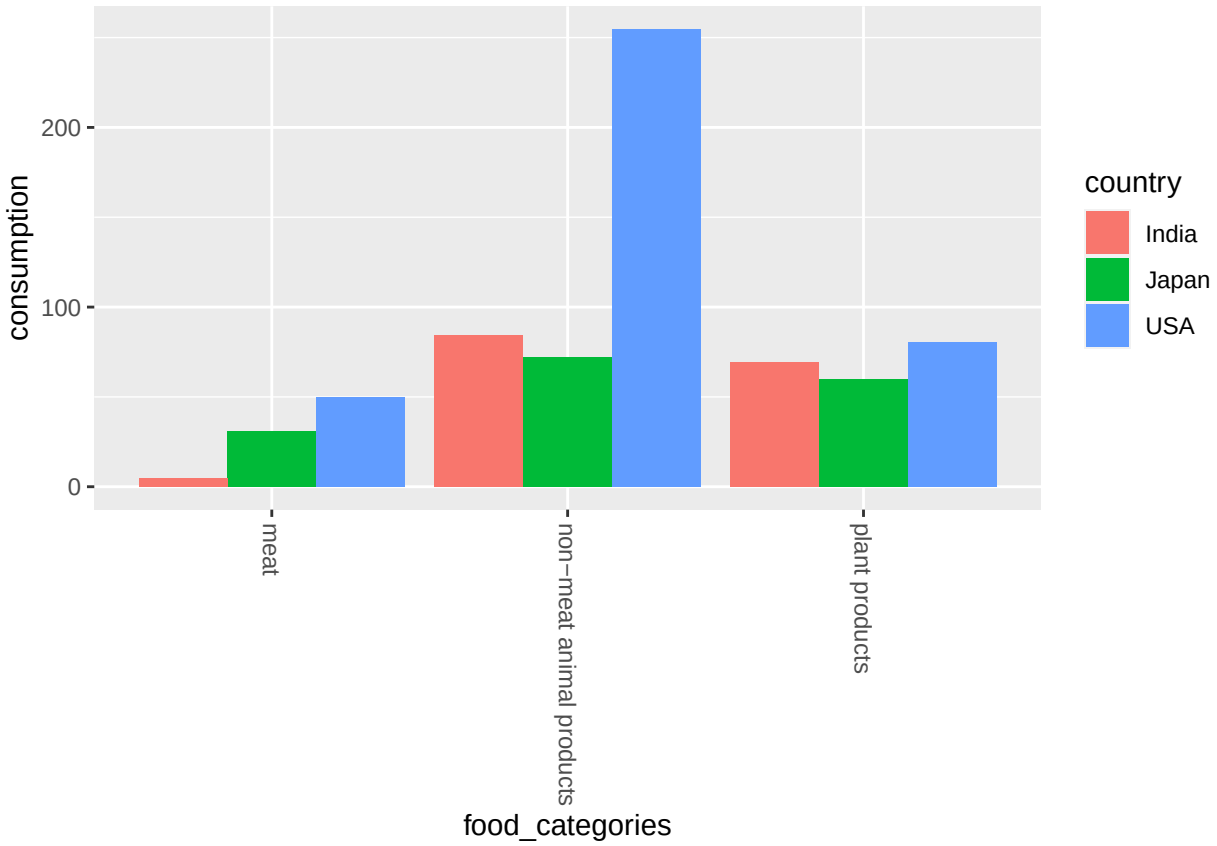
```
food_consumption %>%
  filter(country == "USA" | country == "Sweden" | country == "Japan") %>%
  ggplot(aes(x = food_category, y = consumption, fill = country)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_text(aes(label = consumption), vjust = 1.6, color = "black",
            position = position_dodge(0.9), size = 2) +
  theme(axis.text.x = element_text(angle = -90, vjust = 0.5, hjust = 0))
```



Looking at the food consumption of Japan, US, and Sweden, there is cool to see that the food each country is more known for eating is reflected accurately in this plot. Sweden: cheese; Japan: rice, soybeans, and fish; USA: beef and poultry. It seems that the US and Sweden have similar consumption habits, while Japan seems to be quite different.

b. Create a new column in the dataset that categories the `food_categories` into the following broader categories: meat, non-meat animal products, and plant products. Compare the differences in consumption for these broader categories in the USA and two other countries. Draw some conclusions from the graph.

```
food_categories <- c("meat", "non-meat animal products", "meat", "meat",
                     "non-meat animal products", "plant products", "meat", "meat",
                     "plant products", "plant products", "plant products")

food_category <- c("Beef", "Eggs", "Fish", "Lamb & Goat", "Milk - inc. cheese",
                   "Nuts inc. Peanut Butter", "Pork", "Poultry", "Rice", "Soybeans",
                   "Wheat and Wheat Products")
df <- data.frame(food_category, food_categories)
food_consumption <- food_consumption %>%
  left_join(df)

food_consumption %>%
  filter(country == "USA" | country == "India" | country == "Japan") %>%
  ggplot(aes(x = food_categories, y = consumption, fill = country)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme(axis.text.x = element_text(angle = -90, vjust = 0.5, hjust = 0))
```



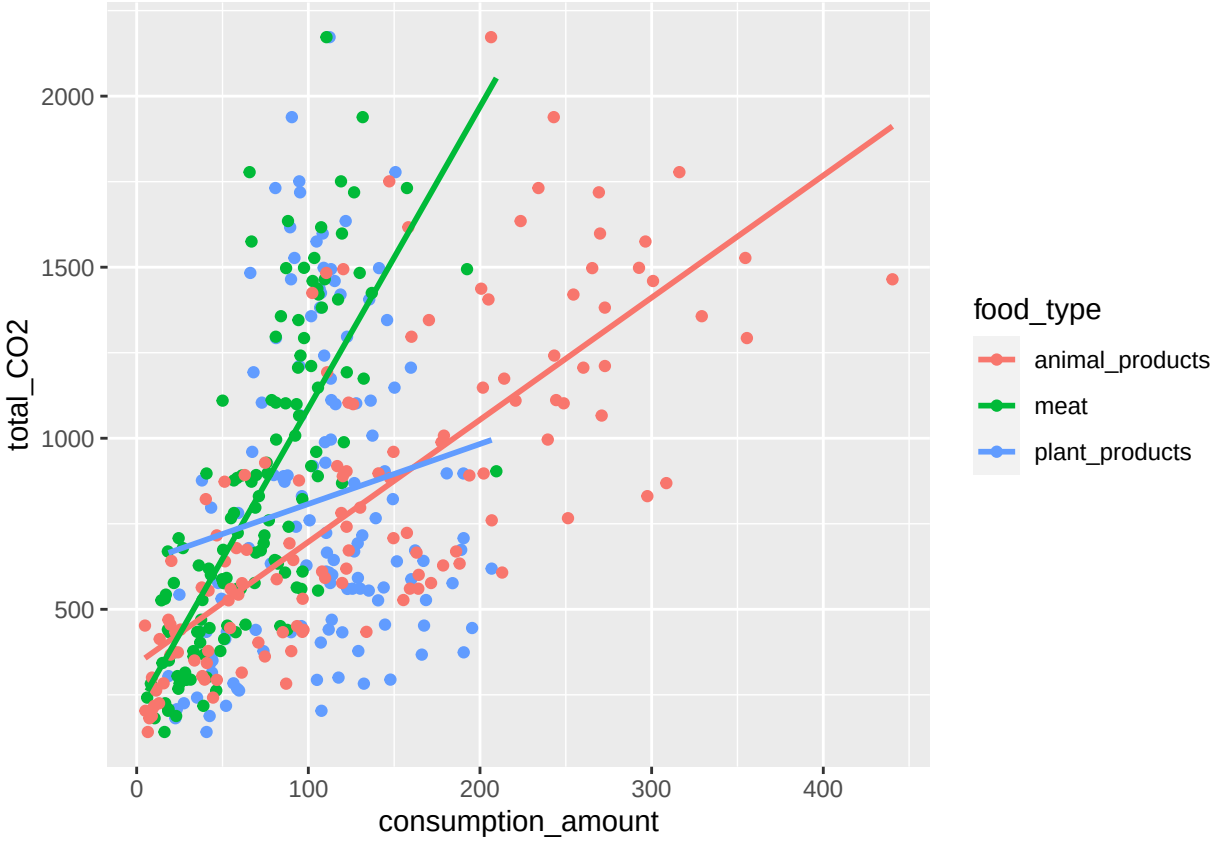
Looking at India, Japan, and the USA, the US consumes the most out of all of these specific food categories. It would be interesting to include fruits and vegetables to this dataset, since I am not sure how the US is consuming the most food despite having a smaller population size than India. This could mean that the US eats less of fruits and vegtables as a whole, which is why their consumption of meat, non-meat animal products, and plant products is so high.

