Stat 292 - Recitation 9

Linear Models

Orçun Oltulu

26 / 5 / 2021

Load Required Packages

```
inst_pack_func <- function(list.of.packages){
  new.packages <- list.of.packages[!(list.of.packages %in%
                                       installed.packages()[,"Package"])]
  if(length(new.packages)) install.packages(new.packages)
  lapply(list.of.packages,function(x){library(x,character.only=TRUE)})
}

list.of.packages <- c("tidyverse","magrittr",
                      "ISLR","psych","dplyr",
                      "ggplot2","knitr", "modelr",
                      "bindrcpp","gapminder",
                      "purrr","broom")

inst_pack_func(list.of.packages)
```

Exercise 1:

Part A:

Load Auto data set from ISLR package.

```
data("Auto")
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
```

```
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161
```

Part B:

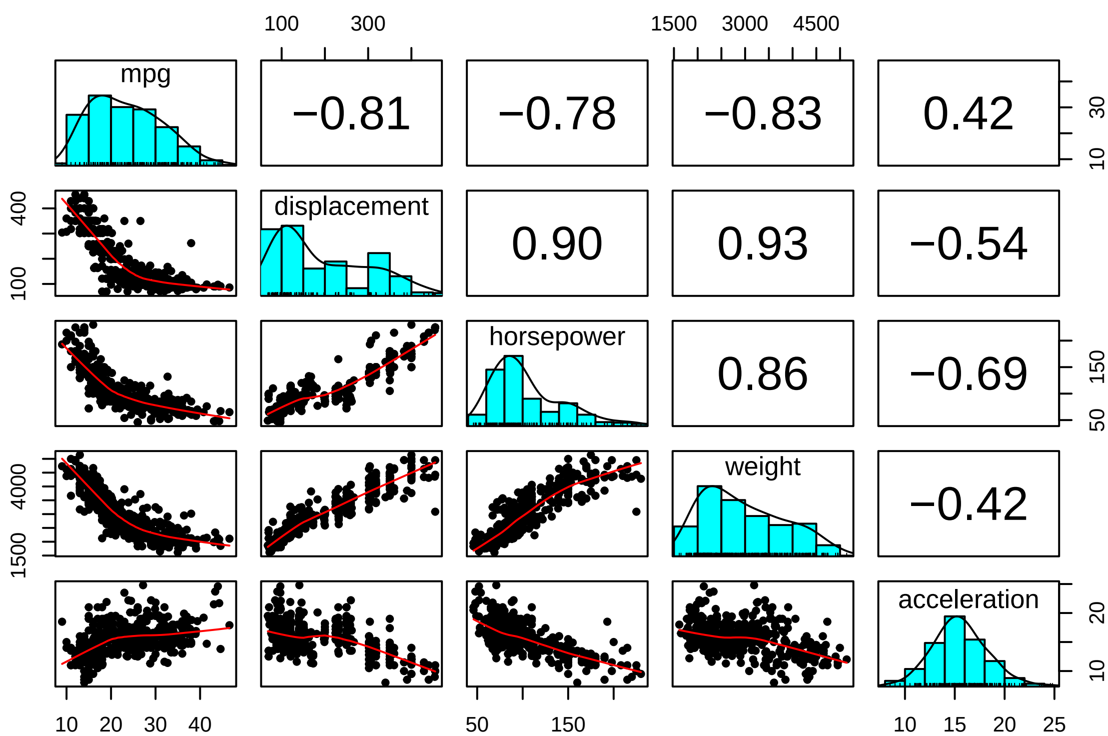
Drop ‘name’ and year variables, convert cylinders and origin to factors.

```
Auto %<>%
  select(-name, -year) %>%
  mutate(origin = factor(origin), cylinders = factor(cylinders))
```

Part C:

Using ‘pairs.panels’ function from ‘psych’ package, obtain a scatter-plot & correlation matrix for numeric variables.

```
Auto %>%
  select_if(is.numeric) %>%
  pairs.panels(.,ellipses = FALSE)
```



Part D:

Fit a simple linear regression model for estimating mpg, using only weight variable. Comment on the outputs.

