Data Wrangling Assignment 5

Yaniv Bronshtein

2/28/2021

Import the necessary libraries

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

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

```
library(broom)
library(gapminder)
library(rsample)
```

Problem 1

In this exercise we will work with the total number of words spoken by characters of different races and genders in the Lord of the Rings movies.
1.Get the data in a single data frame
Create 3 data frames (or tibbles) from these files:

```
fellowship_t <- read_csv("https://raw.githubusercontent.com/jennybc/lotr-tidy/master/data/The_Fellowship
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Film = col_character(),
##   Race = col_character(),
##   Female = col_double(),
##   Male = col_double()
## )
```

```
two_towers_t <- read_csv("https://raw.githubusercontent.com/jennybc/lotr-tidy/master/data/The_Two_Towers
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Film = col_character(),
##   Race = col_character(),
##   Female = col_double(),
##   Male = col_double()
## )
```

```
return_t <- read_csv("https://raw.githubusercontent.com/jennybc/lotr-tidy/master/data/The_Return_Of_The_
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Film = col_character(),
##   Race = col_character(),
##   Female = col_double(),
##   Male = col_double()
## )
```

2. Tidy the combined data frame by creating new variables "Gender" and "Words"
   Use bind_rows to sequentially append all the tibbles to each other Then, use pivot_longer() to convert the Male and Female columns to values in the Gender column In turn, move the values originally in these columns to a new column called Words

```
lotr_t <- fellowship_t%>%bind_rows(two_towers_t) %>% bind_rows(return_t)
lotr_t <- lotr_t %>% pivot_longer(cols = c("Female", "Male"), names_to = "Gender", values_to = "Words")
```

3.Use the combined data frame to answer the following questions
a)How many words were spoken in each movie?
First, create a tibble by grouping lotr_t by Film and then using summarise() to create a new column total_words based on the film groups

```
total_movie_wc_t <- lotr_t %>% group_by(Film) %>% summarise(total_words=sum(Words))
```

Next, derive the count in the first film by using str_detect on the Film column to filter the dataframe, pulling out the column in vector form, and then converting the result to a numeric value Repeat thuis step for the other two movies

```
count_fellowship <- total_movie_wc_t %>% filter(str_detect(Film, "Fellow")) %>% pull(total_words) %>% a
cat("The Fellowship of the Ring Total Word Count", count_fellowship)
```

```
## The Fellowship of the Ring Total Word Count 7853
```

```
count_towers <- total_movie_wc_t %>% filter(str_detect(Film, "Tower")) %>% pull(total_words) %>% as.num
cat("The Two Towers Total Word Count", count_towers)
```

```
## The Two Towers Total Word Count 7297
```

```
count_return <- total_movie_wc_t %>% filter(str_detect(Film, "Return")) %>% pull(total_words) %>% as.nu
cat("The Return of the King Total Word Count", count_return)
```

```
## The Return of the King Total Word Count 6095
```

b)How many words were spoken by each gender in total?
First, create a tibble by grouping lotr_t by Gender and then using summarise() to create a new column words_by_gender based on the Gender groups

```
words_by_gender_t <- lotr_t %>% group_by(Gender) %>% summarise(words_by_gender = sum(Words))
```

Next, derive the count of males and females by filtering on Male and Female respectively pulling out the column in vector form, and then converting the result to a numeric value

```
male_wc <- words_by_gender_t %>% filter(Gender=="Male") %>% pull(words_by_gender) %>% as.numeric()
cat("Total words spoken by men", male_wc)
```

```
## Total words spoken by men 18817
```

```
female_wc <- words_by_gender_t %>% filter(Gender=="Female") %>% pull(words_by_gender) %>% as.numeric()
cat("Total words spoken by women", female_wc)
```

```
## Total words spoken by women 2428
```

c)How many words were spoken by each race in total?
First, create a tibble by grouping lotr_t by Race and then using summarise() to create a new column words_by_race based on the Race groups

```
words_by_race_t <- lotr_t %>% group_by(Race) %>% summarise(words_by_race = sum(Words))
```

