Lab 3

Alyssa Andrichik

Math 241, Week 4

```r
# Put all necessary libraries here
library(tidyverse)
library(dplyr)
```

Due: Thursday, February 25th at ~~8:30am~~ 6:00pm

Goals of this lab

1. Practice using GitHub.
2. Practice wrangling data.

Data Notes:

- For Problem 2, we will continue to dig into the SE Portland crash data but will use two datasets:
  - `CRASH`: crash level data
  - `PARTIC`: participant level data

```r
# Crash level dataset
crash <- read_csv("/home/courses/math241s21/Data/pdx_crash_2018_CRASH.csv")

# Participant level dataset
partic <- read_csv("/home/courses/math241s21/Data/pdx_crash_2018_PARTIC.csv")
```

- For Problem 3, we will look at chronic illness data from the CDC along with the regional mapping for each state.

```r
# CDC data
CDC <- read_csv("/home/courses/math241s21/Data/CDC2.csv")

# Regional data
USregions <- read_csv("/home/courses/math241s21/Data/USregions.csv")
```

- For Problem 4, we will use polling data from FiveThirtyEight.com.

```r
# Note I only want us to focus on a subset of the variables
polls <- read_csv("/home/courses/math241s21/Data/generic_topline.csv") %>%
  select(subgroup, modeldate, dem_estimate, rep_estimate)
```

- For Problem 6, we will use several datasets that came from `pdxTrees` but good messed up a bit:

```r
# Data on trees in a few parks in Portland
treez <- read_csv("/home/courses/math241s21/Data/treez.csv")
treez_loc <- read_csv("/home/courses/math241s21/Data/treez_loc.csv")
treez_park <- read_csv("/home/courses/math241s21/Data/treez_park.csv")
```

Problems

Problem 1: Git Control

In this problem, we will practice interacting with GitHub on the site directly and from the RStudio Server. Do this practice on **your labwork_username repo**, not your group's Project 1 repo, so that the graders can check your progress with Git.

a. Let's practice creating and closing **Issues**. In a nutshell, **Issues** let us keep track of our work. Within your repo on GitHub.com, create an Issue entitled "Complete Lab 3". Once Lab 3 is done, close the **Issue**. (If you want to learn more about the functionalities of Issues, check out this page.)
b. Edit the ReadMe of your repo to include your name and a quick summary of the purpose of the repo. You can edit from within GitHub directly or on the server. If you edit on the server, make sure to push your changes to GitHub.
c. Upload both your Lab 3 .Rmd and .pdf to your repo on GitHub.

Problem 2: `dplyr` madness

Each part of this problem will require you to wrangle the data and then do one or both of the following:

- Display the wrangled data frame. To ensure it displays the whole data frame, you can pipe `as.data.frame()` at the end of the wrangling.
- Answer a question(s).

Some parts will require you to do a data join but won't tell you that.

a. Produce a data frame that provides the frequency of the different collision types, ordered from most to least common. What type is most common? What type is least common?

```r
collision_df <- crash %>%
  select(COLLIS_TYP_CD)

collision_tidy <- as.data.frame(table(collision_df$COLLIS_TYP_CD), stringsAsFactors=FALSE)

collision_tidy <- collision_tidy %>%
  rename(collision_type = Var1) %>%
  rename(frequency = Freq)

collision_tidy$collision_type[collision_tidy$collision_type=="1"] <- "Angle"
collision_tidy$collision_type[collision_tidy$collision_type=="2"] <- "Head-On"
collision_tidy$collision_type[collision_tidy$collision_type=="3"] <- "Rear-End"
collision_tidy$collision_type[collision_tidy$collision_type=="4"] <- "Sideswipe-meeting"
collision_tidy$collision_type[collision_tidy$collision_type=="5"] <- "Sideswipe-overtaking"
collision_tidy$collision_type[collision_tidy$collision_type=="6"] <- "Turning Movement"
collision_tidy$collision_type[collision_tidy$collision_type=="7"] <- "Parking Maneuver"
collision_tidy$collision_type[collision_tidy$collision_type=="8"] <- "Non-collision"
collision_tidy$collision_type[collision_tidy$collision_type=="9"] <- "Fixed-Object or Other-Object"
collision_tidy$collision_type[collision_tidy$collision_type=="0"] <- "Pedestrian"
collision_tidy$collision_type[collision_tidy$collision_type=="-"] <- "Backing"
collision_tidy$collision_type[collision_tidy$collision_type=="&"] <- "Miscellaneous"

collision_tidy <- collision_tidy %>%
  arrange(desc(frequency))
collision_tidy
```

```
##                  collision_type frequency
## 1                      Rear-End       671
```

```
## 2              Turning Movement       365
## 3                         Angle       241
## 4          Sideswipe-overtaking        89
## 5                    Pedestrian        86
## 6  Fixed-Object or Other-Object        51
## 7             Sideswipe-meeting        17
## 8                       Head-On        16
## 9                       Backing        12
## 10             Parking Maneuver        10
## 11                Non-collision         6
## 12                Miscellaneous         3
```

The most common collision is rear-ends and, technically, miscellaneous is the least common collision. But, of the actual collision types provided, parking maneuvers are the least common type of collisions.

b. For the three most common collision types, create a table that contains:
   - The frequencies of each collision type and weather condition combination.
   - The proportion of each collision type by weather condition.

Arrange the table by weather and within type, most to least common collision type.

```
coll_weath_df <- crash %>%
  select(WTHR_COND_SHORT_DESC, COLLIS_TYP_CD)

coll_weath_df <- coll_weath_df[coll_weath_df$COLLIS_TYP_CD %in% c(3,6,1),]

coll_weath_1 <- coll_weath_df %>%
  group_by(WTHR_COND_SHORT_DESC, COLLIS_TYP_CD) %>%
  summarise(n = n()) %>%
  mutate(proportion = n / sum(n)) %>%
  group_by(WTHR_COND_SHORT_DESC) %>%
  arrange(desc(n), .by_group = TRUE) %>%
  rename(collision_type = COLLIS_TYP_CD) %>%
  rename(frequency = n) %>%
  rename(weather_type = WTHR_COND_SHORT_DESC)

coll_weath_1$collision_type[coll_weath_1$collision_type=="1"] <- "Angle"
coll_weath_1$collision_type[coll_weath_1$collision_type=="3"] <- "Rear-End"
coll_weath_1$collision_type[coll_weath_1$collision_type=="6"] <- "Turning Movement"

coll_weath_1
```

```
## # A tibble: 19 x 4
## # Groups:   weather_type [8]
##    weather_type collision_type   frequency proportion
##    <chr>        <chr>                <int>      <dbl>
##  1 CLD          Rear-End                29      0.468
##  2 CLD          Turning Movement        20      0.323
##  3 CLD          Angle                   13      0.210
##  4 CLR          Rear-End               549      0.535
##  5 CLR          Turning Movement       290      0.282
##  6 CLR          Angle                  188      0.183
##  7 FOG          Angle                    2      0.667
##  8 FOG          Turning Movement         1      0.333
##  9 RAIN         Rear-End                71      0.497
## 10 RAIN         Turning Movement        44      0.308
```

```
## 11 RAIN          Angle                     28       0.196
## 12 SLT           Angle                      1       1
## 13 SMOK          Rear-End                   1       1
## 14 SNOW          Angle                      3       0.5
## 15 SNOW          Turning Movement           2       0.333
## 16 SNOW          Rear-End                   1       0.167
## 17 UNK           Rear-End                  20       0.588
## 18 UNK           Turning Movement           8       0.235
## 19 UNK           Angle                      6       0.176
```