c. Ask a new question of these data. Generate a graph to answer that question and draw some conclusions from the graph.

How does meat consumption relate to a country's total CO2 emissions from food consumption?

```
food_consumption <- food_consumption %>%
  group_by(country)
food_consumption$total_CO2 <- ave(food_consumption$co2_emmission,
                                  food_consumption$country, FUN = sum)

food_consumption_meat <- food_consumption %>%
  filter(food_categories == "meat")
food_consumption_meat$meat <- ave(food_consumption_meat$consumption,
                                  food_consumption_meat$country, FUN = sum)

food_consumption_plant <- food_consumption %>%
  filter(food_categories == "plant products")
food_consumption_plant$plant_products <- ave(food_consumption_plant$consumption,
                                             food_consumption_plant$country, FUN = sum)

food_consumption_aniprod <- food_consumption %>%
  filter(food_categories == "non-meat animal products")
```

```r
food_consumption_aniprod$animal_products <- ave(food_consumption_aniprod$consumption,
                                                food_consumption_aniprod$country, FUN = sum)

food_consumption_plant <- food_consumption_plant %>%
  full_join(food_consumption_meat) %>%
  full_join(food_consumption_aniprod)

consumption_compare <- pivot_longer(food_consumption_plant,
                                    cols = c(meat, plant_products, animal_products),
                                  names_to = "food_type",
                                  values_to = "consumption_amount") %>%
  select(country, food_type, consumption_amount, total_CO2) %>%
  drop_na() %>%
  distinct()

ggplot(consumption_compare, aes(x = consumption_amount, y = total_CO2, color = food_type)) +
  geom_point() +
  geom_smooth(aes(group = food_type),
              method = "lm", se = FALSE)
```



I created this plot specifically because I think it is necessary to compare each food category to each other. This plot not only shows that countries that consume a lot of meat have much higher total CO2 emissions, but specifically that meat plays a large role in CO2 emissions. Plant products have the smallest slope (though it should be noted that the regression line is not an accurate representation), meaning that even as consumption of plant products goes up, it does not dramatically contribute to the total CO2 emissions from food consumption of a country. Animal products have the second highest slope, indicating that they contribute more to the total CO2 emissions, but not as drastically as meat does. This plot tells us that meat

consumption contributes greatly to total CO2 emissions.

Problem 5: What's in a Name? (You'd Be Surprised!)

1. Load the `babynames` dataset, which contains yearly information on the frequency of baby names by sex and is provided by the US Social Security Administration. It includes all names with at least 5 uses per year per sex. In this problem, we are going to practice pattern matching!

```
library(babynames)
data("babynames")
?babynames
```

   a. For 2000, find the ten most popular female baby names that start with the letter Z.

```
babynamesA <- babynames %>%
  filter(year == 2000) %>%
  filter(str_detect(name, "\\b(Z)[:alpha:]*")) %>%
  arrange(desc(n)) %>%
  top_n(10, n)
babynamesA
```

```
## # A tibble: 10 x 5
##     year sex   name          n     prop
##    <dbl> <chr> <chr>     <int>    <dbl>
##  1  2000 M     Zachary   19849 0.00951
##  2  2000 F     Zoe        3785 0.00190
##  3  2000 M     Zane       1368 0.000655
##  4  2000 M     Zackary    1351 0.000647
##  5  2000 M     Zachery    1150 0.000551
##  6  2000 M     Zion       1004 0.000481
##  7  2000 M     Zackery     710 0.000340
##  8  2000 M     Zachariah   700 0.000335
##  9  2000 F     Zoey        691 0.000346
## 10  2000 F     Zaria       568 0.000285
```

   b. For 2000, find the ten most popular female baby names that contain the letter z.

```
babynamesB <- babynames %>%
  filter(year == 2000, sex == "F") %>%
  filter(str_detect(name, "[Z|z]")) %>%
  arrange(desc(n)) %>%
  top_n(10, n)
babynamesB
```

```
## # A tibble: 10 x 5
##     year sex   name          n     prop
##    <dbl> <chr> <chr>     <int>    <dbl>
##  1  2000 F     Elizabeth 15094 0.00757
##  2  2000 F     Mackenzie  6348 0.00318
##  3  2000 F     Zoe        3785 0.00190
##  4  2000 F     Mckenzie   2526 0.00127
##  5  2000 F     Makenzie   1613 0.000809
##  6  2000 F     Jazmin     1391 0.000697
##  7  2000 F     Jazmine    1353 0.000678
##  8  2000 F     Lizbeth     817 0.000410
##  9  2000 F     Eliza       759 0.000380
## 10  2000 F     Litzy       722 0.000362
```

c. For 2000, find the ten most popular female baby names that end in the letter z.

```
babynamesC <- babynames %>%
  filter(year == 2000, sex == "F") %>%
  filter(str_detect(name, ".(?=[z]$)")) %>%
  arrange(desc(n)) %>%
  top_n(10, n)
babynamesC
```

```
## # A tibble: 11 x 5
##     year sex   name         n       prop
##    <dbl> <chr> <chr>    <int>      <dbl>
##  1  2000 F     Luz        489 0.000245
##  2  2000 F     Beatriz    357 0.000179
##  3  2000 F     Mercedez   141 0.0000707
##  4  2000 F     Maricruz    96 0.0000481
##  5  2000 F     Liz         72 0.0000361
##  6  2000 F     Inez        69 0.0000346
##  7  2000 F     Odaliz      24 0.0000120
##  8  2000 F     Marycruz    23 0.0000115
##  9  2000 F     Cruz        19 0.00000952
## 10  2000 F     Deniz       16 0.00000802
## 11  2000 F     Taiz        16 0.00000802
```

d. Between your three tables in 1.a - 1.c, do any of the names show up on more than one list? If so, which ones? (Yes, I know you could do this visually but use some joins!)

```
babynamesD <- rbind(babynamesA, babynamesB, babynamesC)
dup_idx <- duplicated(babynamesD)
dup_rows <- babynamesD[dup_idx, ]
dup_rows
```

```
## # A tibble: 1 x 5
##    year sex   name      n    prop
##   <dbl> <chr> <chr> <int>   <dbl>
## 1  2000 F     Zoe    3785 0.00190
```

Zoe is in both the ten most popular female baby names that start with the letter Z and the ten most popular female baby names that contain the letter z.

e. Verify that none of the baby names contain a numeric (0-9) in them.

```
babynames %>%
  filter(str_detect(name, "[0-9]"))
```

```
## # A tibble: 0 x 5
## # ... with 5 variables: year <dbl>, sex <chr>, name <chr>, n <int>, prop <dbl>
```

f. While none of the names contain 0-9, that doesn't mean they don't contain "one", "two", ..., or "nine". Create a table that provides the number of times a baby's name contained the word "zero", the word "one", ... the word "nine".

