Descriptive Statistic, ggplot2, and a Little Bit of LM

Cecilia Y. Sui, and all other TAs

Contents

1 Descriptive Statistics

1.1 Name Commands

- names() – It works on matrix or data frame objects.

```
# Load the data
worldTFR <- read.csv("./worldTFR.csv")

names(worldTFR)
```

```
##  [1] "Country"       "Uncode"        "Year"          "TFR"
##  [5] "InfMRateCME"   "InfMRateUN"    "U5MRateCME"    "U5MRateUN"
##  [9] "LifeExpB"      "MtoFbirth"     "MtoF04"        "Pop1564"
## [13] "Pop1564Female" "GDPpc"         "GDPpcGrowth"   "Yschooling"
## [17] "YschoolF1549"  "GenrollPrim"   "ChildBearing"  "CountryCode"
```

- rownames() – It works on matrix or data frame objects and is used to give names to rows.

```
rownames(worldTFR)
```

```
##     [1] "1"     "2"     "3"     "4"     "5"     "6"     "7"     "8"     "9"
```

```
##    [10] "10"    "11"    "12"    "13"    "14"    "15"    "16"    "17"    "18"
##    [19] "19"    "20"    "21"    "22"    "23"    "24"    "25"    "26"    "27"
##    [28] "28"    "29"    "30"    "31"    "32"    "33"    "34"    "35"    "36"
##    [37] "37"    "38"    "39"    "40"    "41"    "42"    "43"    "44"    "45"
##    [46] "46"    "47"    "48"    "49"    "50"    "51"    "52"    "53"    "54"
##    [55] "55"    "56"    "57"    "58"    "59"    "60"    "61"    "62"    "63"
##    [64] "64"    "65"    "66"    "67"    "68"    "69"    "70"    "71"    "72"
##    [73] "73"    "74"    "75"    "76"    "77"    "78"    "79"    "80"    "81"
##    [82] "82"    "83"    "84"    "85"    "86"    "87"    "88"    "89"    "90"
...
```

- colnames() – It works on matrix or data frame objects and is used to give names to columns.

```
colnames(worldTFR)
```

```
##  [1] "Country"       "Uncode"        "Year"          "TFR"
##  [5] "InfMRateCME"   "InfMRateUN"    "U5MRateCME"    "U5MRateUN"
##  [9] "LifeExpB"      "MtoFbirth"     "MtoF04"        "Pop1564"
## [13] "Pop1564Female" "GDPpc"         "GDPpcGrowth"   "Yschooling"
## [17] "YschoolF1549"  "GenrollPrim"   "ChildBearing"  "CountryCode"
```

- dimnames() – Gets row and column names for matrix or data frame objects, that is, it is used to see dimensions of the data frame.

```
dimnames(worldTFR)[2]
```

```
## [[1]]
##  [1] "Country"       "Uncode"        "Year"          "TFR"
##  [5] "InfMRateCME"   "InfMRateUN"    "U5MRateCME"    "U5MRateUN"
##  [9] "LifeExpB"      "MtoFbirth"     "MtoF04"        "Pop1564"
## [13] "Pop1564Female" "GDPpc"         "GDPpcGrowth"   "Yschooling"
...
```

```
dim(worldTFR)
```

```
## [1] 12342    20
```

```
dimnames(worldTFR)[2]
```

```
## [[1]]
##  [1] "Country"       "Uncode"        "Year"          "TFR"
##  [5] "InfMRateCME"   "InfMRateUN"    "U5MRateCME"    "U5MRateUN"
##  [9] "LifeExpB"      "MtoFbirth"     "MtoF04"        "Pop1564"
## [13] "Pop1564Female" "GDPpc"         "GDPpcGrowth"   "Yschooling"
## [17] "YschoolF1549"  "GenrollPrim"   "ChildBearing"  "CountryCode"
```

1.2 Summarizing Commands

- max(x, na.rm = FALSE) – It shows the maximum value. By default, NA values are not removed. NA is considered the largest unless na.rm=true is used.
- min(x, na.rm = FALSE) – Shows minimum value in a vector. If there are na values, NA is returned unless na.rm=true is used.
- length(x) – Gives length of the vector and includes na values. Na.rm=instruction does not work with this command.
- sum(x, na.rm = FALSE) – Shows the sum of the vector elements.
- mean(x, na.rm = FALSE) – We obtain an arithmetic mean with this.

- median(x, na.rm = FALSE) – Shows the median value of the vector.
- sd(x, na.rm = FALSE) – Shows the standard deviation.
- var(x, na.rm = FALSE) – Shows the variance.
- mad(x, na.rm = FALSE) – Shows the median absolute deviation. This first finds the median of the data set, then computes the absolute differences between each data point and the median, and finally takes the median of these absolute differences.
- log(dataset) – Shows log value for each element.
- summary(dataset) – We have seen how it shows a summary of dataset like maximum value, minimum value, mean, etc.
- quantile() – Shows the quantiles by default—the 0%, 25%, 50%, 75%, and 100% quantiles. You can select other quantiles also.

```
x <- sample(1:100000, 2000)
quantile(x, probs = seq(0, 1, 0.1), na.rm = FALSE, names = TRUE)
```

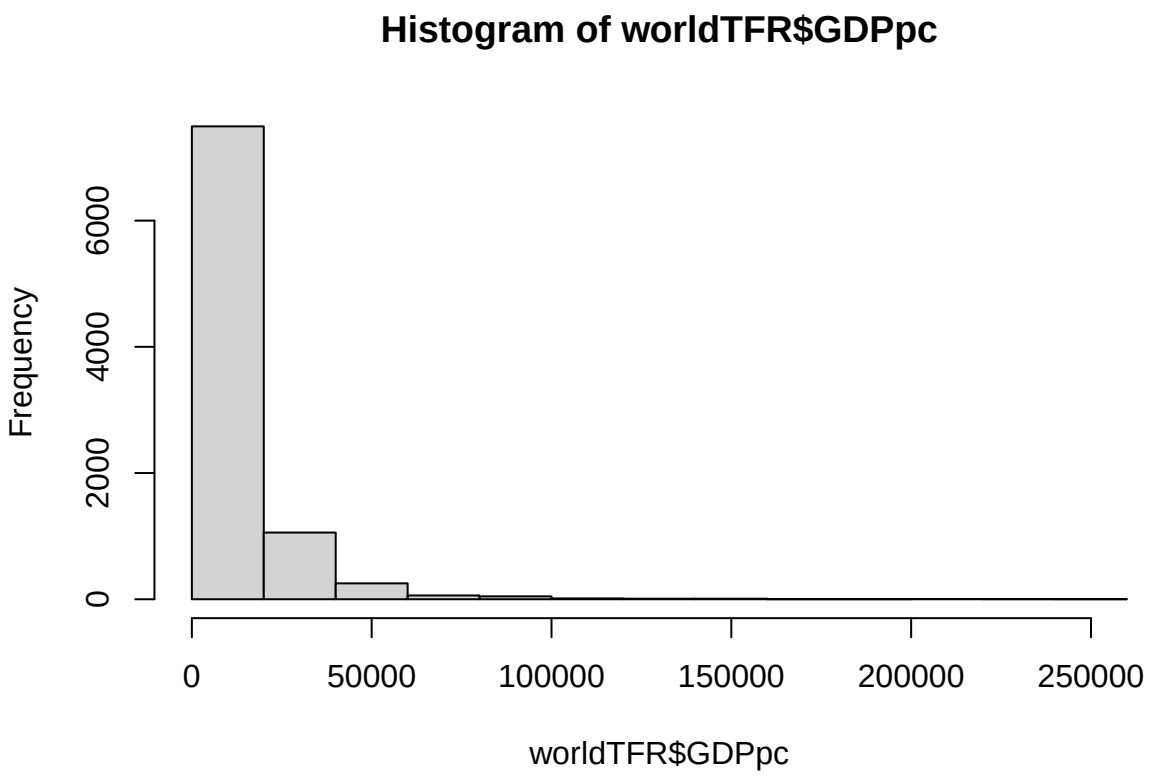
```
##      0%     10%     20%     30%     40%     50%     60%     70%     80%     90%
##     1.0  9135.4 18368.8 27312.4 39128.8 48567.0 57404.0 67926.0 77770.6 89214.7
##    100%
## 99954.0
```

```
?quantile
```

1.3 Histograms and Scatter Plots

One helpful graph in describing the data is histograms. R provides a very quick function to draw histogram: `hist()`

```
hist(worldTFR$GDPpc)
```



As we have seen in

1.4 In-class exercises:

We will continue using the worldTFR dataset.

1. Load the dataset and omit all NA's in the dataframe.
2. What are the column names and row names?
3. What is the max and min of Years?
4. What is the average Life Expectancy at Birth (LifeExpB)?
5. What is the median Life Expectancy at Birth (LifeExpB)?
6. What is the 25% and 75% quantile of Life Expectancy at Birth (LifeExpB)?

2 Introduction to ggplot2

`ggplot2` is a powerful visualization package and is one of the areas where R truly shines. It is so highly regarded that many researchers use `ggplot2` to create publication-quality figures even if they perform their primary data analysis in other languages like Python. As a core component of the `tidyverse`(an ecosystem of packages that includes `dplyr`), `ggplot2` shares a consistent design philosophy. This creates a natural synergy between the two, allowing them to work together seamlessly in a single, fluid workflow.

We need to first install and load the package.

