Problem Set - 4

Data Science with R

More on Visualizations

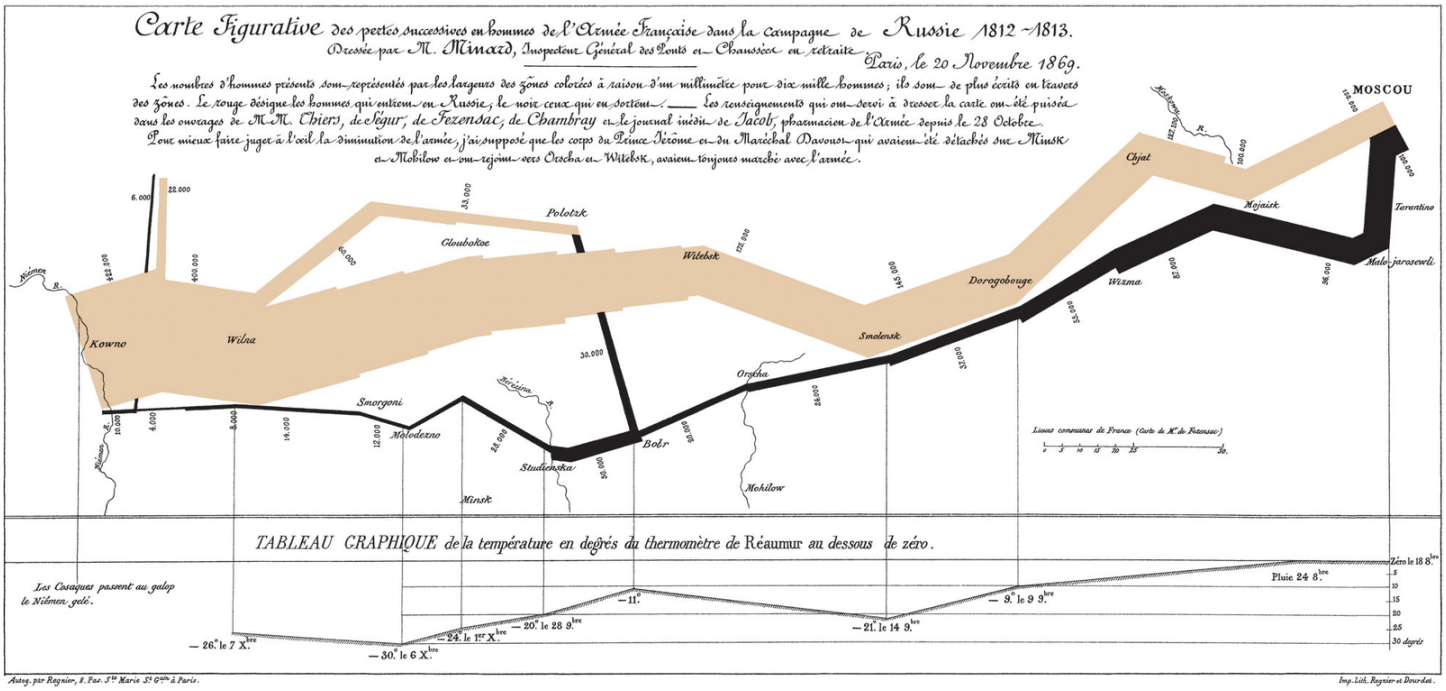
We are going to learn a little more about the importance of visualizations and telling a story. We will do this through a few case studies.

