Homework 3

Name: Arthur Viegas Eguia

I worked with: Allison, Jacob Aronson (stats help center), Piper Dean (stats help center)

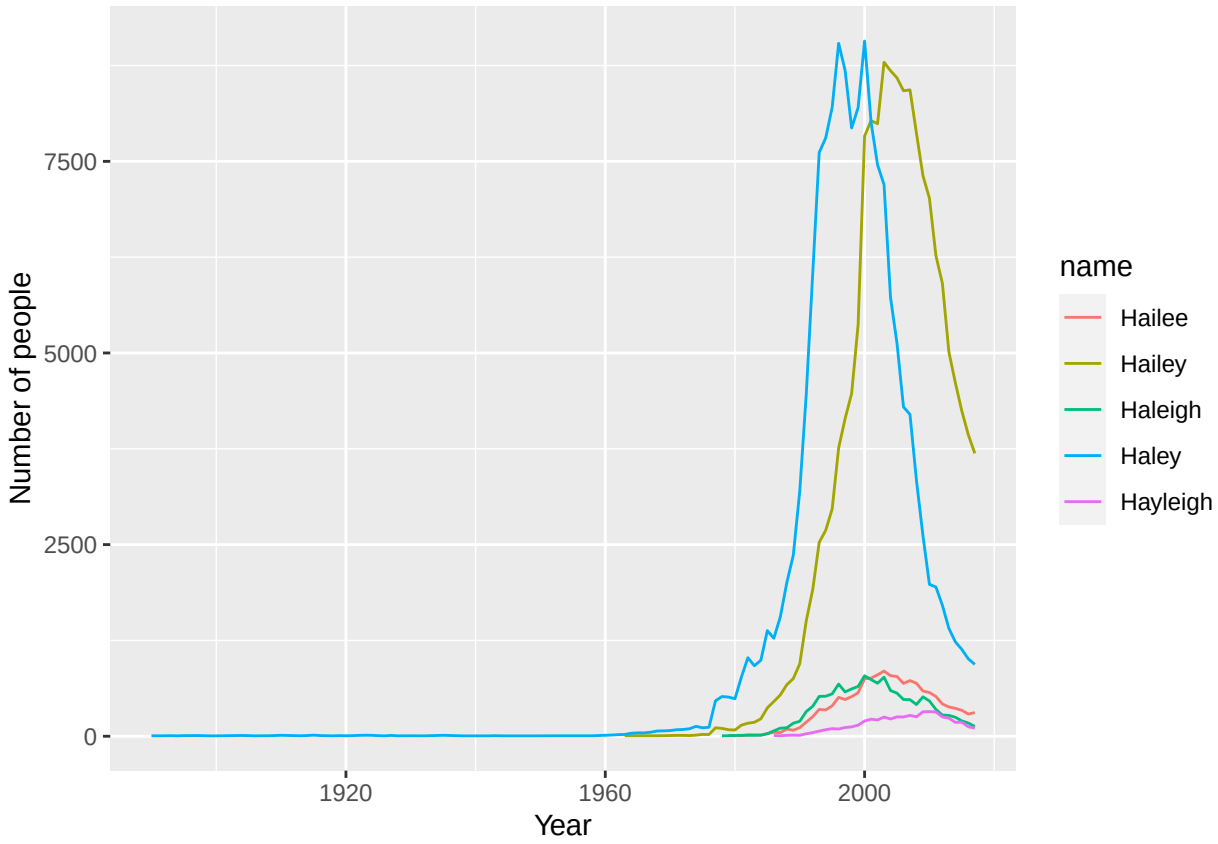
Click the "Knit" button in RStudio to knit this file to a pdf.

Problem 1: babynames

a.

answer: The names Haley and Hailey started becming popular in the 70s, the popularity of these names greatly improved peaking in the 90s and early 2000s. The names Hailee, Haleigh and Hayleigh arised in the 90s, peaking in the 2000s, but never got the same popularity as Haley or Hailey.

```
babies <- babynames

#Gets the number of female babies named Haley
babynames_hailey <- babynames %>%
  filter(name %in% c("Hailey", "Hailee", "Haleigh", "Haley", "Hayleigh"), sex == "F")

#Plots them at a line graph
ggplot(babynames_hailey, aes(y = n, x = year, color = name)) +
  geom_line() +
  labs(y = "Number of people", x = "Year")
```



b.

answer: Blair, with 14470 male babies and 14195 female babies.