Next, derive the count of each race filtering the Race column, pulling out the column in vector form, and then converting the result to a numeric value

```
elf_wc <- words_by_race_t %>% filter(Race == "Elf") %>% pull(words_by_race) %>% as.numeric()
cat("Total words spoken by elves", elf_wc)
```

```
## Total words spoken by elves 3737
```

```
hobbit_wc <- words_by_race_t %>% filter(Race == "Hobbit") %>% pull(words_by_race) %>% as.numeric()
cat("Total words spoken by Hobbits", hobbit_wc)
```

```
## Total words spoken by Hobbits 8796
```

```
man_wc <- words_by_race_t %>% filter(Race == "Man") %>% pull(words_by_race) %>% as.numeric()
cat("Total words spoken by Man", man_wc)
```

```
## Total words spoken by Man 8712
```

4.Create a data frame with totals by race and movie, calling it by_race_film.

```
by_race_film_t <- lotr_t %>% group_by(Film, Race) %>% summarise(words_by_race_movie = sum(Words))
```

```
## `summarise()` has grouped output by 'Film'. You can override using the `.groups` argument.
```

```
by_race_film_t
```

```
## # A tibble: 9 x 3
## # Groups:   Film [3]
##   Film                       Race   words_by_race_movie
##   <chr>                      <chr>                <dbl>
## 1 The Fellowship Of The Ring Elf                   2200
## 2 The Fellowship Of The Ring Hobbit                3658
## 3 The Fellowship Of The Ring Man                   1995
## 4 The Return Of The King     Elf                    693
## 5 The Return Of The King     Hobbit                2675
## 6 The Return Of The King     Man                   2727
## 7 The Two Towers             Elf                    844
## 8 The Two Towers             Hobbit                2463
## 9 The Two Towers             Man                   3990
```

Problem 2

1.Split/group the gapminder data by country. For each country, fit an ARIMA(0,0,1) or MA(1) model to lifeExp, and produce a tibble that the country-wise values of AIC and BIC, two measures of goodness of model fit. Obtain a scatter plot of AIC versus BIC and comment.

Convert gapminder data set to tibble form

```
gapminder_t <- gapminder %>% as_tibble()
```