c. Create a column for whether or not a crash happened on a weekday or on the weekend and then create a data frame that explores if the distribution of collision types varies by whether or not the crash happened during the week or the weekend.

```
collision_c <- crash %>%
  select(COLLIS_TYP_CD, CRASH_WK_DAY_CD)
weekday_df <- data.frame(day_of_week = c(1, 2, 3, 4, 5, 6, 7),
                         day = c("weekend", "weekday", "weekday", "weekday", "weekday",
                                 "weekday", "weekend"))
collision_c <- left_join(collision_c, weekday_df,
                         by = c("CRASH_WK_DAY_CD" = "day_of_week"))

collision_c <- collision_c %>%
  select(COLLIS_TYP_CD, day) %>%
  group_by(COLLIS_TYP_CD, day) %>%
  summarise(n = n()) %>%
  group_by(COLLIS_TYP_CD, day) %>%
  arrange(desc(n), .by_group = TRUE) %>%
  rename(collision_type = COLLIS_TYP_CD) %>%
  rename(frequency = n)

collision_c$collision_type[collision_c$collision_type=="1"] <- "Angle"
collision_c$collision_type[collision_c$collision_type=="2"] <- "Head-On"
collision_c$collision_type[collision_c$collision_type=="3"] <- "Rear-End"
collision_c$collision_type[collision_c$collision_type=="4"] <- "Sideswipe-meeting"
collision_c$collision_type[collision_c$collision_type=="5"] <- "Sideswipe-overtaking"
collision_c$collision_type[collision_c$collision_type=="6"] <- "Turning Movement"
collision_c$collision_type[collision_c$collision_type=="7"] <- "Parking Maneuver"
collision_c$collision_type[collision_c$collision_type=="8"] <- "Non-collision"
collision_c$collision_type[collision_c$collision_type=="9"] <- "Fixed-Object or Other-Object"
collision_c$collision_type[collision_c$collision_type=="0"] <- "Pedestrian"
collision_c$collision_type[collision_c$collision_type=="-"] <- "Backing"
collision_c$collision_type[collision_c$collision_type=="&"] <- "Miscellaneous"

collision_c
```

```
## # A tibble: 23 x 3
## # Groups:   collision_type, day [23]
##    collision_type day     frequency
##    <chr>          <chr>       <int>
##  1 Backing        weekday        10
##  2 Backing        weekend         2
##  3 Miscellaneous  weekday         3
##  4 Pedestrian     weekday        67
##  5 Pedestrian     weekend        19
```

```
## 6 Angle          weekday       180
## 7 Angle          weekend        61
## 8 Head-On        weekday        11
## 9 Head-On        weekend         5
## 10 Rear-End      weekday       510
## # ... with 13 more rows
```

d. First determine what proportion of crashes involve pedestrians. Then, for each driver license status, determine what proportion of crashes involve pedestrians. What driver license status has the highest rate of crashes that involve pedestrians?

```r
crash_d <- crash %>%
  select(CRASH_ID, COLLIS_TYP_SHORT_DESC)
ped_prop_d <- crash_d %>%
  group_by(COLLIS_TYP_SHORT_DESC) %>%
  summarise(n = n()) %>%
  mutate(proportion = n / sum(n)) %>%
  filter(COLLIS_TYP_SHORT_DESC == "PED")
ped_prop_d
```

```
## # A tibble: 1 x 3
##   COLLIS_TYP_SHORT_DESC     n proportion
##   <chr>                 <int>      <dbl>
## 1 PED                      86     0.0549
```

What proportion of crashes involve pedestrians? 0.054881940 or 5.4881940% of all crashes involved pedestrians.

```r
partic_d <- partic %>%
  select(CRASH_ID, DRVR_LIC_STAT_SHORT_DESC)

ped_d <- left_join(crash_d, partic_d)

ped_d <- ped_d %>%
  group_by(DRVR_LIC_STAT_SHORT_DESC, COLLIS_TYP_SHORT_DESC) %>%
  summarise(n = n()) %>%
  mutate(proportion = n / sum(n)) %>%
  filter(COLLIS_TYP_SHORT_DESC == "PED")
ped_d
```

```
## # A tibble: 5 x 4
## # Groups:   DRVR_LIC_STAT_SHORT_DESC [5]
##   DRVR_LIC_STAT_SHORT_DESC COLLIS_TYP_SHORT_DESC     n proportion
##   <chr>                    <chr>                 <int>      <dbl>
## 1 OR-Y                     PED                      72    0.0290
## 2 OTH-Y                    PED                       6    0.0193
## 3 SUSP                     PED                       7    0.121
## 4 UNK                      PED                       2    0.00769
## 5 <NA>                     PED                       7    0.00873
```

For each driver license status, determine what proportion of crashes involve pedestrians?

OR-Y: 0.029008864 or 2.9%; OTH-Y: 0.019292605 or 1.9%; SUSP: 0.120689655 or 12.1%; UNK: 0.007692308 or 0.8%; NA: 0.008728180 or 0.9%

What driver license status has the highest rate of crashes that involve pedestrians? Suspended, or revoked drivers license, has the highest rate of pedestrian involved crashes.

e. Create a data frame that contains the age of drivers and collision type. (Don't print it.) Complete the following:

- Find the average and median age of drivers.
- Find the average and median age of drivers by collision type.
- Create a graph of driver ages.
- Create a graph of driver ages by collision type.