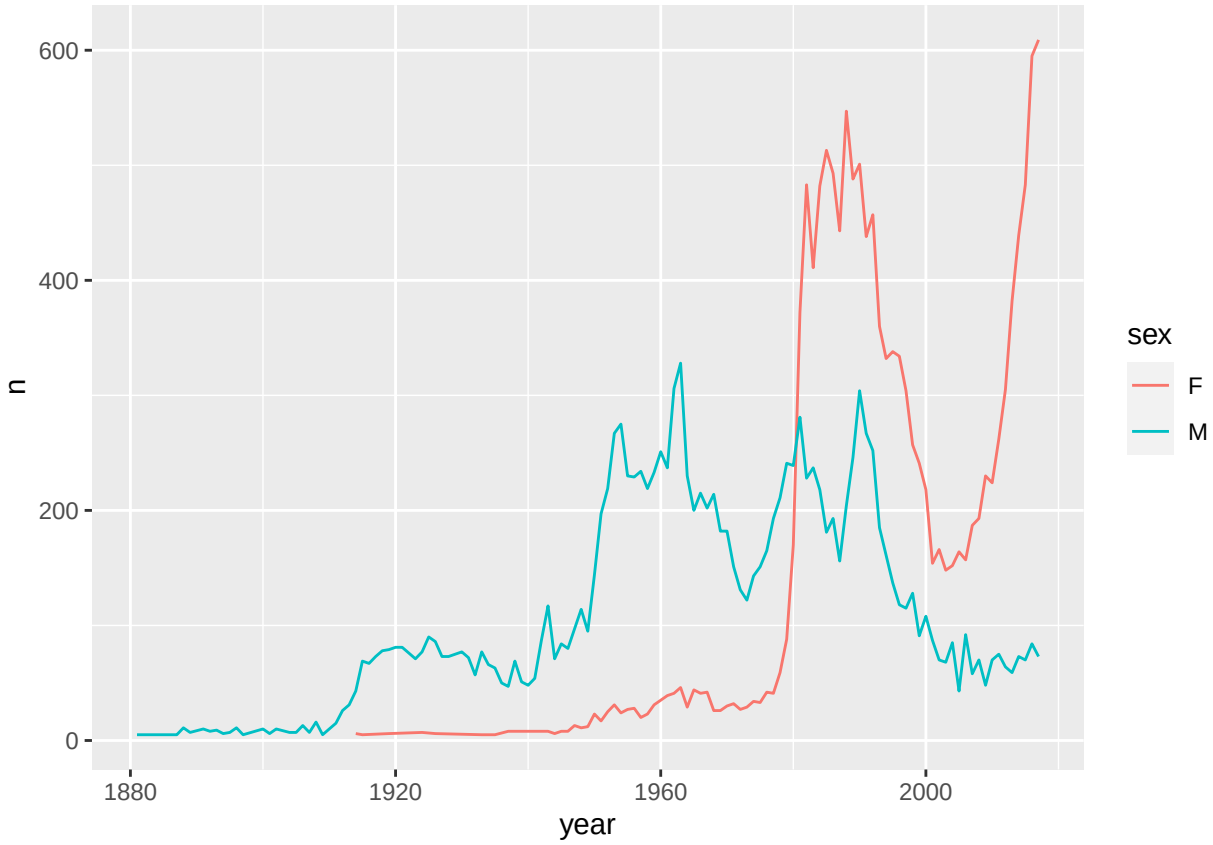
```
#Groups the items based on name and sex. Gets names that have at least 10000 occurrences
popular_names <- babynames %>%
  group_by(sex, name) %>%
  summarise(total = sum(n)) %>%
  filter(total >= 10000)

#Gets Male and Female names
male_names <- popular_names %>% filter(sex == "M")
female_names <- popular_names %>% filter(sex == "F")


#Joins the names together, so we pair "name" (M) and "name" (F)
common_names <- male_names %>%
  inner_join(female_names, by = "name")


#Calculates the proportion of male and female names
common_names <- common_names %>%
  mutate(proportion_male = total.x/(total.x + total.y),
         proportion_female = total.y/(total.x + total.y))

#special thanks to Allison and Jacob Aronson
#Gets the most balanced name
```

```r
most_balanced <- common_names %>%
  filter((proportion_male - 0.5) == min(abs(common_names$proportion_male - 0.5)))


most_balanced
## # A tibble: 1 x 7
##   sex.x name  total.x sex.y total.y proportion_male proportion_female
##   <chr> <chr>   <int> <chr>   <int>           <dbl>             <dbl>
## 1 M     Blair   14470 F       14195           0.505             0.495
```

c.

answer: Blair was most popular as a male name between the 40s and 60s. It became a popular female name in the late 70s decreasing in the early 200s. Right now, its popularity as a female name is increasing, while there are few males named Blair.

