dominicwhite Initial commit acc5a93 · 8 months ago

img	Initial commit	8 months ago
.gitignore	Initial commit	8 months ago
DESCRIPTION	Initial commit	8 months ago
Makefile	Initial commit	8 months ago
README.md	Initial commit	8 months ago
penguins_visualization.Rmd	Initial commit	8 months ago

README

Assignment 2: Visualization by example

Instructions

This assignment will help you become more familiar with creating visualizations using R by guiding you through a few simple examples.

This GitHub repository contains a starter file named `penguins_visualization.Rmd`. Follow the instructions on Blackboard/Canvas to "clone" (i.e. import) the GitHub repository into RStudio. Click on the file in the RStudio *Files* tab (bottom right pane) to open it.

Read the About the dataset section to get some background information on the dataset that you'll be working with.

Each of the below exercises are to be completed in the provided spaces within your starter file `penguins_visualization.Rmd`.

When you're ready to submit, follow the directions in the **How to submit** section below.

About the dataset

In this assignment we will be working with a dataset of penguin measurements. These observations were measured from 344 penguins at Palmer Station in Antarctica.

The dataset contains 344 rows, each of which represents one of the 344 penguins. There are 8 columns, each of which represents a variable measured on the penguins, such as their species or their flipper length.

The dataset is included in the `palmerpenguins` package, and is a commonly used dataset to learn data visualization and exploration.

Exercises

Start working on these exercises **after** you have successfully cloned your repository into RStudio.

Use to proper grammar, punctuation, etc. when answering the exercises, as well as full sentences where appropriate.

And don't forget to change your name in the header section at the top of the `Rmd` file!

1. In this exercise we will write code to load two R *packages*.

   Every computer programming language includes *base* functions. When you install R, you automatically get these *base functions* as well. However, specialized work requires additional *packages*, or collections of functions that need to be installed separately. It is very important for data scientists to know how to use common packages and stay up-to-date on them, as this is vital to do good work.

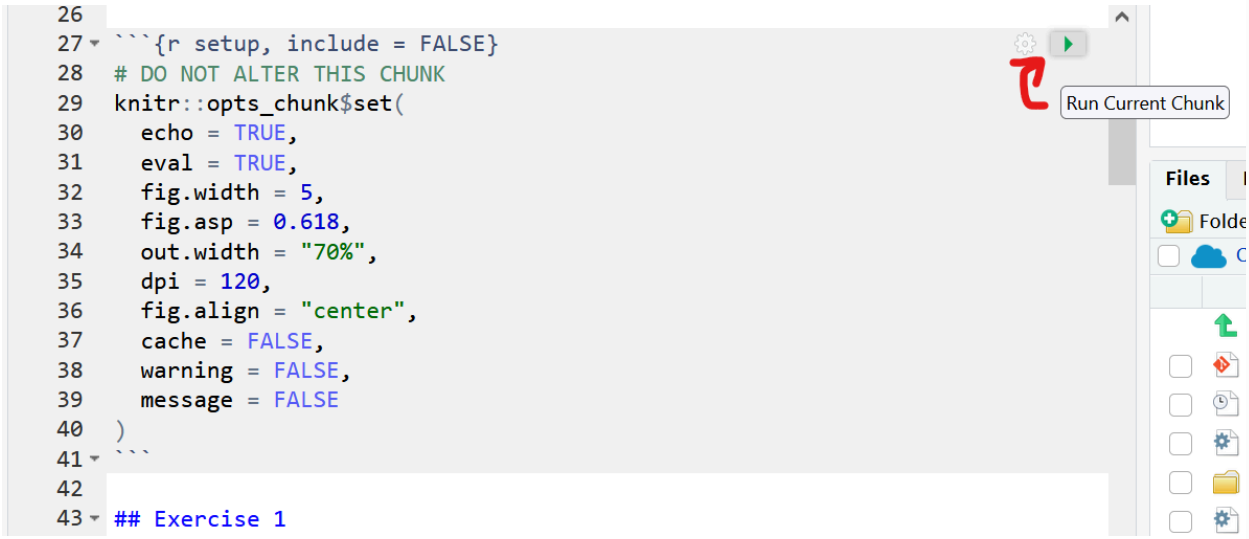
   Some common data science packages you might come across in R are `ggplot2` (for graphs) and `dplyr` (for data wrangling).

