

ASSIGNMENT 1

The goal of this assignment is to familiarize you with the fundamentals of R, data visualization, basic data wrangling, and exploration. The first part of this assignment is quite easy to complete because you will have a chance to check the solution at each step, interactively. However, I strongly recommend you do **not** check a solution before you attempt to answer a question. Otherwise, you will not learn, and you will struggle with the rest of the course. So, please!

If you don't cut corners by checking solutions even before you attempt, this assignment will take quite some time depending on how familiar you are with R. Please plan ahead and start early.

Assignment Instructions (Part I)

Please complete the following RStudio Cloud Primers and take the screenshots showing both the completion of each Primer and your username together. You can take screenshots on a PC using the [Snipping Tool](#) (Mac users, [see this](#)). A sample submission for Part I is on the last page.

Before you click on the following links and start working on the assignment, please visit [rstudio.cloud](#) and [log in](#) first. If you don't log in, your name will not come up in the screenshots.

This assignment has five mandatory parts:

1. [Data Visualization Basics](#)
2. [Programming Basics](#)
3. [Working with Tibbles](#)
4. [Isolating Data with dplyr](#)
5. [Deriving Information with dplyr](#)

and two highly recommended sections¹:

1. [Visualize Data](#)
2. [Tidy Your Data](#)

and two more recommended sections:

1. [Iterate](#)
2. [Write Functions](#)

¹ In the past, students who completed the recommended sections [increased the probability of getting an A by 39%](#), on average. Don't attempt to finish them all at once. You will **not** submit the optional sections in your assignment.

Assignment Instructions (Part II)

In this part of the assignment, you will apply what you learned in the first part. You will work on the Gapminder dataset, which is available as a library in R (how convenient!). You can read more about the source of the dataset [here](#), and enjoy [this talk](#) from 2007 if you have not yet done so. See [How to submit labs/assignments](#) to learn how to submit the second part of the assignment.

Use the following R libraries in this assignment: tidyverse, tidymodels, plotly, skimr, gapminder

Data Overview

1. country: Name of the country
2. continent: Name of the continent a country belongs in
3. year: Year for which the data is collected
4. lifeExp: Average life expectancy of the people in a country in a year
5. pop: Total population of a country in a year
6. gdpPercap: GDP per capita of a country in a year

Questions

1. Create an R Notebook

- a. Start with [the assignment template](#) and set your working directory using `setwd`
- b. From here on, use a new chunk for each question to make your code readable

2. Load the data

- a. In a new chunk, load the `gapminder` library, and use this line to create `dfGap`:
`dfGap <- gapminder`

3. Explore the data

- a. Use the `skim` function on the `dfGap` dataframe to get summary statistics in a nice format. I suggest you use the widest screen possible for the best reading.
- b. Filter `dfGap` for the year 2007 and sort it in descending order of life expectancy. Don't forget to use pipes!
 - i. What are the names of the countries with a life expectancy over 81?
- c. Add a calculated column `totalGDP` to `dfGap` showing the total GDP per country, filter the dataframe for 2007, and sort in descending order for `totalGDP`. If you like, save the new dataframe as a new one for repeated use.
 - i. What are some names of the countries with the top levels of total GDP?
 - ii. Which ones of these countries overlap with the countries from **3-b**?
 - iii. What if you selected only the two columns `country` and `gdpPercap` and sorted the dataframe in descending order for `gdpPercap`? Do you observe more of an overlap now? What do you infer from this difference?
- d. Filter `dfGap` for 2007, group it by continent, and then calculate the median life expectancy and *median* total GDP (so you need to have `totalGDP` already). Remember, you will pipe the filtered and grouped dataframe into `summarize()`

to get the medians. Then, sort it in descending order for the median life expectancy. Before you sort it, don't forget to use `ungroup()` to ungroup.

- i. What continent has the highest median of life expectancy?
- ii. Does it seem to be correlated with the median total GDP?

4. Visualize the data

- a. Now that you have explored the relationship between life expectancy and totalGDP in a table format, let's also visualize it to see a bigger picture.
 - i. Create a scatter plot to understand the relationship between life expectancy (y-axis) and totalGDP (x-axis) in 2007. Does this plot help?
 - ii. Copy the same code, but this time also filter for countries with a totalGDP of over a billion (use the scientific notation $1e+12$). What about now?
 - iii. Copy the same code, and add labels this time. Do you see a cluster now? What are the names of the countries that are outside of the cluster?
 - iv. Here is a pro tip. The labels you used in (iii) overlap and hide the points. This causes poor visibility. Install and load the `ggrepel` library. After that, copy the same code and use `geom_label_repel()` function instead of `geom_label()`. Does it look better now? Describe what has changed.
 - v. Copy the same code. This time, add a color for the continent. What are the continents that are missing from your visual? Why do you think so?
- b. You have an idea about the relationship between life expectancy and totalGDP even though you have not tested it statistically. Now, let's examine a more realistic relationship between life expectancy and gdpPercap (GDP per capita). Plot life expectancy (y-axis) against gdpPercap (x-axis) for 2007, add a smoothed line (no need to define any parameters, use the defaults). What do you observe about the overall relationship? Don't use any labels, just focus on the aggregate.
- c. Now let's find out the variations in life expectancy across different continents. Create box plots for each continent (in the same plot) and add a title this time. What do you observe? Describe your observations and answer the questions:
 - i. Which continent has the highest median life expectancy?
 - ii. Which continent has the largest range of life expectancy?
 - iii. Save your plot as `boxPlotsForAll` and put it into the `ggplotly()` function. More useful, right? Report the actual medians per continent by reading from the new interactive plot `ggplotly()` has created for you.
- d. Finally, it is time to create a more advanced (and likely more helpful) plot. Create a line plot to show how **median GDP per capita** by continent changes **over time**. [Hint: For the continents, use the color parameter]. Describe what you observe.
 - i. What continents have a clearer trend than others? Why do you think so?
 - ii. Change the summary metric from median to mean. What has changed? Why do you think so?
 - iii. Finally, don't you think these plots would be much more useful in plotly? Pick one and save it as `gdpOverTime` and call `ggplotly()` on it. You