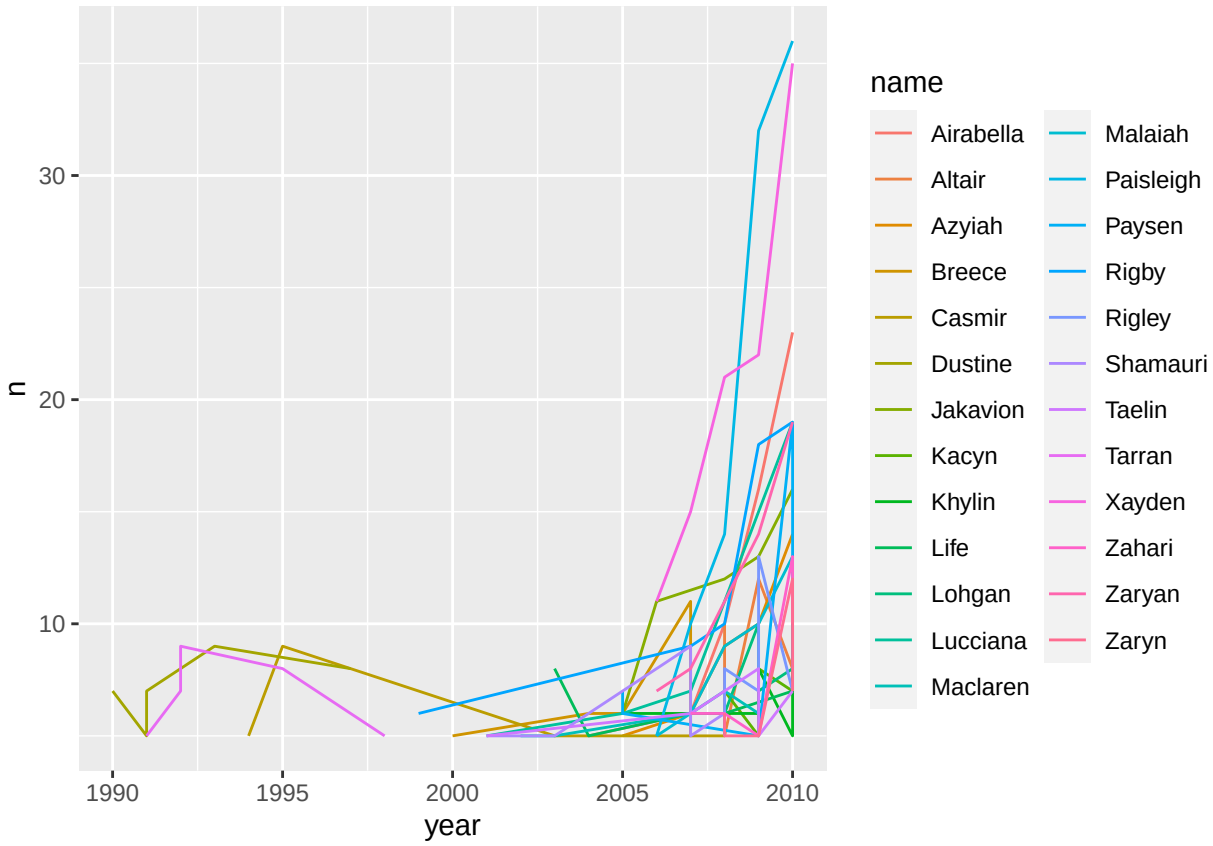
```r
#Gets information about the most balanced name
name_c <- babynames %>%
  filter(name == most_balanced$name)

#Makes a line chart with this information
ggplot(name_c, aes(y = n, x = year, color = sex)) +
  geom_line()
```



d.

