Descriptive Statistic, ggplot2, and a Little Bit of LM

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.

Summarizing Commands

- 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.

Descriptive Summary Table

1. Option 1: `stargazer`

- `stargazer::stargazer(DATAFRAME)`: will generate a nice table
- ONLY DATAFRAME: `as.data.frame()`
- output options: `type = "text" / "html" / "latex"` "latex" is the default
- save as file: `out = {path}` -> extension!

2. Option 2: `modelsummary`

- `modelsummary::datasummary()`

Histograms and Scatter Plots

```
hist(worldTFR$GDPpc)
plot(x = worldTFR$GDPpc, y = worldTFR$Year)
```

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?

Introduction to ggplot2

- ggplot2 is a powerful visualization package
- Shares design philosophy with dplyr -> piping and chaining with +
- geom_graphtype()
  - geom_point()
  - geom_smooth()
  - geom_histogram()
  - geom_line()
- aes(): aesthetics, x and y value, colors
- ggplot cheat sheet: https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf
- We need to first install and load the package.

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

How to Make a Simple Scatterplot

- geom_point()

```
data = USArrests # built-in dataset

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

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

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

With dplyr

```
library(dplyr)

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

Histograms with ggplot

We can use geom_histogram.

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

Adjusting the X and Y Axis Limits

- `xlim()` and `ylim()`
- Accepts a vector of length 2 (minimum and maximum)

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

Change the Title and Axis Labels

- labs(title, subtitle, x, y, legend)

```
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
```

Change the Title and Axis Labels

- xlab()
- ylab()

```
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")
```

Change the Color and Size of Points

- aes()

```
ggplot(data, aes(x=UrbanPop, y=Assault)) +
  geom_point(col="pink", size=2) +  # Set static color and size for points
  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") +
```

Saving your plot

- ggsave() is a convenient function for saving a plot.
- By default, save the last plot using the current size

```
# The following code chunk is not executable.
ggsave(
  filename,
  plot = last_plot(), # you can provide plot as an object
  device = NULL, # automatically determined by path
  path = NULL, # include EXTENSION!
  scale = 1,
  width = NA,
  height = NA,
  units = c("in", "cm", "mm", "px"),
  dpi = 300,
  limitsize = TRUE,
  bg = NULL,
  ...
)
```

Saving your plot

```
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")
```

Saving your plot

```
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)
```

In-class exercises:

1. Omit all NA's from worldTFR.
2. Make a plot where the horizontal axis is LifeExpB and vertical axis is TFR using `ggplot`
3. Set the color to red, and size for points as 0.5
4. Add appropriate label for the axis and the plot.
5. Save your plot

Simple Linear Regression

- Foundations for linear regressions will be covered more in QPM 1.
- `lm(formula, data)`
- `formula` has a form `y ~ x1 + x2`
- interaction term: `x1 * x2` -> `x1 + x2 + x1:x2`

Simple Linear Regression

- As always, we can store the output as an object.
- summary()

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

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

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

Plot the Fitted Line

- geom_smooth() with method = "lm"

```
g1 <- ggplot(data, aes(x = UrbanPop, y = Assault)) +
  geom_point() +
  geom_smooth(
    method = "lm", # default is LOESS
    formula = "y ~ x",
    data = df,
    se = TRUE,
  )
```

Robust Standard Errors

- Robust standard errors are not the default.
- sandwich and lmtest

```
library(sandwich)
library(lmtest)

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

# summary may not work here
print(model1_robust)

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

# summary may not work here
print(model1_robust_clustered)
```

Visual Diagnosis of Linear Model

- `plot()` function provides a quick way to visually check Gauss-Markov assumptions and outliers / high leverage points.
- This 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
```

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)
```

```
# 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))
```

Visualizing Regression Outputs with `modelsummary`

- `modelsummary` package also provides easy and beautiful ways to plot regression outputs.
- `modelplot()`
- a nice coefficient plot in a `ggplot` object

```
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"))
```

More on `modelsummary`

- `modelsummary` package has more useful features
  - `datasummary()` to generate a beautiful descriptive statistics table
- Check more details here!
  https://modelsummary.com/vignettes/modelsummary.html

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`.