Assignment 2: Summarize global cesarean delivery rates and GDP across 137 countries

Your name and student ID

Today's date

- Due date: Friday, September 13 5:00pm.
- Late penalty: 50% late penalty if submitted within 24 hours of due date, no marks for assignments submitted thereafter.
- This assignment is marked out of 27. Marks are indicated for each question. There are 15 questions total.
- Submission process: Unlike previous labs and assignments, please submit your assignment *directly* to Gradescope. You can do this by knitting your file and downloading the PDF to your computer. Then navigate to Gradescope.com to submit your assignment. Here is a tutorial if you need help: https://www.gradescope.com/get_started. Scroll down on that page to "For students:submitting homework".

Helpful hints:

- Every function you need to use was taught during lecture! So you may need to revisit the lecture code to help you along by opening the relevant files on data hub. Alternatively, you may wish to view the code in the condensed PDFs posted on bCourses (under Files > Lectures). Good luck!
- Knit your file early and often to minimize knitting errors! If you copy and paste code for the slides, you are bound to get an error that is hard to diagnose. Typing out the code is the way to smooth knitting! We recommend knitting your file each time after you write a few sentences/add a new code chunk, so you can detect the source of knitting errors more easily. This will save you and the GSIs from frustration!
- If your code runs off the page of the knitted PDF then you will LOSE POINTS! To avoid this, have a look at your knitted PDF and ensure all the code fits in the file. When it doesn't, go back to your .Rmd file and add spaces (new lines) using the return or enter key so that the code runs onto the next line.

Summarizing global cesarean delivery rates and GDP across 137 countries

Introduction

Recall from this week's lab that we constructed bar charts and histograms to explore a data set that looked at global rates of cesarean delivery and GDP. If you need a refresher, you can view your knitted file from lab and remind yourself what you found.

In this week's assignment, you will describe these distributions using numbers. You will investigate the **mean** and **median** of the distribution of GDP. You will also examine the distribution of cesarean delivery separately for countries of varying income levels. Lastly, you will describe the **spread** of the distributions using **quartiles** and make a **box plot**.

Execute this code chunk to load the required libraries:

```
library(readr)
library(dplyr)
library(ggplot2)
```

Just like last time, read in the data that is stored as a .csv file and assign it to an object called `CS_data`. We will also use dplyr's `mutate()` to create the new cesarean delivery variable that ranges between 0 and 100:

```
CS_data <- read_csv("../Data/Cesarean.csv")
```

```
## Parsed with column specification:
## cols(
##   Country_Name = col_character(),
##   CountryCode = col_character(),
##   Births_Per_1000 = col_double(),
##   Income_Group = col_character(),
##   Region = col_character(),
##   GDP_2006 = col_double(),
##   CS_rate = col_double()
## )
```

```
# this code reorders the Factor variable `Income_Group` in the
# order specified in this function. This will affect the order the ggplot
# panels are shown in question 8 when we use `facet_wrap()`.
CS_data$Income_Group <- forcats::fct_relevel(CS_data$Income_Group,
                                             "Low income", "Lower middle income",
                                             "Upper middle income", "High income: nonOECD",
                                             "High income: OECD")

CS_data_new <- CS_data %>% mutate(CS_rate_100 = CS_rate*100)
```

1. [1.5 marks] Fill in the blanks indicated by "—". Please do not remove the asterisks because they bold your answers.

The function `mutate()` takes the old variable called — and multiplies it by — to make a new variable called —.