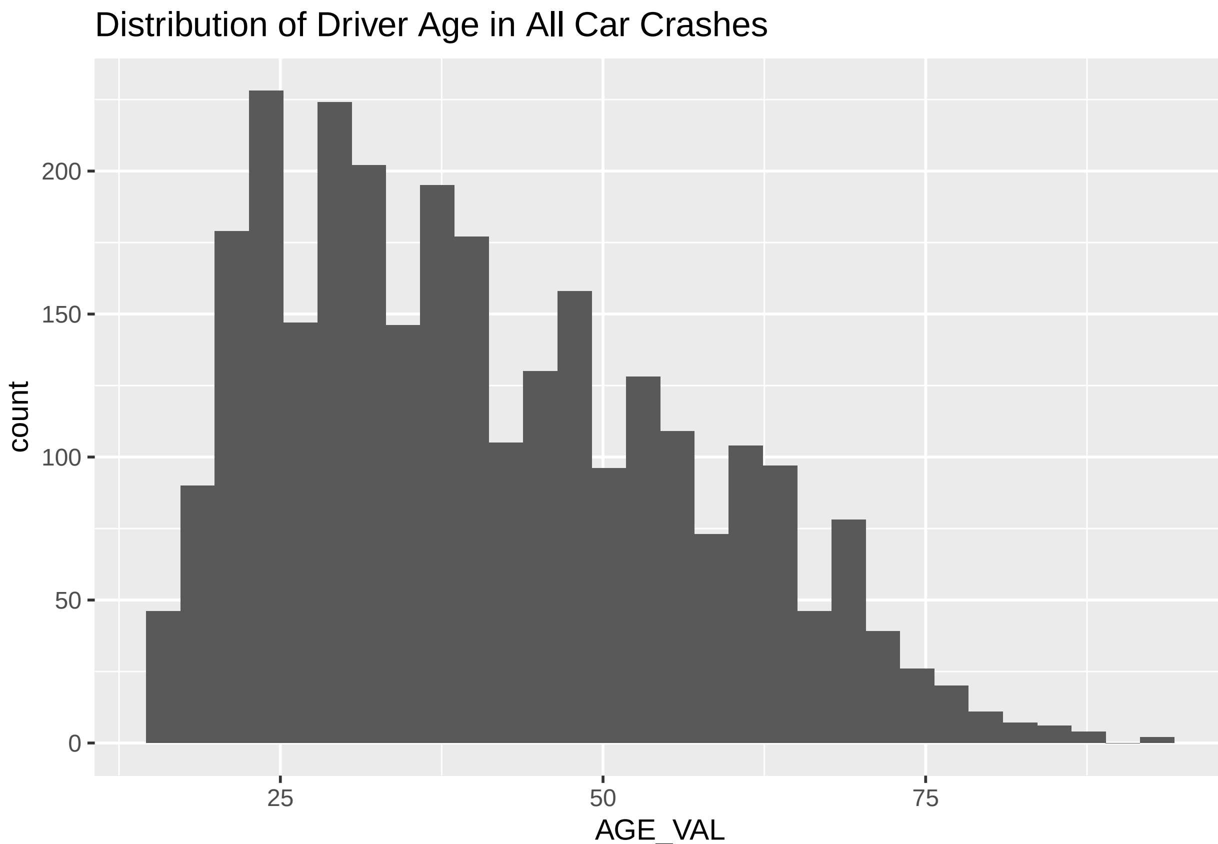
```
ages <- partic%>%
  select(CRASH_ID, PARTIC_TYP_SHORT_DESC, AGE_VAL) %>%
  filter(PARTIC_TYP_SHORT_DESC == "DRVR")

ages$AGE_VAL[ages$AGE_VAL == "00"] <- NA

crash_e <- left_join(crash_d, ages)

crash_e <- crash_e %>%
  select(COLLIS_TYP_SHORT_DESC, AGE_VAL)%>%
  transform(AGE_VAL = as.numeric(AGE_VAL),
            COLLIS_TYP_SHORT_DESC = as.character(COLLIS_TYP_SHORT_DESC))%>%
  na.omit(crash_e)

crash_onlyage <- crash_e %>%
  summarise(mean = mean(AGE_VAL), median = median(AGE_VAL), n = n())
#the average (mean) and median age of drivers
crash_onlyage
```

```
##       mean median    n
## 1 40.90184     38 2873
```

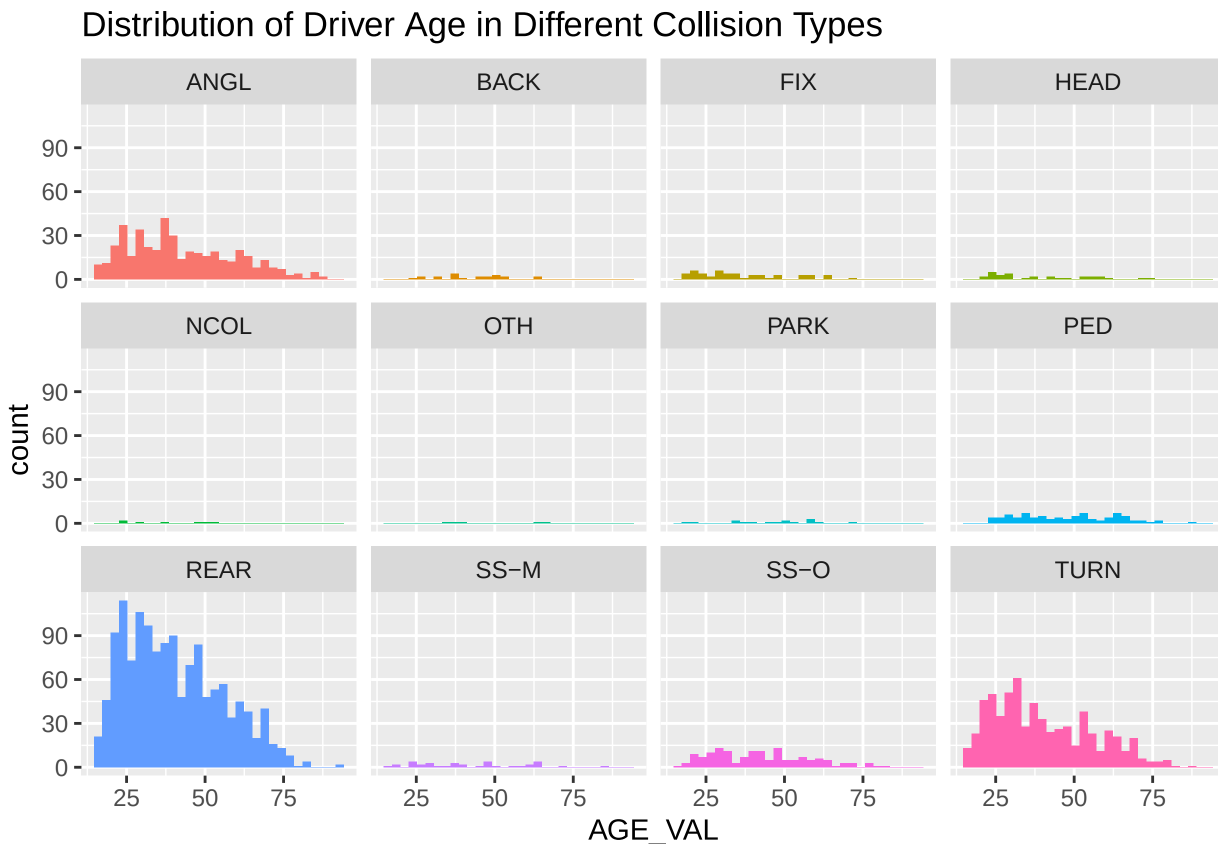
```
crash_avg <- crash_e %>%
  group_by(COLLIS_TYP_SHORT_DESC) %>%
  summarise(mean = mean(AGE_VAL), median = median(AGE_VAL))
#the average (mean) and median age of drivers by collision type
crash_avg
```

```
## # A tibble: 12 x 3
##    COLLIS_TYP_SHORT_DESC  mean median
##  * <chr>                 <dbl>  <dbl>
##  1 ANGL                   42.5     39
##  2 BACK                   42.8     44
##  3 FIX                    36.7     33
##  4 HEAD                   39.4     36
##  5 NCOL                   37.4     36
##  6 OTH                    48.6     40
##  7 PARK                   46.4     50
##  8 PED                    48.0     47
##  9 REAR                   40.2     38
## 10 SS-M                   42.5     40
## 11 SS-O                   42.4     41
## 12 TURN                   40.0     37
```

```
ggplot(crash_e, aes(x = AGE_VAL)) +
  geom_histogram() +
  ggtitle("Distribution of Driver Age in All Car Crashes")
```




```r
ggplot(crash_e, aes(x = AGE_VAL, fill = COLLIS_TYP_SHORT_DESC)) +
  geom_histogram() +
  facet_wrap(~ COLLIS_TYP_SHORT_DESC) +
  theme(legend.position = "none") +
  ggtitle("Distribution of Driver Age in Different Collision Types")
```



Draw some conclusions.