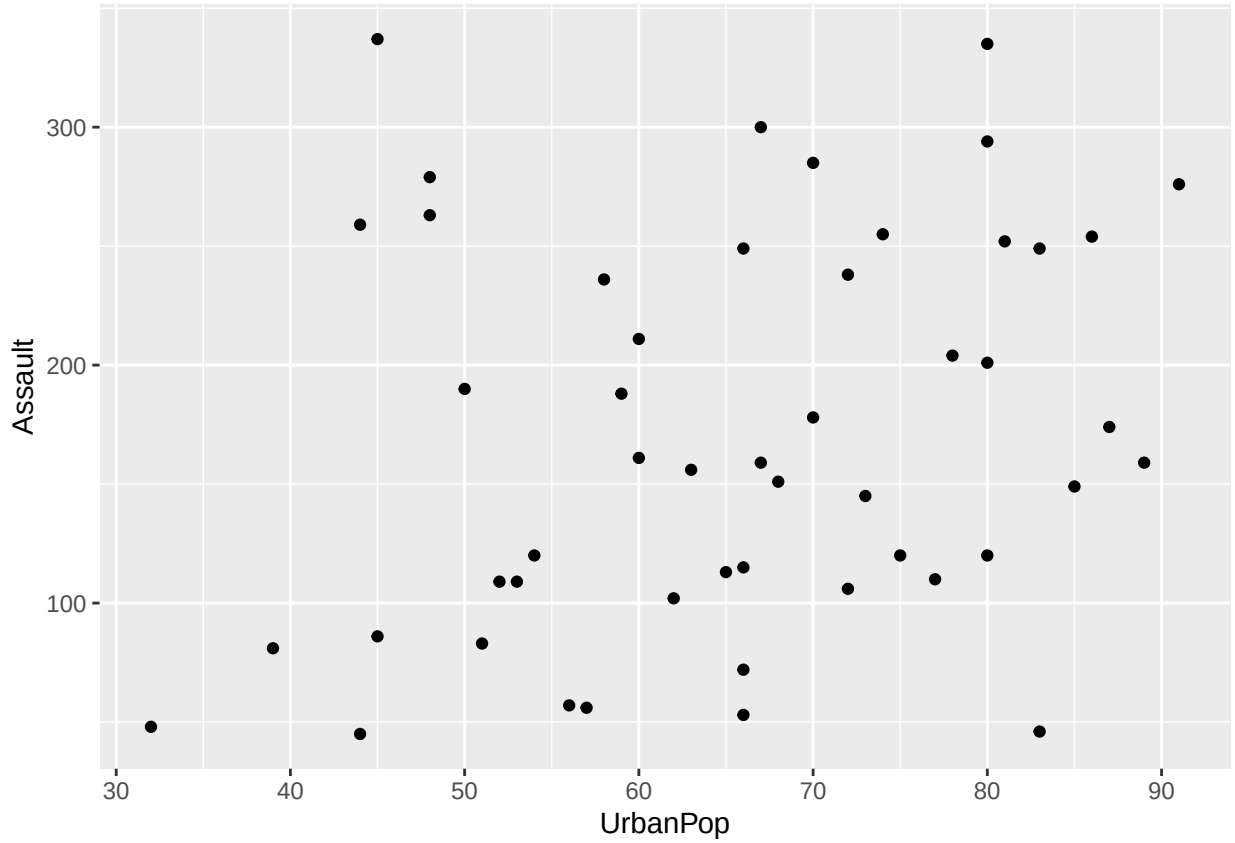
```
# install.packages("ggplot2")
library(ggplot2)
```

2.1 How to Make a Simple Scatterplot

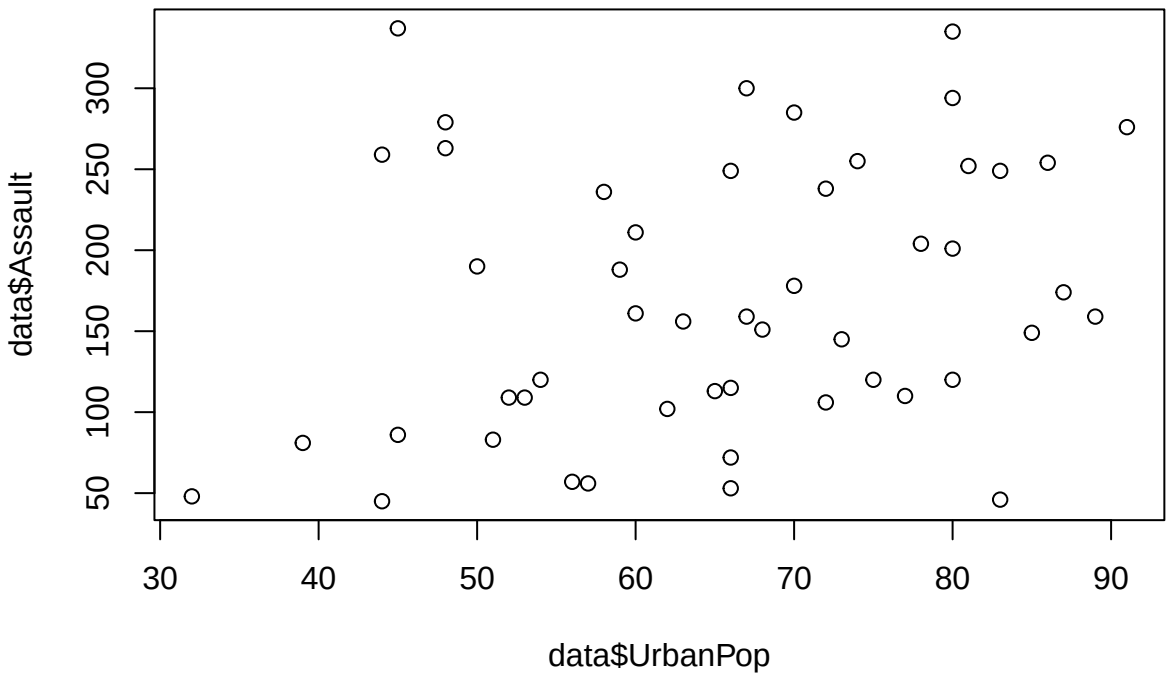
```
data = USArrests # built-in dataset
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```
# ?USArrests

ggplot(data, aes(x = UrbanPop, y = Assault)) + geom_point()
```




```
# sort of equivalent to doing this with the base R plot
plot(data$UrbanPop, data$Assault)
```



We got a basic scatterplot, where each point represents a US state. However, it lacks some basic components such as the plot title, meaningful axis labels, etc.

As mentioned, we can chain `ggplot` with `dplyr`:

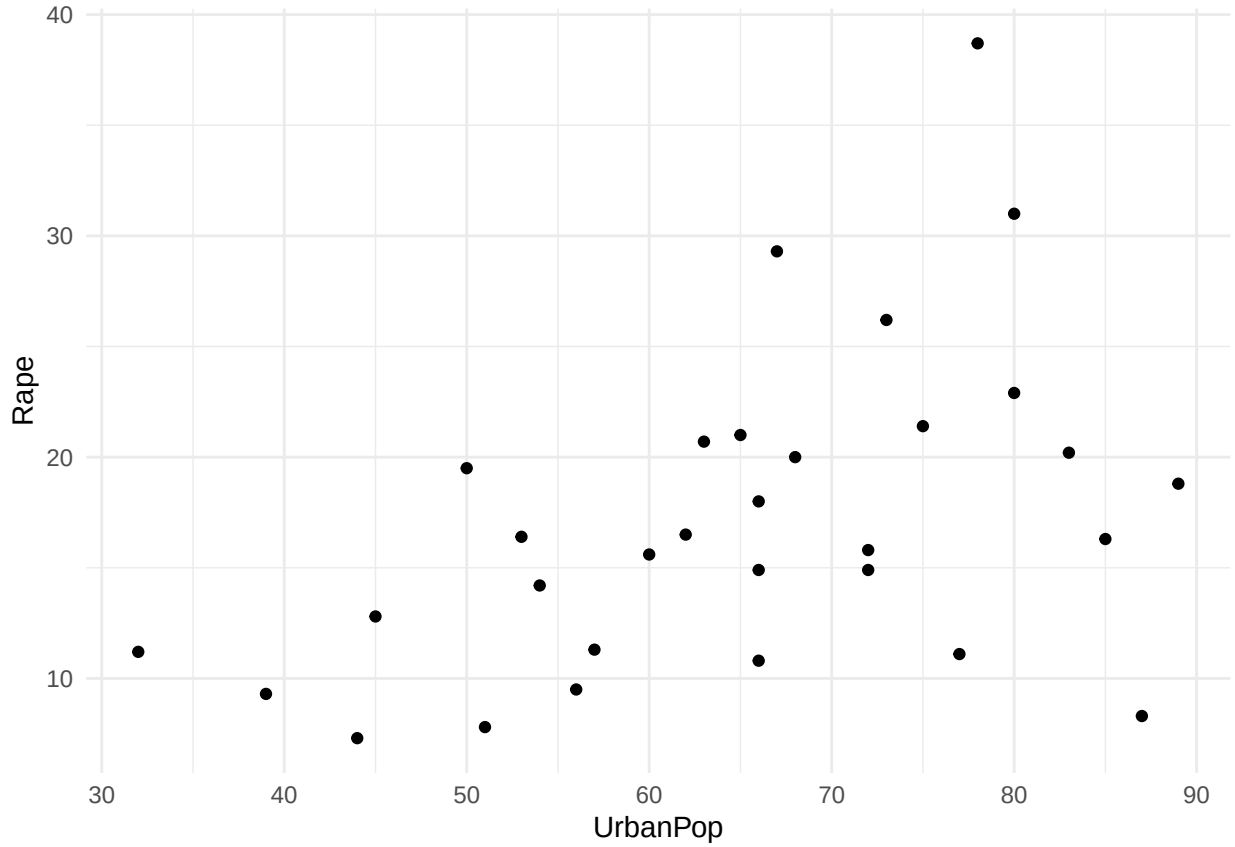
```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

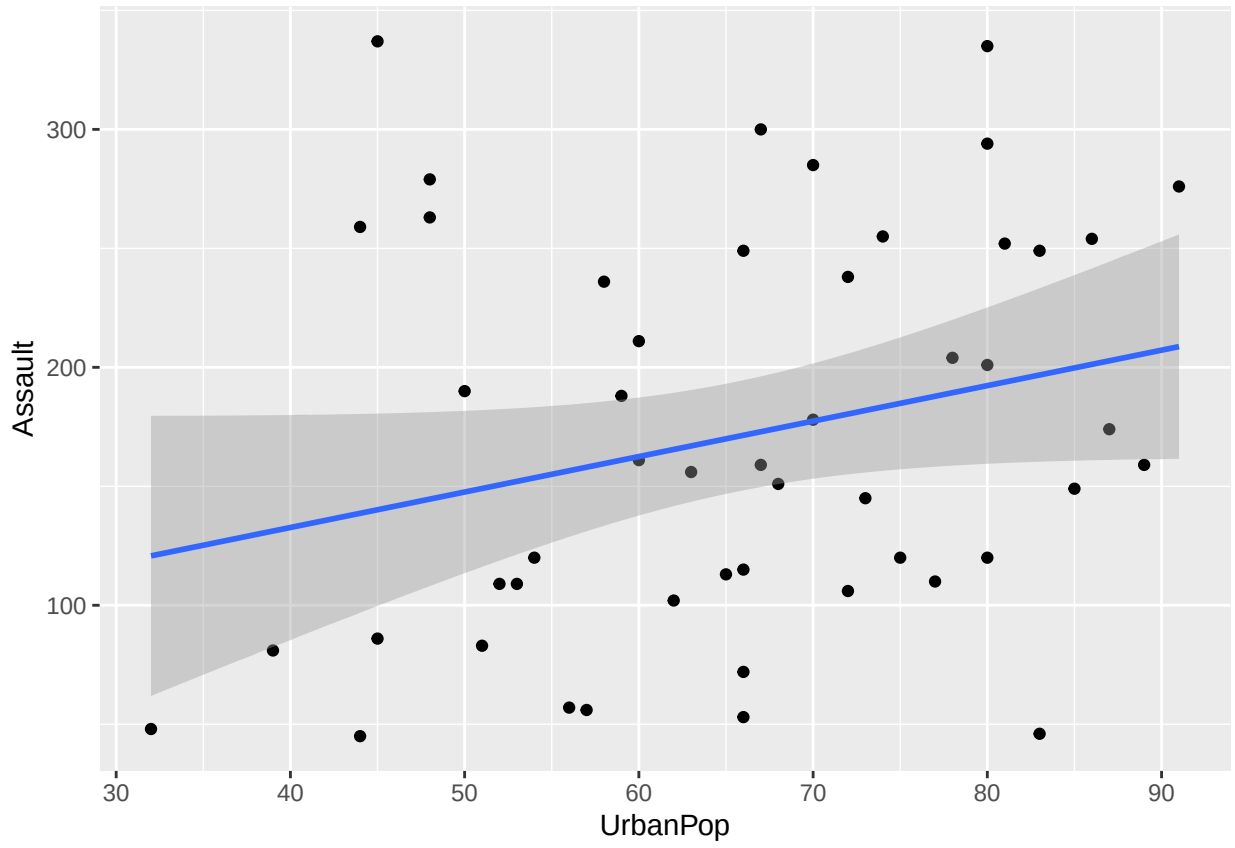
```
USArrests %>%
  group_by(row.names(USArrests)) %>%
  filter(Murder < 9) %>%
  ggplot() +
  geom_point(aes(x = UrbanPop, y = Rape)) +
  theme_minimal()
```



Like `geom_point()`, there are many such geom layers to use for visualization. For now, let's just add a smoothing layer using **geom_smooth(method = 'lm')**. Since the method is set as lm (short for linear model), it draws the line of best fit. The line of best fit is in blue by default. The shaded area is the confidence intervals.