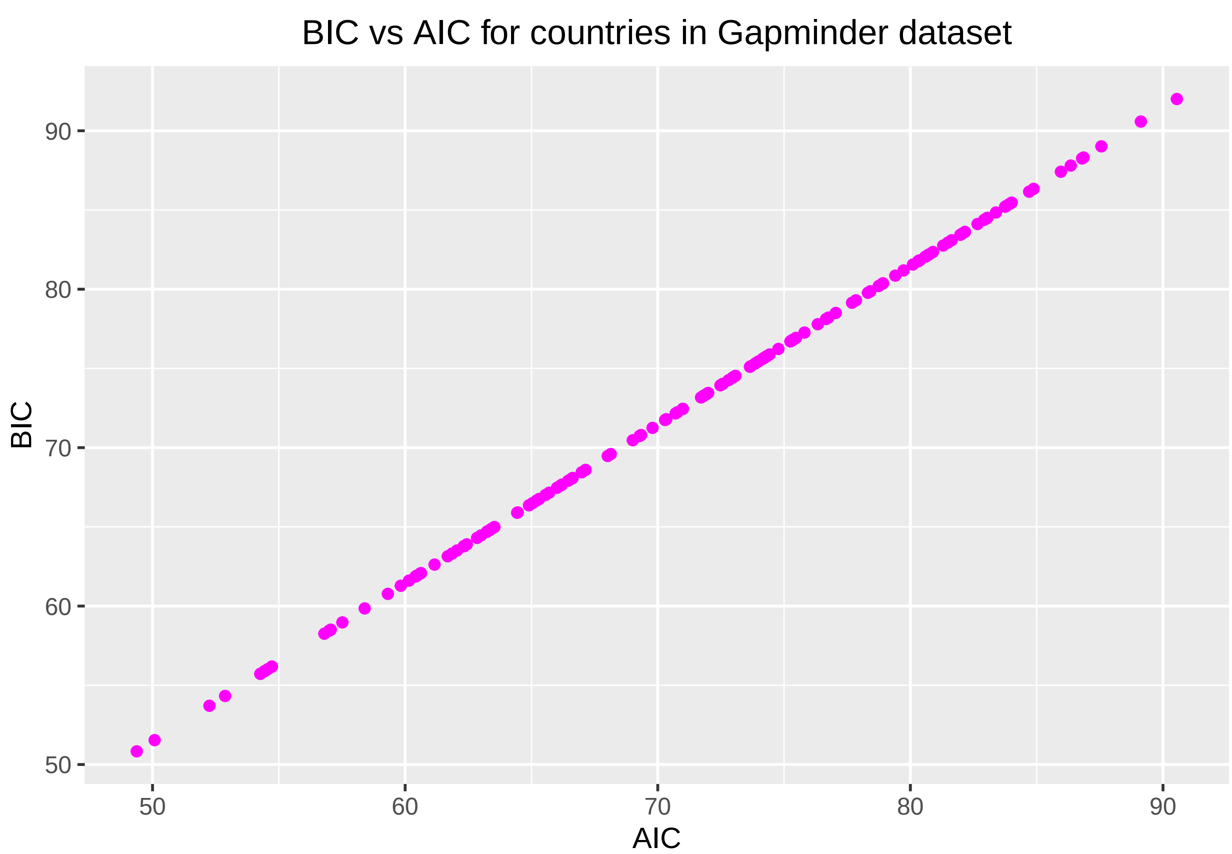
Create a function compute_aic_bic() to generate a tibble containing the country name, AIC, and BIC in that order

```
#This function returns a tibble containing the country, AIC, BIC
# Params:
#p1: number of AR coefficients
#p2: number of differences
#p3: number of MA coefficients
compute_aic_bic = function(p1, p2, p3) {

  #Split the gapminder_t by country using group_split().
  #Use map to apply arima() with order based on function input
  #Use map again to apply the broom function glance() to receive model level information
  #(AIC and BIC)
  countries_arima <- gapminder_t %>%
  group_by(country) %>%
  group_split() %>%
  map(~arima(.$lifeExp, order = c(p1, p2, p3))) %>%
  map(glance)
```

```r
  #Extract AIC and BIC in vector form using map_dbl()
  countries_aic <- countries_arima %>% map_dbl(~.$AIC)
  countries_bic <- countries_arima %>% map_dbl(~.$BIC)
  #Create a tibble by combining the countries(removing duplicates using the unique())
  #function, and adding the extracted AIC and BIC to the end
  countries_t <- gapminder_t %>% select(country) %>%
  unique() %>%
  mutate(AIC = countries_aic) %>%
  mutate(BIC = countries_bic)

  return(countries_t)
}
```

Generate the ggplot of AIC vs BIC for ARIMA(0,0,1)

```r
m1 <- compute_aic_bic(0, 0, 1)
ggplot(data = m1 , mapping = aes(x = AIC, y = BIC)) +
  geom_point(color = "Magenta") +
  labs(
    title = "BIC vs AIC for countries in Gapminder dataset"
  ) +
  theme(plot.title = element_text(hjust = 0.5))
```



2.Now repeat the previous step for four other models: ARIMA(0,0,1), ARIMA(0,0,2), ARIMA(0,0,3), ARIMA(0,1,0), ARIMA(0,1,1), and in a single plot, show boxplots of AIC values for the five models. Based on the boxplot, which of these five models do you think fits the data best for most countries?

In the code below, we will call compute__aic__bic() for the required models And create an additional column filled with the corresponding string to make plotting easier