The average (40.90184) and median (38) ages over all is very close to the average (40.2) and median (38) of rear-end collisions. Rear-end collisions are the most common collisions, which helps explain why the total average and mean is so similar to the rear-end-specific data. Angle and turning movement collisions are the next most common collisions; their median and means are also very close to the total average and median. The distribution of ages for the other, far less common collisions break away from the total average and median and have a median younger or older. The fact that 40ish year olds contribute the most common collisions is because they are the most likely age group to be driving anyway: old enough to have enough money to by a car and experienced enough, and young enough to not have any heath issues that would result in needing to drive less over all.

Problem 3: Chronically Messy Data

a. Turning to the CDC data, let's get a handle of what is represented there. For 2016 (use `YearStart`), how many distinct topics were tracked?

```
CDC_a <- CDC %>%
  filter(YearStart == "2016")
length(table(CDC_a$Topic))
```

```
## [1] 16
```

16 topics were tracked.

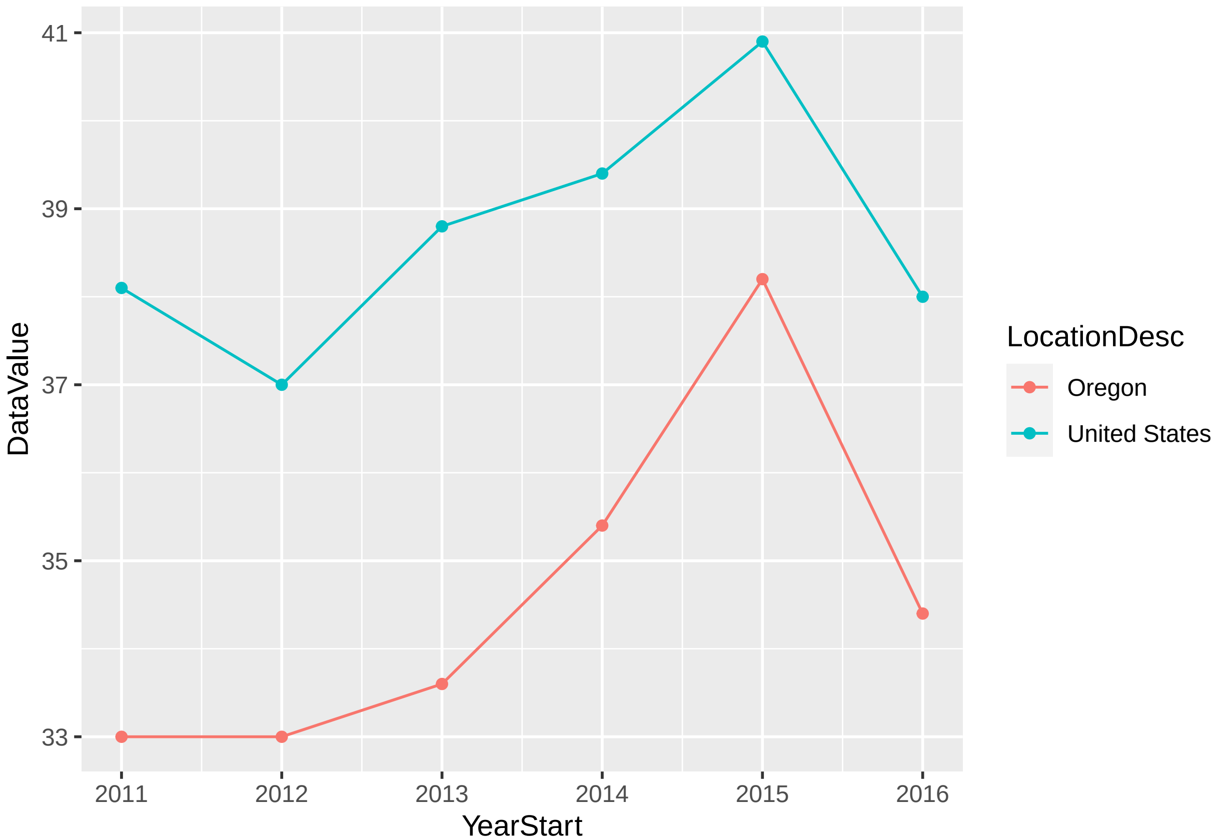
b. Let's study influenza vaccination patterns! Create a dataset that contains the age adjusted prevalence of the "Influenza vaccination among noninstitutionalized adults aged >= 18 years" for Oregon and the US from 2010 to 2016.

```
CDC_b <- CDC %>%
  filter(DataValueType == "Age-adjusted Prevalence",
         Question == "Influenza vaccination among noninstitutionalized adults aged >= 18 years")

CDC_b <- CDC_b[CDC_b$LocationDesc %in% c("Oregon", "United States"),]
CDC_b
```

```
## # A tibble: 12 x 34
##    YearStart YearEnd LocationAbbr LocationDesc  DataSource Topic Question
##        <dbl>   <dbl> <chr>        <chr>         <chr>      <chr> <chr>
##  1      2016    2016 US           United States BRFSS      Immu~ Influenza vacc~
##  2      2016    2016 OR           Oregon        BRFSS      Immu~ Influenza vacc~
##  3      2015    2015 OR           Oregon        BRFSS      Immu~ Influenza vacc~
##  4      2015    2015 US           United States BRFSS      Immu~ Influenza vacc~
##  5      2012    2012 OR           Oregon        BRFSS      Immu~ Influenza vacc~
##  6      2012    2012 US           United States BRFSS      Immu~ Influenza vacc~
##  7      2011    2011 OR           Oregon        BRFSS      Immu~ Influenza vacc~
##  8      2011    2011 US           United States BRFSS      Immu~ Influenza vacc~
##  9      2014    2014 OR           Oregon        BRFSS      Immu~ Influenza vacc~
## 10      2014    2014 US           United States BRFSS      Immu~ Influenza vacc~
## 11      2013    2013 OR           Oregon        BRFSS      Immu~ Influenza vacc~
## 12      2013    2013 US           United States BRFSS      Immu~ Influenza vacc~
## # ... with 27 more variables: Response <lgl>, DataValueUnit <chr>,
## #   DataValueType <chr>, DataValue <dbl>, DataValueAlt <dbl>,
## #   DataValueFootnoteSymbol <chr>, DatavalueFootnote <chr>,
## #   LowConfidenceLimit <dbl>, HighConfidenceLimit <dbl>,
## #   StratificationCategory1 <chr>, Stratification1 <chr>,
## #   StratificationCategory2 <lgl>, Stratification2 <lgl>,
## #   StratificationCategory3 <lgl>, Stratification3 <lgl>, GeoLocation <chr>,
## #   ResponseID <lgl>, LocationID <chr>, TopicID <chr>, QuestionID <chr>,
## #   DataValueTypeID <chr>, StratificationCategoryID1 <chr>,
## #   StratificationID1 <chr>, StratificationCategoryID2 <lgl>,
## #   StratificationID2 <lgl>, StratificationCategoryID3 <lgl>,
## #   StratificationID3 <lgl>
```

c. Create a graph comparing the immunization rates of Oregon and the US. Comment on the observed trends in your graph

```
ggplot(CDC_b, mapping = aes(y = DataValue, x = YearStart, color = LocationDesc)) +
  geom_point() +
  geom_line()
```