Notes:

- I recommend first converting all the names to lower case.
- If none of the baby's names contain the written number, there you can leave the number out of the table.
- Use `str_extract()`, not `str_extract_all()`. (We will ignore names where more than one of the words exists.)

Hint: You will have two steps that require pattern matching:

1. Subset your table to only include the rows with the desired words.
2. Add a column that contains the desired word.

```
babynames %>%
  mutate(name = str_to_lower(name)) %>%
  filter(str_detect(string = name,
                    pattern = "(zero|one|two|three|four|five|six|seven|eight|nine)")) %>%
  mutate(number =
           str_extract(name,
                       pattern = "(zero|one|two|three|four|five|six|seven|eight|nine)")) %>%
  count(number)
```

```
## # A tibble: 9 x 2
##   number     n
## * <chr>  <int>
## 1 eight    356
## 2 four       2
## 3 nine     807
## 4 one    10142
## 5 seven     50
## 6 six      106
## 7 three     58
## 8 two      288
## 9 zero       4
```

g. Which written number or numbers don't show up in any of the baby names?

one, two, three, eight, and nine show up, but four, five, six, and seven do not.

h. Create a table that contains the names and their frequencies for the two least common written numbers.

```
babynames %>%
  mutate(name = str_to_lower(name)) %>%
  filter(str_detect(string = name, pattern = "(zero|four)")) %>%
  group_by(name) %>%
  count(name)
```

```
## # A tibble: 4 x 2
## # Groups:   name [4]
##   name        n
##   <chr>   <int>
## 1 balfour     2
## 2 luzero      2
## 3 zero        1
## 4 zeron       1
```

i. Redo f. but this time produce a table that counts the number of babies named "zero", "one", "two", ... "nine" (instead of just containing the number). How does this table compare to the table in f.?

```
babynames %>%
  mutate(name = str_to_lower(name)) %>%
  filter(str_detect(
    string = name,
    pattern = "\\b(zero|one|two|three|four|five|six|seven|eight|nine)")) %>%
  mutate(
    number = str_extract
```

```
    (name, pattern = "\\b(zero|one|two|three|four|five|six|seven|eight|nine)")) %>%
  count(number)
```

```
## # A tibble: 6 x 2
##   number     n
## * <chr>  <int>
## 1 nine     100
## 2 one      884
## 3 seven     50
## 4 six      105
## 5 two        2
## 6 zero       2
```

The count drastically decreases for almost all (Six only goes from 106 to 105 and Zero only goes from 4 to 2), but "One" still dominates as the most names. The only numbers with full names are Zero, One, Two, Six, Seven, and Nine.

Problem 6: Tidying the "Call of the Wild"

Did you read "Call of the Wild" by Jack London when you were growing up? If not, read the first paragraph of its wiki page for a quick summary and then let's do some text analysis on this classic! The following code will pull the book into R using the `gutenbergr` package.

```
library(gutenbergr)
wild <- gutenberg_download(215, mirror = "http://www.gutenberg.org/dirs/")
```

a. Create a tidy text dataset where you tokenize by words.

```
tokenize_wild <- wild %>%
  unnest_tokens(word, text)
```

b. Find the frequency of the 20 most common words. First, remove stop words.

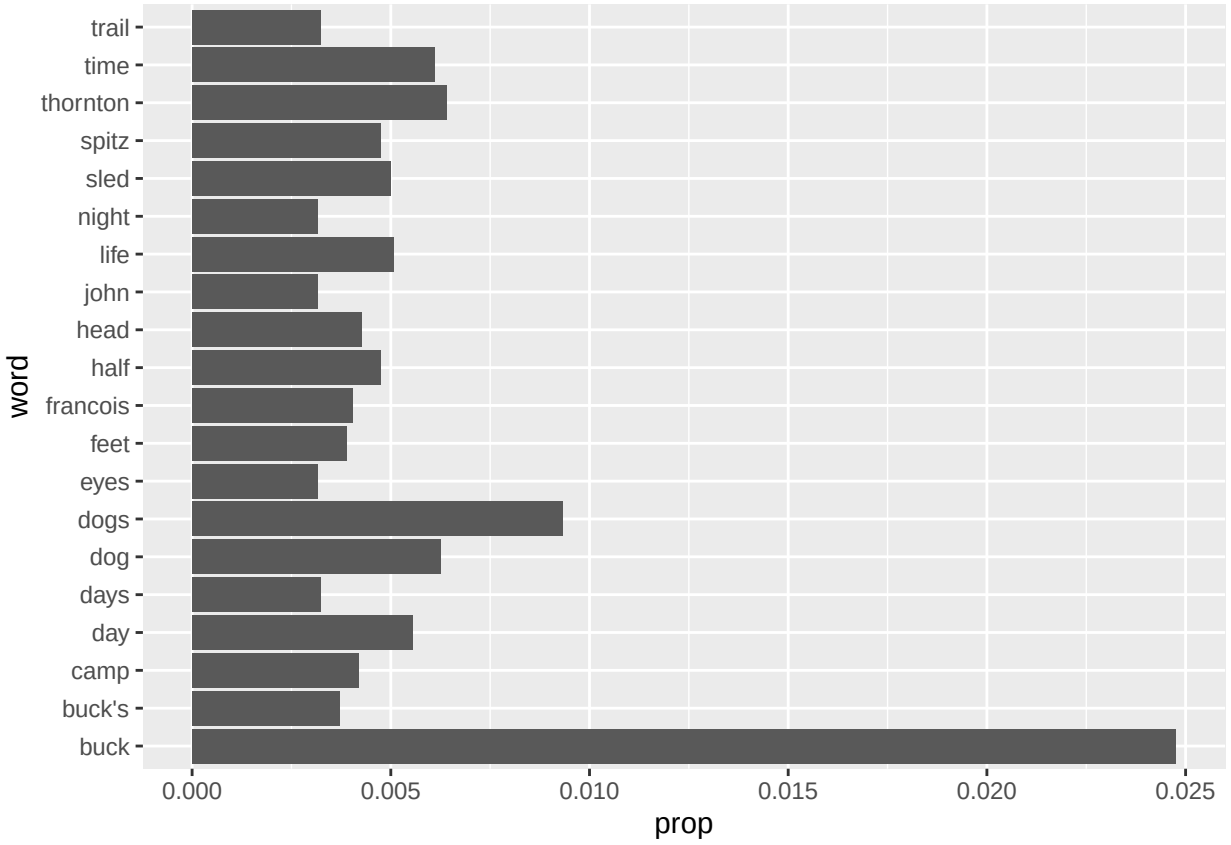
```
frequent_wild <- tokenize_wild %>%
  anti_join(stop_words, by = "word") %>%
  count(word) %>%
  mutate(prop = n/sum(n)) %>%
  arrange(desc(n)) %>%
  slice(1:20)
frequent_wild
```

```
## # A tibble: 20 x 3
##    word         n    prop
##    <chr>    <int>   <dbl>
##  1 buck       313 0.0248
##  2 dogs       118 0.00933
##  3 thornton    81 0.00641
##  4 dog         79 0.00625
##  5 time        77 0.00609
##  6 day         70 0.00554
##  7 life        64 0.00506
##  8 sled        63 0.00498
##  9 half        60 0.00475
## 10 spitz       60 0.00475
## 11 head        54 0.00427
## 12 camp        53 0.00419
## 13 francois    51 0.00403
```

```
## 14 feet        49 0.00388
## 15 buck's      47 0.00372
## 16 days        41 0.00324
## 17 trail       41 0.00324
## 18 eyes        40 0.00316
## 19 john        40 0.00316
## 20 night       40 0.00316
```

c. Create a bar graph and a word cloud of the frequencies of the 20 most common words.

```
ggplot(frequent_wild, aes(x = word, y = prop)) +
  geom_col() +
  coord_flip()
```