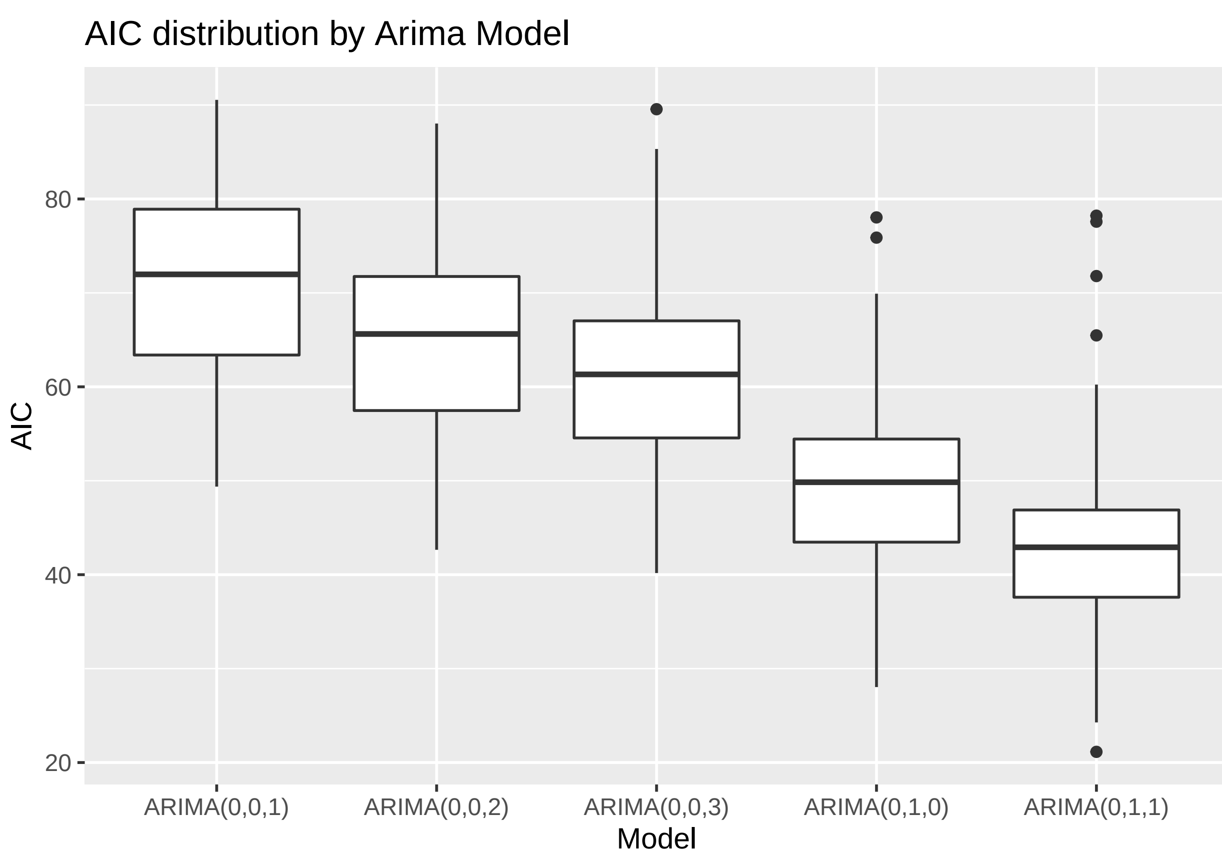
```
m1 <- m1 %>% mutate(Model = "ARIMA(0,0,1)" )
m2 <- compute_aic_bic(0, 0, 2) %>% select(AIC) %>% mutate(Model = "ARIMA(0,0,2)")
m3 <- compute_aic_bic(0, 0, 3) %>% select(AIC) %>% mutate(Model = "ARIMA(0,0,3)")
m4 <- compute_aic_bic(0, 1, 0) %>% select(AIC) %>% mutate(Model = "ARIMA(0,1,0)")
m5 <- compute_aic_bic(0, 1, 1) %>% select(AIC) %>% mutate(Model = "ARIMA(0,1,1)")
```

Next, we stack each model on top of one another to create a unified tibble with the following structure: Country|AIC|BIC|Model

```
models_t <- bind_rows(m1, m2, m3, m4, m5)
```

Generate the boxplot

```
ggplot(data = models_t, mapping = aes(x = Model, y = AIC)) +
  geom_boxplot() +
  labs(title = "AIC distribution by Arima Model")
```



The best model has the minimal AIC, so ARIMA(0,1,1) wins

3.Filter the data only for continent Europe.
For the best model identified in step 2, create a tibble showing the

country-wise model parameters (moving average coefficients) and their errors using the broom package.