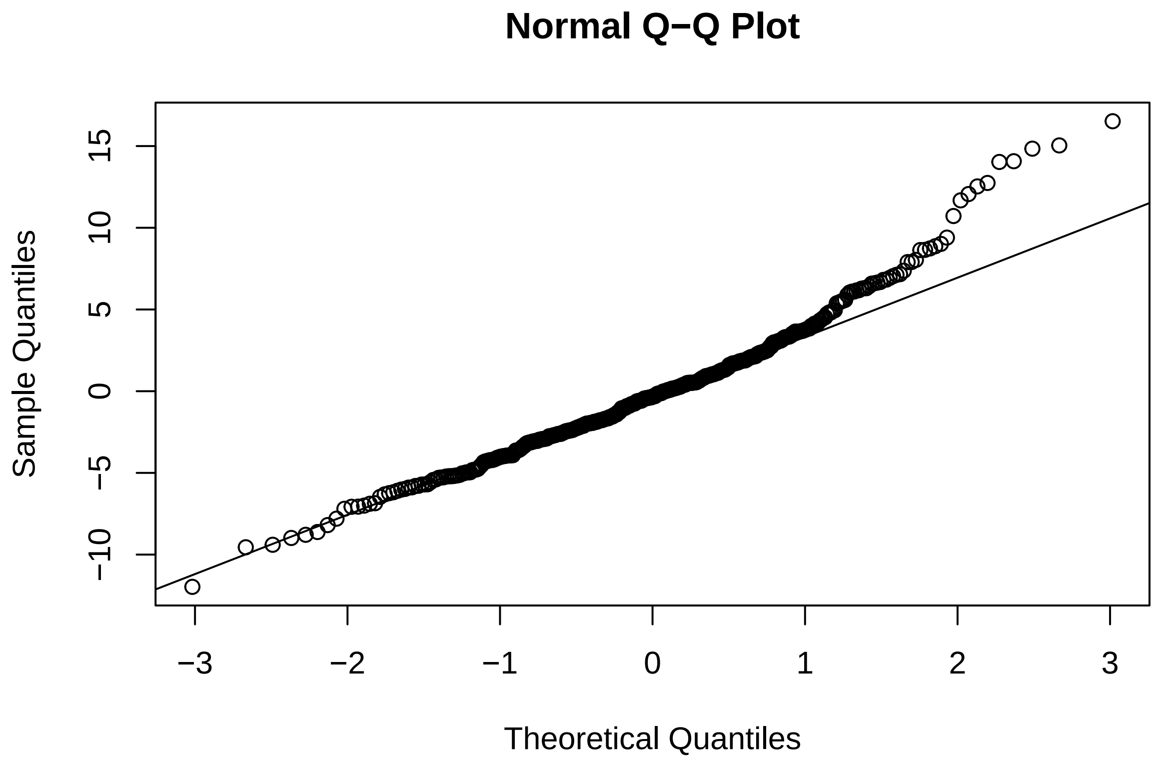
```
fit1 <- lm(mpg ~ weight, data = Auto)
summary(fit1)
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9736  -2.7556  -0.3358   2.1379  16.5194
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.216524   0.798673   57.87   <2e-16 ***
## weight      -0.007647   0.000258  -29.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.333 on 390 degrees of freedom
## Multiple R-squared:  0.6926, Adjusted R-squared:  0.6918
## F-statistic: 878.8 on 1 and 390 DF,  p-value: < 2.2e-16
```

Part E:

Obtain a Quantile-Quantile Plot (QQ-Plot) for residuals. Comment on the plot.

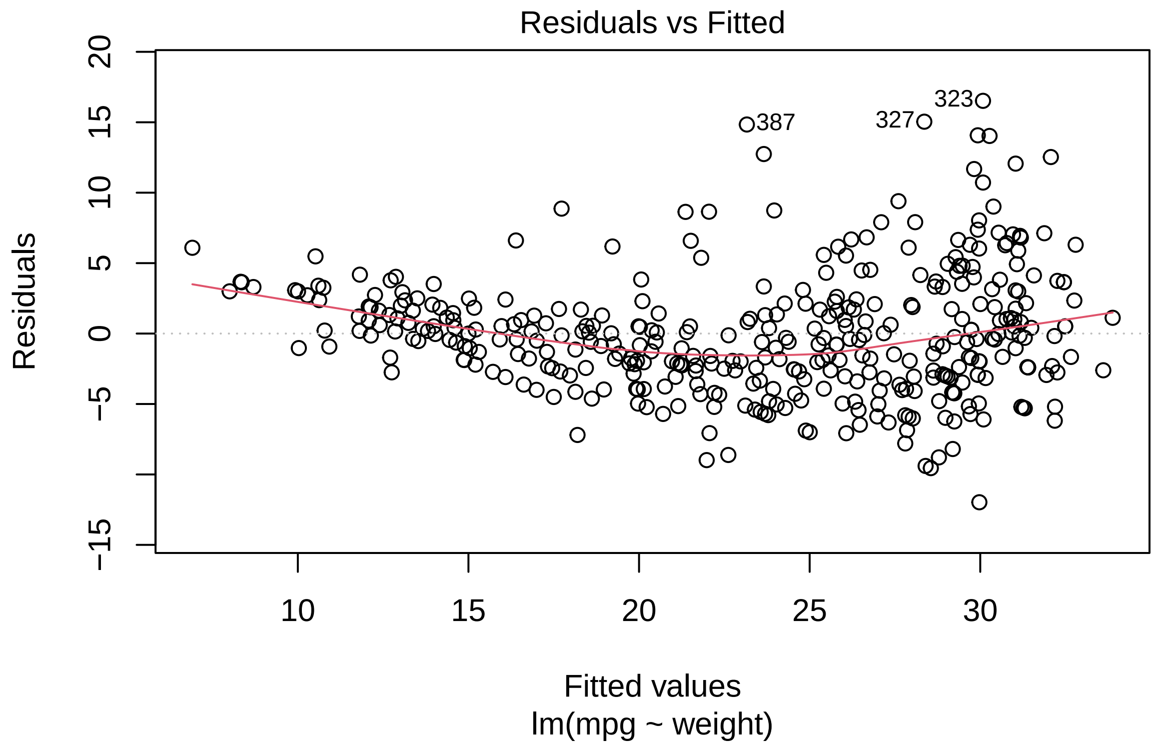
```
fit1$residuals %T>%
  qqnorm() %>%
  qqline()
```