Investigate what would have happened had we not assigned the data using <- to CS_data_new? Re-run the code without the assignment and see what happens.

```
rm(CS_data_new) #first, remove using rm() CS_data_new from the environment
CS_data %>% mutate(CS_rate_100 = CS_rate*100) #re-run code without assignment
```

```
## # A tibble: 137 x 8
##    Country_Name CountryCode Births_Per_1000 Income_Group Region GDP_2006
##    <chr>        <chr>                <dbl> <fct>        <chr>     <dbl>
##  1 Albania      ALB                     46 Upper middl~ Europ~    3052.
##  2 Andorra      AND                      1 High income~ Europ~   42417.
##  3 United Arab~ ARE                     63 High income~ Middl~   42950.
##  4 Argentina    ARG                    689 High income~ Latin~    6649.
##  5 Armenia      ARM                     47 Lower middl~ Europ~    2127.
##  6 Australia    AUS                    267 High income~ East ~   36101.
##  7 Austria      AUT                     76 High income~ Europ~   40431.
##  8 Azerbaijan   AZE                    166 Upper middl~ Europ~    2473.
##  9 Belgium      BEL                    119 High income~ Europ~   38936.
## 10 Benin        BEN                    342 Low income   Sub-S~     557.
## # ... with 127 more rows, and 2 more variables: CS_rate <dbl>,
## #   CS_rate_100 <dbl>
```

```
names(CS_data)
```

```
## [1] "Country_Name"    "CountryCode"     "Births_Per_1000" "Income_Group"
## [5] "Region"          "GDP_2006"        "CS_rate"
```

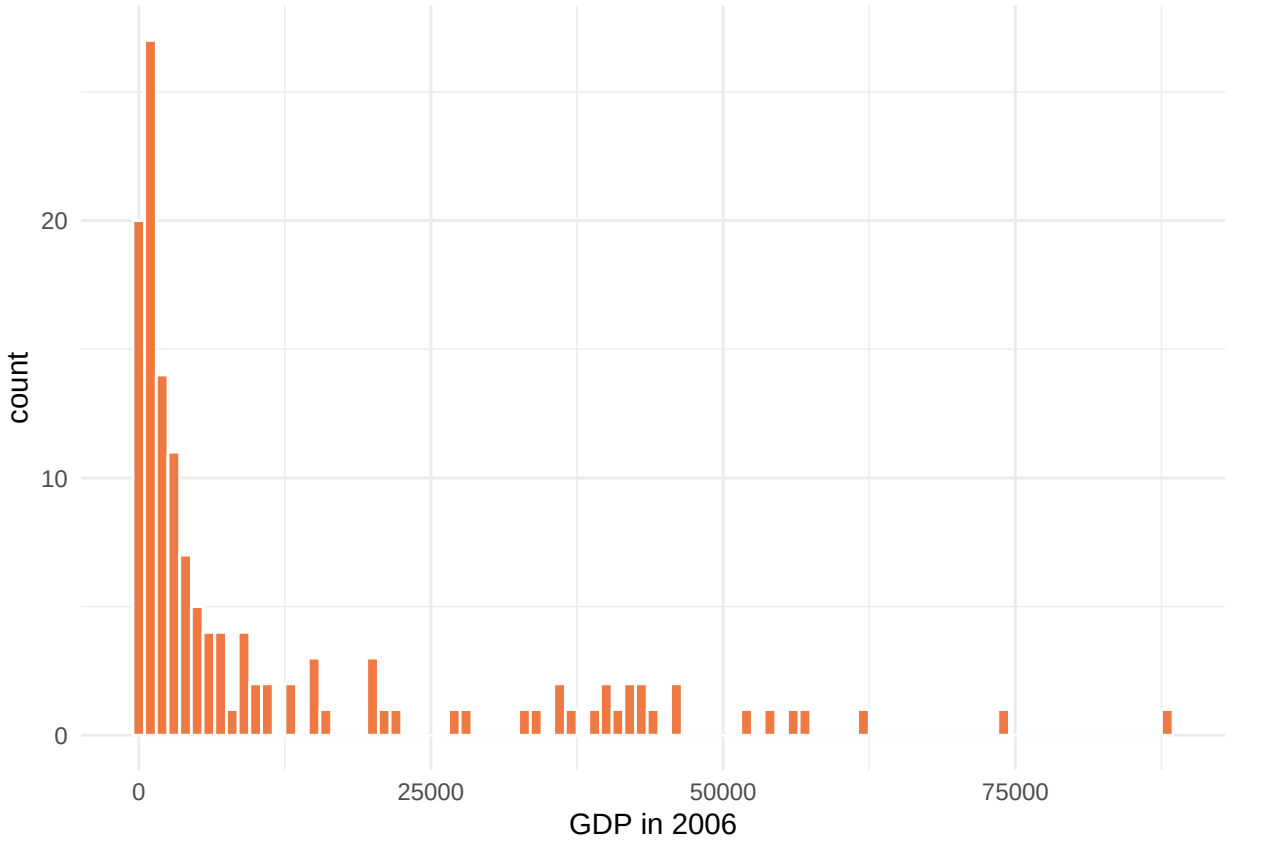
Did CS_rate_100 get added to the data set? No. You can tell by using `head(CS_data)` to view the first few rows and notice that the variable hasn't been added. This is because when we don't assign the output to anything, it just prints it out for us to see. Nothing is saved. Because we are only making a small change, we can overwrite the original object (CS_data) by assigning the transformed data to the same name. Otherwise, you would want to assign the variable to a different name so you can keep two separate copies of the data set.

```
CS_data <- CS_data %>% mutate(CS_rate_100 = CS_rate*100)
```