Create a separate dataframe on gapminder just for Europe

```
gapminder_europe <- gapminder_t %>% filter(continent == "Europe")
```

Apply similar logic to that of compute_aic_bic(), replacing glance() with tidy() to generate coefficient level data

```
countries_arima_europe_tidy <- gapminder_europe %>%
  group_by(country) %>%
  group_split() %>%
  map(~arima(.$lifeExp, order = c(0, 1, 1))) %>%
  map(tidy)

countries_ma_estimate <- countries_arima_europe_tidy %>%
  map_dbl(~.$estimate)
countries_ma_std.error <- countries_arima_europe_tidy %>%
  map_dbl(~.$std.error)

countries_list <- gapminder_europe %>% select(country) %>% unique()
countries_ma_estimate_error_t <-
  tibble(countries_list,
         estimate = countries_ma_estimate, error = countries_ma_std.error)

countries_ma_estimate_error_t
```

```
## # A tibble: 30 x 3
##    country                estimate error
##    <fct>                     <dbl> <dbl>
##  1 Albania                   1.00  0.353
##  2 Austria                   0.708 0.263
##  3 Belgium                   0.645 0.183
##  4 Bosnia and Herzegovina    1.00  0.353
##  5 Bulgaria                  1.00  0.411
##  6 Croatia                   0.676 0.199
##  7 Czech Republic            0.606 0.203
##  8 Denmark                   0.494 0.204
##  9 Finland                   0.778 0.227
## 10 France                    0.706 0.189
## # ... with 20 more rows
```

4.Now filter the data only for year 1992. Plot lifeExp against log10(gdpPercapita). Fit a linear model of lifeExp on log10(gdpPercapita) using population as weights and obtain (i) bootstrapped 95% confidence intervals for the slope coefficient and (ii) bootstrapped 90% prediction intervals for each data point using 500 bootstrapped samples (show a plot of the prediction intervals). Compare the bootstrapped 95% confidence intervals for the estimated slope coefficient with those generated automatically by the lm() function. Which one is wider?

Step 1: create a filtered tibble on gapminder_t for the year 1992

```r
gapminder_1992_t <- gapminder_t %>% filter(year == 1992)
```

Step 2: Generate a linear model to be used as reference and display both coefficient and confidence interval data

```r
lm_1992 <- gapminder_1992_t %>%
  lm(lifeExp ~ log10(gdpPercap), weights = pop, data = .)

lm_1992_coef <- coef(lm_1992)
cat("LM 1992 coefficients:\n", lm_1992_coef,"\n")
```

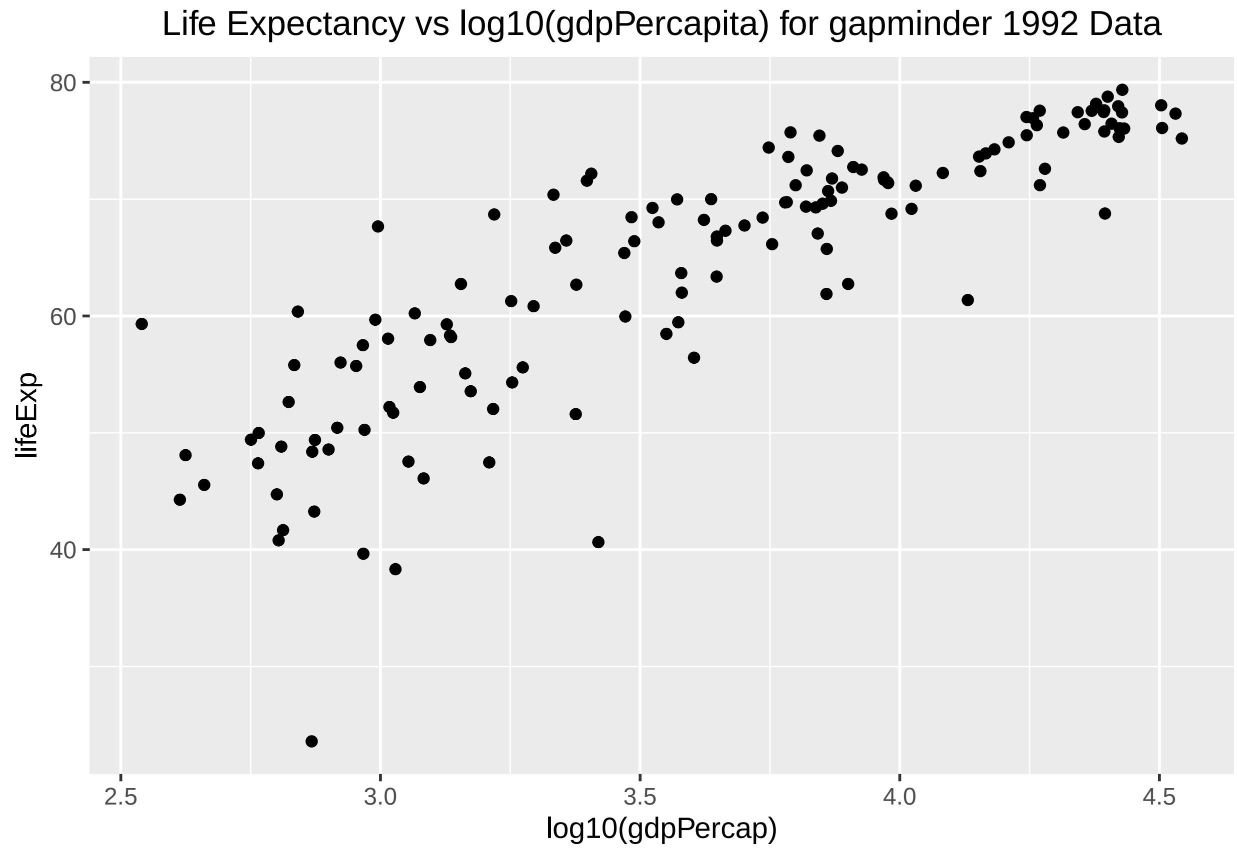
```
## LM 1992 coefficients:
##  23.72062 12.04351
```

```r
lm_1992_confint <- confint(lm_1992, level = 0.95)
cat("LM 1992 confidence interval:\n", lm_1992_confint,"\n")
```

```
## LM 1992 confidence interval:
##  17.87865 10.38391 29.5626 13.70311
```