Napolean March

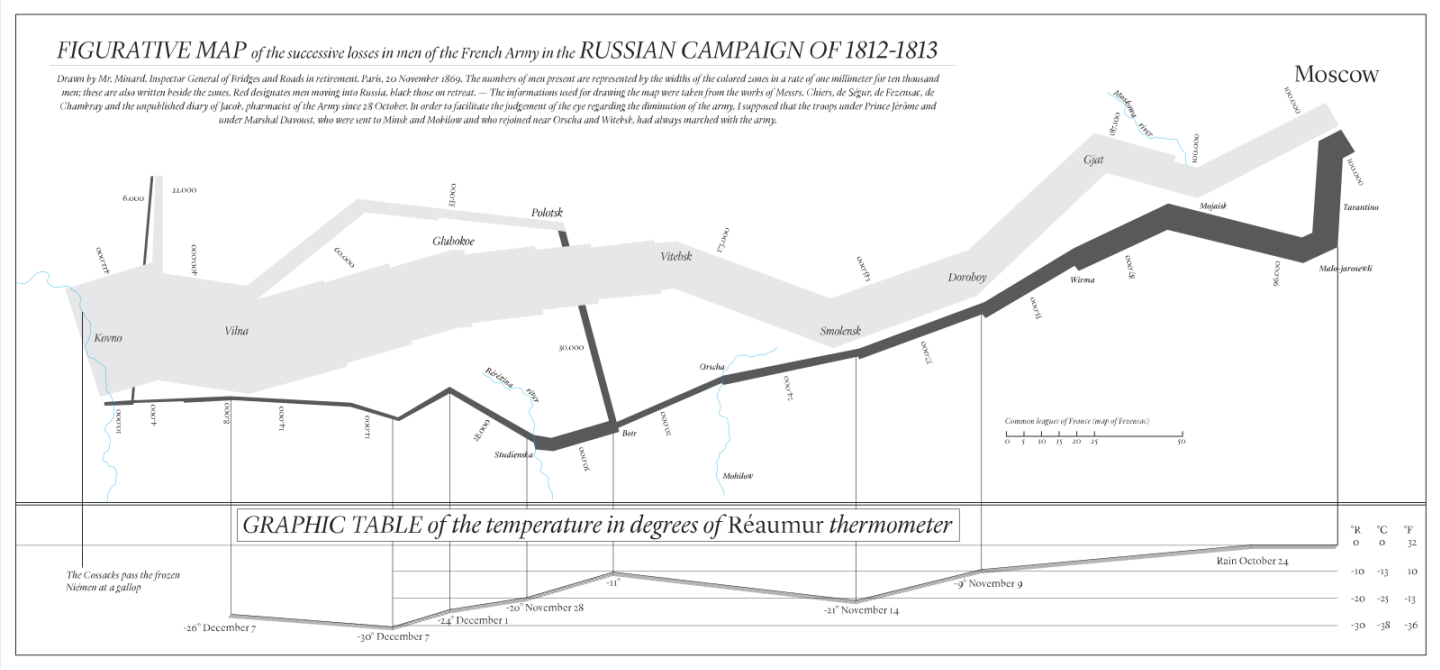
In June of 1812, Napolean's army of over 400,000 solders entered the Polish-Russian border and marched to Moscow to invade Russia. This is one of the most lethal military operations in history, killing about 98% of the soldiers.

A natural visualization of the survival of the people

However, in order to tell the story of the journey and magnitude of destruction, the following visualization by Charles Minard (1869) is astoundingly informative



The following is an English translation of the map:



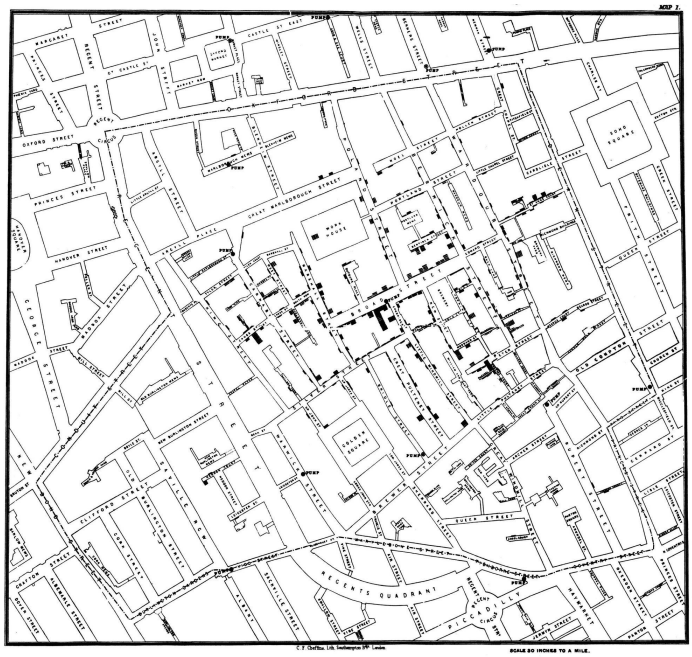
Impressively, the graph shows many variables in these two-dimensions:

1. the number of soldiers
2. distance
3. temperature
4. direction of travel
5. local relative to dates

This plot pains us a story of the invasion and the devastation incurred.