GDP: Summarizing measures of centrality

Recall your histogram of GDP from lab:

```
ggplot(data = CS_data, aes(x = GDP_2006)) +
  geom_histogram(col = "white", fill = "sienna2", binwidth = 1000) +
  xlab("GDP in 2006") +
  theme_minimal()
```



2. [1 mark] Describe the shape of this distribution. Is it "skewed left", "skewed right", "symmetric", or "bimodal"?

Replace this text with your answer.

3. [1 mark] Based on your answer, will the mean be approximately the "same", "larger than", or "smaller than" the median?

Replace this text with your answer.

4. [3 marks] Describe, in words, how the mean and median are calculated:

Replace this text with your answer.

To calculate the mean and median in R, we can use the `summarize()` function from the `dplyr` package. The `summarize()` function is used anytime we want to take a variable and summarize something about it into one number, like it's mean or median. Here is the code to summarize `GDP_2006`'s mean and print it out to the screen. In the code, we name the mean `mean_GDP` and output the result to the screen:

```
GDP_summary <- CS_data %>% summarize(mean_GDP = mean(GDP_2006))
GDP_summary
```

```
## # A tibble: 1 x 1
##   mean_GDP
##      <dbl>
## 1   11791.
```

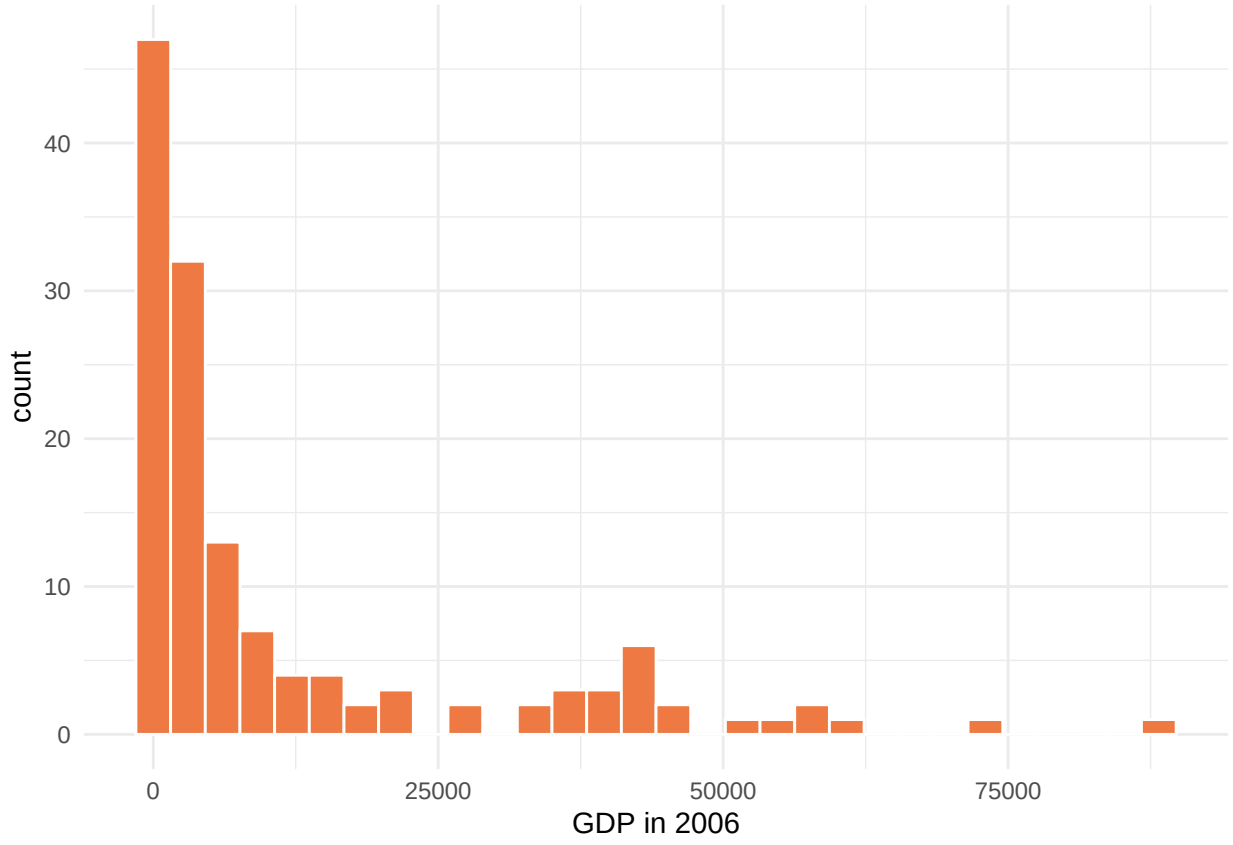
5. [1 mark] Extend the above code to also summarize the median. Call the median summary `median_GDP`. Assign this summary to `GDP_summary` (it will overwrite the previous version):

```
# use this code chunk to write your code.
```

