612 Week 4 Homework: dplyr

Yunting

2/8/2020

Exercise 1: Pygmalion Data

```r
library(Sleuth3)
library(tidyverse)
```

```
## -- Attaching packages --- tidyverse 1.3.0 --

## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0

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

```r
data(case1302)
head(case1302)
```

```
##   Company     Treat Score
## 1      C1 Pygmalion  80.0
## 2      C1   Control  63.2
## 3      C1   Control  69.2
## 4      C2 Pygmalion  83.9
## 5      C2   Control  63.1
## 6      C2   Control  81.5
```

1. What are the units? What are the variables?
Units : platoons
Variables: Company, Treat and Score

```r
library(Sleuth3)
library(tidyverse)
library(scales)
```

2. Calculate the average score within each company by treatment combination.

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

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

```
case1302 %>%
  group_by(Company,Treat) %>% #calculate the mean Score of Company and Treat
 summarize(average_Score = mean(Score, na.rm = TRUE), n= n())
```

```
## # A tibble: 20 x 4
## # Groups:   Company [10]
##    Company Treat     average_Score     n
##    <fct>   <fct>             <dbl> <int>
##  1 C1      Control            66.2     2
##  2 C1      Pygmalion          80       1
##  3 C10     Control            70.7     2
##  4 C10     Pygmalion          83.7     1
##  5 C2      Control            72.3     2
##  6 C2      Pygmalion          83.9     1
##  7 C3      Control            76.2     1
##  8 C3      Pygmalion          68.2     1
##  9 C4      Control            66.5     2
## 10 C4      Pygmalion          76.5     1
## 11 C5      Control            76.2     2
## 12 C5      Pygmalion          87.8     1
## 13 C6      Control            81.8     2
## 14 C6      Pygmalion          89.8     1
## 15 C7      Control            65.1     2
## 16 C7      Pygmalion          76.1     1
## 17 C8      Control            70.5     2
## 18 C8      Pygmalion          71.5     1
## 19 C9      Control            73.1     2
## 20 C9      Pygmalion          69.5     1
```

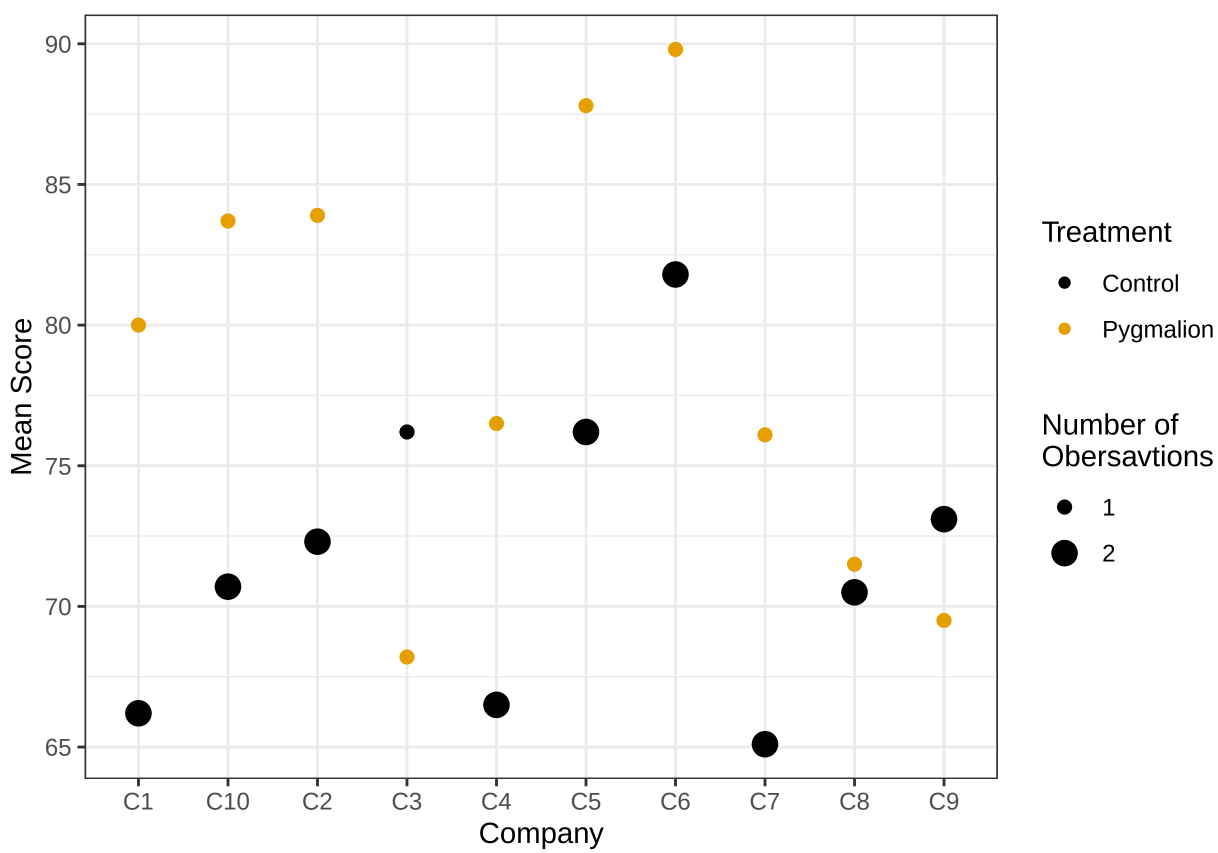
```