Oregon is below the US average on immunization rates, but Oregon does generally follow the US immunization trend. For example, from 2015 to 2016 that is a distinct drop in immunization rates for both the US and Oregon. The years prior generally showed a slight increase (a couple percentage points) in immunization rates for both Oregon and the United States.

d. Let's see how immunization rates vary by region of the country. Join the regional dataset to our CDC dataset so that we have a column signifying the region of the country.

```
CDC_d <- left_join(CDC, USregions, by = c("LocationDesc" = "State"))
CDC_d
```

```
## # A tibble: 74,811 x 35
##    YearStart YearEnd LocationAbbr LocationDesc   DataSource Topic Question
##        <dbl>   <dbl> <chr>        <chr>          <chr>      <chr> <chr>
##  1      2016    2016 US           United States  BRFSS      Alco~ Binge drinkin~
##  2      2016    2016 AL           Alabama        BRFSS      Alco~ Binge drinkin~
##  3      2016    2016 AK           Alaska         BRFSS      Alco~ Binge drinkin~
##  4      2016    2016 AZ           Arizona        BRFSS      Alco~ Binge drinkin~
##  5      2016    2016 AR           Arkansas       BRFSS      Alco~ Binge drinkin~
##  6      2016    2016 CA           California     BRFSS      Alco~ Binge drinkin~
##  7      2016    2016 CO           Colorado       BRFSS      Alco~ Binge drinkin~
##  8      2016    2016 CT           Connecticut    BRFSS      Alco~ Binge drinkin~
##  9      2016    2016 DE           Delaware       BRFSS      Alco~ Binge drinkin~
## 10      2016    2016 DC           District of C~ BRFSS      Alco~ Binge drinkin~
## # ... with 74,801 more rows, and 28 more variables: Response <lgl>,
## #   DataValueUnit <chr>, DataValueType <chr>, DataValue <dbl>,
## #   DataValueAlt <dbl>, DataValueFootnoteSymbol <chr>, DatavalueFootnote <chr>,
## #   LowConfidenceLimit <dbl>, HighConfidenceLimit <dbl>,
## #   StratificationCategory1 <chr>, Stratification1 <chr>,
```

```
## #   StratificationCategory2 <lgl>, Stratification2 <lgl>,
## #   StratificationCategory3 <lgl>, Stratification3 <lgl>, GeoLocation <chr>,
## #   ResponseID <lgl>, LocationID <chr>, TopicID <chr>, QuestionID <chr>,
## #   DataValueTypeID <chr>, StratificationCategoryID1 <chr>,
## #   StratificationID1 <chr>, StratificationCategoryID2 <lgl>,
## #   StratificationID2 <lgl>, StratificationCategoryID3 <lgl>,
## #   StratificationID3 <lgl>, Region <chr>
```

e. Why are there NAs in the region column of the new dataset?

The region column describes the region a state resides in within the US. The NA is there when the LocationDesc is in the US, or a territory of the US, but is not a state and cannot be categorized as a part of a region. The region is NA when it is the average US immunization rate since the US encompasses all regions.

f. Create a dataset that contains the age adjusted influenza immunization rates in 2016 for each state in the country and sort it by highest immunization to lowest. Which state has the highest immunization?

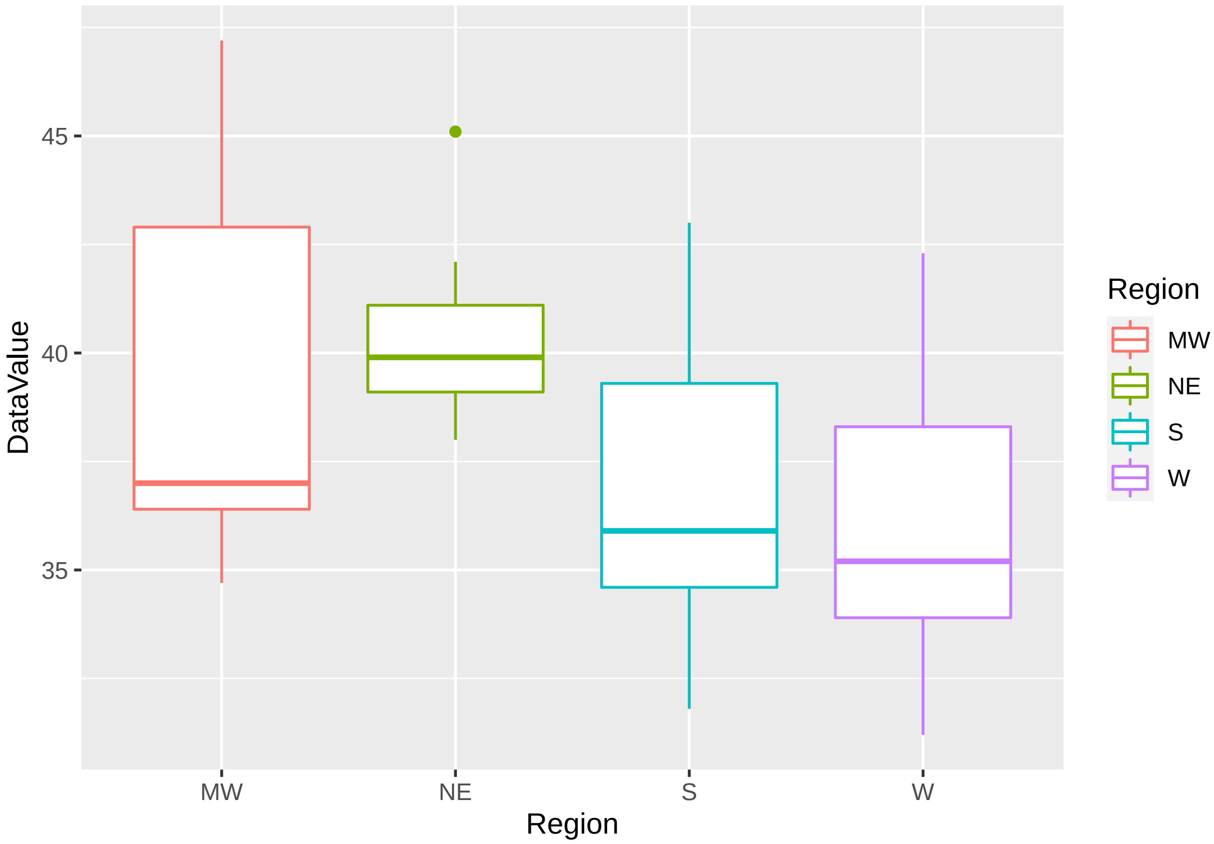
```
CDC_f <- CDC %>%
  filter(YearStart == "2016",
         DataValueType == "Age-adjusted Prevalence",
         Question == "Influenza vaccination among noninstitutionalized adults aged >= 18 years") %>%
  arrange(desc(DataValue))
CDC_f <- left_join(CDC_f, USregions, by = c("LocationDesc" = "State"))
CDC_f
```

```
## # A tibble: 55 x 35
##    YearStart YearEnd LocationAbbr LocationDesc  DataSource Topic Question
##        <dbl>   <dbl> <chr>        <chr>         <chr>      <chr> <chr>
##  1      2016    2016 SD           South Dakota  BRFSS      Immu~ Influenza vacc~
##  2      2016    2016 RI           Rhode Island  BRFSS      Immu~ Influenza vacc~
##  3      2016    2016 IA           Iowa          BRFSS      Immu~ Influenza vacc~
##  4      2016    2016 NE           Nebraska      BRFSS      Immu~ Influenza vacc~
##  5      2016    2016 NC           North Caroli~ BRFSS      Immu~ Influenza vacc~
##  6      2016    2016 MN           Minnesota     BRFSS      Immu~ Influenza vacc~
##  7      2016    2016 CO           Colorado      BRFSS      Immu~ Influenza vacc~
##  8      2016    2016 MD           Maryland      BRFSS      Immu~ Influenza vacc~
##  9      2016    2016 VA           Virginia      BRFSS      Immu~ Influenza vacc~
## 10      2016    2016 CT           Connecticut   BRFSS      Immu~ Influenza vacc~
## # ... with 45 more rows, and 28 more variables: Response <lgl>,
## #   DataValueUnit <chr>, DataValueType <chr>, DataValue <dbl>,
## #   DataValueAlt <dbl>, DataValueFootnoteSymbol <chr>, DatavalueFootnote <chr>,
## #   LowConfidenceLimit <dbl>, HighConfidenceLimit <dbl>,
## #   StratificationCategory1 <chr>, Stratification1 <chr>,
## #   StratificationCategory2 <lgl>, Stratification2 <lgl>,
## #   StratificationCategory3 <lgl>, Stratification3 <lgl>, GeoLocation <chr>,
## #   ResponseID <lgl>, LocationID <chr>, TopicID <chr>, QuestionID <chr>,
## #   DataValueTypeID <chr>, StratificationCategoryID1 <chr>,
## #   StratificationID1 <chr>, StratificationCategoryID2 <lgl>,
## #   StratificationID2 <lgl>, StratificationCategoryID3 <lgl>,
## #   StratificationID3 <lgl>, Region <chr>
```