Cholera in London

In 1854, a severe outbreak of cholera killed 616 people near Bond Street in London. At the time, not much was known on how cholera spread.



John Snow (not the Game of Throne's one) studied the cases in the area and made a modification of a *dot map* - a map with dots placed representing data.

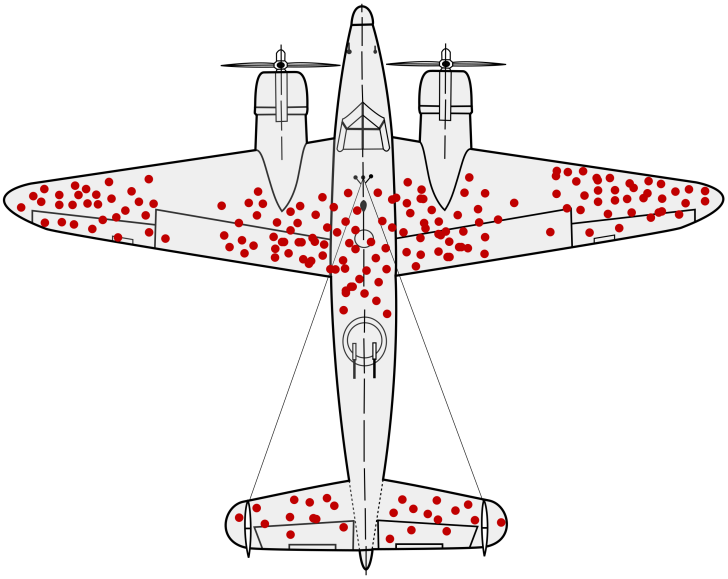
The above plot was a revelation as it was able to get to the solution of the problem. Can you guess?

Bullet Holes

Reading plots is as important as visualizing plots.

During the World War II, the US military was trying to understand how to minimize their aircraft loss in battle, and where to reinforce more armor in their aircraft. Naturally, they didn't want to put too much armor, as that increases the weight of the aircraft.

They analyzed the placement of the bullet holes on the planes that returned and arrived a map similar to the one below



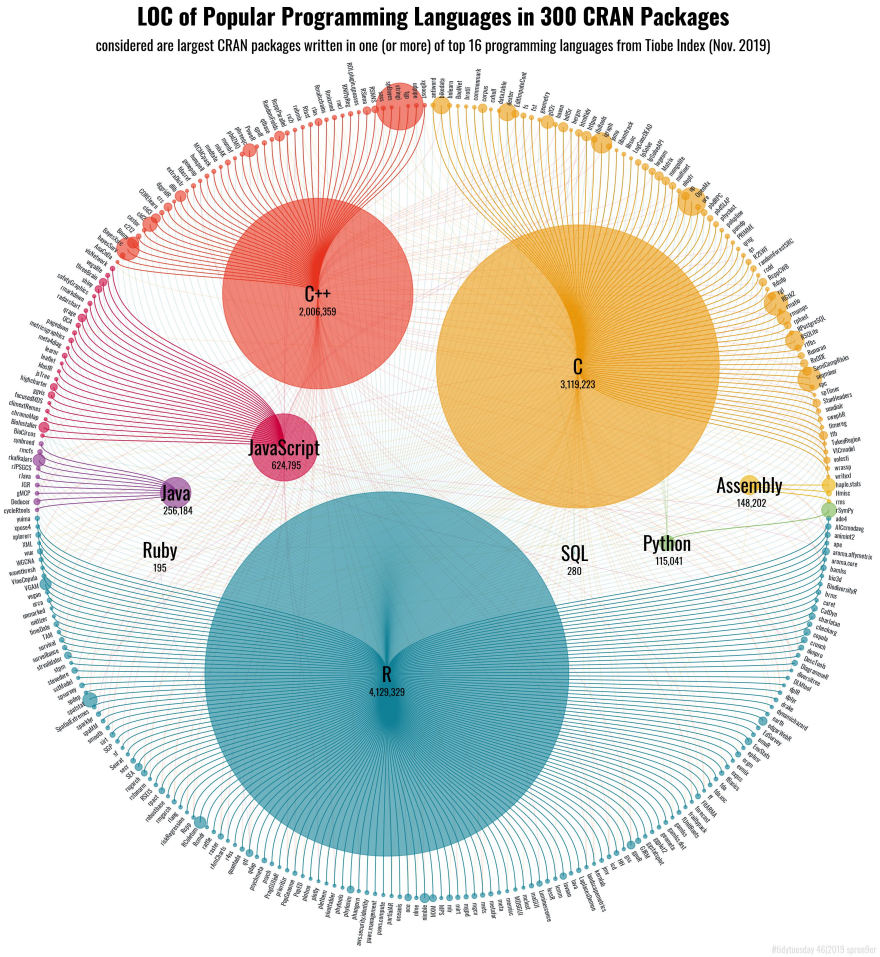
They went on to put more armor on where the red dots were found. However, this turned out to be not very helpful.