   The `tidyverse` is a collections of related packages that are all designed to work together, such as `ggplot` and `dplyr`. For this assignment, since we will need to use the `qplot()` function from the `ggplot2` package, we will load the `tidyverse` package that contains `ggplot2`.

   We will also need a package called `palmerpenguins`, which contains the dataset for this assignment.

   To do this assignment in RStudio, **first run the setup code chunk** (lines 27-41 of the `penguins_visualization.Rmd` file) that is already included at the top of your `penguins_visualization.Rmd` file. To run a code chunk, either:

- Click the green "play" button in the top right of the code chunk.



- Alternatively, click anywhere inside the code chunk (so that your cursor is flashing on one of the lines of code) and press the following keys at the same time: *Control* + *Shift* + *Enter* (or *Cmd* + *Option* + *Enter* on a Mac).

Next, install the packages by running this code **in your *Console* tab** of RStudio (do not put the code in the RMarkdown file): `install.packages("tidyverse", "palmerpenguins")` .

> **How to run code in the Console**
>
> The angle bracket `>` in the Console is called the *prompt*. Type (or copy-and-paste) code after the prompt, and then press Enter to run it.

This installs the packages into your local R library. However, when you want to use a package it needs to be *loaded* into R using the `library()` function. Since we will need to load the libraries to run the code in our `penguins_visualization.Rmd` RMarkdown document, we also need to add the `library()` functions here too.

Add the following code chunk under the Exercse 1 heading in the `penguins_visualization.Rmd` file.

```
```{r}
library(tidyverse)
library(palmerpenguins)
```
```

Excellent. Now run the code block, and observe what happens in the Console window.

As we learned in Assignment 1, it is a good idea to commit your work frequently (i.e. make a "checkpoint" with Git). In this assignment we will commit our work after exercise.

Once you have finished adding the code for this exercise in the `penguins_visualization.Rmd`:

- Save the `penguins_visualization.Rmd` file in RStudio, and then commit those changes. If you need a refresher, you can find instructions on how to do so in the course textbook.

- When making the commit, write a *commit message* that summarizes the work that you completed. Examples of good messages might be: "Completed Exercise 1" or "Answered Ex. 1 questions about the dataset".

You do not need to push to GitHub at this stage (i.e. Steps 4-6 of the instructions linked above), although you can do so if you wish.

> ## For posit.cloud users
>
> If you are using the online version of posit.cloud, then you will be asked for your GitHub credentials when you try to *Push* to GitHub for the first time. When this happens, type your GitHub username in the pop-up that asks for your username, but in the second pop-up, copy-and-paste your GitHub token into the "Password" box. If you do not remember your token, you will need to create another one (remember to check the `Repo` permissions box).

2. You should always inspect a new dataset to become more familiar with what it contains. The `View()` function invokes a spreadsheet like data viewer in a new tab in Rstudio.

   Type the following command **in your *Console* window** (not in the R Markdown file):

   ```
   View(penguins)
   ```

   (If you get an error message that says `object penguins not found` then you have not run the code from Exercise 1. Go back to Exercise 1 and rerun the code chunk. When you close RStudio, the output of any code chunks is forgotten, included loaded packages, so you will need to rerun all code chunks whenever you reopen an exisiting assignment.)

   Inspect the table that comes up and use it to answer the following questions (remember to write you answers in the `penguins_visualization.Rmd` file in the space provided for Exercise 2).

   i. How many rows and columns does this dataset have?

   ii. What does a row in this dataset represent (i.e. what is the unit of observation)?

   iii. What are three categorical variables in the `penguins` dataset?

   iv. What are four continuous variables in the `penguins` dataset?

   v. Which variable in the `penguins` dataset could be treated as either continuous or categorical, depending on the context in which it is used?

   vi. What are the three species of penguin in the dataset?

   You never need to copy the exercise instructions into your answer file - you can just write the answers (we know what the questions are! However, you may find it helpful to put the answers in a matching list - see the hint below for a reminder on how to do so.)