South Dakota has the highest immunization.

g. Construct a graphic of the 2016 influenza immunization rates by region of the country. Don't include locations without a region. Comment on your graphic.

```r
ggplot(data = subset(CDC_f, !is.na(Region)), aes(y = DataValue, x = Region, color = Region)) +
 # geom_violin() +
 # geom_point() +
  geom_boxplot()
```



The region with the highest mean immunization rate is the North East, then the Mid West, then the South, and the West is last. I chose to visualize this data with a boxplot cause it gives the mean as well as the range of the points in each region.

Problem 4: Tidying Data Like a Boss

I was amazed by the fact that many of the FiveThirtyEight datasets are actually not in a perfectly *tidy* format. Let's tidy up this dataset related to polling.

a. Why is this data not currently in a tidy format? (Consider the three rules of tidy data!)

```r
polls
```

```
## # A tibble: 1,529 x 4
##    subgroup  modeldate dem_estimate rep_estimate
##    <chr>     <chr>            <dbl>        <dbl>
##  1 All polls 9/18/2018         48.8         39.8
##  2 All polls 9/17/2018         49.0         39.9
##  3 All polls 9/16/2018         49.0         39.9
##  4 All polls 9/15/2018         49.0         39.9
##  5 All polls 9/14/2018         48.9         39.8
##  6 All polls 9/13/2018         48.8         39.7
##  7 All polls 9/12/2018         48.8         39.6
```

```
##   8 All polls 9/11/2018         48.5         39.9
##   9 All polls 9/10/2018         48.4         39.9
## 10 All polls 9/9/2018          48.4         39.9
## # ... with 1,519 more rows
```

The data is not tidy because the columns dem_estimate and rep_estimate are currently storing a variable (party). For complete tidy observations, there should be one column addressing party affiliation and another addressing the estimate.

b. Create a tidy dataset of the `All polls` subgroup.

```
all_polls <- polls %>%
  filter(subgroup == "All polls") %>%
  select(modeldate, dem_estimate, rep_estimate)
all_polls <- pivot_longer(all_polls,
                          cols = c(dem_estimate, rep_estimate),
                          names_to = "affiliation",
                          values_to = "estimates")
all_polls
```

```
## # A tibble: 1,044 x 3
##    modeldate affiliation  estimates
##    <chr>     <chr>            <dbl>
##  1 9/18/2018 dem_estimate      48.8
##  2 9/18/2018 rep_estimate      39.8
##  3 9/17/2018 dem_estimate      49.0
##  4 9/17/2018 rep_estimate      39.9
##  5 9/16/2018 dem_estimate      49.0
##  6 9/16/2018 rep_estimate      39.9
##  7 9/15/2018 dem_estimate      49.0
##  8 9/15/2018 rep_estimate      39.9
##  9 9/14/2018 dem_estimate      48.9
## 10 9/14/2018 rep_estimate      39.8
## # ... with 1,034 more rows
```

c. Now let's create a new untidy version of `polls`. Focusing just on the estimates for democrats, create a data frame where each row represents a subgroup (given in column 1) and the rest of the columns are the estimates for democrats by date.

```
dem_polls <- polls %>%
  select(subgroup, modeldate, dem_estimate)
dem_polls$modeldate <- as.Date(dem_polls$modeldate, "%m/%d/%Y")
untidy_polls <- pivot_wider(dem_polls,
                            names_from = modeldate,
                            values_from = dem_estimate)
untidy_polls
```

```
## # A tibble: 3 x 523
##   subgroup  `2018-09-18` `2018-09-17` `2018-09-16` `2018-09-15` `2018-09-14`
##   <chr>            <dbl>        <dbl>        <dbl>        <dbl>        <dbl>
## 1 All polls         48.8         49.0         49.0         49.0         48.9
## 2 Voters            NA           NA           NA           NA           NA
## 3 Adults            NA           NA           NA           NA           NA
## # ... with 517 more variables: 2018-09-13 <dbl>, 2018-09-12 <dbl>,
## #   2018-09-11 <dbl>, 2018-09-10 <dbl>, 2018-09-09 <dbl>, 2018-09-08 <dbl>,
## #   2018-09-07 <dbl>, 2018-09-06 <dbl>, 2018-09-05 <dbl>, 2018-09-04 <dbl>,
## #   2018-09-03 <dbl>, 2018-09-02 <dbl>, 2018-09-01 <dbl>, 2018-08-31 <dbl>,
```

```
## #   2018-08-30 <dbl>, 2018-08-29 <dbl>, 2018-08-28 <dbl>, 2018-08-27 <dbl>,
## #   2018-08-26 <dbl>, 2018-08-25 <dbl>, 2018-08-24 <dbl>, 2018-08-23 <dbl>,
## #   2018-08-22 <dbl>, 2018-08-21 <dbl>, 2018-08-20 <dbl>, 2018-08-19 <dbl>,
## #   2018-08-18 <dbl>, 2018-08-17 <dbl>, 2018-08-16 <dbl>, 2018-08-15 <dbl>,
## #   2018-08-14 <dbl>, 2018-08-13 <dbl>, 2018-08-12 <dbl>, 2018-08-11 <dbl>,
## #   2018-08-10 <dbl>, 2018-08-09 <dbl>, 2018-08-08 <dbl>, 2018-08-07 <dbl>,
## #   2018-08-06 <dbl>, 2018-08-05 <dbl>, 2018-08-04 <dbl>, 2018-08-03 <dbl>,
## #   2018-08-02 <dbl>, 2018-08-01 <dbl>, 2018-07-31 <dbl>, 2018-07-30 <dbl>,
## #   2018-07-29 <dbl>, 2018-07-28 <dbl>, 2018-07-27 <dbl>, 2018-07-26 <dbl>,
## #   2018-07-25 <dbl>, 2018-07-24 <dbl>, 2018-07-23 <dbl>, 2018-07-22 <dbl>,
## #   2018-07-21 <dbl>, 2018-07-20 <dbl>, 2018-07-19 <dbl>, 2018-07-18 <dbl>,
## #   2018-07-17 <dbl>, 2018-07-16 <dbl>, 2018-07-15 <dbl>, 2018-07-14 <dbl>,
## #   2018-07-13 <dbl>, 2018-07-12 <dbl>, 2018-07-11 <dbl>, 2018-07-10 <dbl>,
## #   2018-07-09 <dbl>, 2018-07-08 <dbl>, 2018-07-07 <dbl>, 2018-07-06 <dbl>,
## #   2018-07-05 <dbl>, 2018-07-04 <dbl>, 2018-07-03 <dbl>, 2018-07-02 <dbl>,
## #   2018-07-01 <dbl>, 2018-06-30 <dbl>, 2018-06-29 <dbl>, 2018-06-28 <dbl>,
## #   2018-06-27 <dbl>, 2018-06-26 <dbl>, 2018-06-25 <dbl>, 2018-06-24 <dbl>,
## #   2018-06-23 <dbl>, 2018-06-22 <dbl>, 2018-06-21 <dbl>, 2018-06-20 <dbl>,
## #   2018-06-19 <dbl>, 2018-06-18 <dbl>, 2018-06-17 <dbl>, 2018-06-16 <dbl>,
## #   2018-06-15 <dbl>, 2018-06-14 <dbl>, 2018-06-13 <dbl>, 2018-06-12 <dbl>,
## #   2018-06-11 <dbl>, 2018-06-10 <dbl>, 2018-06-09 <dbl>, 2018-06-08 <dbl>,
## #   2018-06-07 <dbl>, 2018-06-06 <dbl>, ...
```

