Lab 2: Visualization of global rates of cesarean delivery

Your name and student ID

today's date

Visualizing global cesarean delivery rates and GDP across 137 countries

The proportion of pregnant women who undergo cesarean deliveries varies dramatically across countries. In the US, the cesarean delivery rate is around 33%. A 2018 study in the British Medical Journal found that cesarean delivery rates ranged between 0.6% in South Sudan and 58.9% in the Dominican Republic.[1]

Over the course of this lab and the first assignment, we will study the relationship between countries' cesarean delivery rates and their gross domestic products (GDPs).[2,3]

In today's lab we will:

- import the data,
- familiarize ourselves with the dataset,
- explore the univariate distributions of the key variables.

If you haven't already, begin by knitting this document by pushing the "Knit" button above. As you fill it out, you can re-knit (push the button again) and see how the document changes.

Let's get started!

Hit the green arrow icon in the line below to execute the three lines of code in the code chunk, or execute them line by line by placing your cursor on the first l line and hitting cmd + enter on Mac or ctrl + enter on PC. If you do not want messages produced by loading `dplyr` to print in your knitted document, toggle `message = TRUE` to `message = FALSE` in the code chunk options. Re-knit your file and notice the different in the output:

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

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

```
##
## 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
```

```
library(ggplot2)
```

- The `library` command loads the libraries `dplyr` and `ggplot2` into memory.
- The `dplyr` library contains functions we will use to manipulate data.
- The `ggplot2` library contains functions to visualize data.

The data on each country's cesarean delivery rate is stored in a comma separate values (csv) file. We can load that data and assign it to an object called `CS_data` by running the code below. Hit the green arrow in the line below to run the code. (Note that the green arrow will run all the code in this code chunk, not just the first line of code.):

```
CS_data <- read_csv("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 re-orders the variable Income_Group in the specified order.
# Note that it *does not* change the order of the data frame (like arrange() does)
# Rather, it specifies the order the data will be plotted.
# This will make more sense when we plot the data using Income_Group, and then
# again using Income_Group_order
CS_data <- CS_data %>%
  mutate(Income_Group_order = forcats::fct_relevel(Income_Group, "Low income",
                                                   "Lower middle income", "Upper middle income",
                                                   "High income: nonOECD", "High income: OECD"))
```

Look over at the environment tab (on the top right pane) and make sure you see your dataset there. Click the icon to the right of its name. This will open the data viewer tab. Take a quick look at the data and then switch back to the "Lab02.Rmd" file and move to the next step.

Often, when you receive a dataset, you also receive a data dictionary that tells you what each variable means. Below, is an example of such a dictionary for this dataset:

Variable	Description
Country_Name	The name of the country
CountryCode	3-digit unique country identifier
Births_Per_1000	The number of births per 1000 people in the country
Income_Group	The income group the country is categorized into
Region	The geographic region the country belongs to
GDP_2006	The gross domestic product (GDP) of the country in 2006
CS_rate	The percent of births that were delivered by cesarean section
Income_Group_order	The "ordered" version of Income_Group

Look at your newly imported data

In four separate code chunks, type `str(CS_data)`, `dim(CS_data)`, `names(CS_data)`, and `head(CS_data)`. Execute each code chunk and study the output.

```
# write your code here
```

```
# write your code here
```

```
# write your code here
```

```
# write your code here
```

In your own words, what do these functions do: `dim()`: `str()`: `names()`: `head()`:

Based on these functions' outputs, who/what are the individuals in this dataset? How many of them are there?

Add a new variable

Right now, `CS_rate` is presented as a number between 0 and 1. For plotting, it will be nice to have the proportion displayed as a number between 0 and 100. We can use `dplyr`'s `mutate()` function to add a new variable called `CS_rate_100` to the dataset that takes this desired format. Write one line of code to add this variable to the dataset:

```
# write your code here
```

Examine the distribution of income and geographic regions

These countries represent a wide range of income and geographic regions. Which variable groups the countries by income regions? By geographic regions?

The variable that groups the countries by income regions is:

The variable that groups the countries by geographic regions is:

These variables are categorical variables. So far in lecture, we've seen categorical variables stored as `chr` variables when we've examined `str(data)`. Here however, `Income_Group_order` is stored as a `Factor` variable, which is another way to store categorical data. One benefit of storing it as `Factor` is that we can re-order the `Factor` variable in a way that makes sense for plotting and later analysis. To see the values and order of a factor variable, we can use the following code.

```
CS_data %>% pull(Income_Group_order) %>% levels
```

```
## [1] "Low income"           "Lower middle income"  "Upper middle income"
## [4] "High income: nonOECD" "High income: OECD"
```

```
# You won't be tested on writing the line of code above. Try and gain an understanding of what it is sh
```

Because `Income_Group` and `Region` are categorical, we make bar charts to represent their distributions. Using the `ggplot()` code you learnt during lecture, make a bar chart showing the number of countries per income group using `Income_Group`. Then remake the plot using `Income_Group_order`.

```
# write your code here
```

Do you prefer the plot using `Income_Group` or `Income_Group_order`? Why?