Mathematician/Statistician, Abraham Wald had a different suggestion. Can you guess?

The figure of an aircraft with bullet markings is synonymous with **Survivorship Bias.**

CRAN Packages

Dr. Torsten Sprenger studied the lines of code of top 300 packages on CRAN, and produced the following visualization:



This is an example of an organized bubble/cloud plot. The code for the visualization is available here.

ggplot2

In addition to the basic R plots we learned to make in Week-3, R provides another very useful graphical package for modern plots called `ggplot2`. Depending on what you want to plot and how you want to plot it, you can choose to make plots in either base R or `ggplot2`. We first load the package and load the IMDB dataset (this is in the worksheet repo).

```
library(ggplot2)
load("IMDB_movies.Rdata")
```

`ggplot2` works a little differently. It works in layers. First, we define the data are studying, then the variables, then the type of plot etc. For example, the following creates an empty plot with the right axes on the x axes.

```
ggplot(dat, aes(x = rating))
```

Below are one variable plots

```
ggplot(dat, aes(x = rating)) +
  geom_histogram()

# also run this
```

```r
ggplot(dat, aes(x = rating)) +
  geom_boxplot()

ggplot(dat, aes(x = rating)) +
  geom_bar()
```

Two variable plots

```r
ggplot(dat, aes(x = year, y = over.votes)) +
  geom_point()

# zooming in to some part
ggplot(dat, aes(x = year, y = over.votes)) +
  geom_point() +
  coord_cartesian(xlim = c(1996, 2025))
```

Aesthetics can be added using `aes()` options

```r
Year <- dat$year < 2000
Year <- as.factor(Year)
levels(Year) <- c("Before 2000", "After 2000")
ggplot(dat, aes(x = over.votes, y = rating)) +
  geom_point(aes(shape = Year, col = Year)) +
  labs(title = "Votes vs Rating", y = "Rating", x = "Number of Votes")
```

There are a number of options and features of `ggplot2()` and a lot of it can be found in their home website https://ggplot2.tidyverse.org/.

Here are some practice problems:

1. Load the `covid.Rdata` object in your repository which has updated Covid data from India.
2. Using examples from the website below
   http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html,

   take inspiration and make different useful visualizations for this dataset.

Using Quarto and RMarkdown

All of your Problem Set documents have been written in Quarto — Quarto is an improved version of RMarkdown.

RMarkdown is a R-integrated language interface that allows the development of reproducible documents and website. Quarto is a modified visual editor that allows you to integrate all sorts of code into the document.

A clear purpose of RMarkdown is the following: If there is code involved to generate any plot/tables etc in document, then to ensure that all work is reproducible, document should be able to contain the code used to make the document.

Additionally, RMarkdown is TeX enabled, we are able to write equations simply and beautifully.

Your job in this assignment will be to recreate the document `sample.pdf`.