library(ggthemes)
case1302 %>%
  group_by(Company,Treat) %>%
 summarize(mean_Score = mean(Score, na.rm = TRUE), n = n()) ->


  sumdf
head(sumdf)
```

3. Copy this plot:

```
## # A tibble: 6 x 4
## # Groups:   Company [3]
##   Company Treat     mean_Score     n
##   <fct>   <fct>          <dbl> <int>
## 1 C1      Control         66.2     2
## 2 C1      Pygmalion       80       1
## 3 C10     Control         70.7     2
## 4 C10     Pygmalion       83.7     1
## 5 C2      Control         72.3     2
## 6 C2      Pygmalion       83.9     1
```

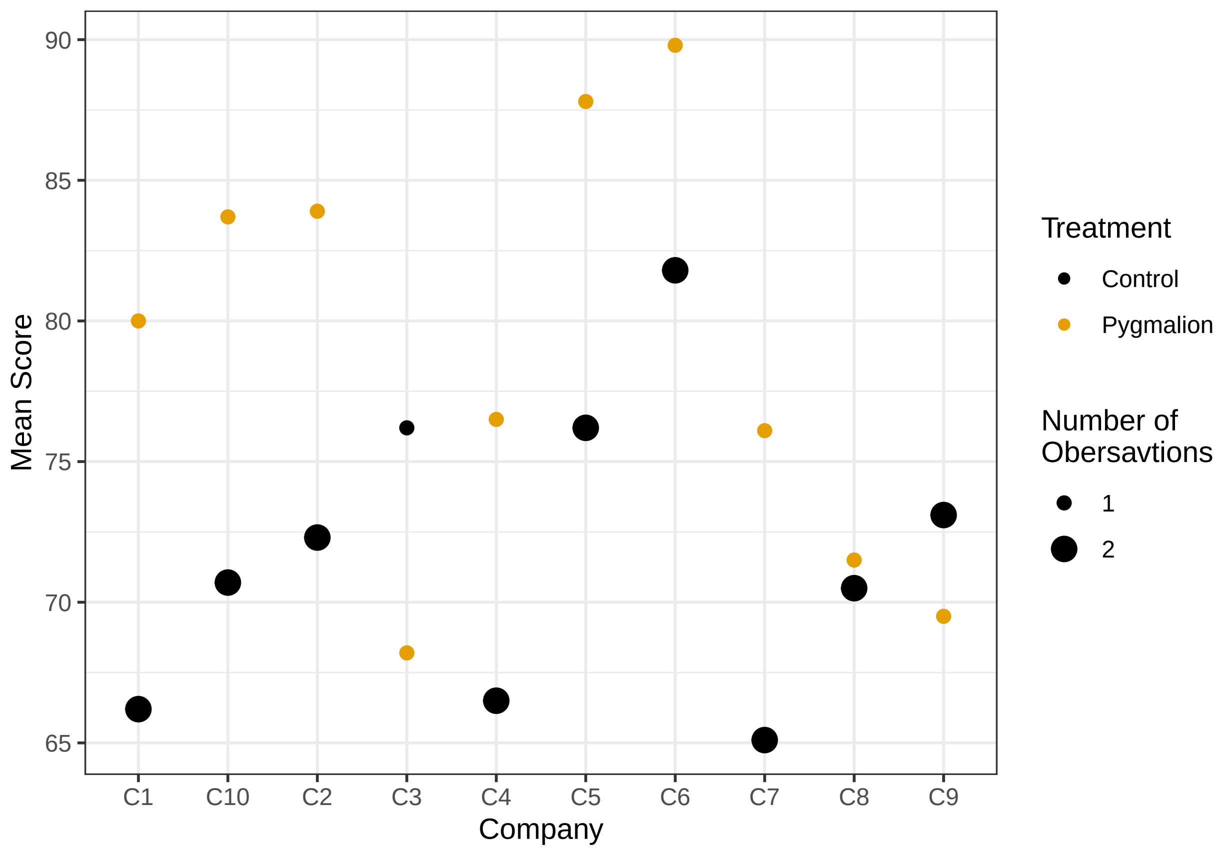
```
sumdf %>%
  ggplot(aes(x = Company, y = mean_Score)) +
  theme_bw() +
  geom_point(aes(size = n, color = Treat)) +
  scale_color_colorblind(name = "Treatment") +
  scale_size(name = "Number of \nObersavtions", range = c(2,4), breaks = c(1,2)) +
  guides("Treatment", "Number of Obersavtions") +
  ylab("Mean Score") +
  guides(color = guide_legend(order=1),size = guide_legend(order=2))
```