```
ggplot(data, aes(x=UrbanPop, y=Assault)) +
  geom_point() +
  geom_smooth(method="lm", se = TRUE)
```

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



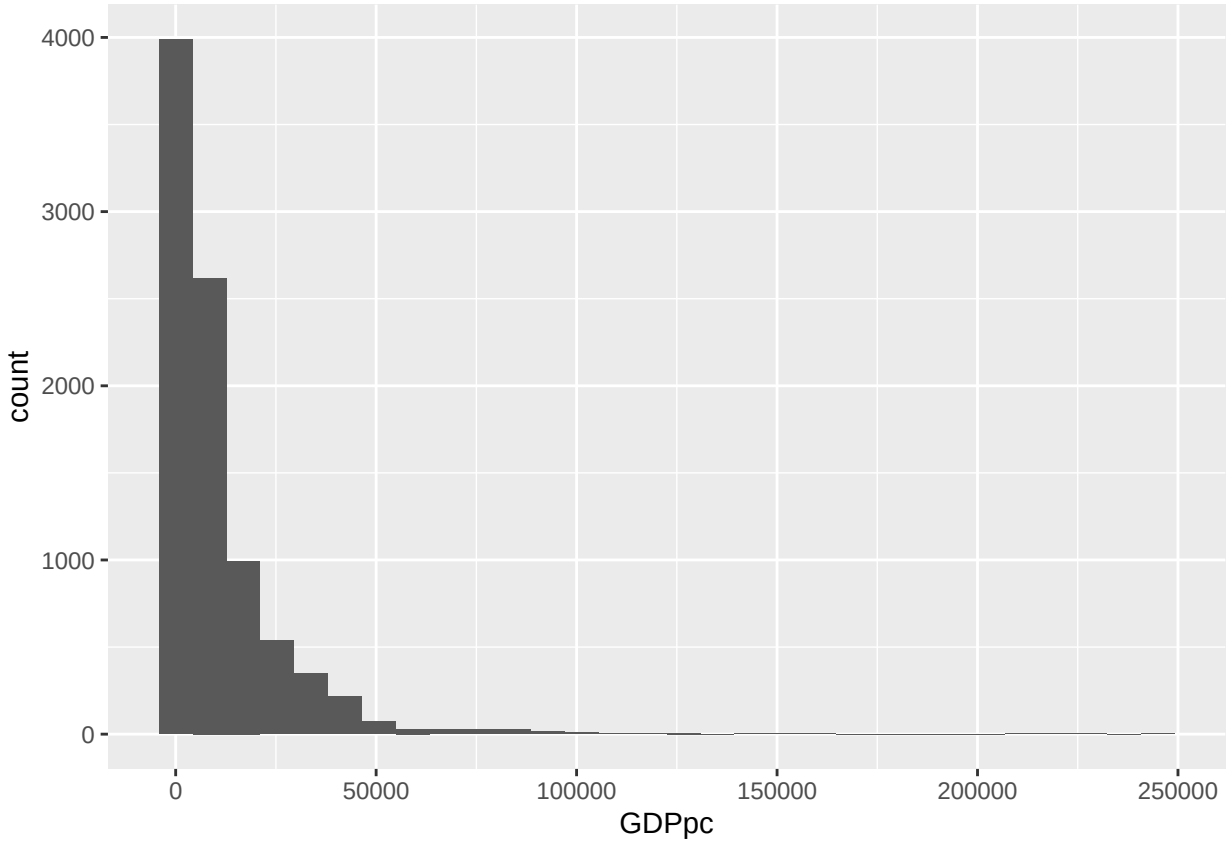
2.2 Histograms with ggplot

We can use geom_histogram.

```
ggplot(data = worldTFR) +
  geom_histogram(aes(GDPpc))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 3389 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



2.3 Adjusting the X and Y Axis Limits

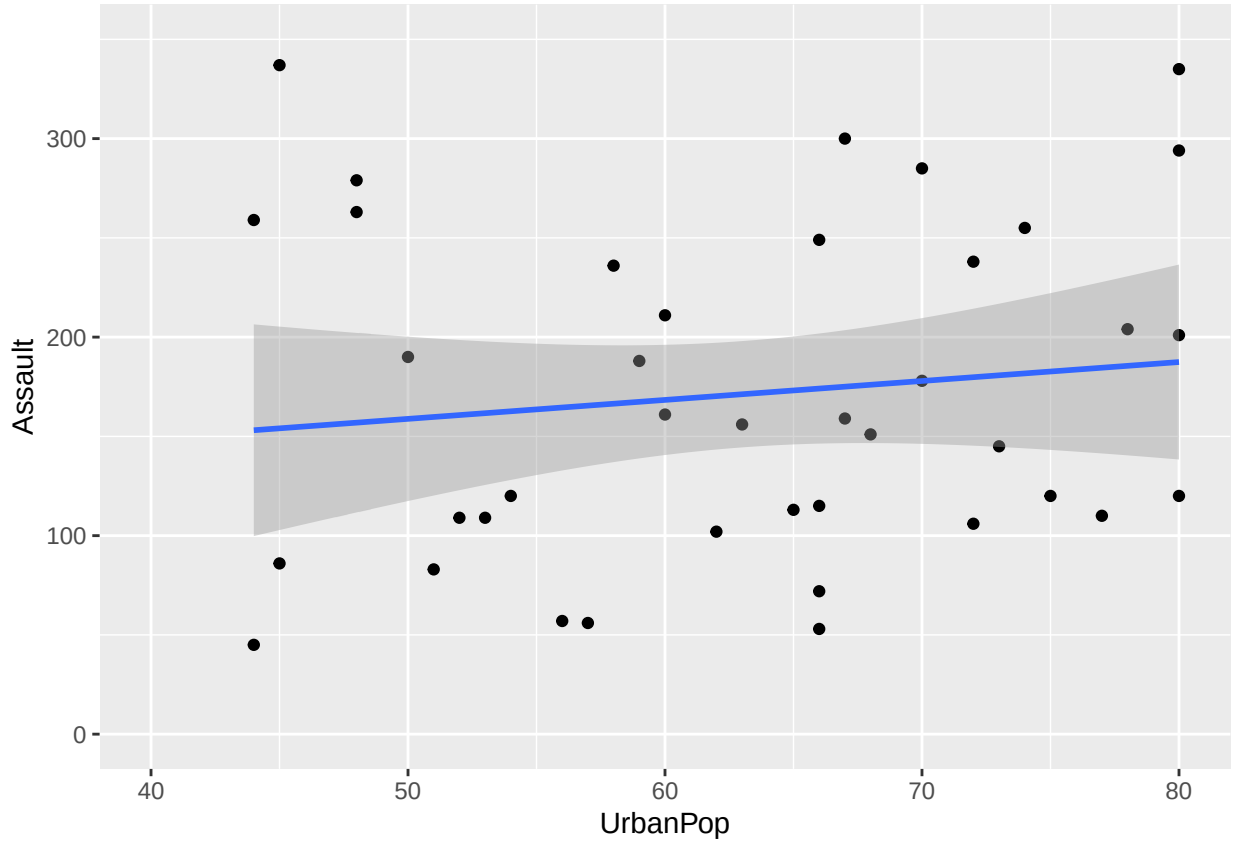
We can delete the points outside our range. This will change the lines of best fit or smoothing lines as compared to the original data. This can be done by **xlim()** and **ylim()**. You can pass a numeric vector of length 2 (with min and max values) or just the max and min values itself.

```
ggplot(data, aes(x=UrbanPop, y=Assault)) +
  geom_point() +
  geom_smooth(method="lm")  +
  xlim(c(40, 80)) + ylim(c(0, 350))   # deletes points
```