6. [2 marks] `geom_vline()` can be used to add the mean and the median to the histogram shown above. This `geom` adds a vertical line to the graph. You need to specify where to add the line by passing it an "xintercept" argument. Remove the comment indicator (i.e., the three "#") from the code below and update the `geom_vline()` code to plot lines at the mean and median by telling it the mean and median estimates. The argument `lty=1` (standing for line type) will plot a solid line and `lty=2` will plot a dashed line.

```
ggplot(data = CS_data, aes(x = GDP_2006)) +
  geom_histogram(col = "white", fill = "sienna2") +
  xlab("GDP in 2006") +
  theme_minimal() #+
```

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

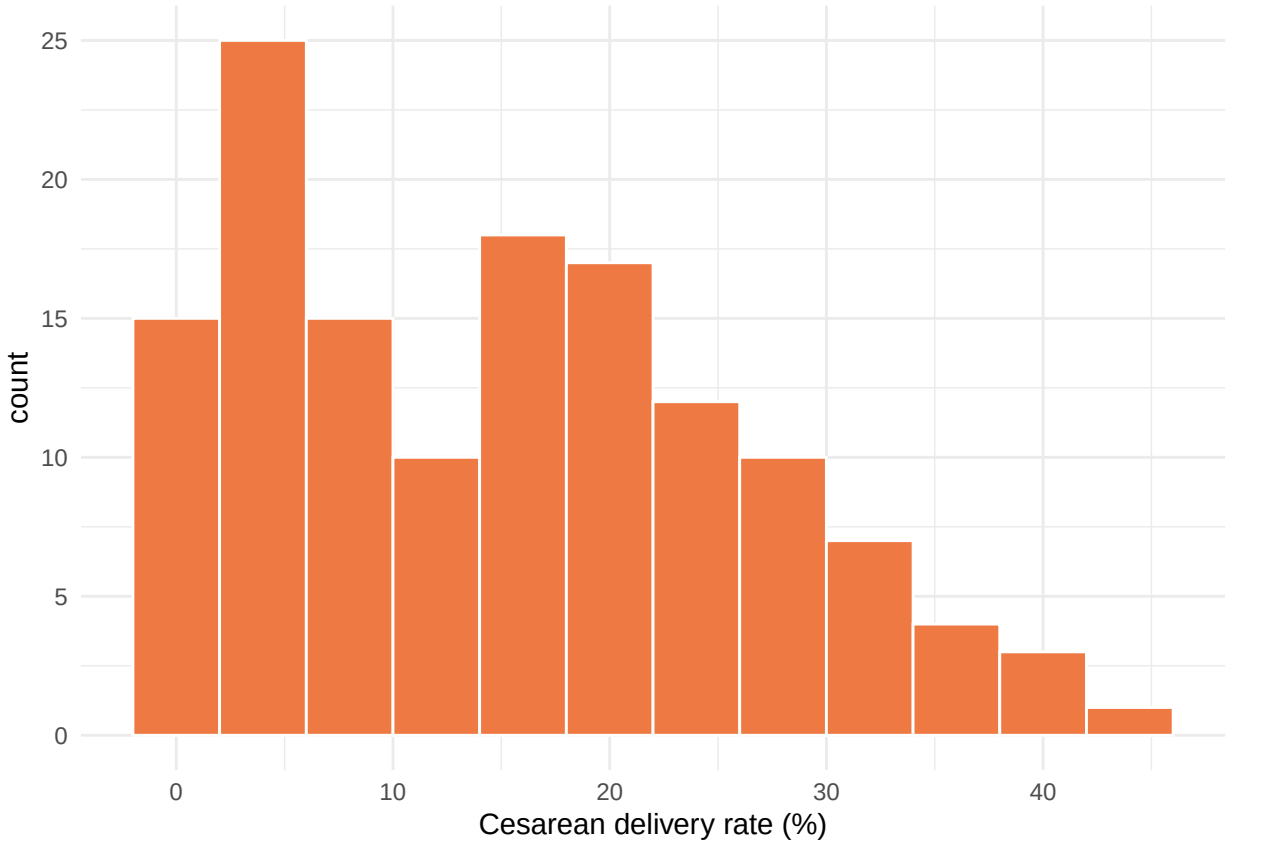



```
  #geom_vline(aes(xintercept = ), lty=1) +
  #geom_vline(aes(xintercept = ), lty=2)
```

Summarizing the distribution of cesarean delivery

Recall the distribution of cesarean delivery rates across countries:

```
ggplot(data = CS_data, aes(x = CS_rate_100)) +
  geom_histogram(binwidth = 4, col = "white", fill = "sienna2") +
  xlab("Cesarean delivery rate (%)") +
  theme_minimal()
```



7. [1 mark] Describe the shape of this distribution. Is it "skewed left", "skewed right", "symmetric", or "bimodal"?

Replace this text with your answer.

There appears to be multiple peaks which sometimes points to there being another variable that might explain the peaks. We can make a separate histogram for each income group using the `facet_wrap()` function.