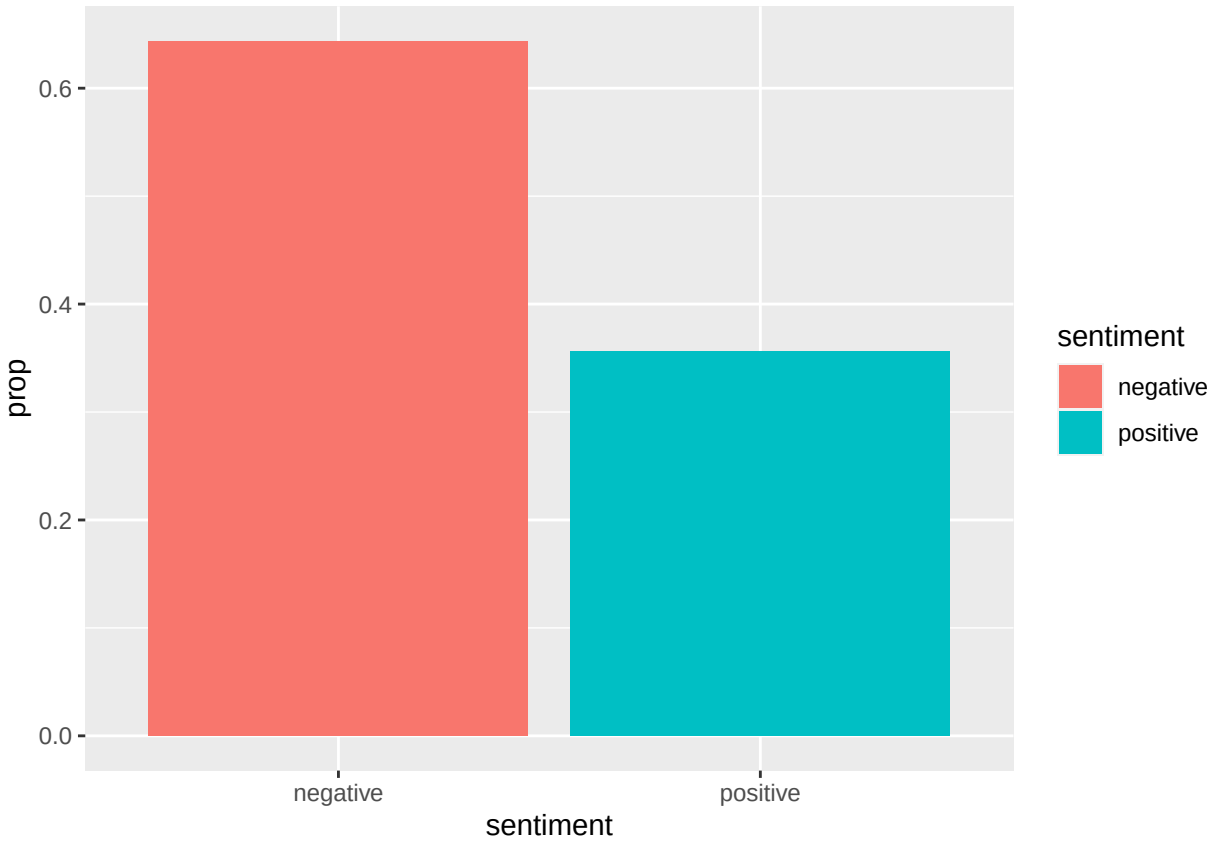

```
library(wordcloud)
library(viridis)
pal <- magma(n = 30, direction = -1)
frequent_wild %>%
  with(wordcloud(word, n, colors = pal,
          min.freq = 7, random.order = FALSE,
          scale = c(4, 1)))
```

d. Explore the sentiment of the text using two of the sentiment lexicons in `tidytext`. What does your analysis say about the sentiment of the text?

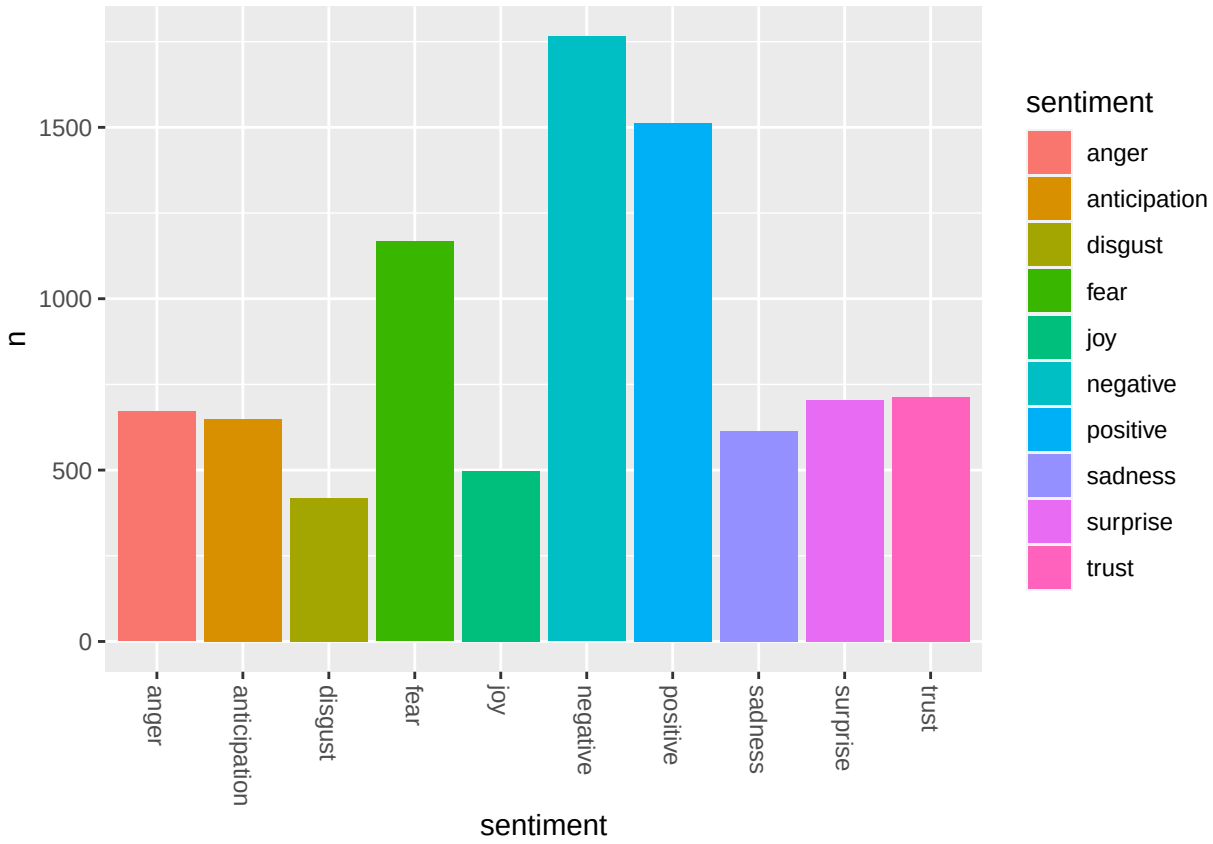
Notes:

- Make sure to NOT remove stop words this time.
- afinn is a numeric score and should be handled differently than the categorical scores.

```
bing <- get_sentiments("bing")
nrc <- get_sentiments("nrc")

tokenize_wild %>%
  inner_join(bing, by = "word") %>%
  group_by(sentiment) %>%
  count(word) %>%
  summarize(n = sum(n)) %>%
  mutate(prop = n/sum(n)) %>%
  ggplot(aes(x = sentiment, y = prop, fill  = sentiment)) +
  geom_col()
```




```
tokenize_wild %>%
  inner_join(nrc, by = "word") %>%
  group_by(sentiment) %>%
  count(word) %>%
  summarize(n = sum(n)) %>%
  mutate(prop = n/sum(n)) %>%
  ggplot(aes(x = sentiment, y = n, fill  = sentiment)) +
  geom_col() +
  theme(axis.text.x = element_text(angle = -90, vjust = 0.5, hjust = 0))
```



The sentiment of the text is definitely more negative than positive. Notably, the third most common sentiment was fear according the nrc lexicon. The nrc lexicon gives words multiple sentiments, for example, a word is negative, but also represents fear and disgust. Fear would not go along with a positive sentiment, meaning that most of the words that are described as negative also contain the sentiment of fear. This means that the book articulates many cases of fear, according to the lexicon sentiment scores.

e. If you didn't do so in 6.d, compute the average sentiment score of the text using `afinn`. Which positive words had the biggest impact? Which negative words had the biggest impact?

```
afinn <- get_sentiments("afinn")

tokenize_wild_afinn <- tokenize_wild %>%
  inner_join(afinn, by = "word") %>%
  group_by(value) %>%
  count(word) %>%
  mutate(sentiment_impact = (n*value))

#the average sentiment score of the text
avg_sentiment <- mean(tokenize_wild_afinn$sentiment_impact)
avg_sentiment
```

```
## [1] -1.119318
```

```
#positive words had the biggest impact (top 10 of sentiment_impact)
tokenize_wild_afinn %>%
  arrange(desc(sentiment_impact))
```

```
## # A tibble: 528 x 4
## # Groups:   value [9]
```

```
##    value word         n sentiment_impact
##    <dbl> <chr>    <int>            <dbl>
##  1     3 great       49              147
##  2     2 like        58              116
##  3     3 good        28               84
##  4     3 love        21               63
##  5     2 strength    21               42
##  6     3 best        10               30
##  7     3 greater     10               30
##  8     3 delight      8               24
##  9     2 save        10               20
## 10     2 strong      10               20
## # ... with 518 more rows
```

```
#negative words had the biggest impact (top 10 of sentiment_impact)
tokenize_wild_afinn %>%
  arrange(sentiment_impact)
```

```
## # A tibble: 528 x 4
## # Groups:   value [9]
##    value word         n sentiment_impact
##    <dbl> <chr>    <int>            <dbl>
##  1    -1 no          95              -95
##  2    -3 dead        22              -66
##  3    -2 fire        33              -66
##  4    -2 cried       21              -42
##  5    -3 lost        13              -39
##  6    -2 fear        17              -34
##  7    -3 killed      11              -33
##  8    -3 mad         11              -33
##  9    -3 terrible    11              -33
## 10    -2 death       16              -32
## # ... with 518 more rows
```