Part F:

Obtain Fitted vs Residuals plot. Comment on the plot.

```
#plot(fit1$fitted.values, fit1$residuals)
plot(fit1, which = 1)
```



Part G:

Now add another variable, 'horsepower' to the model and compare summary results with the previous model.

```
fit2 <- lm(mpg ~ weight + horsepower, data = Auto)
summary(fit2)
```

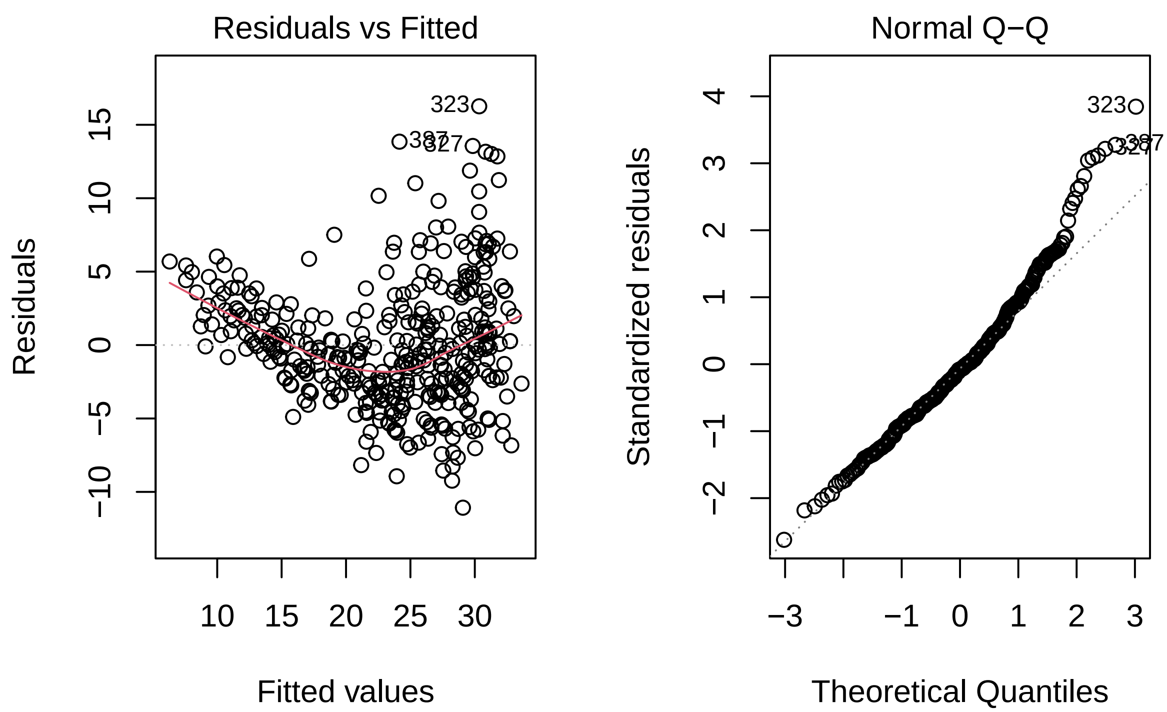
```
##
## Call:
## lm(formula = mpg ~ weight + horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0762  -2.7340  -0.3312   2.1752  16.2601
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 45.6402108  0.7931958  57.540  < 2e-16 ***
## weight      -0.0057942  0.0005023 -11.535  < 2e-16 ***
## horsepower  -0.0473029  0.0110851  -4.267 2.49e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.24 on 389 degrees of freedom
## Multiple R-squared:  0.7064, Adjusted R-squared:  0.7049
## F-statistic: 467.9 on 2 and 389 DF,  p-value: < 2.2e-16
```