answer: There are around 2000 names that increased in popularity from 1990 until 2012. The graph shows some examples

```r
#Special thanks to Allison and Jacob Aronson
#Groups babies from the desired time period by name and sex, gets the names that increased in popularity
increased_popularity <- babies %>%
  group_by(name, sex) %>%
  filter(1990 <= year & year <= 2010) %>%
  arrange(name, sex, year) %>%
  mutate(last_year = lag(n)) %>%
  mutate(year_difference = n - last_year) %>%
  tidyr::drop_na() %>%
  summarise(min_difference = min(year_difference)) %>%
  filter(min_difference > 0)

#Prints the information on the screen
increased_popularity
## # A tibble: 2,887 x 3
## # Groups:   name [2,880]
##    name      sex   min_difference
##    <chr>     <chr>          <int>
##  1 Aaban     M                  1
##  2 Aadon     M                  1
##  3 Aahana    F                  1
##  4 Aania     F                  1
##  5 Aanshi    F                  5
##  6 Aarav     M                 16
##  7 Aariyanna F                  4
##  8 Aarna     F                  4
##  9 Aaronn    M                  2
## 10 Aashritha F                  1
## # i 2,877 more rows

#Treats the data so it can be printed in a chart
graph_data <- increased_popularity %>%
  inner_join(babies, by = "name") %>%
  filter(year %in% 1990:2010) %>%
  filter(n() == 5)

#Graphs the names that increased in popularty
ggplot(graph_data, aes(x = year, y = n, color = name)) +
  geom_line()
```




```
increased_popularity
## # A tibble: 2,887 x 3
## # Groups:   name [2,880]
##    name      sex   min_difference
##    <chr>     <chr>          <int>
##  1 Aaban     M                  1
##  2 Aadon     M                  1
##  3 Aahana    F                  1
##  4 Aania     F                  1
##  5 Aanshi    F                  5
##  6 Aarav     M                 16
##  7 Aariyanna F                  4
##  8 Aarna     F                  4
##  9 Aaronn    M                  2
## 10 Aashritha F                  1
## # i 2,877 more rows
```

Problem 2: consumption

```
food_consumption <- read.csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/foodconsumptio
```

a.

answer: The top free food category types per person in 2018 were Milk - inc. cheese, Wheat and Wheat Products and Rice

```
#Groups the food by category
food_consumption.df <- food_consumption %>%
  group_by(food_category) %>%
  summarise(total_consumption = sum(consumption)) %>% #Gets the sum of consumption by category
  arrange(desc(total_consumption)) %>%
  slice(1:3) #Gets teh top 3 categories
food_consumption.df
## # A tibble: 3 x 2
##   food_category            total_consumption
##   <chr>                                <dbl>
## 1 Milk - inc. cheese                  16351.
## 2 Wheat and Wheat Products             9301.
## 3 Rice                                 3819.
```

b.

answer: The top 5 countries by Milk - inc. cheese consumption are Finland, Netherlands, Botswana, Sweden, Switzerland