d. Why might someone want to transform the data like we did in part c?

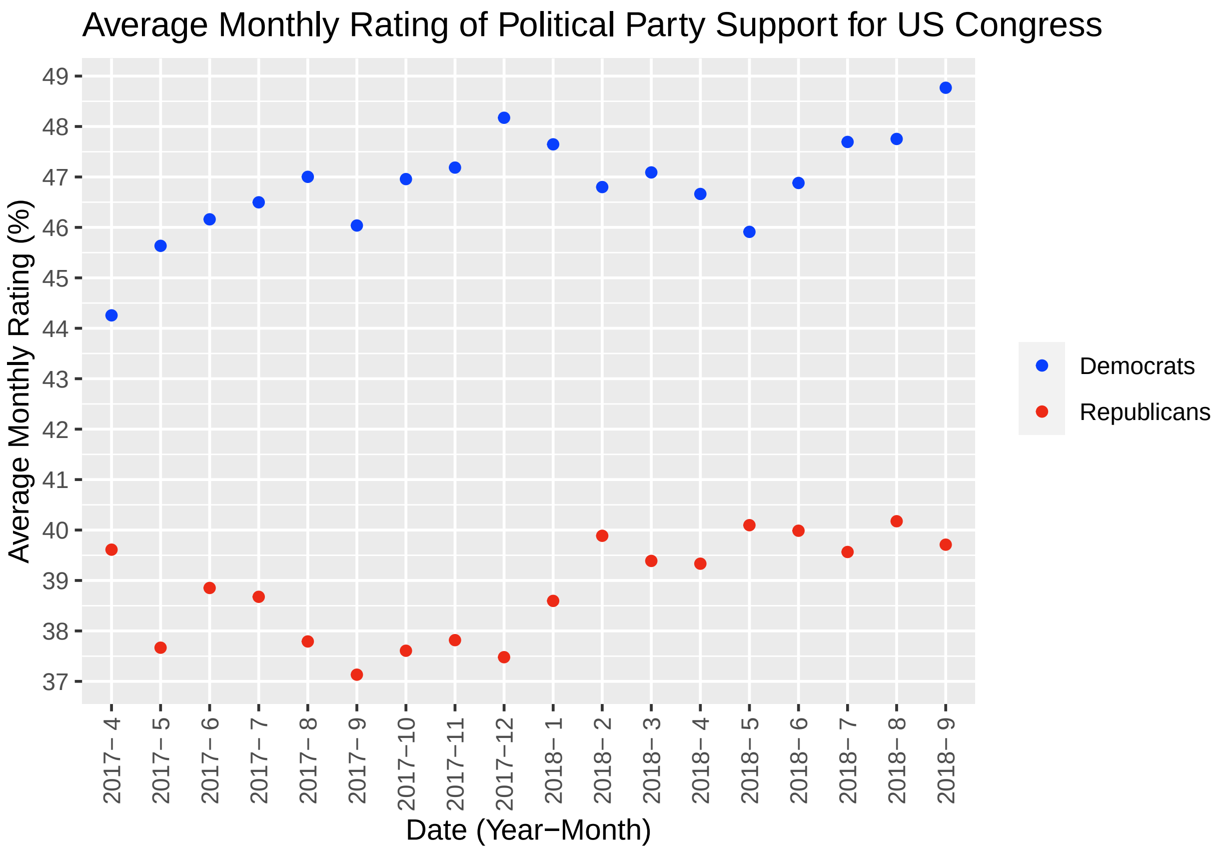
This form of the data makes missing values more clear. It also made comparing the different poll estimates for each day easier to see when looking at the dataset.

Problem 5: YOUR TURN!