Step 3: Generate a scatter plot showing Life Expectancy vs log10(gdpPercapita)

```r
ggplot(data = gapminder_1992_t,
       mapping = aes(x = log10(gdpPercap), y = lifeExp)) +
  geom_point() +
  labs(
    title = "Life Expectancy vs log10(gdpPercapita) for gapminder 1992 Data"
  ) +
  theme(plot.title = element_text(hjust = 0.5))
```



Step 4: As part of the requirement for part (i) provide bootstrapped 95% confidence intervals for the slope coefficient. Tidy is necessary for this

```
set.seed(1)
alpha1 = 0.05
boot_lm <- gapminder_1992_t %>% bootstraps(500)
boot_lm1 <- map(boot_lm$splits, ~as_tibble(.)) %>%
  map(~tidy(lm(lifeExp ~ log10(gdpPercap), weights = pop, data = .))) %>%
  bind_rows(.)
```

Step 4a. Display the confidence interval together with the median for part (i)

```
conf_int_95 <- boot_lm1 %>%
  group_by(term) %>%
  summarise(conf.low = quantile(estimate, alpha1 /2),
                                    conf.high = quantile(estimate, 1 - alpha1 /2),
                                    median = median(estimate))
conf_int_95
```

```
## # A tibble: 2 x 4
##   term             conf.low conf.high median
## * <chr>               <dbl>     <dbl>  <dbl>
## 1 (Intercept)          4.27      34.5   22.7
## 2 log10(gdpPercap)     9.51      16.6   12.3
```

Step 5: As part of the requirement for part (ii), provide the bootstrapped 90% prediction intervals Augment() is now necessary for this

```r
alpha2 = 0.1
boot_lm2 <- map(boot_lm$splits, ~as_tibble(.)) %>%
  map(~augment(lm(lifeExp ~ log10(gdpPercap), weights = pop, data = .))) %>%
  # Needed to rename because of special characters
  bind_rows(.) %>% rename(log_gdp_percap = names(.)[2])
```

Step 5a. Display the confidence interval together with the median for part (ii)