Part H:

Obtain Fitted vs Residuals plot and Quantile-Quantile Plot for the second model.

```r
par(mfrow = c(1,2))
plot(fit2, which = 1:2)
```



Part I:

Now, split the data set into 2 parts, train and test. Keep 80% (roughly) of the observations for train set, and assign rest of the data to test set. (You can use 'sample' function. Also use set.seed(292))

```r
set.seed(292)
index <- sample(1:nrow(Auto),
                size = floor(nrow(Auto) * 0.8),
                replace = FALSE)
```

```r
train <- Auto[index,]
test <- Auto[-index,]
```

Part J:

After you split the data into 2 parts, using all the variables you have in the training data, fit a multiple linear regression model.

```r
fit3 <- lm(mpg ~ ., data = train)
summary(fit3)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5363 -2.2117 -0.3517  1.8516 15.4383
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  34.9918128  3.1169637  11.226  < 2e-16 ***
## cylinders4    8.7714446  1.9725694   4.447 1.23e-05 ***
## cylinders5   11.8872304  2.9972828   3.966 9.13e-05 ***
## cylinders6    4.6144602  2.2097490   2.088 0.037615 *
## cylinders8    6.3211745  2.6029940   2.428 0.015748 *
## displacement -0.0002436  0.0093765  -0.026 0.979291
## horsepower   -0.0661583  0.0168639  -3.923 0.000108 ***
## weight       -0.0033385  0.0008028  -4.158 4.18e-05 ***
## acceleration -0.1604157  0.1205478  -1.331 0.184284
## origin2      -0.6804321  0.7335845  -0.928 0.354385
## origin3       2.2749845  0.7020968   3.240 0.001327 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.648 on 302 degrees of freedom
## Multiple R-squared:  0.7636, Adjusted R-squared:  0.7558
## F-statistic: 97.54 on 10 and 302 DF,  p-value: < 2.2e-16
```

Part K:

Calculate Mean Squared Error value for the last model. To do this, you can to use 'predict' command to get prediction $\hat{y}_i$ values for test set and compare with real values y_i.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2$$

```
pred_val <- predict(fit3, newdata = test)
mse <- (pred_val - test$mpg)^2 %>% mean
```

Exercise 2:

Part A:

Import population.csv data into R.

```
population <- read.csv("Population.csv")
head(population)
```

```
##   Year           City Population In.migration Out.migration Net.migration
## 1 2020          ADANA    2258718        47088         45832          1256
## 2 2020       ADIYAMAN     632459        16163         16936          -773
## 3 2020 AFYONKARAHISAR     736912        18439         19434          -995
## 4 2020           AGRI     535435        14626         24353         -9727
## 5 2020         AMASYA     335494        12241         11347           894
## 6 2020         ANKARA    5663322       153162        141165         11997
```

Part B:

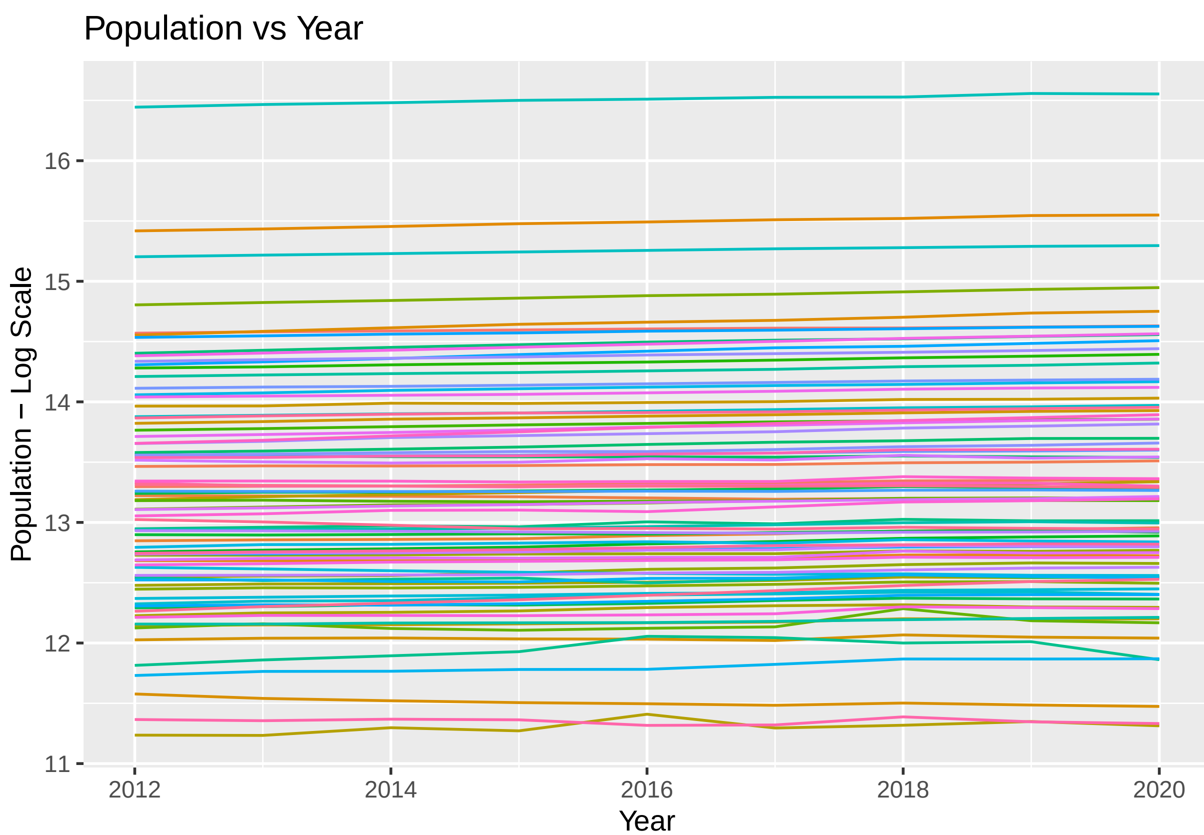
Arrange the data by city.

```
population %<>%
  arrange(City)
```