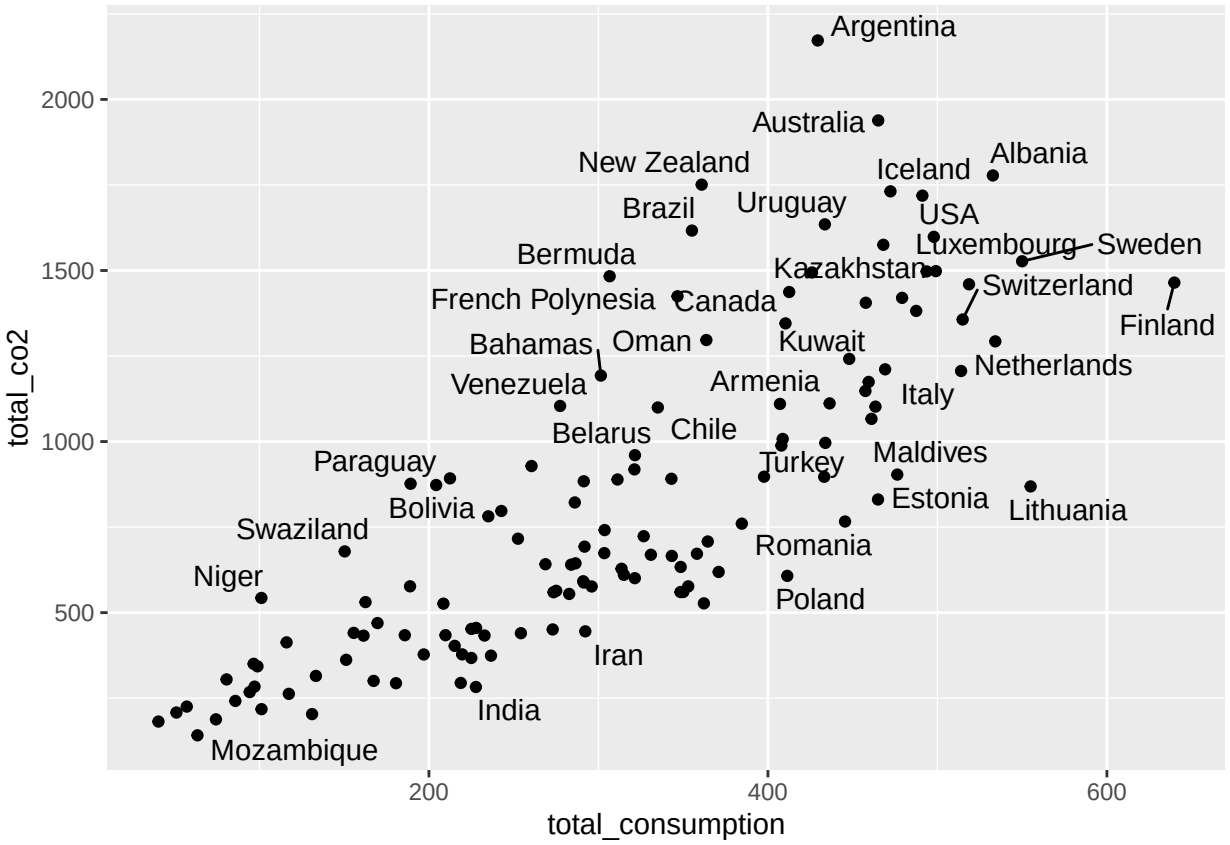
```
#Computes the percentage of consumption for each food category by country
food_consumption_by_category <- food_consumption %>%
  group_by(country) %>%
  mutate(total_consumption = sum(consumption),
         proportion_consumption = consumption/total_consumption)


#Creates a data frame with the 5 countries with the highest Milk consumption
food_consumption_by_category %>%
  filter(food_category == "Milk - inc. cheese") %>%
  arrange(desc(proportion_consumption)) %>%
  ungroup() %>%
  slice(1:5)
## # A tibble: 5 x 7
##       X country     food_category    consumption co2_emmission total_consumption
##   <int> <chr>       <chr>                  <dbl>         <dbl>             <dbl>
## 1   150 Finland     Milk - inc. che~         431.          614.              640.
## 2   260 Netherlands Milk - inc. che~         341.          486.              534.
## 3   788 Botswana    Milk - inc. che~         118.          168.              189.
## 4   117 Sweden      Milk - inc. che~         341.          486.              550
## 5   249 Switzerland Milk - inc. che~         319.          454.              515.
## # i 1 more variable: proportion_consumption <dbl>
```

c.

answer: The plot is shown below

```
#Gets the total food consumption across all categories and co2 emission per person
consumption_and_co2 <- food_consumption %>%
  group_by(country) %>%
  summarise(total_consumption = sum(consumption),
            total_co2 = sum(co2_emmission))
consumption_and_co2
## # A tibble: 130 x 3
##    country    total_consumption total_co2
##    <chr>                  <dbl>     <dbl>
```

```
##  1 Albania                533.      1778.
##  2 Algeria                365.       708.
##  3 Angola                 116.       413.
##  4 Argentina              429.      2172.
##  5 Armenia                407.      1110.
##  6 Australia              465.      1939.
##  7 Austria                469.      1211.
##  8 Bahamas                301.      1193.
##  9 Bangladesh             237.       374.
## 10 Barbados               311.       889.
## # i 120 more rows

#Creates a scatterplot with names points
ggplot(consumption_and_co2, aes(y = total_co2, x = total_consumption)) + geom_point() +geom_text_repel(a
```



Problem 3: restaurant violations

```
data("Violations", package = "mdsr")
```

a.