```
## Warning: Duplicated aesthetics after name standardisation: NA
```



```r
library(ggthemes)
case1302 %>%
  group_by(Company,Treat) %>%
 summarize(mean_Score = mean(Score, na.rm = TRUE), n = n()) %>%

  ggplot(aes(x = Company, y = mean_Score, size = n, color = Treat)) +
  theme_bw() +
  geom_point() +
  scale_color_colorblind(name = "Treatment") +

  scale_size(name = "Number of \nObersavtions", range = c(2,4), breaks = c(1,2)) +
  guides("Treatment", "Number of Obersavtions") +
  ylab("Mean Score") +
  guides(color = guide_legend(order=1),size = guide_legend(order=2))
```

```
## Warning: Duplicated aesthetics after name standardisation: NA
```



4. Does it make sense to add a loess smoother to the above plot? Why or why not? If so, add one.

I suggest we cannot add a geom_smooth line. Although we added the function, the spot of this chart is too separate. In this case, it is not make sense to add a loess smoother to the above plot.

real ans: because a loess smoother would only make sense if the explanatory variable was also quantitative, but it is categorical

Exercise 2: Midwest Data

- For this exercise, we'll use the midwest data from ggplot2.

```
data(midwest)
head(midwest)
```

1. Load these data into R.

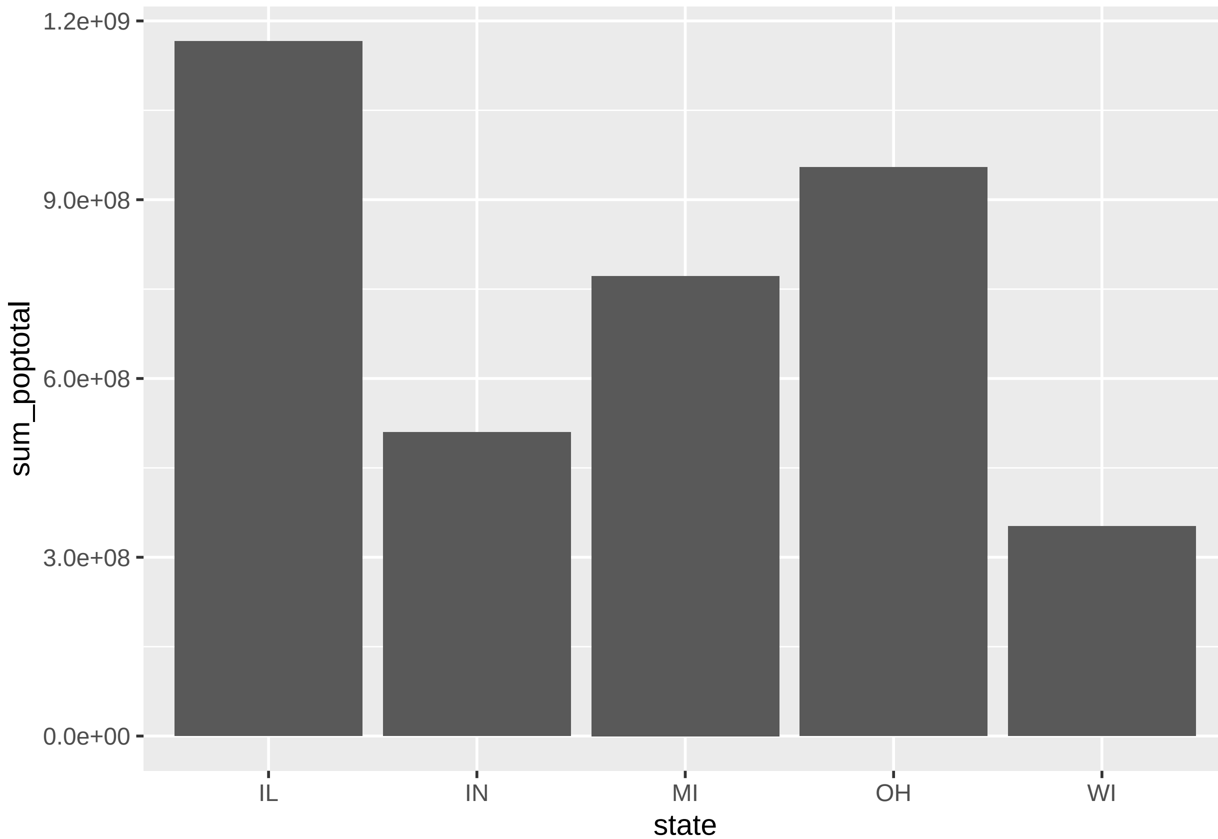
```
## # A tibble: 6 x 28
##     PID county state  area poptotal popdensity popwhite popblack popamerindian
##   <int> <chr>  <chr> <dbl>    <int>      <dbl>    <int>    <int>         <int>
## 1   561 ADAMS  IL    0.052    66090      1271.    63917     1702            98
## 2   562 ALEXA... IL    0.014    10626        759      7054     3496            19
## 3   563 BOND   IL    0.022    14991       681.    14477      429            35
## 4   564 BOONE  IL    0.017    30806      1812.    29344      127            46
## 5   565 BROWN  IL    0.018     5836       324.     5264      547            14
## 6   566 BUREAU IL    0.05     35688       714.    35157       50            65
## # ... with 19 more variables: popasian <int>, popother <int>, percwhite <dbl>,
## #   percblack <dbl>, percamerindan <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>
```

2. What are the observational units?

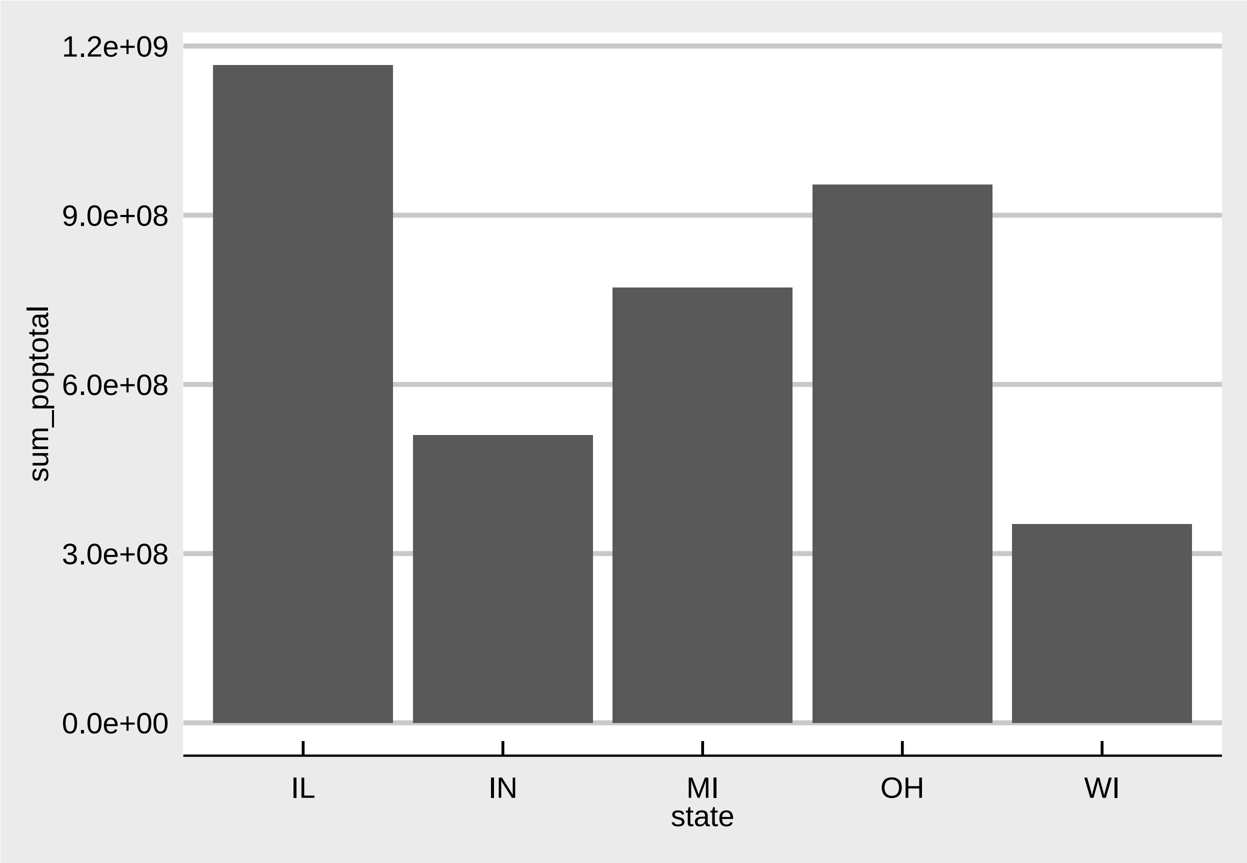
Ans: counties