   When you are finished writing your answers, make another commit (in the same way that you did at the end of Exercise 1).

> **Creating list with Roman numerals in RMarkdown**
>
> If you wish to create a list of items that are numbered with Roman numerals (e.g. `i.`, `ii.` etc.) then you can do so by writing these numerals at the start of the line:
>
> ```
> i. This is the first item.
>
> ii. This is the second item.
> ```

3. Create your first graph by using the `plot()` function, which is usually loaded by default into R. In the Exercise 3 section of your starter file `penguins_visualization.Rmd`, create an R code block by typing:

    ````
    ```{r}

    ```
    ````

    You will **always** need to create an R code block by typing the above if you want to execute code in your R Markdown file. Then, type the following code inside the R code block, like so:

    ````
    ```{r}
    plot(x = penguins$flipper_length_mm)
    ```
    ````

    Run the code by either clicking the green "play" button in the upper right corner of the code chunk or, with your cursor still in the block, by pressing `Ctrl` + `Shift` + `Enter`. The type of plot it creates is called a *scatter plot*.

    Notice the syntax inside the function. `penguins` is the name of the dataset. `flipper_length_mm` is the name of the variable. They are connected by the `$` symbol. This symbol is an example of an *operator*. The `$` operator is used to select an element from a data component. Here, we are telling R to go to the `penguins` dataset, select the column `flipper_length_mm`, and use that variable for the `plot()` function.

    > **Operators**
    >
    > More information on operators can be found in the course textbook

    Once you have finished this exercise, commit save the file, and make another commit. A good commit message might be "Finished Exercise 3" or "Created exercise 3 scatterplot".

    > **Important!**
    >
    > The backticks for creating the R code block **are assumed to be present in all subsequent examples.** You will need to add them to your answer file yourself.
    >
    > So if you see this:
    >
    > ```
    > 1 + 1
    > ```

> You should type **this** in your R Markdown file:
>
> ```
> ```{r}
> 1 + 1
> ```
> ```
>
> If you find that typing out lots of backticks is a nuisance, then you can either use a keyboard shortcut to insert an R code chunk (Windows: Ctrl+Alt+I, Mac: Cmd+Option+I), or you can click on the green "Insert" button at the top of the file editor pane and select the "R" option.

4. i. Next, let's create a graph using the `ggplot2` package we just installed with `tidyverse`. Create another graph of `flipper_length_mm` using the `qplot()` function. Note the syntax inside of the `qplot` function is different from the `plot()` function.

   qplot(x = flipper_length_mm, data = penguins)

   ii. Write a few sentences beow your code comparing and contrasting the two plots in Exercises 3 & 4. For example, What types of graphs are they? How is the syntax different? What operators did you use in each?

   When you have finished this exercise, make another commit.

5. The histogram in the last exercise was a little jagged. For example, there are two peaks in the distribution, but their shape is obscured by noise in the data. We can *smooth* the distribution by plotting fewer (wider) bins.

   i. Let's start by setting the number of bins. Use the code snippet from exercise 2 as a starting point. Copy it to a new code chunk in the Exercise 4 answer section, and add `bins = 15` to the list of arguments passed to the `qplot(...)` function.

   Your code should be something like this:

   ```
   qplot(x = flipper_length_mm, ..., data = penguins)
   ```

   ii. Create another new code chunk in your answer file, and copy the histogram code from part (i) of this exercise. Then **change** the `bins=15` parameter to `binwidth=1`.

   iii. What do the `binwidth = ....` and `bins = ....` parameters do? What is the different between the two? You can find help in the documentation of `ggplot2` histograms.

   Once you have finished this exercise, commit your work again with a suitable commit message.

   After making this commit, let's also push our work to GitHub by clicking the green upwards arrow in the Git tab of RStudio.

Finally go to the GitHub repository that you created for Assignment 2 and check that all five of your commits have successfully been pushed to GitHub. (If you refresh the GitHub repository webpage you should notice the most recently pushed commit shows up near the top of the page, as well as the total number of commits.)