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

```
## Warning: Removed 10 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 10 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



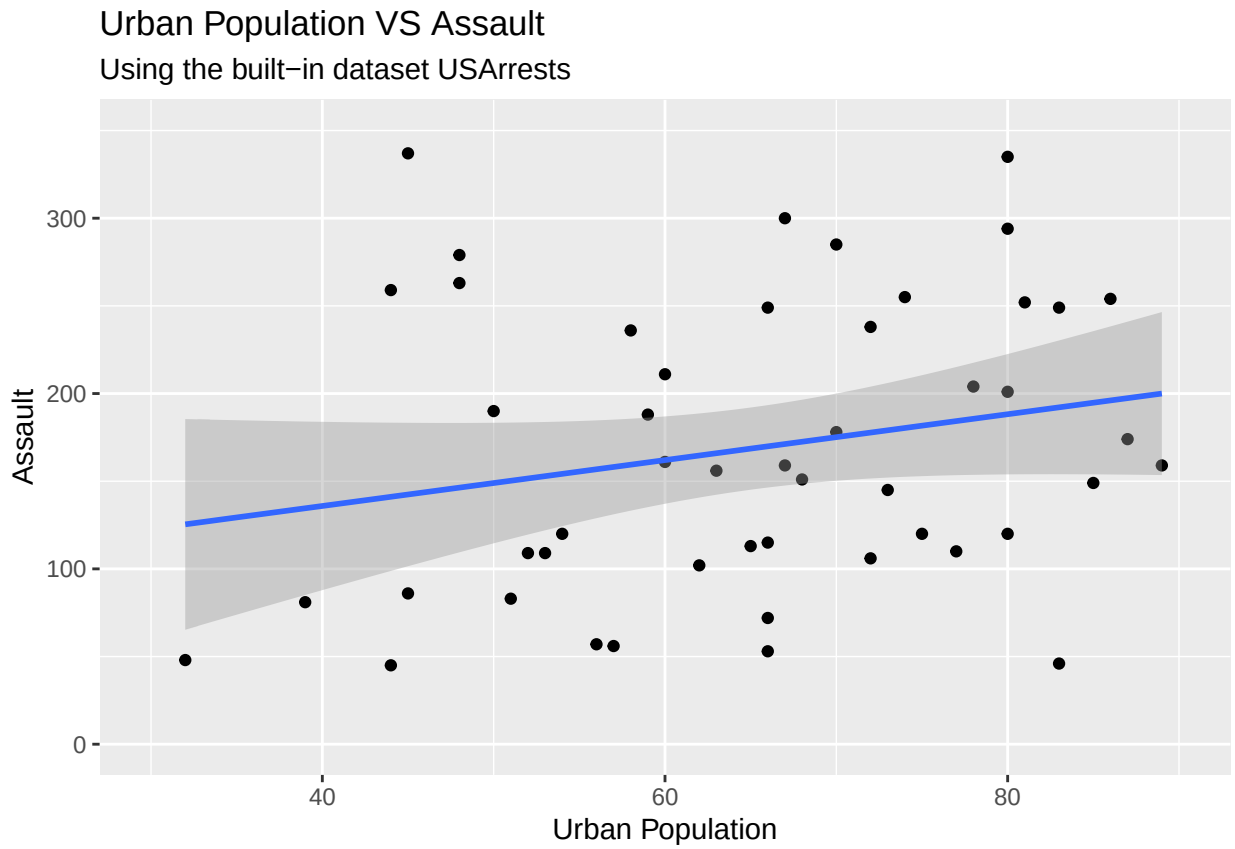
2.4 Change the Title and Axis Labels

```
ggplot(data, aes(x=UrbanPop, y=Assault)) +
  geom_point() +
  geom_smooth(method="lm") +
  xlim(c(30, 90)) + ylim(c(0, 350)) +
  labs(title="Urban Population VS Assault",
         subtitle="Using the built-in dataset USArrests",
       y="Assault", x="Urban Population") # adding labels here
```

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

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



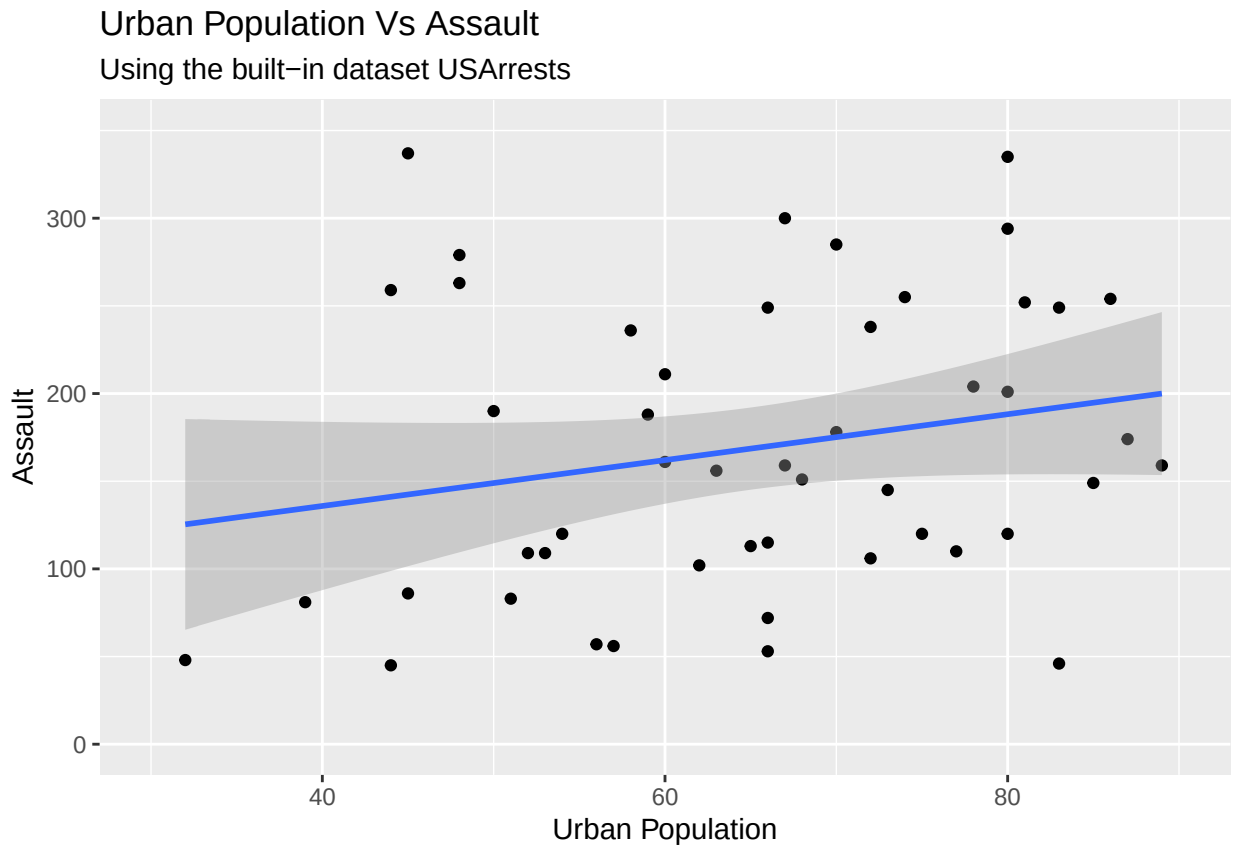
You can also get the same plot by adding the labels using an alternative way:

```
ggplot(data, aes(x=UrbanPop, y=Assault)) +
  geom_point() +
  geom_smooth(method="lm") +
  xlim(c(30, 90)) + ylim(c(0, 350)) +
  ggtitle("Urban Population Vs Assault",
            subtitle="Using the built-in dataset USArrests") +
  xlab("Urban Population") +
  ylab("Assault")
```

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

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



2.5 Change the Color and Size of Points

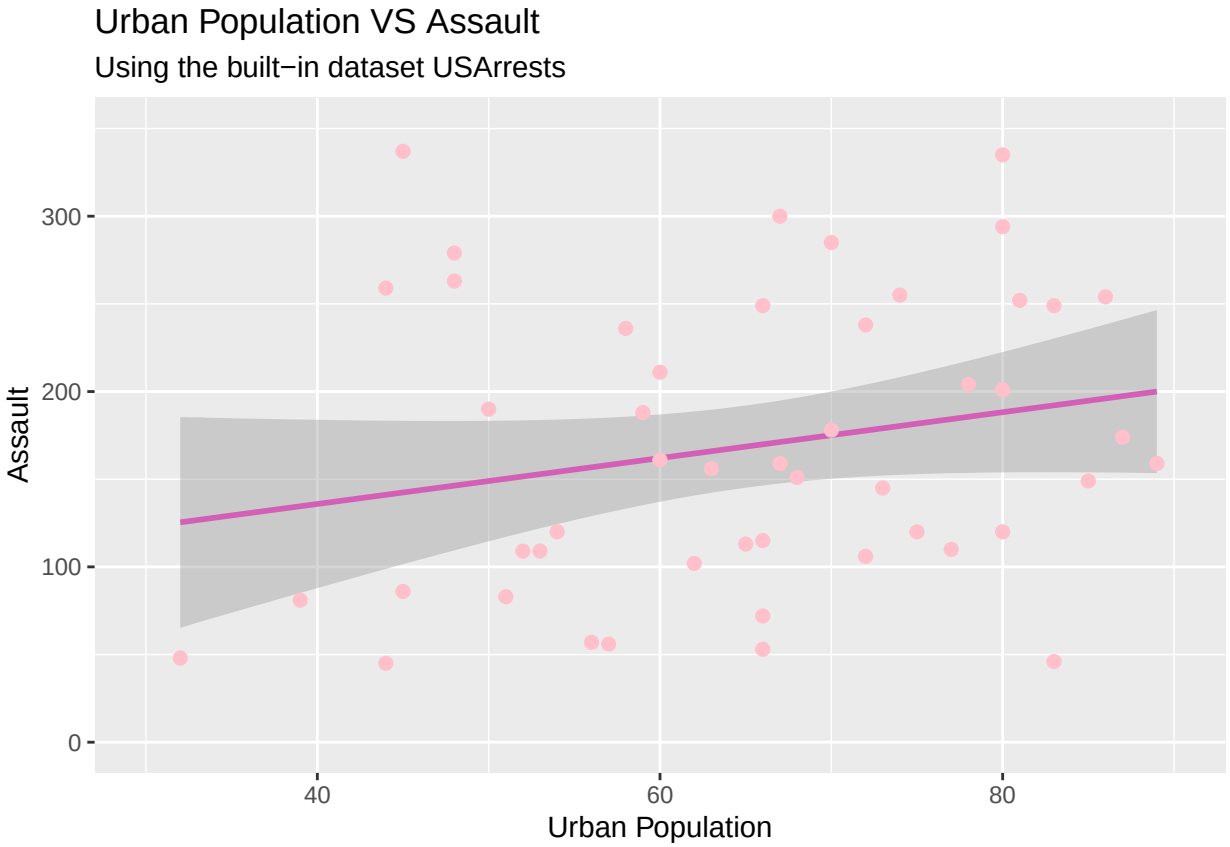
We can change the aesthetics of a geom layer by modifying the respective **geoms**. Let's change the color of the points and the line to a static value. R has a limited number of colors to use by their name, such as red, blue, yellow, etc. But you can always use html color codes to make your plots more colorful!