8. [1 mark] Extend your ggplot code using the `facet_wrap()` statement to make a separate histogram for each level of the `Income_Group` variable:

```
# copy and paste your old ggplot code here and extend it using facet_wrap()
```

9. [2 marks] Based on this plot and the previous one, describe why the data had two peaks

Replace this text with your answer.

10. [1 mark] Why might lower income countries have lower rates of cesarean delivery?

Replace this text with your answer.

11. [2 marks] Modify the previously used `dplyr summarize()` code (from question 5) to calculate the mean and median of `CS_rate_100` using only one `summarize()` command. Assign this summary to the name `CS_summary` and then print the results by typing `CS_summary` so you can see the contents.

```
# your dplyr code followed by printing CS_summary to the screen
```

Measures of variation

12. [2 marks] Use ggplot2 to make a boxplot of the distribution of `CS_rate_100`

```
# your box plot code
```

Recall that the box plot summarizes the distribution in five numbers: the minimum, the first quartile (with 25% of the data below it), the median, the third quartile (with 75% of the data below it), and the maximum. Each of these numbers has at least one corresponding R function:

Number	R function
Minimum	`min(variable)`
First quartile	`quantile(variable, probs = 0.25)`
Median	`median(variable)` or `quantile(variable, probs = 0.5)`
Third quartile	`quantile(variable, probs = 0.75)`
Maximum	`max(variable)`