f. You should have found that "no" was an important negative word in the sentiment score. To know if that really makes sense, let's turn to the raw lines of text for context. Pull out all of the lines that have the word "no" in them. Make sure to not pull out extraneous lines (e.g., a line with the word "now").

```
str_subset(wild$text, pattern = "\\b(N|n)o\\b")
```

```
##  [1] "Manuel's treachery. No one saw him and Buck go off through the orchard"
##  [2] "solitary man, no one saw them arrive at the little flag station known"
##  [3] "the club, but his madness knew no caution. A dozen times he charged, and"
##  [4] "\"He's no slouch at dog-breakin', that's wot I say,\" one of the men on"
##  [5] "all, that he stood no chance against a man with a club. He had learned"
##  [6] "in the red sweater. \"And seem' it's government money, you ain't got no"
##  [7] "an animal. The Canadian Government would be no loser, nor would its"
##  [8] "he developed no affection for them, he none the less grew honestly to"
##  [9] "The other dog made no advances, nor received any; also, he did not"
## [10] "No lazy, sun-kissed life was this, with nothing to do but loaf and be"
## [11] "were not town dogs and men. They were savages, all of them, who knew no"
## [12] "wolf, though not half so large as she. There was no warning, only a leap"
## [13] "trouble him in his sleep. So that was the way. No fair play. Once down,"
## [14] "turned to run when he saw that appeasement was of no avail, and cried"
## [15] "(still appeasingly) when Spitz's sharp teeth scored his flank. But no"
## [16] "no more trouble. His only apparent ambition, like Dave's, was to be left"
```

```
## [17] "again he returned. Were they in the tent? No, that could not be, else he"
## [18] "unduly civilized dog, and of his own experience knew no trap and so"
## [19] "ice was very thin, and where there was swift water, there was no ice at"
## [20] "his unfinished ration. There was no defending it. While he was fighting"
## [21] "days, no matter what the odds, he had never run from a fight. But"
## [22] "as well as external economy. He could eat anything, no matter how"
## [23] "night in advance. No matter how breathless the air when he dug his"
## [24] "they ran it down. It was no task for him to learn to fight with cut"
## [25] "he betrayed no impatience, shunned all offensive acts."
## [26] "as he circled back and forth for a chance to spring in. Buck was no less"
## [27] "eager, and no less cautious, as he likewise circled back and forth for"
## [28] "irresistible. There was no opposing them. The team-dogs were swept back"
## [29] "there was no hope for him. But he braced himself to the shock of Spitz's"
## [30] "the sled lashings and canvas coverings. In fact, nothing, no matter how"
## [31] "Again, the rim ice broke away before and behind, and there was no escape"
## [32] "Things no longer went right. There was continual bickering and jangling."
## [33] "had destroyed the solidarity of the team. It no longer was as one dog"
## [34] "into all kinds of petty misdemeanors. No more was Spitz a leader greatly"
## [35] "There was no hope for him. Buck was inexorable. Mercy was a thing"
## [36] "make good time. No more Spitz, no more trouble, sure.\""
## [37] "There was no place for Buck save at the front. Once more Francois"
## [38] "\"Nevaire such a dog as dat Buck!\" he cried. \"No, nevaire! Heem worth one"
## [39] "there was no new-fallen snow with which to contend. It was not too cold."
## [40] "Dawson. It was no light running now, nor record time, but heavy toil"
## [41] "distant, and such memories had no power over him. Far more potent were"
## [42] "ate, and no man sought his sleeping-robe till he had seen to the feet of"
## [43] "locate no broken bones, could not make it out."
## [44] "tried to drive him away with the whip; but he paid no heed to the"
## [45] "They were all terribly footsore. No spring or rebound was left in them."
## [46] "slow and prolonged strength drainage of months of toil. There was no"
## [47] "power of recuperation left, no reserve strength to call upon. It had"
## [48] "but no businesslike method. The tent was rolled into an awkward bundle"
## [49] "\"Whoa! whoa!\" but they gave no heed. He tripped and was pulled off his"
## [50] "nothing lively about it, no snap or go in him and his fellows. They were"
## [51] "Buck felt vaguely that there was no depending upon these two men and the"
## [52] "at all. And on no day did they succeed in making more than half the"
## [53] "that for love or money no additional dog-food was to be obtained. So"
## [54] "ration of the husky, so the six Outside dogs under Buck could do no less"
## [55] "and kindly, did not come to these two men and the woman. They had no"
## [56] "she made their lives unendurable. She no longer considered the dogs, and"
## [57] "a nightmare. He pulled when he could; when he could no longer pull, he"
## [58] "still at the head of the team, but no longer enforcing discipline or"
## [59] "to Thornton's warning to take no more chances on the rotten ice. \"They"
## [60] "made no effort. He lay quietly where he had fallen. The lash bit into"
## [61] "no longer felt anything, though very faintly he could hear the impact of"
## [62] "the club upon his body. But it was no longer his body, it seemed so far"
## [63] "Thornton stood between him and Buck, and evinced no intention of getting"
## [64] "Hal had no fight left in him. Besides, his hands were full with his"
## [65] "To Buck's surprise these dogs manifested no jealousy toward him. They"
## [66] "names. Buck knew no greater joy than that rough embrace and the sound of"
## [67] "had come into the Northland had bred in him a fear that no master could"
## [68] "but the strange dog, no matter what the breed or valor, swiftly"
## [69] "there was no middle course. He must master or be mastered; while to show"
## [70] "Thornton shook his head. \"No, it is splendid, and it is terrible, too."
```

```
## [71] "a stretch of wild water in which no swimmer could live."
## [72] "permitting no slack, while Pete kept it clear of coils. Buck held on"
## [73] "of a dozen men fixed upon him, silent and waiting. Further, he had no"
## [74] "There were no takers. Not a man believed him capable of the feat."
## [75] "were no more than in proportion with the rest of the body, where the"
## [76] "frankly down his cheeks. \"Sir,\" he said to the Skookum Bench king, \"no,"
## [77] "shrouded in mystery. No one knew of the first man. The oldest tradition"
## [78] "But no living man had looted this treasure house, and the dead were"
## [79] "in no haste, Indian fashion, he hunted his dinner in the course of the"
## [80] "uncharted vastness, where no men were and yet where men had been if"
## [81] "where wildfowl had been, but where then there was no life nor sign of"
## [82] "packed flat, And that was all--no hint as to the man who in an early day"
## [83] "They sought no farther. Each day they worked earned them thousands of"
## [84] "He had made no noise, yet it ceased from its howling and tried to sense"
## [85] "that no harm was intended, finally sniffed noses with him. Then they"
## [86] "again he took to wandering in the woods, but the wild brother came no"
## [87] "who would quarrel no more."
## [88] "secrecy of the forest. He no longer marched. At once he became a thing"
## [89] "proceeded to cut the bull out from the herd. It was no slight task. He"
## [90] "there throughout the summer. No longer was this fact borne in upon him"
## [91] "no withstanding him. He plunged about in their very midst, tearing,"
## [92] "Thornton; for Buck followed his trace into the water, from which no"
## [93] "to kill a husky dog than them. They were no match at all, were it"
## [94] "dead. The last tie was broken. Man and the claims of man no longer bound"
```

g. Draw some conclusions about how "no" is used in the text. No is not used in the text often as a yes/no dichotomy which tends to be related to the positive/negative dichnotomy and I imagine is how the sentiment score defines every no in the text. This means that no is probably not fully representing what each "no"'s true sentiment is. Looking at the text, no is often used to explain where there was nothing left, "no more trouble"/"no longer was"/"no longer pull", which I see as having a negative sentiment.

h. Let's look at the bigrams (2 consecutive words). Tokenize the text by bigrams.

```
bigrams <- wild %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
```

i. Produce a sorted table that counts the frequency of each bigram and notice that stop words are still an issue.

```
bigrams %>%
  count(bigram) %>%
  mutate(prop = n/sum(n)) %>%
  arrange(desc(n))
```

```
## # A tibble: 20,538 x 3
##    bigram       n    prop
##    <chr>    <int>   <dbl>
##  1 of the     246 0.00766
##  2 in the     176 0.00548
##  3 he was     131 0.00408
##  4 to the     122 0.00380
##  5 it was     111 0.00346
##  6 and the    103 0.00321
##  7 on the      86 0.00268
##  8 he had      83 0.00258
##  9 at the      71 0.00221
```

```
## 10 into the    69 0.00215
## # ... with 20,528 more rows
```

j. Put each of the bigram words in its own column (hint: Use a `tidyr` function) and then remove any row where either the first word or the second word is a stop word.

```
sep_bigram <- bigrams %>%
  separate(col = bigram, into = c("first_word", "second_word"), sep = "\\s")
sep_bigram <- sep_bigram %>%
  anti_join(stop_words, by = c("first_word" = "word"))
```

k. Produce a sorted table that counts the frequency of each bigram. (Do you remember "the man in the red sweater?")

```
sep_bigram$bigram = paste(sep_bigram$first_word, sep_bigram$second_word)

sep_bigram %>%
  count(bigram) %>%
  mutate(prop = n/sum(n)) %>%
  arrange(desc(n))
```

```
## # A tibble: 10,595 x 3
##    bigram            n     prop
##    <chr>         <int>    <dbl>
##  1 john thornton    34 0.00269
##  2 buck was         29 0.00229
##  3 sol leks         27 0.00214
##  4 buck and         20 0.00158
##  5 dogs and         18 0.00142
##  6 till he          14 0.00111
##  7 dog and          12 0.000949
##  8 red sweater      12 0.000949
##  9 buck had         11 0.000870
## 10 half breed       11 0.000870
## # ... with 10,585 more rows
```

Problem 7: Your Sentiment Analysis

Now it is your turn. Pick 4 of the texts from `gutenbergr` or 4 albums from `genius` and do the following:

- Create wordclouds for word frequency for each text/album.
- Create a graphic that compares the sentiments of the texts/albums in some way.
- Create a graphic that presents the tf_idf's for the important words in each text.