Now it is your turn. Pick one (or multiple) of the datasets used on this lab. Ask a question of the data. Do some data wrangling to produce statistics (use at least two wrangling verbs) and a graphic to answer the question. Then comment on any conclusions you can draw about your question.

```r
# How does the polling average (mean) each month change for both democrats and republicans?
newpolls <- polls %>%
  filter(subgroup == "All polls")
newpolls$modeldate <- as.Date(newpolls$modeldate, "%m/%d/%Y")

eighteen_poll <- newpolls %>% #only 2018
  filter(modeldate >= as.Date("2018-01-01"), modeldate <= as.Date("2018-12-31"))
eighteen_poll$Month <- months(eighteen_poll$modeldate)
eighteen_poll$Year <- format(eighteen_poll$modeldate, format = "%Y")
dem_eighteen <- aggregate(dem_estimate ~ Month + Year, eighteen_poll, mean)
rep_eighteen <- aggregate(rep_estimate ~ Month + Year, eighteen_poll, mean)
eighteen_avg <- left_join(dem_eighteen, rep_eighteen)
eighteen_avg$Month[eighteen_avg$Month == "January"] <- "1"
eighteen_avg$Month[eighteen_avg$Month == "February"] <- "2"
eighteen_avg$Month[eighteen_avg$Month == "March"] <- "3"
eighteen_avg$Month[eighteen_avg$Month == "April"] <- "4"
eighteen_avg$Month[eighteen_avg$Month == "May"] <- "5"
eighteen_avg$Month[eighteen_avg$Month == "June"] <- "6"
eighteen_avg$Month[eighteen_avg$Month == "July"] <- "7"
eighteen_avg$Month[eighteen_avg$Month == "August"] <- "8"
eighteen_avg$Month[eighteen_avg$Month == "September"] <- "9"
eighteen_avg <- within(eighteen_avg, Date <- sprintf("%s-%02s", Year, Month))
```

```r
seventeen_poll <- newpolls %>% #only 2017
  filter(modeldate >= as.Date("2017-01-01"), modeldate <= as.Date("2017-12-31"))
seventeen_poll$Month <- months(seventeen_poll$modeldate)
seventeen_poll$Year <- format(seventeen_poll$modeldate, format = "%Y")
dem_seventeen <- aggregate(dem_estimate ~ Month + Year, seventeen_poll, mean)
rep_seventeen <- aggregate(rep_estimate ~ Month + Year, seventeen_poll, mean)
seventeen_avg <- left_join(dem_seventeen, rep_seventeen)
seventeen_avg$Month[seventeen_avg$Month == "April"] <- "4"
seventeen_avg$Month[seventeen_avg$Month == "May"] <- "5"
seventeen_avg$Month[seventeen_avg$Month == "June"] <- "6"
seventeen_avg$Month[seventeen_avg$Month == "July"] <- "7"
seventeen_avg$Month[seventeen_avg$Month == "August"] <- "8"
seventeen_avg$Month[seventeen_avg$Month == "September"] <- "9"
seventeen_avg$Month[seventeen_avg$Month == "October"] <- "10"
seventeen_avg$Month[seventeen_avg$Month == "November"] <- "11"
seventeen_avg$Month[seventeen_avg$Month == "December"] <- "12"
seventeen_avg <- within(seventeen_avg, Date <- sprintf("%s-%02s", Year, Month))


month_avg <- full_join(seventeen_avg, eighteen_avg)
month_avg <- month_avg %>%
  select(Date, dem_estimate, rep_estimate) %>%
  arrange(Date)%>%
  rename(Democrats = dem_estimate) %>%
  rename(Republicans = rep_estimate)
month_avg <- pivot_longer(month_avg, cols = c(Democrats, Republicans),
                          names_to = "PartyAvg",
                          values_to = "Rating")

ggplot(month_avg, aes(x = Date, y = Rating, color = PartyAvg)) +
  geom_point()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  scale_y_continuous(breaks = c(35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
                                45, 46, 47, 48, 49, 50)) +
  ylab("Average Monthly Rating (%)") + xlab("Date (Year-Month)") +
  ggtitle("Average Monthly Rating of Political Party Support for US Congress")+
  scale_color_manual(values = c("#093FFD", "#ED2A16")) +
  theme(legend.title = element_blank())
```



I wanted to see what the polls would average monthly so it was easier to see change by month in ratings for each political party. It is interesting to see that while the change in ratings seem don't seem super correlated (as one party's ratings gets higher, the other's gets lower) by this monthly basis. There a bit of correlation, not drastic enough to really say direct causality that the parties' support are greatly connected when you average to monthly averages. Maybe median would show something more interesting?

Problem 6: Channeling your Inner Marie Kondo

In this problem, I am going to ask you to wrangle/clean up some data and then compare your "cleaned data" with a peer to see how your final versions differ.

a. Join `treez`, `treez_park`, and `treez_loc` to create one data frame where:

- Each row represents one tree (and there are no duplicates) from the following parks: Mt Tabor Park, Laurelhurst Park, Columbia Park
- All missing values (including suspicious values) are appropriately coded as `NA`.
- Each variable has a suitable `class`.
- Categories of categorical variables are appropriated encoded.
- And, any other cleaning is done.

It might take a little sleuthing to figure out which variables are your keys and what makes these datasets messy.

```
parks <- treez_park
parks <- left_join(treez, parks)
parks <- parks[parks$Park %in% c("Mt Tabor Park", "Laurelhurst Park", "Columbia Park"), ]

location <- treez_loc %>%
  select(IDUser, Latitude, Longitude) %>%
```

```r
  rename(UserID = IDUser)

parks <- left_join(parks, location)

parks <- parks %>%
  mutate(Crown_Width_NS = na_if(Crown_Width_NS, 99999)) %>%
  mutate(Crown_Width_EW = na_if(Crown_Width_EW, 99999)) %>%
  mutate(Crown_Base_Height = na_if(Crown_Base_Height, "missing"))

parks$Crown_Base_Height <- as.numeric(parks$Crown_Base_Height)
parks$DBH <- as.numeric(parks$DBH)

parks$Collected_By[parks$Collected_By == "STAFF"] <- "Staff"
parks$Collected_By[parks$Collected_By == "volunteer"] <- "Volunteer"

parks <- unique(parks)
parks
```

```
## # A tibble: 3,088 x 12
##    UserID   DBH Common_Name          Tree_Height Crown_Width_NS Crown_Width_EW
##     <dbl> <dbl> <chr>                      <dbl>          <dbl>          <dbl>
##  1   6855  14   Magnolia                      27             27             27
##  2   6856  23.2 Deodar Cedar                  66             45             37
##  3   6857  25.8 European White Birch          76             47             51
##  4   6858  21.4 Norway Maple                  45             45             47
##  5   6859  22.9 Norway Maple                  53             53             48
##  6   6860  11.5 Lacebark Pine                 31             19             17
##  7   6861  30.6 London Plane Tree             72             54             69
##  8   6862  24.8 London Plane Tree             69             53             66
##  9   6863  25.1 Deodar Cedar                  69             47             45
## 10   6864   9.3 Ornamental Crabapple          19             30             21
## # ... with 3,078 more rows, and 6 more variables: Crown_Base_Height <dbl>,
## #   Collected_By <chr>, Edible <chr>, Park <chr>, Latitude <dbl>,
## #   Longitude <dbl>
```

b. Export your dataset to a csv file using `write_csv()`.

```r
# I recommend leaving in eval = FALSE
write_csv(parks, file = "parks.csv")
```

c. Find a classmate (maybe a project group member?) and share your cleaned datasets with each other. Save their data on RStudio and import it in the R chunk below. Also, state who you shared data with. (Feel free to share your data with multiple people but you only need to load one classmate's dataset.)

```r
# Import their dataset
blaise_trees <- read_csv("blaise_trees.csv")
```

d. Compare your dataset and their dataset. In your comparison, answer the following questions:

- Do your datasets have the same number of rows? Same number of columns?
  We have the same number of columns, but not rows. He has 4,057 and I have 3,088.
- Use `setequal()` to determine if they are exactly the same.

```r
setequal(parks, blaise_trees)
```

```
## [1] FALSE
```

- How are they different? I think he did not filter out for the specific parks asked for and we coded the "Collected_By" observations differently.

e. A goal of this exercise with to experience both the **subjectivity** and **iterative nature** of data cleaning. Any time we clean data, we are making choices and often we don't catch all the bugs in our data the first (or second time around).

Based on your explorations of a classmate's cleaned dataset, do you think your dataset needs further wrangling? If not, justify. If so, do that now.

Mine does not need to be further wrangled. I went through every step mentioned for the problem, plus I have fewer observations so I know I didn't forget to do something.