```
midwest %>%
  group_by(state) %>%
  mutate(sum_poptotal = sum(poptotal, na.rm = TRUE))%>%


  ggplot(aes(x = state, y = sum_poptotal)) +
  geom_col()
```

3. Calculate the total population in each state. Display the data in an appropriate plot

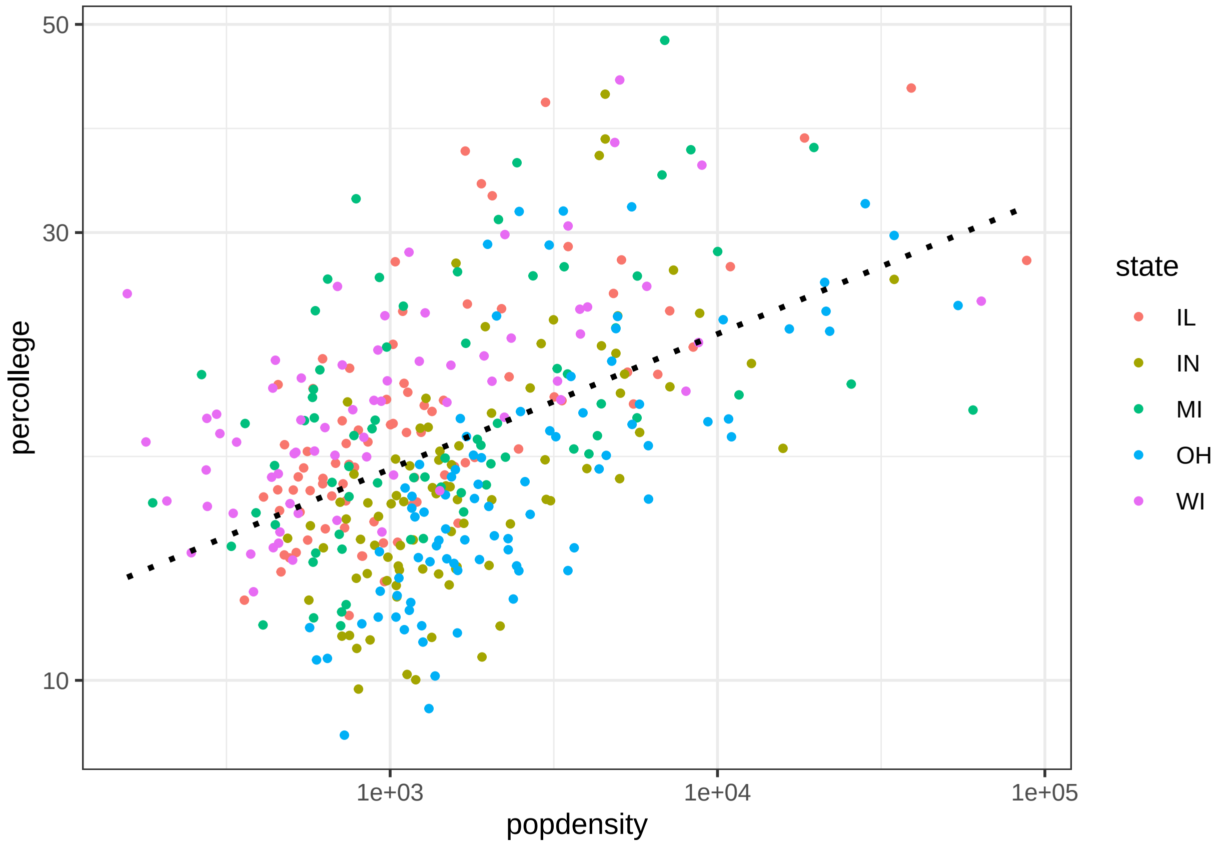