Did you use the argument `stat="identity"` in the code to make this bar chart? Why or why not?

Make a bar chart showing the number of countries per geographic region.

```
# write your code here
```

Based on your plot, which world region has the most countries in the data set?

Which geographic region had the fewest number of countries in the data set

Note that the above code chunk had the chunk option "fig.width = 11". Try changing the setting to a lower number, like 9 and re-running the chunk. Notice how the width of the rendered plot changes. Change it back to 11 so that the x-axis labels don't look too cramped.

Examine the distribution of `GDP_2006` and `CS_rate_100` across countries

`GDP_2006` and `CS_rate_100` are quantitative, meaning that we use histograms rather than bar charts to examine their distributions. Using code you learnt in class, make a histogram of `GDP_2006`. Pay attention to the message that R outputs when running your code. It is telling you that by default the data is grouped into 30 bins for plotting.

```
# write your code here
```

Rather than letting R choose the number of `bins` we can set `binwidth = number` within the geom function to choose the width of the bins. Try setting the binwidth to different amounts and see how it changes the look of the plot.

To choose a binwidth, look at the x-axis of the histogram you just made. Because we are plotting GDP across a wide variety of countries, it might make sense to choose a bin width in the 1000s. Thus, the range of the variable matters when choosing a good bin width.

```
# write your code here
```

Next, make a histogram for cesarean delivery rates using the variable `CS_rate_100`. First run your code without binwidth set and then add in the binwidth argument to choose the size.

```
# write your code here
```

Polish up your plots

Your plots look great so far, but let's get them ready to present at a meeting or in a report. To do this please update the plot and axes titles (`labs`), and change the `fill` or `color` of the bars. Here is a big list of colors that you can set col and fill equal to. For example col = "orchid1" will change the color to a pinky-purple.

Below present your final plots with updated titles, labels, colors, and fills. To do this, extend the ggplot code you wrote above with the functions and arguments just described:

```
# write your code here
```

Save your plots to separate files

Sometimes, you need to save plots as PNG or JPEG to include in a PowerPoint presentation or submit to a research journal for publication. The `ggsave()` function is used to save plots as separate files. Create a "Plots" folder in your project folder by clicking the New Folder button. Be sure to make the folder in the main PH142 directory, rather than inside your Code/ or Data/ folders.

Quickly read the documentation of the `ggsave()` function by executing this code which opens a help window in the bottom right pane:

```
?ggsave
```

Try figuring out how to save a plot based on the documentation and running your attempts in a code chunk. Don't worry about producing an error or saving your plot in the wrong place – that is all part of coding and learning new functions.

Hint: To specify the Plots/ folder as the `path`, modify the path we sent to `read_csv()` when we specified the Data/ folder.

When you are done, knit your R document one last time and ensure it looks good! Change anything you don't like. You can choose to knit to .docx or .pdf by selecting these options

form the dropdown menu next to knit, in the case that you'd like to save the completed file onto your computer.

References

1. Boatin AA, Schlotheuber A, Betran AP, Moller AB, Barros AJ, Boerma T, Torloni MR, Victora CG, Hosseinpoor AR. Within country inequalities in caesarean section rates: observational study of 72 low and middle income countries. BMJ. 2018;24(360):k55. DOI: 10.1136/bmj.k55.
2. Gibbons L, Belizán JM, Lauer JA, Betrán AP, Merialdi M, Althabe F. The global numbers and costs of additionally needed and unnecessary caesarean sections performed per year: overuse as a barrier to universal coverage. World Health Organization; 2010. Available at: http://www.who.int/healthsystems/topics/financing/healthreport/30C-sectioncosts.pdf. Accessed October 4, 2015.
3. The World Bank. GDP per capita (current US$). Washington, DC; Available at: http://data.worldbank.org/indicator/NY.GDP.PCAP.CD. Accessed Sep 30, 2015.

Submission (Tutorial: https://www.youtube.com/watch?v=NYNDi_zJRGE)

- 1. Before submitting, make sure the file knits properly, otherwise you won't receive credit for completing the lab.
- 2. Then, click on the **Terminal** tab in the panel below.
- 3. Copy and paste the line of code (cd; cd PH142_Fall2019/Labs/Lab02; python3 ok –submit;) into **Terminal** panel below and press "enter".
- 4. Follow the remaining prompts in the terminal to submit your assignment. You can watch this video tutorial online (https://www.youtube.com/watch?v=NYNDi_zJRGE) for a more detailed step-by-step.