Here are some resources you will need:

- Markdown basics: https://quarto.org/docs/authoring/markdown-basics.html

- Chunk options in Quarto https://quarto.org/docs/computations/execution-options.html
- List of mathematical commands: https://oeis.org/wiki/List_of_LaTeX_mathematical_symbols
- List of mathematical symbols: https://www.cmor-faculty.rice.edu/~heinken/latex/symbols.pdf

Building R Shiny Apps

We will finally make some Shiny Apps! The idea of a Shiny App is to make an interactive environment for analyzing a dataset or certain concepts.

A Shiny application is divided into two parts: the **User Interface (UI)** and the **Server**. The UI contains the way the app looks and its presentation. The server is responsible for the logical flow of the app, and the core functions. The Server also controls the data that will be displayed.

To start a new R Shiny App, choose "Shiny Web App" in the upper left corner. This creates a default folder with a single file called `app.R`. There are two aspects: `ui` and `server()`:

```
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(

    # Application title
    titlePanel("Old Faithful Geyser Data"),

    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),

        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("distPlot")
        )
    )
)
```

`ui` saves the characteristics of the visual of the shiny app.

- The style being used is `fluidPage()` which turns the page into column-panels and rows.
- `titlePanel()` sets the main title of the ShinyApp.
- `sidebarLayout()` starts how the sidebar looks.
  - In that, there is a `sidebarPanel()` which has a `sliderInput()`.

  - The slider is tagged as `bins`, has the name "Number of bins". The `min` value is 1, the `max` value is 50 and the default value is 30.

- The `mainPanel()` describes what the main panel will look like
  - It has a `plotOutput` that has the plot `distPlot`, this `distPlot` is plotted using the `server()` function.

The `server()` function looks like the following:

```
# Define server logic required to draw a histogram
server <- function(input, output) {

    output$distPlot <- renderPlot({
        # generate bins based on input$bins from ui.R
        x    <- faithful[, 2]
        bins <- seq(min(x), max(x), length.out = input$bins + 1)

        # draw the histogram with the specified number of bins
        hist(x, breaks = bins, col = 'darkgray', border = 'white',
             xlab = 'Waiting time to next eruption (in mins)',
             main = 'Histogram of waiting times')
    })
}
```

- `input` contains all the input parameters gathered from the UI interface.
- `output` has the output components that will go into the `mainPanel()`.

Go ahead and click on `Run App` to see how the page looks.