```
midwest %>%
  group_by(state) %>%
  mutate(sum_poptotal = sum(poptotal, na.rm = TRUE))  %>%
ggplot(aes(x = state, y = sum_poptotal))+
  theme_economist_white()+
  geom_col()
```



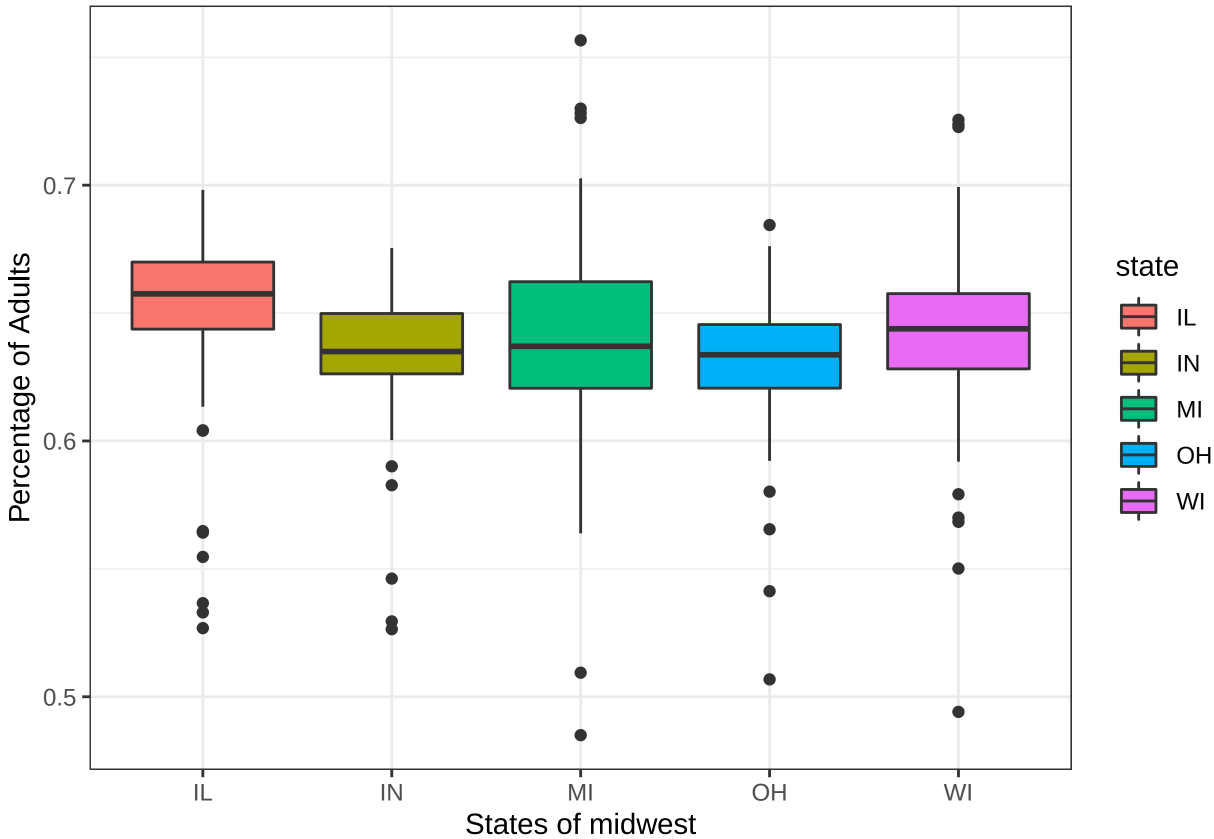

```
midwest %>%
  filter(poptotal > quantile(poptotal, probs = 0.1)) %>%
  ggplot( mapping = aes(x = popdensity, y = percollege, color = state)) +
  geom_point(size = 1) +
  geom_smooth(method = "lm", se = FALSE, linetype = "dotted", color = "black")+
  theme_bw() +
  scale_x_log10() +
  scale_y_log10()
```

4. Make a scatterplot of population density against the percent college educated on a scale that is appropriately linear. You should exclude counties in the bottom tenth percentile of total population. Color code the counties by state. Add the overall OLS line (the one that takes all points into account). Make the OLS line black and dotted.




```
midwest %>%
  mutate(percadults = popadults / poptotal) %>%
  arrange(desc(percadults)) %>%


  ggplot(mapping = aes(x = state, y = percadults, fill = state)) +
  theme_bw() +
  geom_boxplot() +
  ylab("Percentage of Adults") +
  xlab("States of midwest")
```

5. Make an appropriate plot to explore the association between the state and the percentage adults. Make sure the axis labels are nice and use the black-and-white theme.




```r
unique(midwest$state)
```

6. Use an R function to determine the possible values of state.

```
## [1] "IL" "IN" "MI" "OH" "WI"
```

```r
midwest %>%
  mutate(state = recode(state, IL = "Illinois", IN = "Indiana", MI = "Michigan", OH = "Ohio", WI = "Wis

 tail()
```

7. In the state variable, replace or recode the abbreviations with the full state name, e.g., IL with Illinois, IN with Indiana, etc..