answer: Data frame shown below

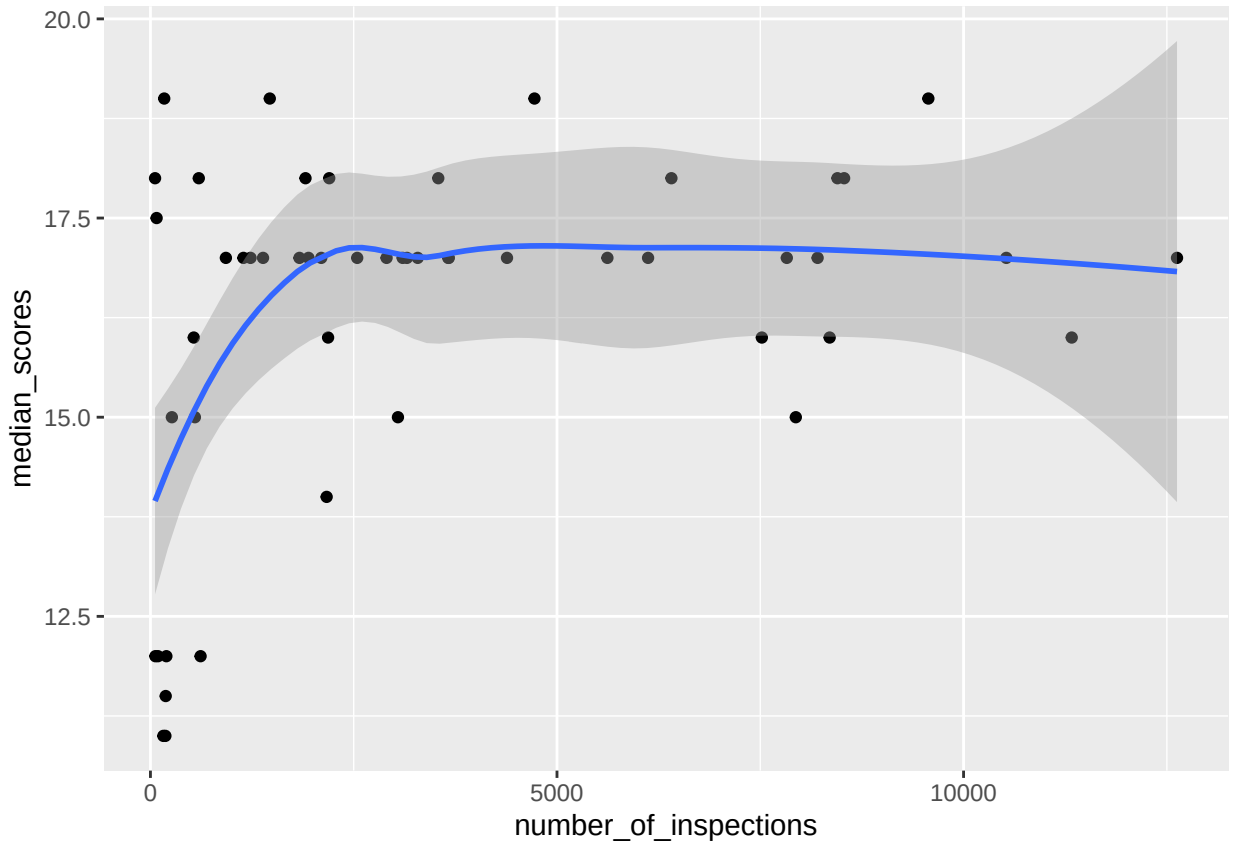
```
food_violations.df <- Violations %>%
  tidyr::drop_na(score) %>% #Drops null values
```

```
  filter(boro == "MANHATTAN") %>% #Makes sure there are only cases from Manhattan
  group_by(zipcode) %>% #Groups by zipcode
  summarise(median_scores = median(score),
            number_of_inspections = n()) %>% #Gets the median and number of of inspections by zipcode
  filter(number_of_inspections >= 50) #Only gets zip codes with 50 or more inspections
food_violations.df
## # A tibble: 51 x 3
##    zipcode median_scores number_of_inspections
##      <int>         <dbl>                 <int>
##  1   10001            15                  7937
##  2   10002            18                  8449
##  3   10003            17                 12625
##  4   10004            14                  2167
##  5   10005            17                  1144
##  6   10006            17                   928
##  7   10007            16                  2185
##  8   10009            17                  5620
##  9   10010            17                  4385
## 10   10011            17                  8205
## # i 41 more rows
```

b.

answer: When there are few inspections the violation score tends to be low. When there are more inspections the violation score increases, until it stabilizes at around 17.

```
food_violations.df
## # A tibble: 51 x 3
##    zipcode median_scores number_of_inspections
##      <int>         <dbl>                 <int>
##  1   10001            15                  7937
##  2   10002            18                  8449
##  3   10003            17                 12625
##  4   10004            14                  2167
##  5   10005            17                  1144
##  6   10006            17                   928
##  7   10007            16                  2185
##  8   10009            17                  5620
##  9   10010            17                  4385
## 10   10011            17                  8205
## # i 41 more rows
ggplot(food_violations.df, aes(x = number_of_inspections, y = median_scores)) + geom_point() + geom_smo
```