```
ggplot(data, aes(x=UrbanPop, y=Assault)) +
  geom_smooth(method="lm",  col="#D35FB7") + # change the color of line
  xlim(c(30, 90)) + ylim(c(0, 350)) +
  labs(title="Urban Population VS Assault",
         subtitle="Using the built-in dataset USArrests",
       y="Assault", x="Urban Population") +
  geom_point(col="pink", size=2) # Set static color and size for points
```

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

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



ggplot2 is a very powerful package for visualizing your data.

If you would like to learn more about what it is capable to do, check out this website:

http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html

2.6 Saving your plot

ggsave() is a convenient function for saving a plot.

It defaults to saving the last plot that you displayed, using the size of the current graphics device. It also guesses the type of graphics device from the extension.

It has many different arguments that you can customize. We will not cover how to use each argument, but you can learn more about them in the documentation for ggplot.

```
# The following code chunk is not executable.
ggsave(
  filename,
  plot = last_plot(),
  device = NULL,
  path = NULL,
  scale = 1,
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  limitsize = TRUE,
  bg = NULL,
```

```
  ...
)
```

For example, you can save the plot above like this:

```
ggplot(data, aes(x=UrbanPop, y=Assault)) +
  geom_point() +
  geom_smooth(method="lm",  col="skyblue") +
  xlim(c(30, 90)) + ylim(c(0, 350)) +
  labs(title="Urban Population VS Assault",
         subtitle="Using the built-in dataset USArrests",
       y="Assault", x="Urban Population") +
  geom_point(col="pink", size=3)

ggsave("myplot.png")
```

When having multiple plots, you can also save your plots to objects and then export to images. For example:

```
g1 <- ggplot(data, aes(x=UrbanPop, y=Assault)) +
          geom_point() +
          geom_smooth(method="lm",  col="skyblue") +
          xlim(c(30, 90)) + ylim(c(0, 350)) +
          labs(title="Urban Population VS Assault",
                 subtitle="Using the built-in dataset USArrests",
               y="Assault", x="Urban Population") +
          geom_point(col="pink", size=3)

ggsave("myplot.pdf", plot = g1)
```

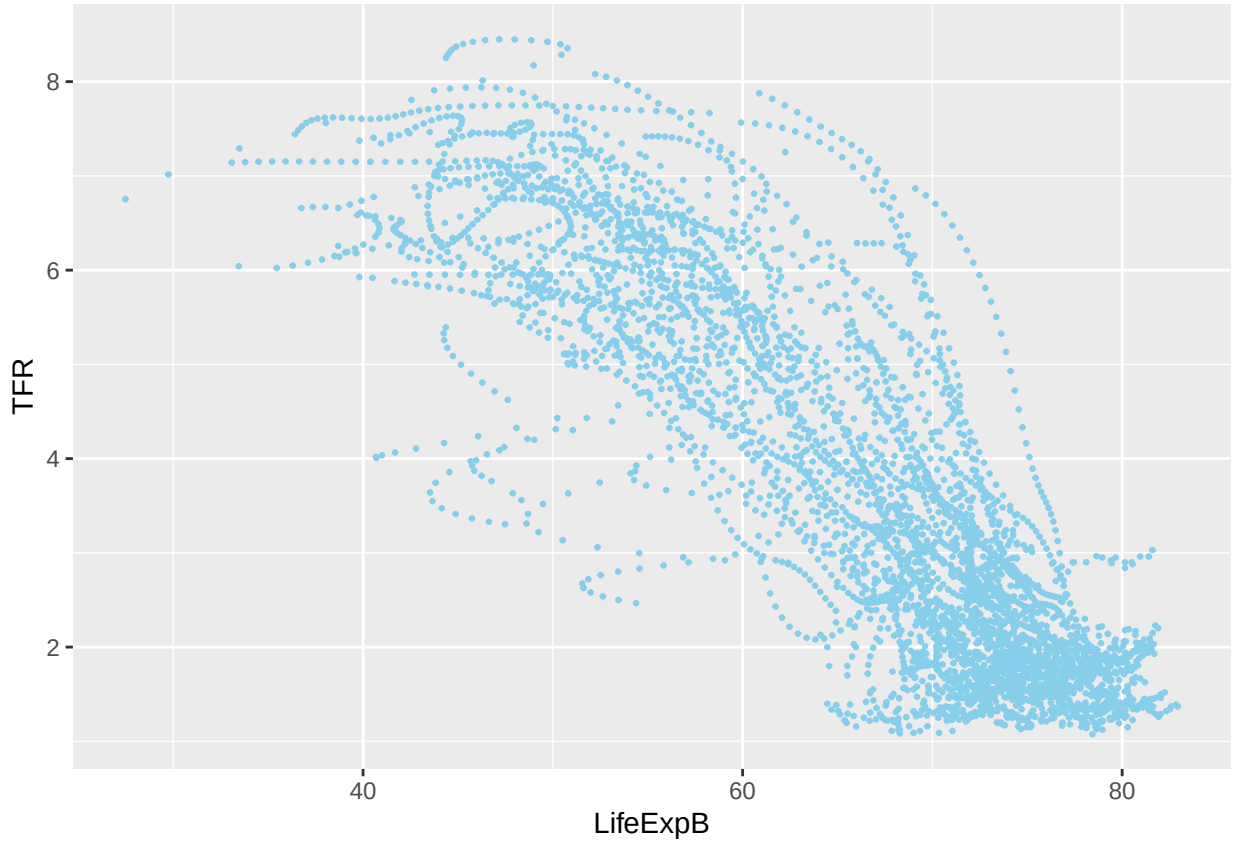
2.7 In-class exercises 5.3:

1. Omit all NA's from worldTFR.

```
df <- na.omit(worldTFR)
```

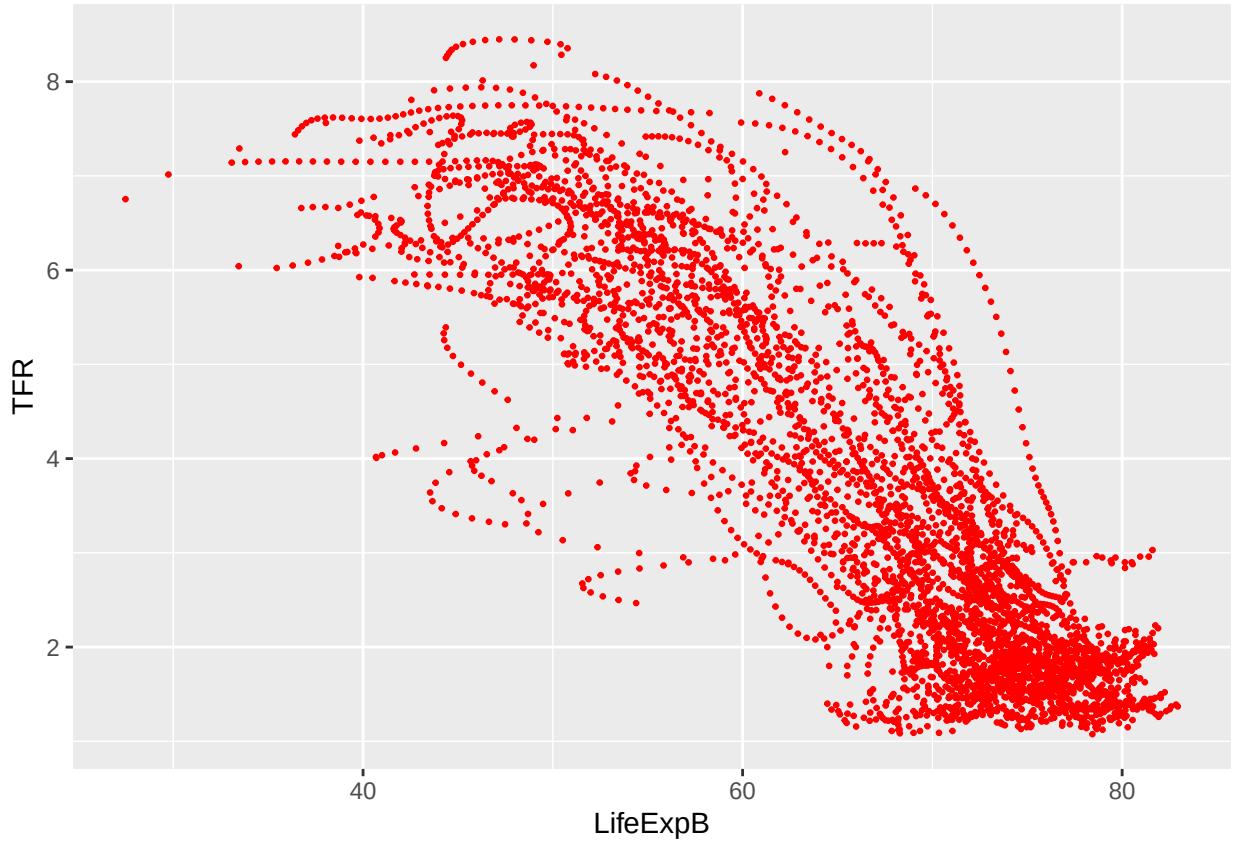
2. Make a plot where the horizontal axis is LifeExpB and vertical axis is TFR.