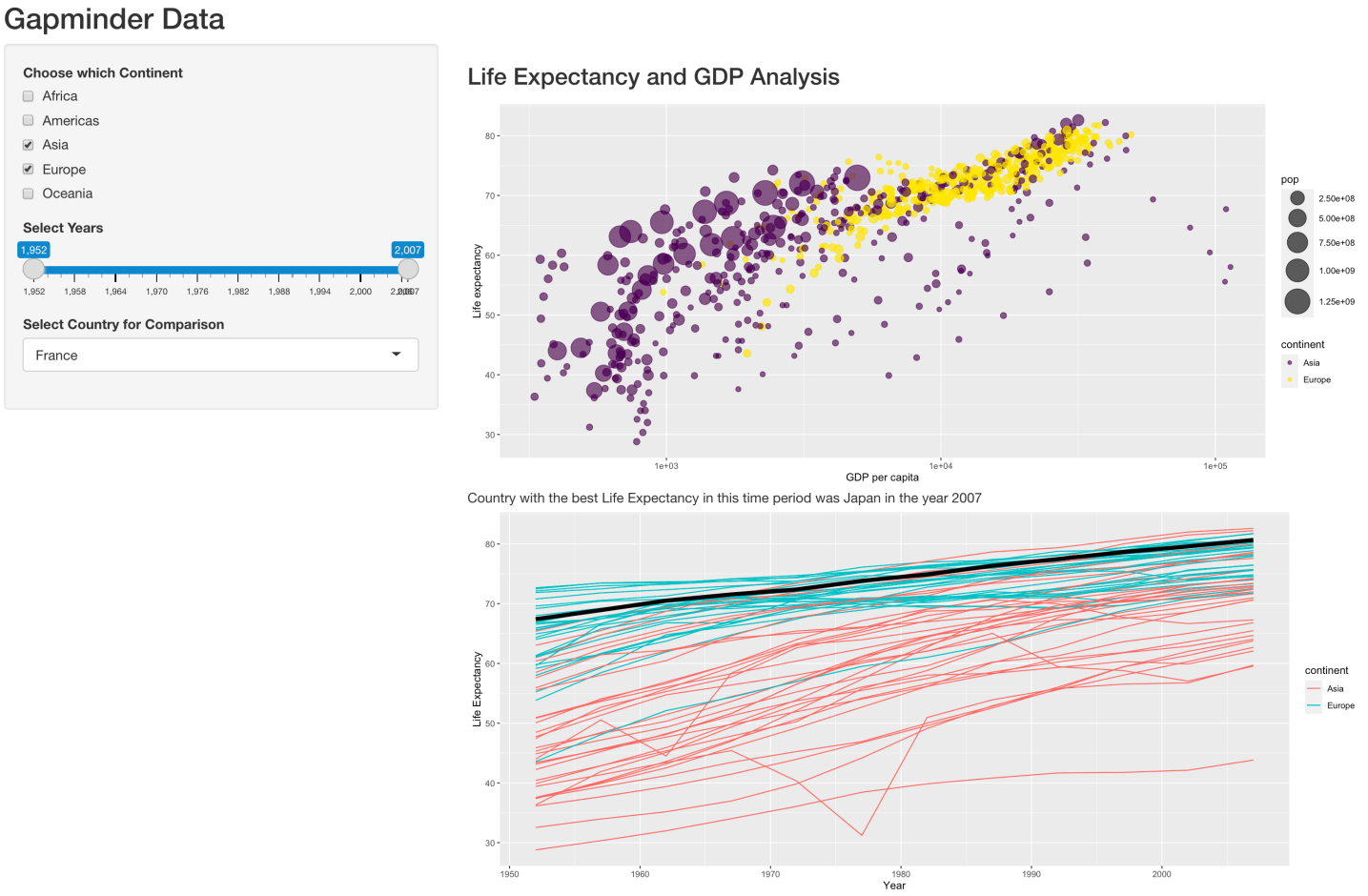
Questions

1. In the GitHub repository, there is another folder: `DataApp` folder, which has another `app.R`. Run that app and study the code carefully.
2. To the previous code, add a fourth dataset to the choice: "faithful".
3. Add a third row panel to the above app which makes a scatterplot using `plot()` for the chosen dataset.

We will do more R Shiny Development now! Note, you may find it very useful for your projects to see the demos provided on this website (with code). https://shiny.rstudio.com/gallery/#demos

There are tutorials available on the shiny website. Go through this Tutorial page and the first 3-4 lessons. This page tells you about the different types of interactive options. https://shiny.rstudio.com/tutorial/written-tutorial/lesson1/

We will use the `gapminder` dataset in `library(gapminder)`. In case this library is not loaded in your machine, you can read the csv provided using `read.csv()` and convert to a tibble using function `as_tibble()` in library `tibble`. This dataset contains information on GDP per capita and population by country. We are going to create an interactive widget to analyze this dataset. Your goal will be to create a shiny app that looks eventually like this:



This image has the following features:

- A title
- Choosing of continents on the side
- Choosing years for which to plot in the second plot
- Choosing the country to highlight in the second plot
- A header for the main panel
- Plot Number 1
- Some text that presents the country within the chosen continents with the highest Life Expectancy in the selected year range.
- Plot Number 2

We will need the following libraries:

```
library(ggplot2)
library(gapminder)
library(shiny)

# do ?gapminder to learn about the data
```

I give you code to make the two plots for the full dataset. For plot number 1 for the full dataset, the code is

```r
# Plot Number 1

# Scatterplot with size = population
# and colour = country
# then changing opacity
# scale ang log scale
p <- ggplot(
  gapminder,
  aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)
) +
  geom_point(show.legend = TRUE, alpha = 0.7) +
  scale_color_viridis_d() +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  labs(x = "GDP per capita", y = "Life expectancy")
p
```

For plot number 2 for the full dataset, the code is:

```r
# Plot Number 2

# year by lifeExp, grouped by country
# and colored by continent
p <- ggplot(gapminder,
            aes(year, lifeExp, group = country, color = continent)) +
  geom_line() +
  labs(x = "Year", y = "Life Expectancy")
p
```