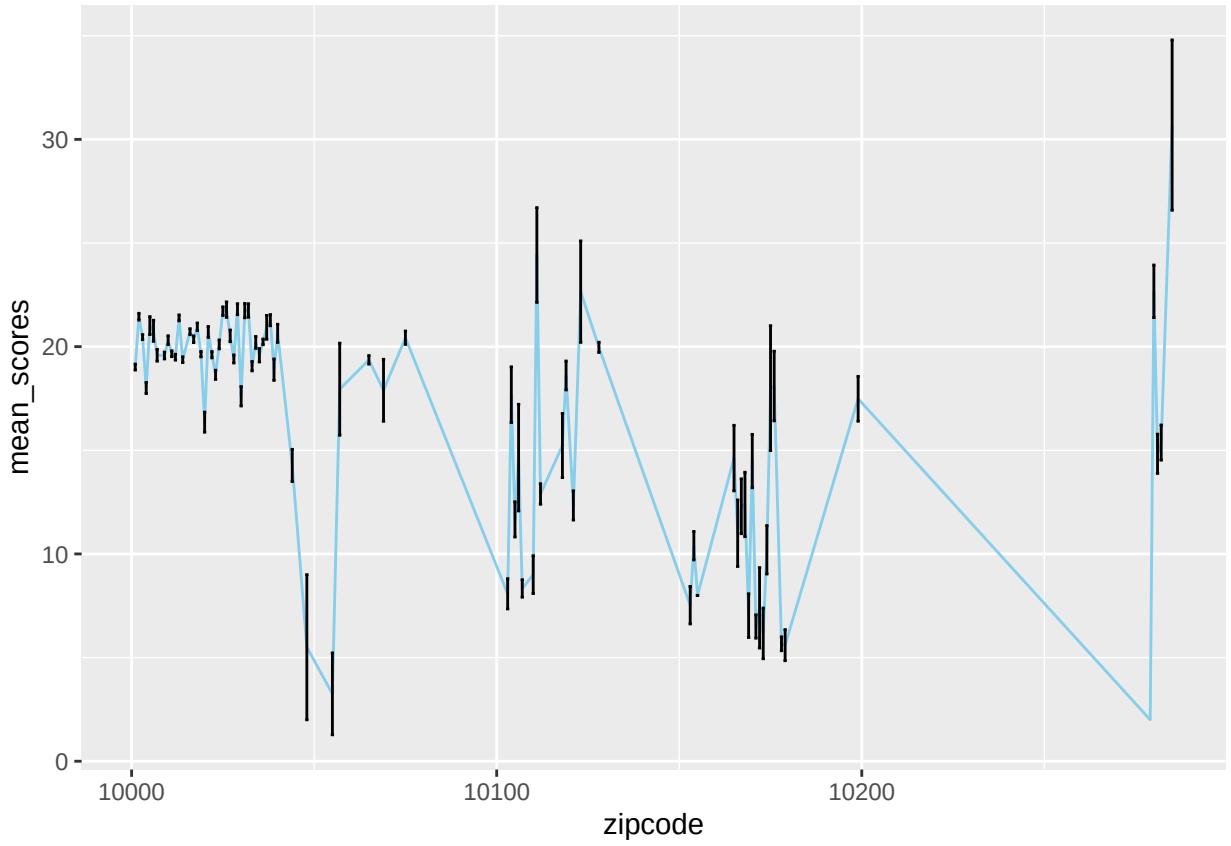


c.

answer: The max mean violation score is in Zipcode 10285. This can be confirmed by the chart, as the highest mean violation score has one of the highest zipcode numbers, it is slightly above 10275. The mean value is around 30. The minimum mean violation score is in zipcode 10279. This can be further confirmed by the graph. The minimum mean violation score is after 1050, most likely after 10275, but before the maximum mean violation score (which is 10285). The minimun mean violation score is around 3.5 (by the graph).

```
food_violations2.df <- Violations %>%
  tidyr::drop_na(score) %>% #Drops NA values
  filter(boro == "MANHATTAN") %>% #Only considers violations from Manhattan
  group_by(zipcode) %>% #gets all possible zipcodes as groups
  summarise(mean_scores = mean(score), #Gets the mean score and standard errpr
            se_scores = sd(score)/sqrt(n())) %>%
  arrange(mean_scores) #Orders mean violations
food_violations2.df
## # A tibble: 81 x 3
##    zipcode mean_scores se_scores
##      <int>       <dbl>     <dbl>
##  1   10279        2       NA
##  2   10055        3.25     1.97
##  3   10048        5.5      3.5
##  4   10179        5.6      0.748
##  5   10178        5.67     0.333
##  6   10173        6.17     1.22
##  7   10171        6.5      0.563
##  8   10169        7.02     1.05
```

```
##  9   10172        7.4      1.94
## 10   10153        7.53     0.904
## # i 71 more rows

#Creates plot plus error bars
ggplot(food_violations2.df, aes(x = zipcode,
                                y = mean_scores)) +
  geom_line(color="skyblue") +
  geom_errorbar(aes(ymin = mean_scores - se_scores,
                    ymax = mean_scores + se_scores))
```




```
max_value <- food_violations2.df %>% ungroup() %>% summarize(max_score = max(mean_scores))
food_violations_max <- food_violations2.df %>% filter(mean_scores == max_value$max_score)
food_violations_max
## # A tibble: 1 x 3
##   zipcode mean_scores se_scores
##     <int>       <dbl>     <dbl>
## 1   10285        30.7      4.11

min_value <- food_violations2.df %>% ungroup() %>% summarize(min_score = min(mean_scores))
food_violations_min <- food_violations2.df %>% filter(mean_scores == min_value$min_score)
food_violations_min
## # A tibble: 1 x 3
##   zipcode mean_scores se_scores
##     <int>       <dbl>     <dbl>
## 1   10279           2        NA
```