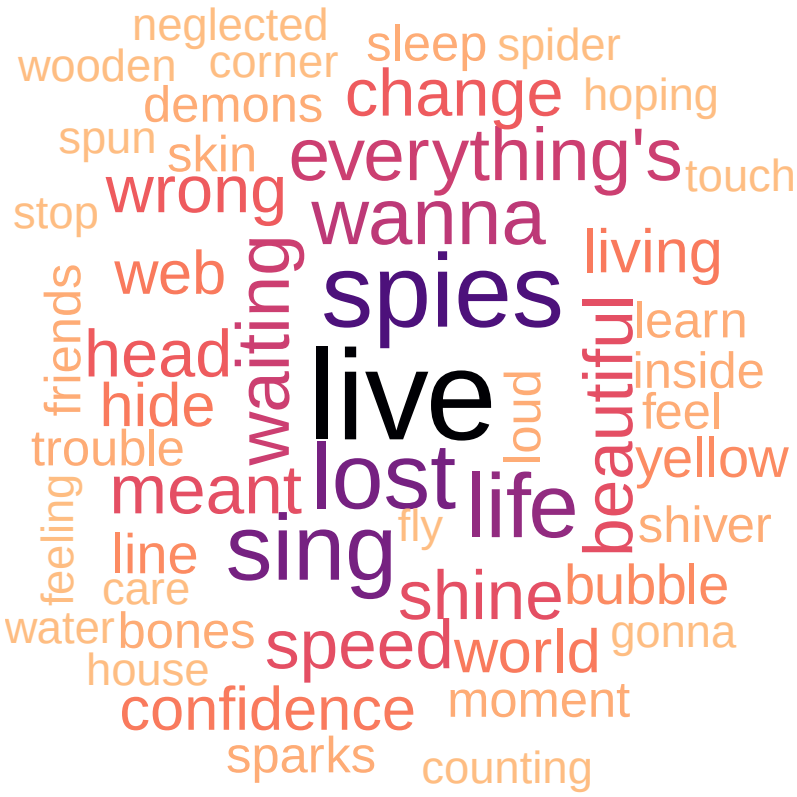
- Write a paragraph or two of key takeaways.

```
library(genius)
MyloXyloto <- genius_album(artist = "Coldplay", album = "Mylo Xyloto") %>%
  mutate(album = "MyloXyloto")
Parachutes <- genius_album(artist = "Coldplay", album = "Parachutes") %>%
  mutate(album = "Parachutes")
Viva_la_vida <- genius_album(artist = "Coldplay", album =
                               "Viva la Vida or Death and All His Friends") %>%
  mutate(album = "Viva_la_vida")
A_Head_Full_of_Dreams <- genius_album(artist = "Coldplay",
                                      album = "A Head Full of Dreams") %>%
  mutate(album = "A_Head_Full_of_Dreams")
```

```
cold <- bind_rows(MyloXyloto, Parachutes, Viva_la_vida,
                      A_Head_Full_of_Dreams) %>%
  unnest_tokens(output = word, input = lyric, token = "words") %>%
  anti_join(stop_words, by = "word") %>%
  filter(!(word %in% c("ooh", "la", "ah", "yeah", "para", "ooo", "oooo", "wa",
                       "aah", "na", "oo", "ooooooh", "woah", "woo", "hoo")))

coldplay <- bind_rows(MyloXyloto, Parachutes, Viva_la_vida,
                      A_Head_Full_of_Dreams) %>%
  unnest_tokens(output = word, input = lyric, token = "words") %>%
  anti_join(stop_words, by = "word") %>%
  filter(!(word %in% c("ooh", "la", "ah", "yeah", "para", "ooo", "oooo", "wa",
                       "aah", "na", "oo", "ooooooh", "woah", "woo", "hoo"))) %>%
  count(album, word) %>%
  group_by(album) %>%
  mutate(prop = n/sum(n)) %>%
  arrange(desc(n))

coldplay_tidy <- bind_rows(MyloXyloto, Parachutes, Viva_la_vida,
                           A_Head_Full_of_Dreams) %>%
  unnest_tokens(output = word, input = lyric, token = "words") %>%
  anti_join(stop_words, by = "word") %>%
  filter(!(word %in% c("ooh", "la", "ah", "yeah", "para", "ooo", "oooo", "wa",
                       "aah", "na", "oo", "ooooooh", "woah", "woo", "hoo"))) %>%
  count(album, word, sort = TRUE) %>%
  bind_tf_idf(word, album, n)

coldplay_wider <- coldplay %>%
  select(album, word, prop) %>%
  pivot_wider(names_from = album, values_from = prop)
```

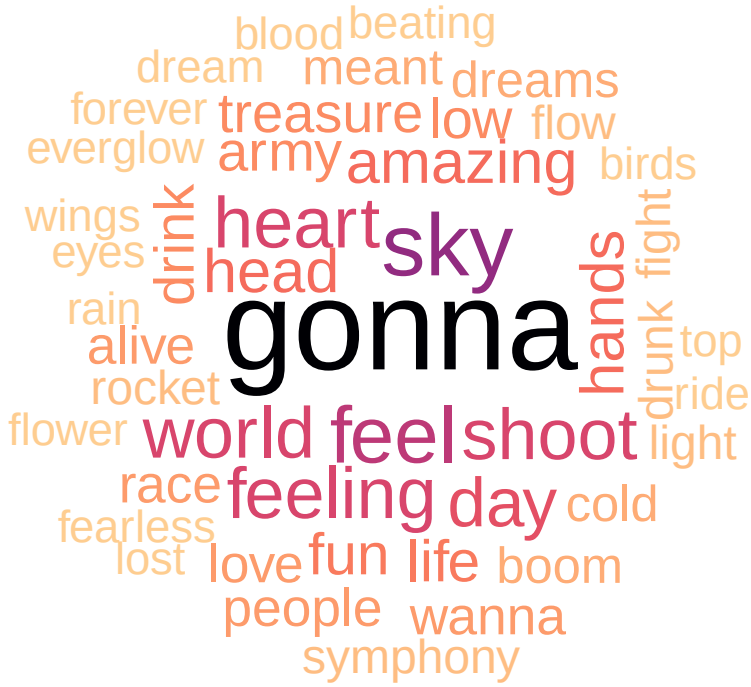
```
#Mylo Xyloto
coldplay %>%
  filter(album == "MyloXyloto") %>%
  with(wordcloud(word, n, scale = c(4, 1),
                 colors = pal, min.freq = 3,
                 random.order = FALSE))
```


```
#Parachutes
coldplay %>%
  filter(album == "Parachutes") %>%
  with(wordcloud(word, n, scale = c(4, 1),
                 colors = pal, min.freq = 3,
                 random.order = FALSE))
```