Part C:

Plot Year vs Population line graphs for each City.

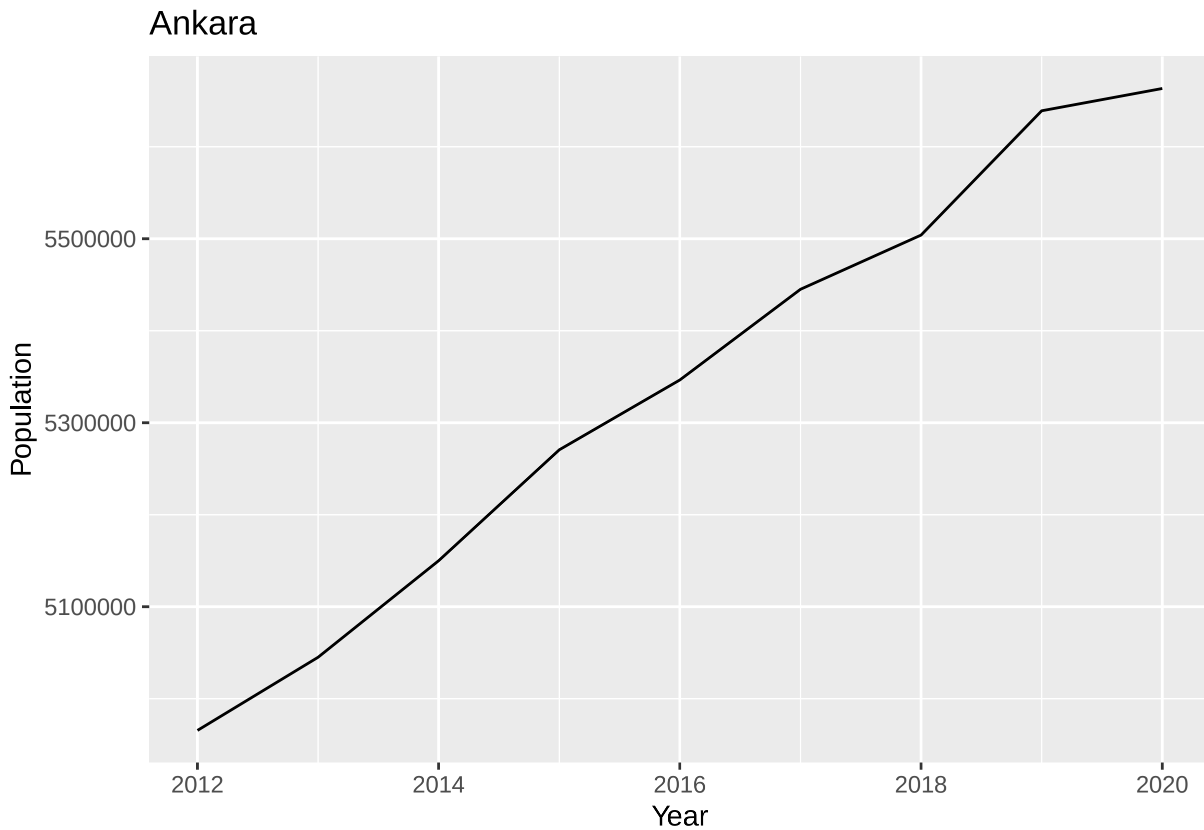
```
population %>%
  ggplot(aes(Year, log(Population), group = City)) +
  geom_line(aes(colour = City), linetype = 1) +
  labs(title = "Population vs Year", y = "Population - Log Scale") +
  theme(legend.position="none")
```