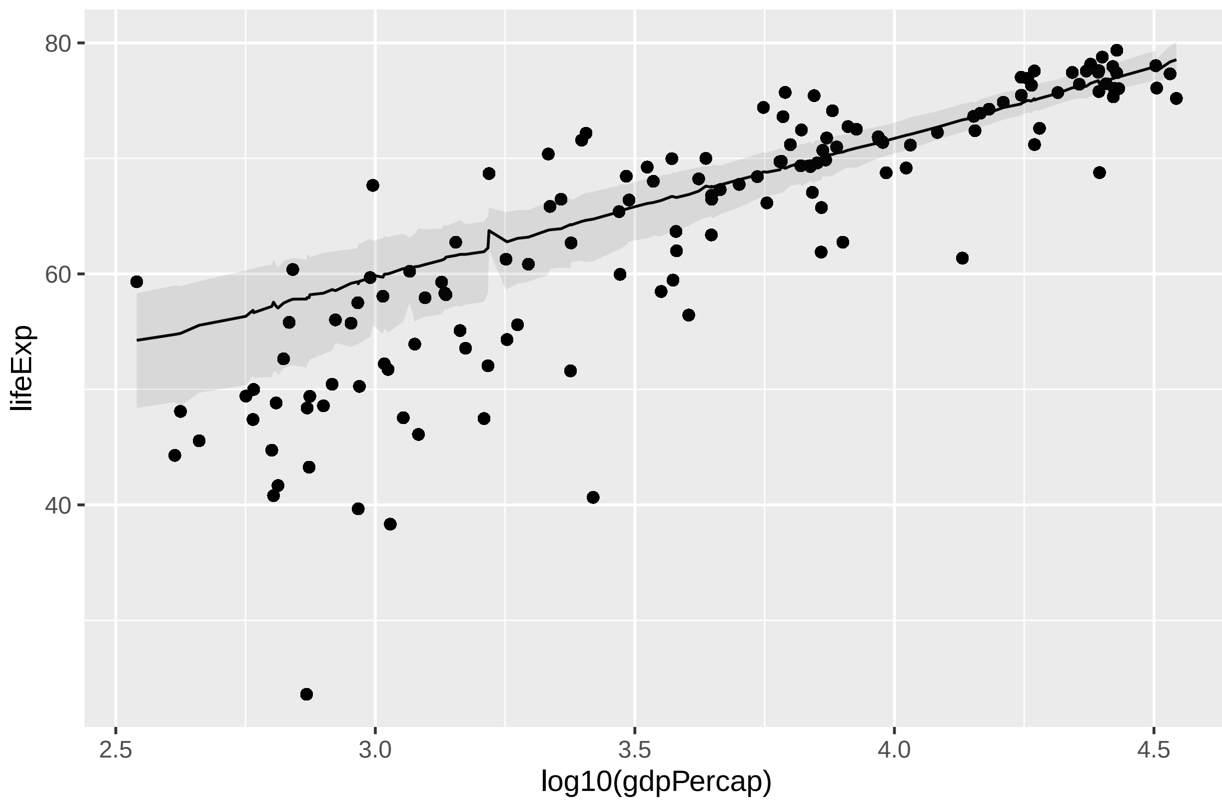
```r
conf_int_90 <- boot_lm2 %>% group_by(log_gdp_percap) %>%
  summarise(conf.low = quantile(.fitted, alpha2 / 2),
            conf.high = quantile(.fitted, 1 - alpha2 / 2),
            median = median(.fitted))

conf_int_90
```

```
## # A tibble: 142 x 4
##    log_gdp_percap conf.low conf.high median
##  *          <dbl>    <dbl>     <dbl>  <dbl>
##  1           2.54     48.4      58.3   54.2
##  2           2.61     48.9      59.0   54.8
##  3           2.62     48.6      58.9   54.8
##  4           2.66     49.7      59.4   55.5
##  5           2.75     50.4      60.3   56.3
##  6           2.76     51.2      60.5   56.8
##  7           2.77     51.0      60.5   56.6
##  8           2.80     51.1      60.8   57.2
##  9           2.80     51.5      61.2   57.5
## 10           2.81     51.6      60.8   57.2
## # ... with 132 more rows
```

Show a plot of the prediction intervals

```r
ggplot(conf_int_90) +
  geom_point(aes(x = log_gdp_percap, y = lifeExp), data = boot_lm2) +
  geom_line(aes(x = log_gdp_percap, y = median)) +
  geom_ribbon(aes(x = log_gdp_percap, ymin = conf.low, ymax = conf.high),
              alpha = 0.1) +
  labs(
    title = "Plot of prediction intervals for bootstraps(500) of a Linear Model",
    x = "log10(gdpPercap)")
```

Plot of prediction intervals for bootstraps(500) of a Linear Model



I believe that the bootstrapped is wider