> **How often should you push to GitHub?**
>
> We can push after every commit, or after several commits - how often you do so does not matter for the purposes of this class.
>
> "Pushing" is the process of syncing the history of checkpoints between your local copy of the project (in RStudio) and the remote copy of the project (on GitHub). By pushing your commits, you back up your work and allow collaborators to download ("Pull") your edits into their own local copy of the project (and, for the purposes of this class, you allow us to see and grade your assignment!).

6. i. Create a scatter plot by fixing this code to a new code chunk in the *Exercise 6* section of your answer file:

```
qplot(x = penguins$flipperlength, y = penguins$bodymass)
```

   Fix the errors in code, run the code block, then look at the output. *hint: there are ~4 mistakes.* Then answer the following questions in your file:

   ii. What variable is on the y-axis of this scatter plot?

   iii. Is there a relationship between these two variables, and if so, is it linear (i.e. straight) or non-linear (i.e. curved)?

   When you have completed this exercise, make another commit.

7. In the *Exercise 7* section of your answer file, we'll create a new code chunk with another scatter plot, but this time we will tell R we want to add a linear trend line.

   Note that when you plot a graph with a trend line, you should almost always show the underlying data as well. This is because a trend line is a summary of the average trend in the data, but that can be misleading if the underlying data contains patterns that are averaged out in the trend line.

   We will add the following two arguments to your `qplot()` code from Exercise 6 (i.e. the one that created a scatter plot of body mass vs bill length): `geom = c("point", "smooth")` and `method = "lm"`

   Fix the errors in broken code template below and run in *Exercise 7* section of your RMarkdown answer file:

```
qplot(
   data = penguins,
   geom = c("point", "smooth")
   method = "lm"
)
```

Hint: there are 2 lines of missing code and 1 other error.

> **What are the `geom` and `method` parameters doing?**
>
> We are passing a vector of two geometries to the `geom` parameter, so you should see both the points of the scatter plot as well as a smoothed trend line through the data.
>
> The `method` parameter allows us to specify that the trend-line should be a straight line ("*lm*" means linear model, i.e. a line-of-best-fit) instead of the default wiggly line.)

Commit your work for this exercise.

8. In this exercise you will create another scatterplot using the `qplot()` function, but this time plot `bill_length_mm` on the x-axis and use the `color` parameter to color the data points by species. By setting `color = ....`, where the ellipsis is some categorical variable, we can break down our plot into different parts depending on the value of that categorical variable.

   Here is a code template to get you started - this is missing a number of parts that you will need to fill in to turn in into the correct `qplot()` scatter plot.

   ```
   x = bill_length_mm
   y = body_mass_g
   color = species
   geom = "point"
   data = penguins
   ```

   Commit your code and answers when you have finished this exercise.

9. Let's investigate the spread in correlation between flipper length and body mass that we say in Exercise 7. (After all, we might expect that bigger penguins have longer flippers.)

   Create a scatterplot from scratch that incorporates elements from Exercises 6, 7, and 8.

   This plot is created with the `qplot()` function, and has these elements:

   - Datapoints are colored by species
   - Pass "smooth" and "point" through the `geom = ...` parameter
   - Create a linear model line using the `method = ...` parameter

   Commit your work and push to GitHub again. If you have made a commit at the end of each exercise you should see at least 9 commits. If you made extra commits (e.g. correcting one of your answers) then you may have even more commits (*it's better to have more commits than fewer*).

How to submit

To submit your assignment, follow the two steps below. Your work will be graded for credit **after** you've completed both steps!

1. If you have any uncommitted work: save, commit, and push your completed R Markdown file so that everything is synchronized to GitHub. If you do this right, then you will be able to view your completed file on the GitHub website.

2. Knit your R Markdown document to the PDF format, *proof-read the PDF*, and then upload it to *Assignment 2* posting on Blackboard/Canvas.

Cheatsheets

You are encouraged to review and keep the following cheatsheets handy while working on this assignment:

- ggplot2 cheatsheet

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- **Makefile** 100.0%

Suggested workflows

Based on your tech stack

Build projects with Make — Configure

Build and test a project using Make.

SLSA Generic generator — Configure

Generate SLSA3 provenance for your existing release workflows

More workflows

Dismiss suggestions