Part D:

Have a closer look at Ankara's population change over years.

```
population %>%
  filter(City == "ANKARA") %>%
  ggplot(aes(Year, Population)) +
  geom_line() +
  ggtitle("Ankara")
```



Part F:

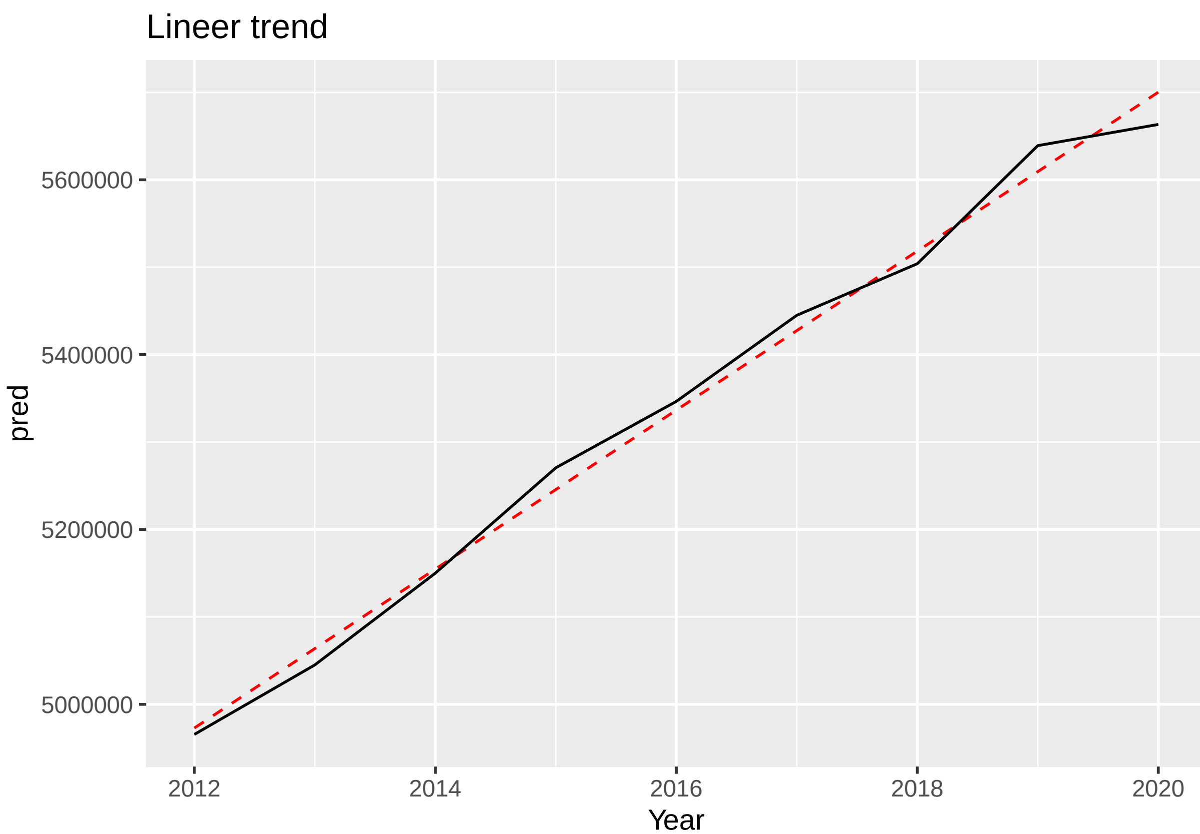
Try to find a linear trend line for Ankara. Then, plot the estimated linear line and real line in a single plot.

```
Ankara <- population %>% filter(City == "ANKARA")

fit_Ankara <- lm(Population ~ Year, data = Ankara)
summary(fit_Ankara)
```

```
##
## Call:
## lm(formula = Population ~ Year, data = Ankara)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -36947 -14439  -4660  17525  29730
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -177964056    6151678  -28.93 1.52e-08 ***
## Year             90923       3051   29.80 1.24e-08 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23640 on 7 degrees of freedom
## Multiple R-squared:  0.9922, Adjusted R-squared:  0.9911
## F-statistic: 887.9 on 1 and 7 DF,  p-value: 1.236e-08
```

```
Ankara %>%
  add_predictions(fit_Ankara) %>%
  ggplot(aes(Year, pred)) +
  geom_line(col = "Red", linetype = 2) +
  ggtitle("Lineer trend ") -> Ankara_trend_plot

Ankara_trend_plot +
  geom_line(aes(Year, Population), data = Ankara)
```



Part G:

Nest the data set wrt. city variable.

```
nested_city <- population %>%
  group_by(City) %>%
```

```
  nest()
```

Part H:

Now, generate a general linear model function.

```
city_model <- function(df) {
  lm(Population ~ Year, data = df)
}
```

Part I:

Apply previously generated model function to each city using 'map' function, then append models to nested tibble version.

```
nested_city %<>%
  mutate(model = map(data,city_model))
nested_city
```

```
## # A tibble: 81 x 3
## # Groups:   City [81]
##    City           data              model
##    <chr>          <list>            <list>
##  1 ADANA          <tibble [9 x 5]> <lm>
##  2 ADIYAMAN       <tibble [9 x 5]> <lm>
##  3 AFYONKARAHISAR <tibble [9 x 5]> <lm>
##  4 AGRI           <tibble [9 x 5]> <lm>
##  5 AKSARAY        <tibble [9 x 5]> <lm>
##  6 AMASYA         <tibble [9 x 5]> <lm>
##  7 ANKARA         <tibble [9 x 5]> <lm>
##  8 ANTALYA        <tibble [9 x 5]> <lm>
##  9 ARDAHAN        <tibble [9 x 5]> <lm>
## 10 ARTVIN         <tibble [9 x 5]> <lm>
## # ... with 71 more rows
```

Part J:

Find the residuals for each country

```
nested_city %<>%
  mutate(resids = map2(data, model, add_residuals))
nested_city
```

```
## # A tibble: 81 x 4
## # Groups:   City [81]
##    City           data              model  resids
```

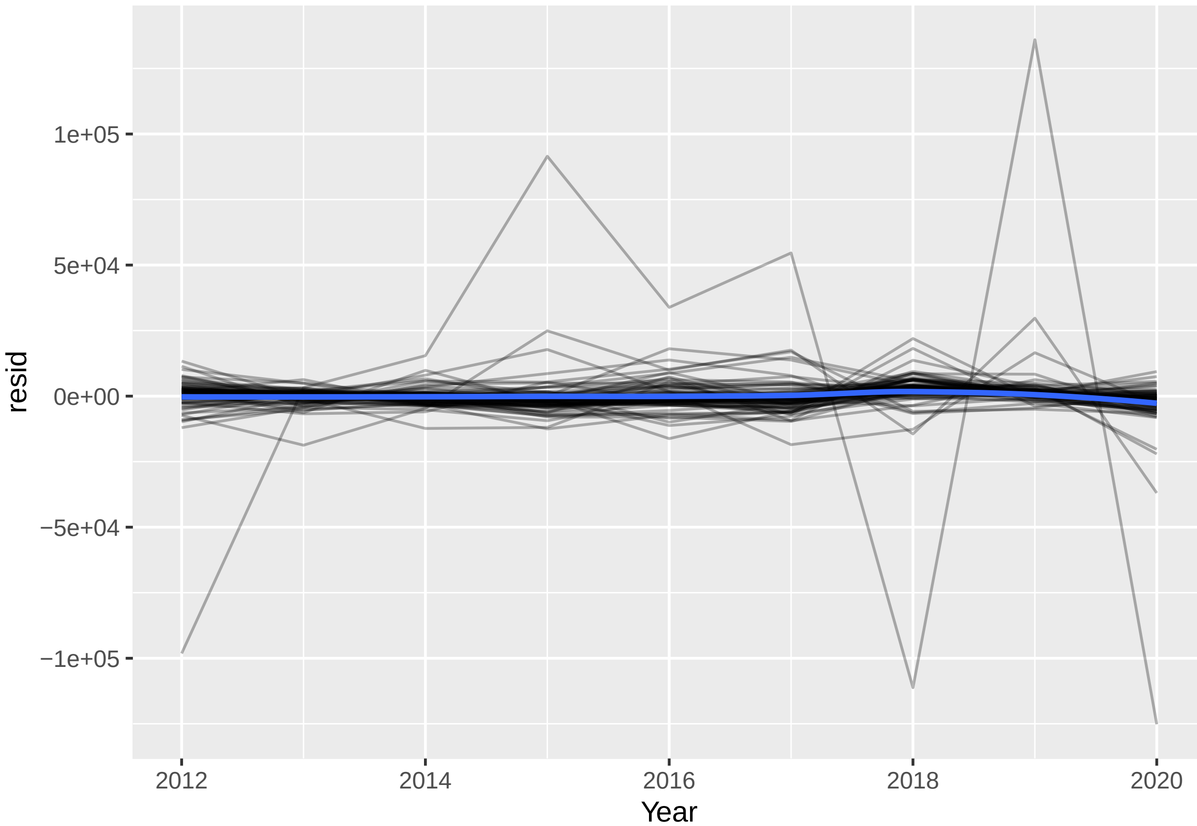
```
##    <chr>          <list>            <list> <list>
##  1 ADANA          <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  2 ADIYAMAN       <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  3 AFYONKARAHISAR <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  4 AGRI           <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  5 AKSARAY        <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  6 AMASYA         <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  7 ANKARA         <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  8 ANTALYA        <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
##  9 ARDAHAN        <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
## 10 ARTVIN         <tibble [9 x 5]> <lm>   <tibble [9 x 6]>
## # ... with 71 more rows
```