```
ggplot(df, aes(x = LifeExpB, y = TFR)) + geom_point(col = "skyblue", size = 0.5)
```



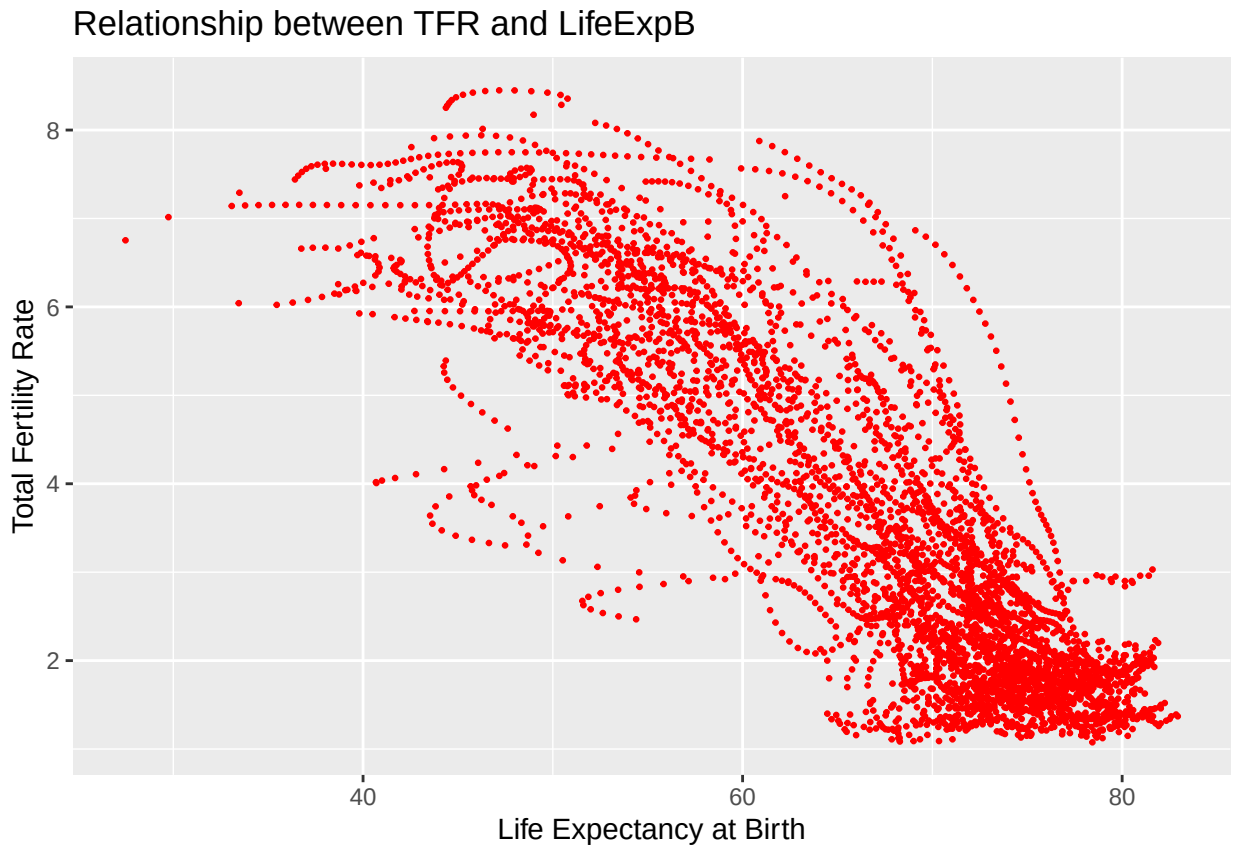
3. Set the color to red, and size for points as 0.5

```
ggplot(df, aes(x = LifeExpB, y = TFR)) + geom_point(col = "red", size = 0.5)
```



4. Add appropriate label for the axis and the plot.

```
ggplot(df, aes(x = LifeExpB, y = TFR)) +
  geom_point(col = "red", size = 0.5) +
  labs(title="Relationship between TFR and LifeExpB",
       y="Total Fertility Rate", x="Life Expectancy at Birth")
```



5. Create a new directory named plots, and save your plot as TFR.png into plots.

```
dir.create("plots")
```

```
## Warning in dir.create("plots"): 'plots' already exists
```

```
ggsave("TFR.png")
```

```
## Saving 6.5 x 4.5 in image
```

3 Simple Linear Regression

Foundations for linear regressions will be covered more in QPM 1. Here I want to briefly touch on how we can run `lm()` function. The `lm()` function mainly takes two arguments: `formula` and `data`.

3.1 `lm()`

`formula` has a form `y ~ x1 + x2` where `y` denotes the dependent variable and `x` denotes independent variable(s). `~` works like an equal sign. For interaction term, we can use `*` and `:`. `:` will add the interaction term only, but `*` will automatically add all main terms. Since it is advised to always include main terms in interaction analysis, I would stick to `*` without special reasons.

As always, we can store the result from `lm()` function as an object. `summary()` function will show the summary of the regression result in a readable way.

```
# Bivariate
model1 <- lm(formula = LifeExpB ~ Pop1564Female, data = worldTFR)
summary(model1)
```

```
##
## Call:
## lm(formula = LifeExpB ~ Pop1564Female, data = worldTFR)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.812  -4.450   1.450   6.448  24.292
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -14.60223    0.79659  -18.33   <2e-16 ***
## Pop1564Female   1.30197    0.01357   95.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.455 on 12340 degrees of freedom
## Multiple R-squared:  0.4271, Adjusted R-squared:  0.4271
## F-statistic:  9200 on 1 and 12340 DF,  p-value: < 2.2e-16
```

```
# Multivariate
model2 <- lm(formula = LifeExpB ~ Pop1564Female + InfMRateUN, data = worldTFR)
summary(model2)
```

```
##
## Call:
## lm(formula = LifeExpB ~ Pop1564Female + InfMRateUN, data = worldTFR)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.9679  -2.2160   0.0648   2.1313  18.6741
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   60.2622913  0.4290565  140.45   <2e-16 ***
## Pop1564Female  0.2503444  0.0067520   37.08   <2e-16 ***
## InfMRateUN    -0.1857975  0.0007215 -257.50   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.745 on 12339 degrees of freedom
## Multiple R-squared:  0.9101, Adjusted R-squared:  0.9101
## F-statistic: 6.247e+04 on 2 and 12339 DF,  p-value: < 2.2e-16
```

```
# Interaction
model3 <- lm(formula = LifeExpB ~ Pop1564Female*InfMRateUN, data = worldTFR)
summary(model3)
```

```
##
## Call:
## lm(formula = LifeExpB ~ Pop1564Female * InfMRateUN, data = worldTFR)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.9945  -2.2198   0.0637   2.1272  18.8226
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               59.6100993  0.6161871  96.740   <2e-16 ***
## Pop1564Female              0.2611471  0.0099626  26.213   <2e-16 ***
## InfMRateUN                -0.1730814  0.0086535 -20.001   <2e-16 ***
## Pop1564Female:InfMRateUN -0.0002246  0.0001523  -1.475     0.14
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.745 on 12338 degrees of freedom
## Multiple R-squared:  0.9101, Adjusted R-squared:  0.9101
## F-statistic: 4.165e+04 on 3 and 12338 DF,  p-value: < 2.2e-16
```

3.1.1 Robust Standard Errors

Robust standard errors are not the default. So we need another package called `sandwich` and `lmtest`. `sandwich` provides functions to calculate robust SE and `lmtest` provides functions to redisplay regression results with the new SE.

```
library(sandwich)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# Robust standard errors with lm()
model1_robust <- coeftest(model1,
                          vcov = vcovHC)

print(model1_robust) # summary may not work here since the result is contained in a different object.
```

```
##
## t test of coefficients:
##
```

```
##                 Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   -14.602227   0.696124 -20.977 < 2.2e-16 ***
## Pop1564Female   1.301971   0.011261 115.621 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

