Homework 03: Creating a Plot-Saving Function

Overview

You are now familiar with using `{ggplot2}` to create basic plot objects. In subsequent topics, you will make more advanced plots but for now, you need a reusable way to save plots as high-quality files which are easy to present on a webpage. The `ggsave()` function will save plots as `.png` but every time you need to save a plot, you will need to specify arguments to parameters in order to ensure uniformity across those visualization (e.g., height, width, resolution, etc.). This approach is inefficient.

For this homework, you will automate this saving step by:

- writing a function to save plot objects; that you will
- `source()` within you your plotting scripts; so that you
- standardize various save settings; in order to
- reproduce all saved plots (at least those using the function), so that
- changes to the function itself will automate updates for all plots

Collaboration

I understand that many of you have not written functions for some time. If you reviewed the **R Basics and Refreshers at the course site**, you know there are reminders about functions and arguments there. Many elements, however, will be reintroduced here.

Due the nature of this task, I encourage you to collaborate with a partner. You are also able to use an AI to help you build the function but you also have to know the difference between parameters and arguments and you are responsible for the function working correctly. Please don't create the function and share it with someone because a) that's not the assignment, b) is an academic integrity violation, c) robs you of the learning experience, and d) will challenge your ability to fix issues that might become part of your project or be related to elements of knowledge assessments.

Steps

1: Review the docs for `ggsave()`

`ggsave()` will be the base function for your function, so you will need to understand its parameters.

Use: `help("ggsave")` or `?ggsave`

An abbreviated version is:

```
ggsave(
  filename,
  plot = last_plot(),
  device = NULL,
  ...
)
```

2: Create a `.R` Function Script

Starting with the starter script, create a function called `save_plot_png()`. Naming your file with the same name as your function would be a good idea so can identify it easily. Save your code file where you think function code should go.

3: Declare the Function and Parameters

A function needs a name, and in this case, parameters too. The `function()` function is used to tell R that `save_plot_png` will be a special object, namely a function object. Parameter names and default arguments will go inside the `()` and your code instructions will go inside `{}`.

Start with:

```
save_plot_png <- function(
   # your parameters, with arguments for any defaults

   ...

   ) {


   }
```

Ensure that your function has these required parameters (some have the same name as `ggsave()`:

- `plot`: a `{ggplot2}` plot object needed to be saved as `.png`
- `file_name`: the name of the output `.png` file (e.g., `"my_plot.png"`)
- `figs_dir`: the path to where you will save your figs for your website, built using `{here}`
- `units`: the units for the dimensions (recommend pixels, "px")
- `width`: the plot width in units
- `height`: the plot height in units
- `dpi`: the resolution in dots per inch

If you do not have default argument settings (e.g., the file name), do not set an argument to that parameter. As you can see in `ggsave()` `filename` has no default argument.

You will want your plots dimensions to look good on the medium you expect your audience to engage. Width and height can be set to `"1600"` and `"1100"` pixels respectively but searching online or with an AI for some advice given how you believe someone would use your website (e.g., computer, phone, etc.) might result in some adjustments.

3: Adding Function Instructions

An Example Parameters alone don't make a function. Without any instructions, your parameters will be fairly useless. Using `ggsave()` as your helper function, you will need to ensure that the arguments that you pass to your function will be used as arguments passed to the parameters of the `ggsave()` function. Seeing an example might help.

An example function based on `mean()`:

```
mean_replicated <- function(
    some_numeric_vector
    ) {
  mean(x = some_numeric_vector, na.rm = TRUE)
}
```

Explanation: `mean()` is a function which calculates the mean of a numeric vector and returns a single value. `x` is one of the parameters of `mean()` and `x` needs an argument passed to it, otherwise `mean()` won't have what it needs to do its work. The argument you need to pass is any numeric vector, for example, `c(1, 2, 11)`.

`mean_replicated()` is a silly function built on `mean()`. `mean_replicated()` has a single parameter, `some_numeric_vector` with no default argument. Because the function has code instructions, `mean(x = some_numeric_vector, ...)`, any vector that is passed as an argument to the `some_numeric_vector` parameter of `mean_replicated()` becomes the argument passed to the `x` parameter of the `mean()` function. Look back at the code flow to see this.

To test the function, you would pass a numeric vector to `some_numeric_vector`, like so:

```
mean_replicated(some_numeric_vector = c(1, 2, 11))
```

Add Code Instructions As with `mean()` use in the above example, use the same process to build your instructions for `save_plot_png()` by leveraging `ggsave()`. Ensure that you include all of the required parameters and that you use a second set of eyes to ensure the arguments are passing correctly.

4: Test Your Function

Define your function by executing the code so that you can use it. Using some data and a ggplot, verify that your plot object will be saved as a `.png` and in the correct directory and with the name you decide.

Tip: When you build your path, you can combine `figs_dir` and `file_name` in order to build the full path.

5: Load Libraries and Document your Script

Once you are do with your function, ensure that and libraries needed are loaded and ensure you document your script so that the elements are clear. If you work on the script with a peer, authorship includes contributors.

Upload your `.R` script to: https://ln5.sync.com/dl/a038628f0/wwfifjxk-f7rfshin-rkedi3y8-77f9zaii