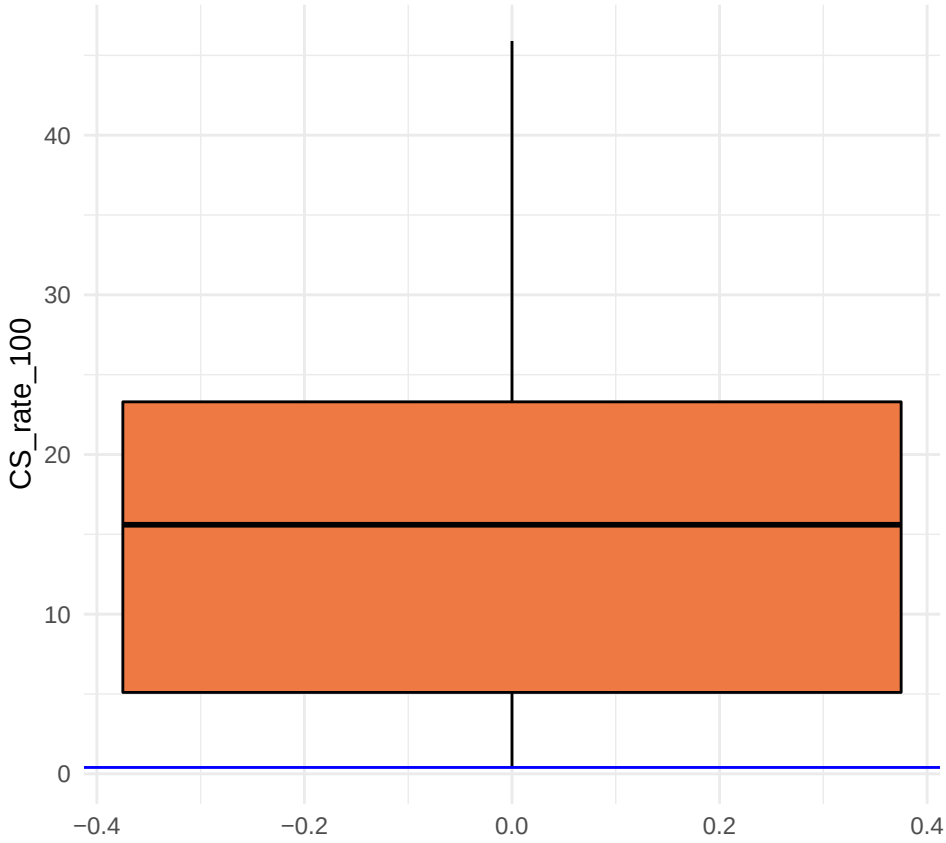
13. [2.5 marks] Use a combination of `dplyr`'s `summarize` function and the above functions to compute the five number summary of `CS_rate_100`. Assign the summary to the name `five_num_summary`. Then, print `five_num_summary` to the screen:

```
# your dplyr code followed by printing five_num_summary to the screen
```

Double check that `geom_boxplot()` is making the box plot correctly. You can do this by adding horizontal lines to the plot at each number in your five number summary using `geom_hline()`. Because horizontal lines intercept the y-axis, `geom_hline()` requires the `yintercept` argument that you can set to each number in your summary.

14. [2 marks] The code below includes one horizontal line at the minimum shown in blue. Add the rest of the lines:

```
ggplot(data = CS_data, aes(y = CS_rate_100)) +
  geom_boxplot(col = "black", fill = "sienna2") +
  theme_minimal() +
  geom_hline(aes(yintercept = 0.4), col = "blue")
```



15. [4 marks] Compile the following code which adds two points to the CS_data, makes a new dataset called `CS_data_plus_2`, and redraws the box plot. How did the box plot change? Perform a calculation to justify why it changed. What are the newly-added features on the plot called?

```
out_data <- tibble::tribble(
  ~Country_Name, ~CS_rate_100,
  "Point 1", 90,
  "Point 2", 60
)

CS_data_plus_2 <- dplyr::full_join(CS_data, out_data)
```

```
## Joining, by = c("Country_Name", "CS_rate_100")
```

```
ggplot(data = CS_data_plus_2, aes(y = CS_rate_100)) +
  geom_boxplot(col = "black", fill = "sienna2")
```