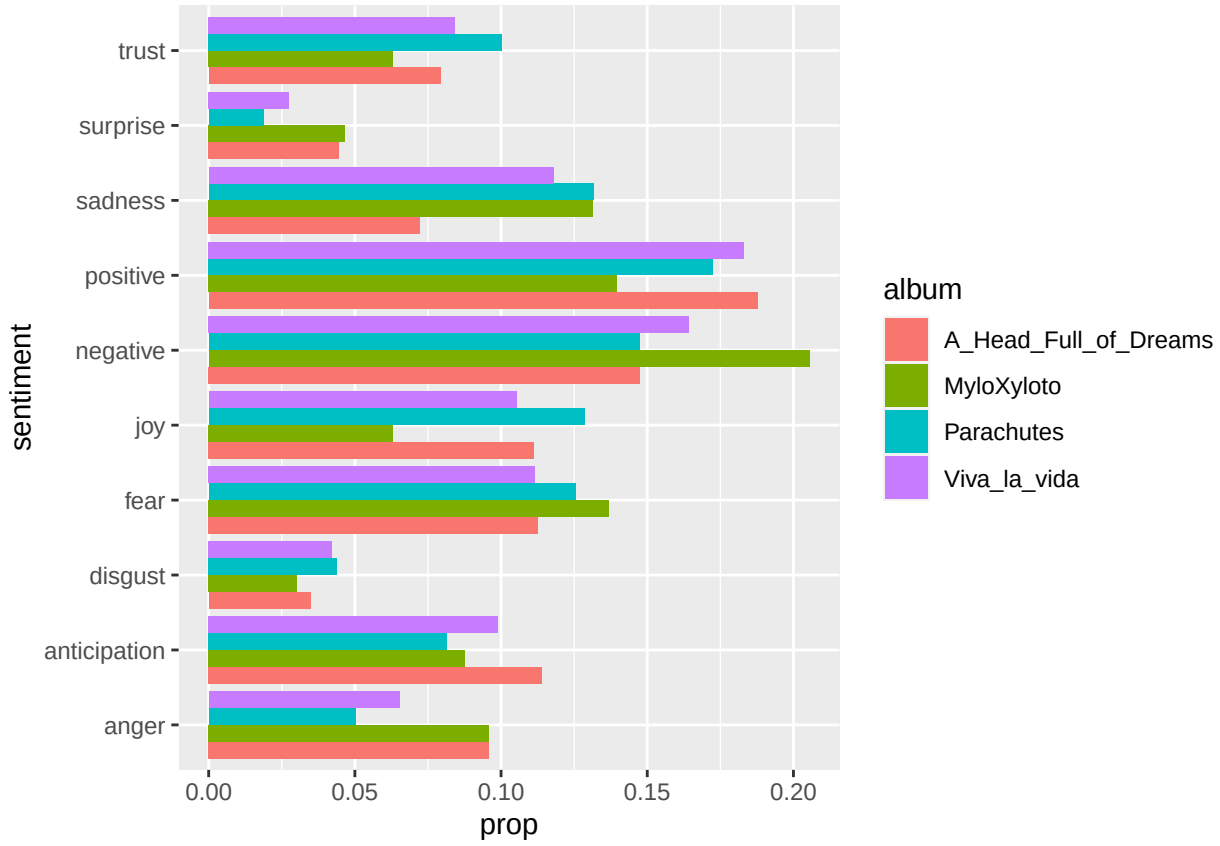



```
#Viva la vida
coldplay %>%
  filter(album == "Viva_la_vida") %>%
  with(wordcloud(word, n, scale = c(4, 1),
                 colors = pal, min.freq = 3,
                 random.order = FALSE))
```

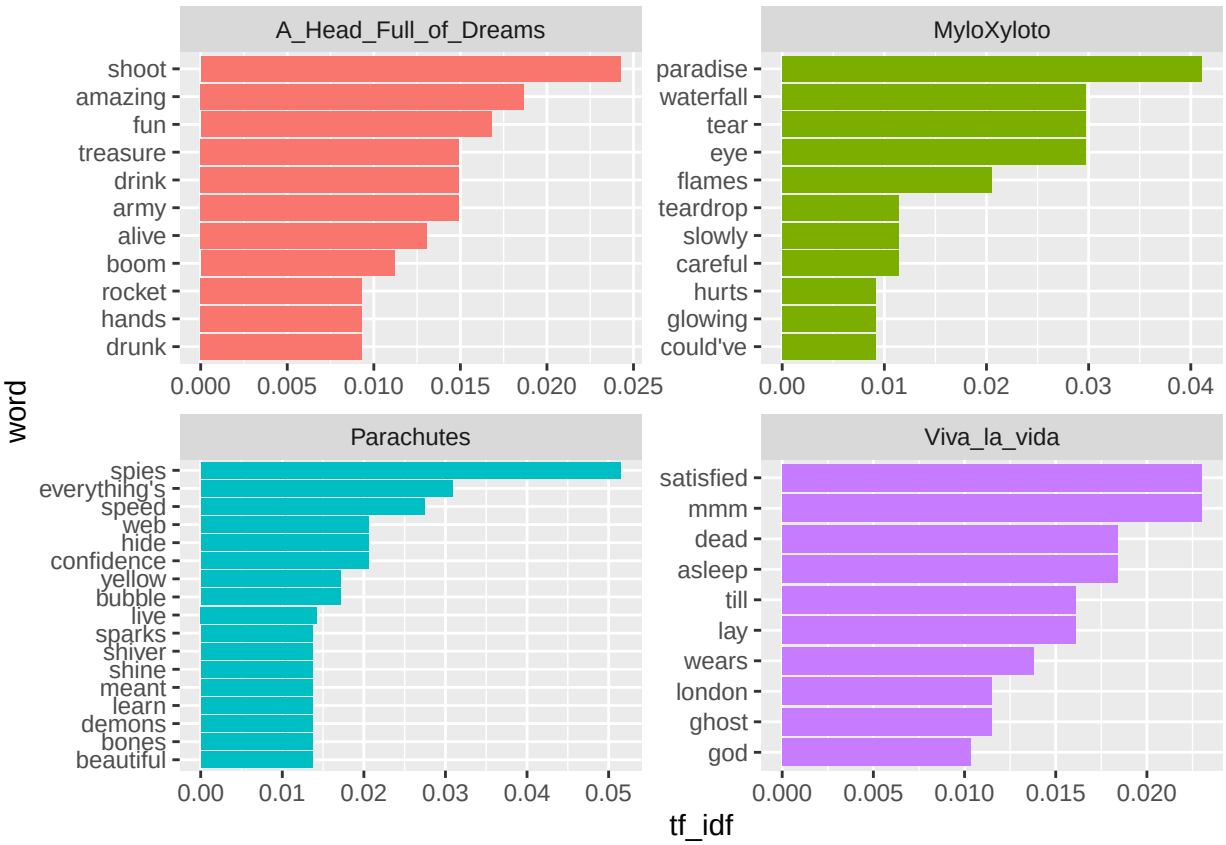

```
#A Rush of Blood to the Head
coldplay %>%
  filter(album == "A_Head_Full_of_Dreams") %>%
  with(wordcloud(word, n, scale = c(4, 1),
                 colors = pal, min.freq = 4,
                 random.order = FALSE))
```



```r
#a graphic that compares the sentiments of the texts/albums in some way
cold %>%
  inner_join(nrc, by = "word") %>%
  group_by(album, sentiment) %>%
  count(word) %>%
  summarize(n = sum(n)) %>%
  mutate(prop = n/sum(n)) %>%
  ggplot(aes(x = sentiment, y = prop, fill  = album)) +
  geom_col(position = "dodge") +
  coord_flip()
```




```r
#a graphic that presents the tf_idf's for the important words in each text
coldplay_tidy %>%
  group_by(album) %>%
  slice_max(tf_idf, n = 10) %>%
  ungroup() %>%
  mutate(word = fct_reorder(word, tf_idf)) %>%
  ggplot(aes(x = word, y = tf_idf, fill = album)) +
  geom_col(show.legend = FALSE) + coord_flip() +
  facet_wrap(~album, ncol = 2, scales = "free")
```



For this problem, I looked at and compared four Coldplay albums. The sentiment chart gives the most interesting comparison of the albums. All albums had minimal sentiments of `surprise` and `disgust`, shows that that the artist does not write much to those emotions. The plot also shows that one album, `Mylo Xyloto`, is quite different from the the other three. Specifically, it is the only album where it has a higher proportion of words that articulate a negative sentiment than positive. The word cloud also points this out. Though the albums most prominant words are `paradise` and `heart`, most of the surrounding words do not have a similar positive sentiment ("tear", "hurt", "dark", "cold", "lost", "flames", "chaos"). The 'A Head Full of Dreams' album, on the other hand, his the highest positive sentiment, but this album also stands out in the anticipation sentiment, it is pointedly higher than the other albums. This checks out when we look at the word clouds, as the most prominent word in the `A Head Full of Dreams` word cloud is "gonna" with other action words around it like "wanna", "race", "dream", and "beating." The tf_idf important words graphic shows the positivity of the album ("amazing", "fun", and "treasure" being listed).