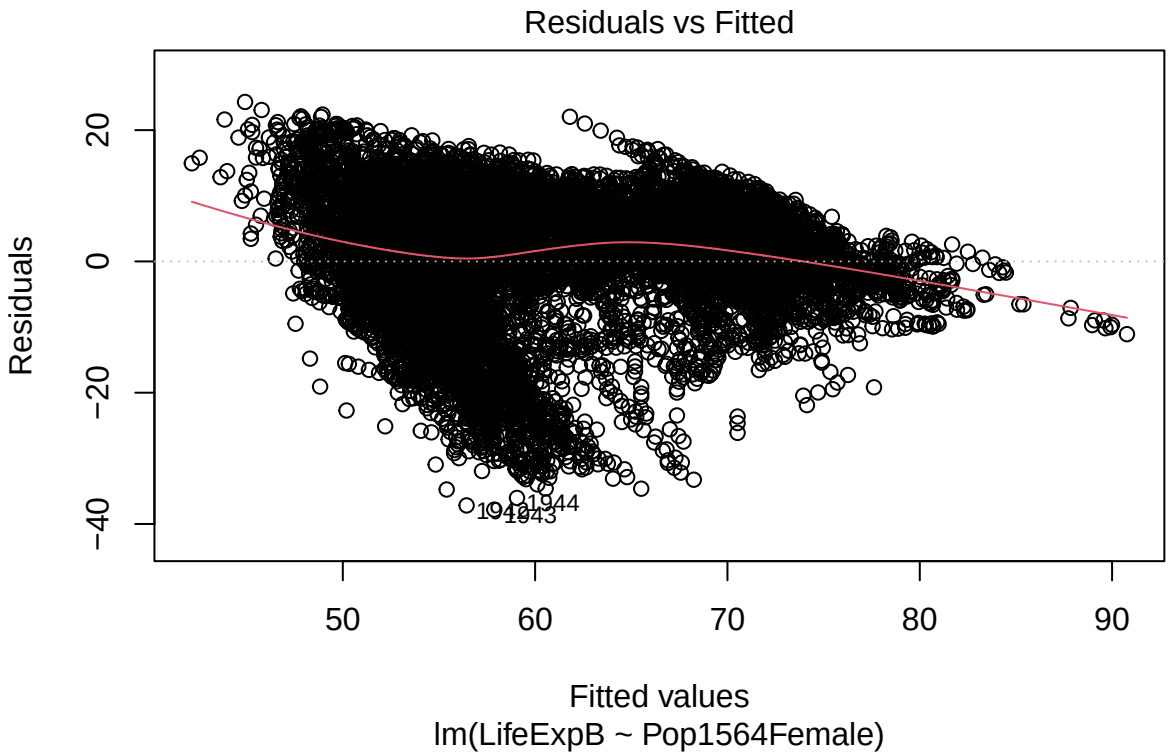
```
# Clustered robust standard errors with lm()
model1_robust_clustered <- coeftest(model1,
                                    vcov = vcovCL,
                                    type = "HC1",
                                    cluster = ~Country)

print(model1_robust_clustered) # summary may not work here since the result is contained in a different
```

```
##
## t test of coefficients:
##
##                 Estimate Std. Error t value  Pr(>|t|)
## (Intercept)   -14.602227   3.381878 -4.3178 1.588e-05 ***
## Pop1564Female   1.301971   0.054126 24.0544 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

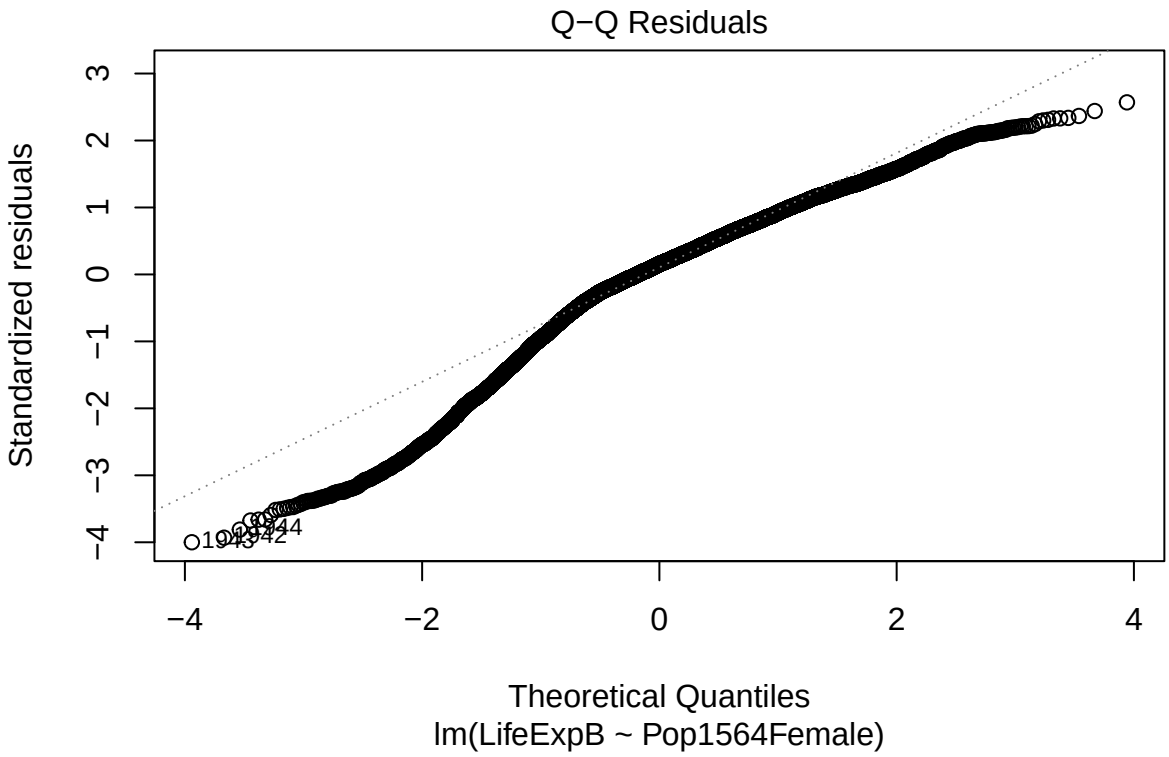
3.2 Visual Diagnosis of Linear Model

`plot()` function provides a quick way to visually check Gauss-Markov assumptions and outliers / high leverage points. These will be covered in more detail in QPM 1.

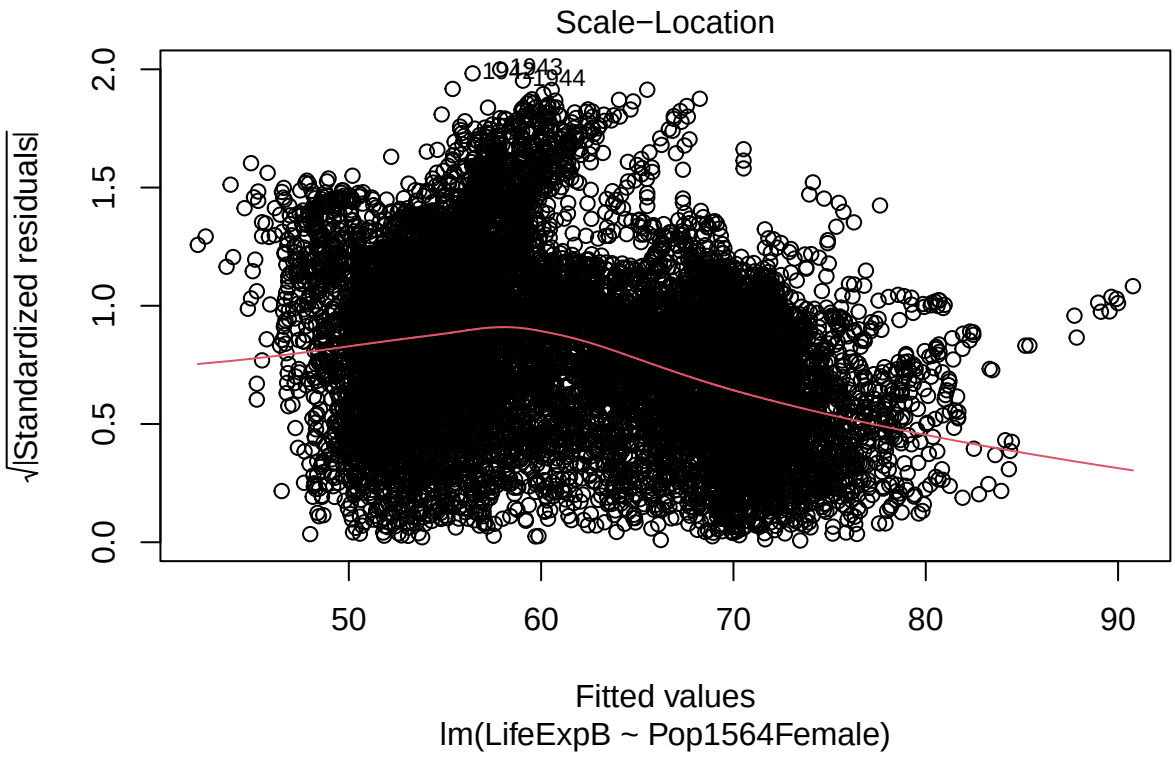
```
plot(model1, which = 1) # Residuals and Fitted values
```



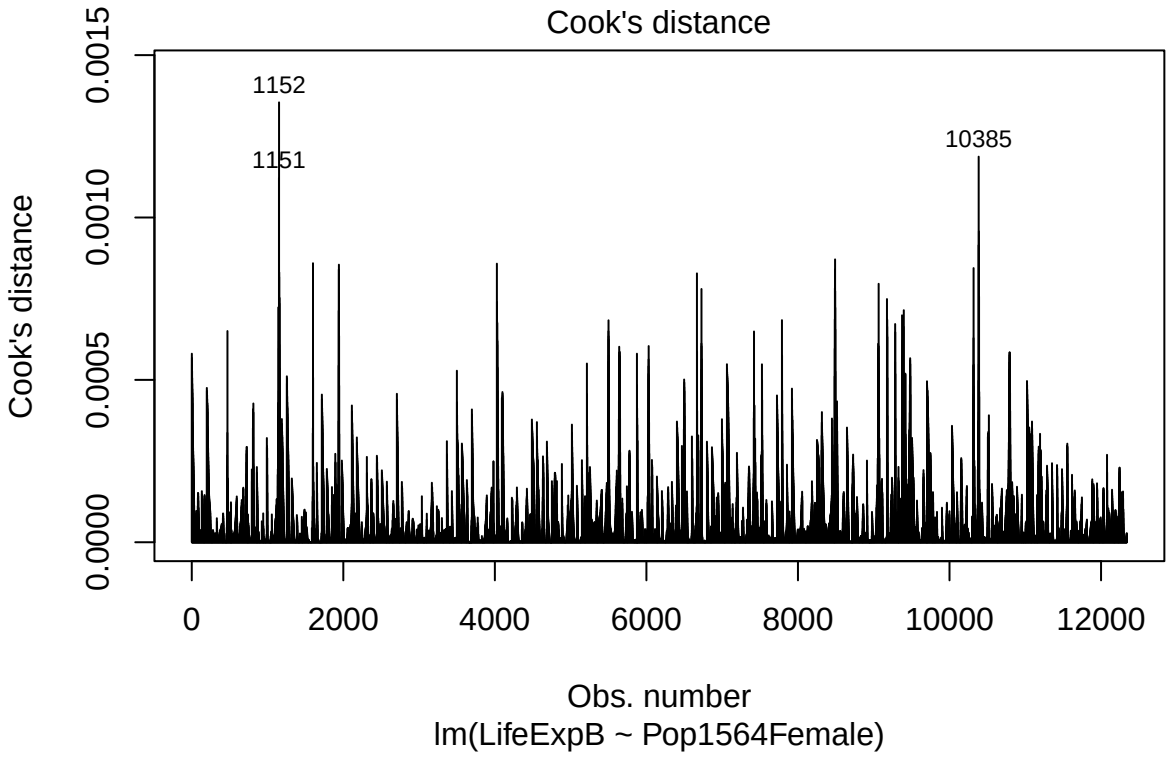