Problem 4: joins

a.

answer: The left table is Class, it will get all the data in Classes and combine them to table to Students. It will combine them when Stud_Id (from Classes) is equal to Id (from Students).

This will add the rows of student to Classes where the student Ids are the same. If there are any values in the left table with no corresponding value in the right table, the code will Append NAs to the columns of the right table (students).

The final result is the following:

Class	Student	Stud_Id	Name	Computer
CS	Jon	4	Jon	m
CS	Arya	1	Arya	m
CS	Cersei	3	Cersei	w
Stats	Gregor	2	Gregor	m
Stats	Jon	4	Jon	m
Stats	Jon	5	Jon	w
Stats	Arya	1	Arya	m

b.

answer: In the code "Classes %>% filter(Class =="CS")" will get all the CS classes.

Then "Stats <- Classes %>% filter(Class =="Stats")" will get all the stats classes.

Then the code will do a join where the Stud_Id is the same on both tables. In other words, it will get all the rows from Stats and match them to all the rows in CS where the Student IDs match.

But instead of creating a new table with the values from CS and Stats, it will just return the values that are on the stats table.

In short If a student ID is both in CS and Stats, it will return the information from Stats of that student ID.

Class	Student	Stud_Id
Stats	Jon	4
Stats	Arya	1

c.

answer: This will do an anti join on the tables of the previous exercise. An anti join is pretty much the opposite of a semi_join.

The anti join will do a join between Stats and CS based on Stud_ID. So it will match all values in stats that have a corresponding value in CS. Then it will return all the values from stats that are not in this join. So it is like Stats - result of join

So this will get all students ID that are in both Stats and CS and return all the information from students who are in stats but not CS.

Class	Student	Stud_Id
Stats	Gregor	2
Stats	Jon	5

Problem 5: restructure

a.

answer: This is an untidy table. The code will get the second, third and fourth columns (which are 2012, 2013, and 2014), and put them as values in a column named Year. Each value will appear twice in Year.

The values associated with 2012, 2013 and 2014 will be put in a column named Clarity. Suppose y was under 2014 (in the original table), then (in the new table) the value in column Year is going to be 2014, and the value in Clarity is going to be y.

LakeId	Year	Clarity
1	2012	6.5
1	2013	5.8
1	2014	5.8
2	2012	2.1
2	2013	3.4
2	2014	2.8

b.

answer: Just like in 1 the code starts with an untidy dataset. The code will start by getting the second, third and fourth columns (which are 2012, 2013, and 2014), and put them as values in a column named Year. Each value will appear twice in Year.

The values associated with 2012, 2013 and 2014 will be put in a column named Clarity. Suppose y was under 2014 (in the original table), then (in the new table) the value in column Year is going to be 2014, and the value in Clarity is going to be y.

After that, the code is going to group the table by LakeId, and sort its values by year. So the final table will alternate LakeIDs (like 1, 2, 1, 2) and years will be grouped together.

Then, the code going to add a new column to the table, the name of the column is Change_in_Clarity and its value is the current clarity value (of a lake) minus the clarity level of the same lake on the previous year.

So, Change_in_clarity for lake 2 in 2013 is the clarity level in 2013 (which is 3.4) minus the clarity level in 2012 (which is 2.1). This will give us 1.3. For 2012, as we have no information from 2011, the Change_in_clarity is going to be NA

LakeId	Year	Clarity	Change_in_clarity
1	2012	6.5	NA
2	2012	2.1	NA
1	2013	5.8	-0.7
2	2013	3.4	1.3
1	2014	5.8	0.0
2	2014	2.8	-0.6