Part J:

Unnest the data and check out the shape of the distributions of residuals for each city.

```
nested_city %>%
  unnest(.,resids) %>%
  ggplot(aes(Year, resid)) +
  geom_line(aes(group = City), alpha = 0.3) +
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Part K:

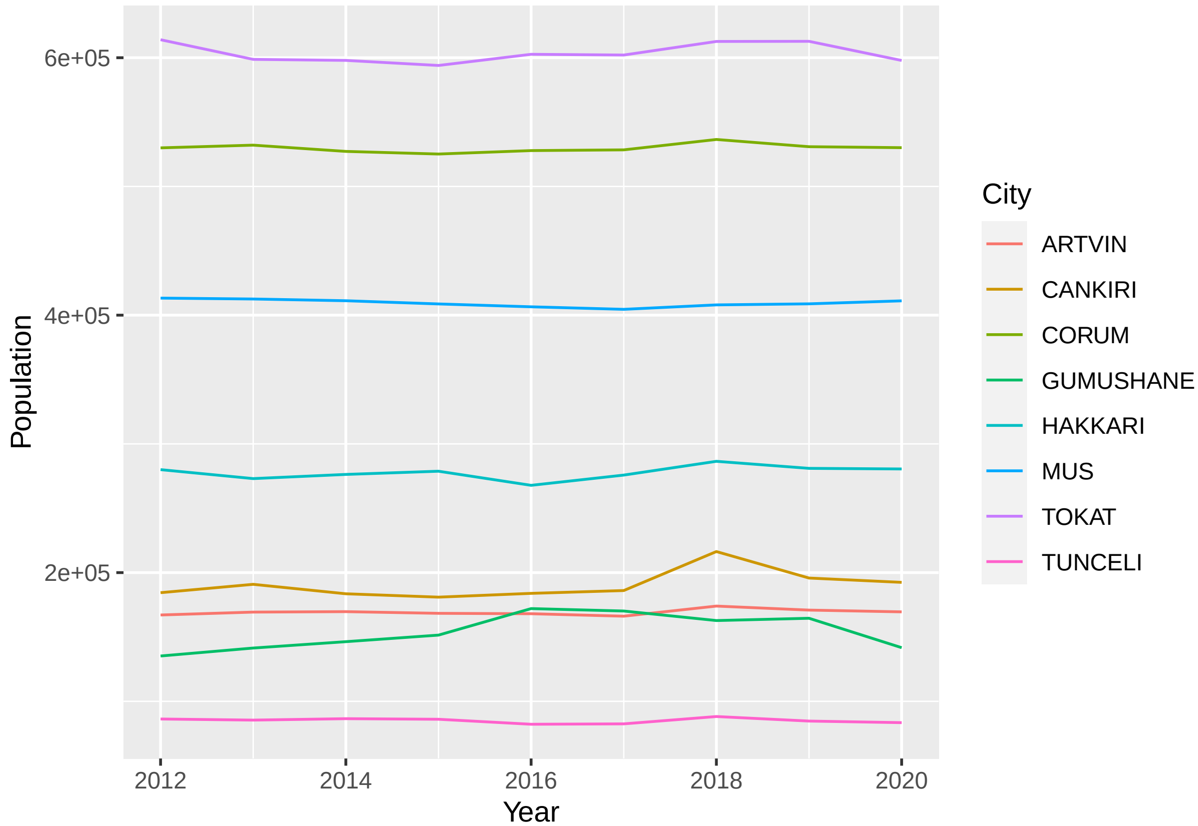
Check out the model indicators for each city, using 'glance' function from 'broom' package.

```
nested_city %>%
  mutate(glance = map(model, broom::glance)) %>%
  unnest(glance) -> glance_city
```

Part L:

Find the cities whose R-squared values are less than 0.25, and plot the population change over years for those cities.

```
low_r2 <- glance_city %>% filter(r.squared < 0.25)

population %>%
  semi_join(low_r2, by = "City") %>%
  ggplot(aes(Year, Population, color = City)) +
  geom_line()
```



6e+05
4e+05
2e+05
Population
2012
2014
2016
2018
2020
Year
City
ARTVIN
CANKIRI
CORUM
GUMUSHANE
HAKKARI
MUS
TOKAT
TUNCELI