```
## # A tibble: 6 x 28
##     PID county state  area poptotal popdensity popwhite popblack popamerindian
##   <int> <chr> <chr> <dbl>    <int>      <dbl>    <int>    <int>         <int>
## 1  3047 WASHI... Wisc... 0.025    95328      3813.    94465      125           208
```

```
## 2  3048 WAUKE... Wisc... 0.034   304715      8962.   298313     1096           672
## 3  3049 WAUPA... Wisc... 0.045    46104      1025.    45695       22           125
## 4  3050 WAUSH... Wisc... 0.037    19385       524.    19094       29            70
## 5  3051 WINNE... Wisc... 0.035   140320      4009.   136822      697           685
## 6  3052 WOOD   Wisc... 0.048    73605      1533.    72157       90           481
## # ... with 19 more variables: popasian <int>, popother <int>, percwhite <dbl>,
## #   percblack <dbl>, percamerindan <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
## #   percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>
```

Exercise 3: Coding Problems

```r
lsumfun <- function(x){
  stopifnot(is.numeric(x))
  if (length(x) %% 2 == 0){
    return(sum(x[x], na.rm = TRUE))
  } else {
      return(sum(x[x %% 2 == 0], na.rm = TRUE))
  }

}

lsumfun(c(1, 2, 3, NA))
```

1. Create a function that takes a vector of numerics as input. It checks the length of the vector. If the length is even, it returns the sum of the vector. If the length is odd, it returns the sum of all of the even numbers of the vector. For example, the following are some outputs of one implementation, calledn lsumfun().

```
## [1] 6
```

```r
lsumfun(c(1, 2, 3))
```

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

```r
lsumfun(c(2, 3, 4, 5, 6))
```

```
## [1] 12
```

```r
lsumfun(c(2, 3, 4, 5, NA))
```

```
## [1] 6
```

```r
lsumfun(c(2, 3, 4, NA, 6))
```

```
## [1] 12
```

2. We add a Leap Day on February 29, almost every four years. In the Gregorian calendar three criteria must be taken into account to identify leap years:

- The year can be evenly divided by 4, is a leap year, unless:
- The year can be evenly divided by 100, it is NOT a leap year, unless:
- The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years

Write a function that takes the year as input and returns TRUE if it is a leap year and FALSE if it is not a leap year. Evaluate your function at 2, 12, 200, 800.

```
Gregorian_calendar <- function(y){
  if (y %% 400 == 0){return(" TRUE")}
     else if (y %% 100 ==0){return(" FALSE")}
     else if (y %% 4 ==0){return(" TRUE")}

     else {return("FALSE")}


}
Gregorian_calendar(2)
```

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

```
Gregorian_calendar(12)
```

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

```
Gregorian_calendar(200)
```

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

```
Gregorian_calendar(800)
```

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

3. Create a function that takes two vectors of numerics as input, a and b.

- Your vector should throw an error if either a or b contain repeated elements.
- If a number is in a and not b you add 1 to your score.
- If a number is in b and not a you subtract 1 from your score.
- If a number is in both a and b you do nothing to your score.
- The function returns the final score.
- Hint: help("%in%")

```
score2 <- function(a,b){
  stopifnot(is.numeric(a))
  stopifnot(is.numeric(b))
```

```r
  if(length(a) != length(unique(a))){
    stop("a has some repeated elements ")
  }else if(length(b) != length(unique(b))){
    stop("b has some repeated elements")
  }

  return(sum(!(a %in% b)) - sum(!(b %in% a)))
  }

score2(c(1, 2, 3), c(3, 4))
```

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

```r
score2(c(1, 2, 3), c(1, 2, 3, 4))
```

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

```r
score2(c(1, 2, 3), c(3, 4, 4))
```

```
## Error in score2(c(1, 2, 3), c(3, 4, 4)): b has some repeated elements
```