can now read the actual GDP values per year. What are some of the breakthrough years (steep changes) for GDP in different continents?

Sample Submission for Part I (For Part II, refer to the assignment instructions above)

Your name and UID: Jane Doe (13474382)

1. Data Visualization Basics

Learn Guide What's New **Primers** Cheat Sheets

GO Gorkem Turgut OZER

≡ The Basics

Data Visualization Basics

- Welcome
- A code template
- Aesthetic mappings
- Geometric objects
- The ggplot2 package**

Start Over

Where to from here

Congratulations! You can use the ggplot2 code template to plot any dataset in many different ways. As you begin exploring data, you should incorporate these tools into your workflow.

There is much more to ggplot2 and Data Visualization than we have covered here. If you would like to learn more about visualizing data with ggplot2, check out RStudio's primer on [Data Visualization](#).

Your new data visualization skills will make it easier to learn other parts of R, because you can now visualize the results of any change that you make to data. you'll put these skills to immediate use in the next tutorial, which will show you how to extract values from datasets, as well as how to compute new variables and summary statistics from your data. See you there.

Previous Topic

2. Programming Basics

Learn Guide What's New **Primers** Cheat Sheets

GO Gorkem Turgut OZER

≡ The Basics

Programming basics

- Welcome
- Functions
- Arguments
- Objects
- Vectors
- Types
- Lists
- Packages**

Start Over

Congratulations!

Congratulations. You now have a formal sense for how the basics of R work. Although you may think of your self as a Data Scientist, this brief Computer Science background will help you as you analyze data. Whenever R does something unexpected, you can apply your knowledge of how R works to figure out what went wrong.

```
1 install.packages("dplyr")
2
3
```

Previous Topic

3. Working with Tibbles

Learn

Guide

What's New

Primers

Cheat Sheets



Gorkem Turgut OZER

Work with Data

Working with Tibbles

Recap

Welcome

babynames

tibbles

tidyverse

Start Over

Tibbles and the tidyverse package are two tools that make life with R easier. Ironically, you may not come to appreciate their value right away: these tutorials pre-load packages for you, and they wrap data frames into an interactive table for display (at least the tutorials in the primers that follow will). However, you will want to utilize tibbles and the tidyverse package when you move out of the tutorials and begin doing your own work with R inside of the RStudio IDE.

This tutorial also introduced the babynames dataset. In the next tutorial, you will use this data set to plot the popularity of *your* name over time. Along the way, you will learn how to filter and subset data sets in R.

Previous Topic

4. Isolating Data with dplyr

Learn

Guide

What's New

Primers

Cheat Sheets



Gorkem Turgut OZER

Work with Data

Isolating Data with dplyr

Recap

Together, `select()`, `filter()`, and `arrange()` let you quickly find information displayed within your data.

The next tutorial will show you how to derive information that is implied by your data, but not displayed within your data set.

In that tutorial, you will continue to use the `%>%` operator, which is an essential part of programming with the

Pipes help make R expressive, like a spoken language. Spoken languages consist of simple words that you combine into sentences to create sophisticated thoughts.

In the tidyverse, functions are like words: each does one simple task well. You can combine these tasks into pipes with `%>%` to perform complex, customized procedures.

Welcome

Your name

select()

filter()

arrange()

%>%

Start Over

Previous Topic

5. Deriving Information with dplyr

Learn

Guide

What's New

Primers

Cheat Sheets



Gorkem Turgut OZER

Work with Data

Derive Information with dplyr

Where to from here

Congratulations! You can use dplyr's grammar of data manipulation to access any data associated with a table—even if that data is not currently displayed by the table.

In other words, you now know how to look at data in R, as well as how to access specific values, calculate summary statistics, and compute new variables. When you combine this with the visualization skills that you learned in [Visualization Basics](#), you have everything that you need to begin exploring data in R.

The next tutorial will teach you the last of three basic skills for working with R:

1. How to visualize data
2. How to work with data
3. How to program with R code

Welcome

The most popular names

summarise()

group_by()

mutate()

Challenges

Start Over

Previous Topic