```
plot(model1, which = 2) # Q-Q plot
```



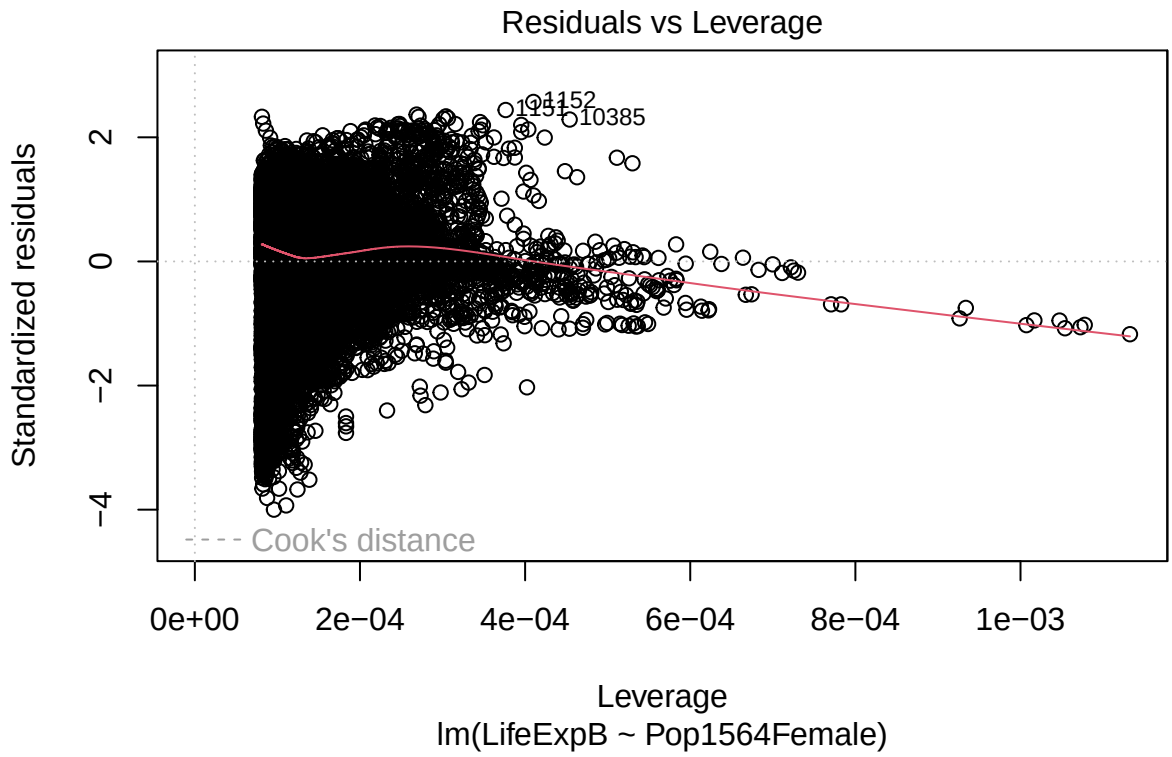

```
plot(model1, which = 3) # Standardized Residuals and Fitted values
```



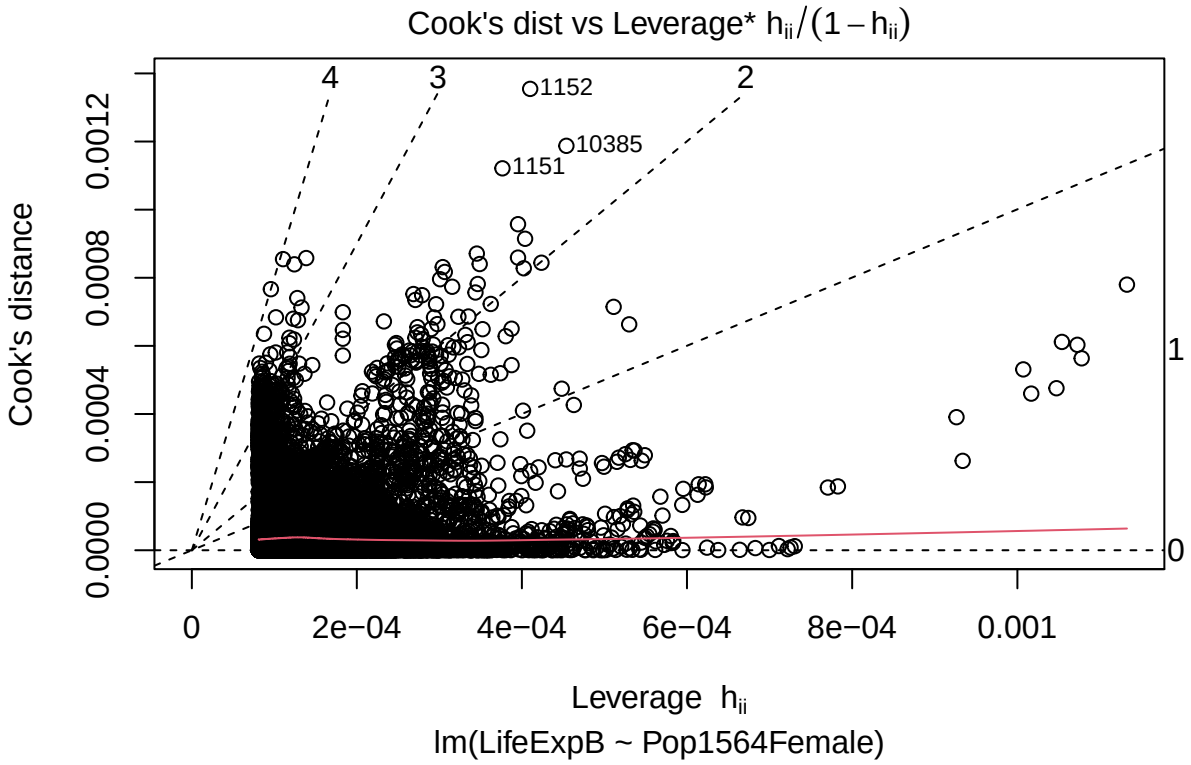

```
plot(model1, which = 4) # Cook's distance
```




```
plot(model1, which = 5) # Residuals and Leverages
```




```
plot(model1, which = 6) # Cook's distance and Leverages
```



3.3 Summarize Regression Outputs with `modelsummary`

`modelsummary::modelsummary()` provides some easy ways to convert the regression model outputs into other formats such as latex and docx.

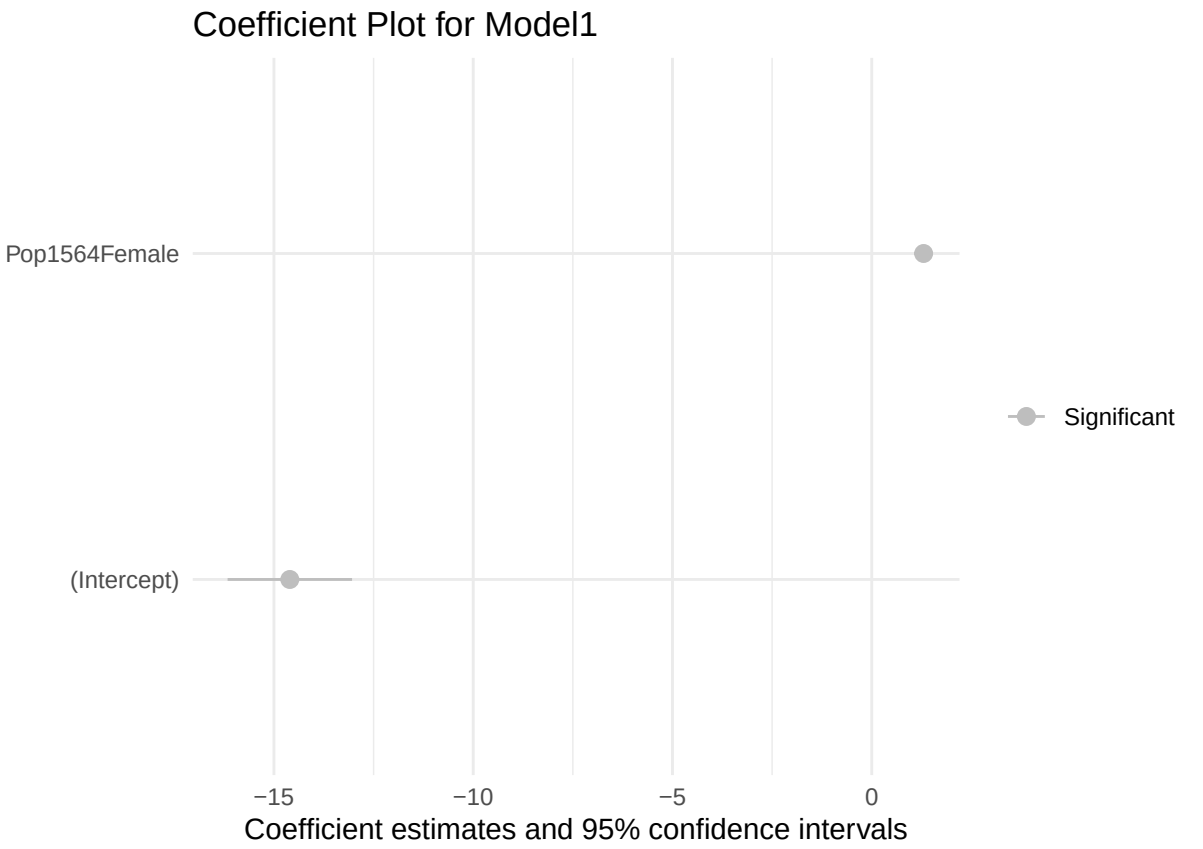
```
library(modelsummary)
library(ggplot2)
```

```
# By default as markdown table
modelsummary(model3)

# Latex
modelsummary(model3, output="latex") # or you can provide output file name such as model3.tex

# HTML
modelsummary(model3, output="html") # or you can provide output file name such as model3.html

# Multiple models as list
modelsummary(list(model1, model2, model3))
```

3.4 Visualizing Regression Outputs with `modelsummary`

`modelsummary` package also provides easy and beautiful ways to plot regression outputs. We will explore very basics of `modelplot()` function today. By default, it gives a nice coefficient plot in a `ggplot` object. We can then edit it just like any other `ggplot` object!

```r
modelplot(model1) +
  labs(title = "Coefficient Plot for Model1") +
  aes(color = ifelse(p.value < 0.001, "Significant", "Not significant")) +
  scale_color_manual(values = c("grey", "black"))
```



`modelsummary` package has more useful features that cannot be touched in this course. Check more details here! modelsummary documentation

3.5 In-class exercises:

We will continue using the worldTFR dataset.

1. Run a simple linear regression where: `GDPpc` as an outcome and `MtoFbirth`, `Pop1564`, and `LifeExpB` as predictors.
2. Get robust SE for the model.
3. Summarize the model in tables and figures using `modelsummary`.