Foundational Skills

Getting started

First, you will need to download the latest versions of R Studio and R. Although you will likely exclusively use R Studio, this software (R Studio) needs to have R installed, as well, as R Studio uses R behind-the-scenes. Both are freely-available, cross-platform, and open-source.

Downloading R Studio

To download R Studio:

- Visit this page here: https://www.rstudio.com/products/rstudio/download/#download
- Beneath "Installers for Supported Platforms", find your operating system and download and install the application.

Downloading R

To download R:

- Visit this page here to download R: https://cran.r-project.org/
- Find your operating system (Mac, Windows, or Linux)
- Download the 'latest release' on the page for your operating system and download and install the application

Don't worry; you will not mess anything up if you download (or even install!) the wrong file. Once you've installed both, you can get started.

If you do have issues, consider this page, and then reach out for help. One good place to start is the R Studio Community is a great place to start.

Check that it worked

Find the console

Configuring R Studio

There are a number of changes you *can* (but do not need to) make to configure R Studio. If you navigate to the Preferences menu in R Studio, you'll see a number of options you can change, from the appearance of the application to which windows appear where.

One important consideration is whether to save your workspace when you close R Studio. By default, R Studio saves all of the objects in your environment. This means that any data that you have loaded–or new data or objects that you have created, such as by merging two datasets together or creating a plot–will, by default, still exist when you open R Studio next. In general, this is not ideal, because it means that you may have taken steps interactively that are not documented your code. This means that when you share your code, or re-run it from the start, it may not work. An easy way to change this is to tell R Studio to start from scratch (in terms of your workspace) each time you open it. You can do that by changing the dropdown menu pointed out in the image below to "Never".

While this may seem like a dramatic step - never saving your workspace - it is the foundation for doing reproducible work and research using R Studio (and R). It also represents one of the biggest shifts from using software like Excel or SPSS, where most steps are not documented in code. This involves a shift from

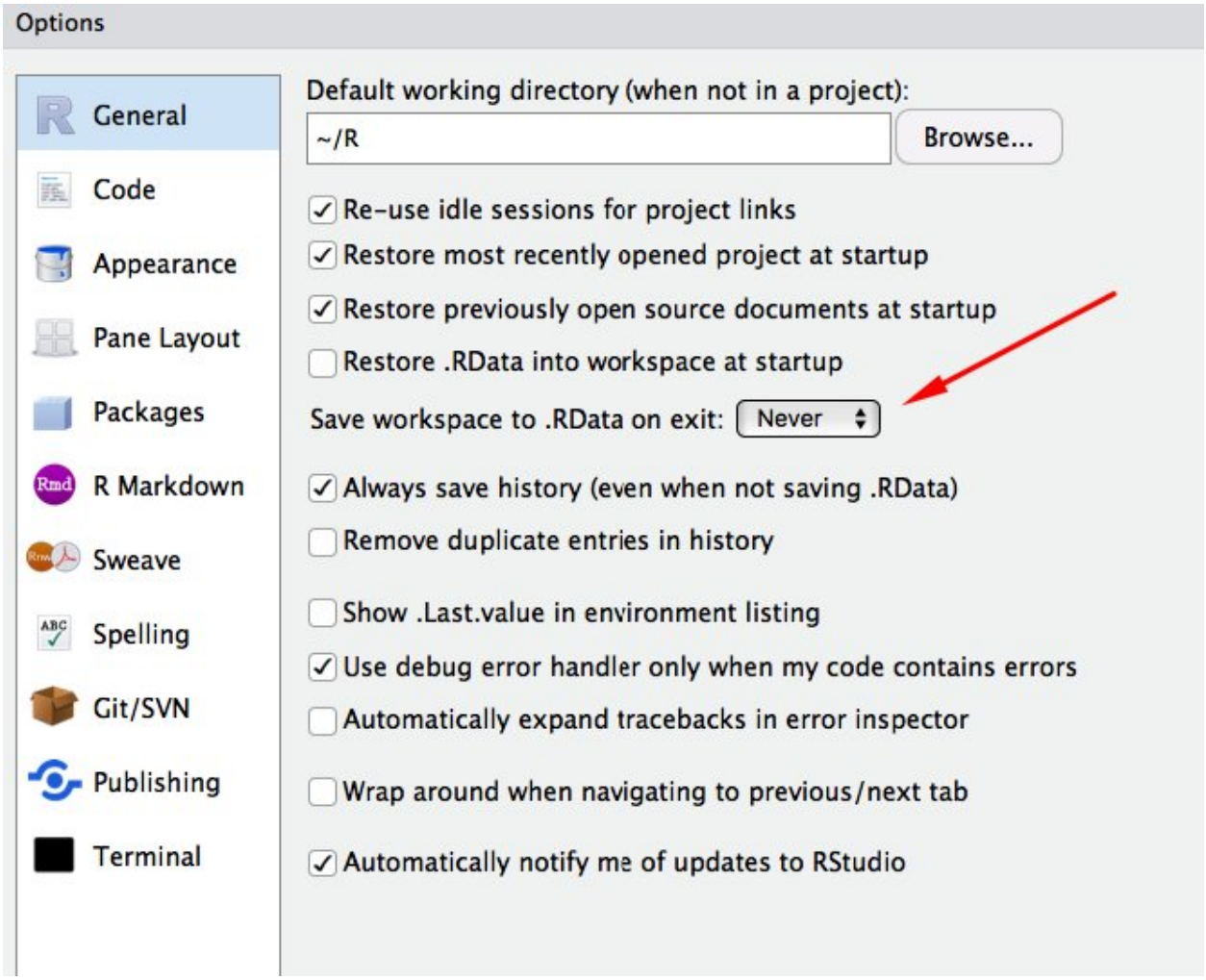

Figure 1: optional caption text

thinking that your most permanent and important part of an analysis is your data to thinking of the most important part as being the code: with the code, you can keep your data in its original form, process it, and then save a processed file, through running code. This also means that when you have to make a change to this code, you can re-run the entire analysis easily.

Help, I'm completely new to using R / R Studio!

If you're completely new, Swirl is a great place to start, as it helps you to learn R *from within R Studio*. Visit this page to see some directions: http://swirlstats.com.

If you have a bit more confidence but still feel like you need some time to get started, Data Camp is another good place to start.

And if you're ready to go, please proceed to the next sections on processing and preparing, plotting, loading, and modeling data and sharing results.

Processing / preparing data

One of the most important steps is to process and prepare data. This walkthrough starts here instead of loading data because of the importance of these steps and because they're exciting places to begin: you can quickly transform an unruly dataset into one ready to be plotted or modeled.

Plotting data

Importing / exporting data from various sources

Loading data

You might be thinking that an Excel file is the first that we would load, but there happens to be a format which you can open and edit in Excel that is even easier to use between Excel and R and among Excel, R, as well as SPSS and other statistical software, like MPlus, and even other programming languages, like Python. That format is CSV, or a comma-separated-values file.

The CSV file is useful because you can open it with Excel and save Excel files as CSV files. Additionally, and as its name indicates, a CSV file is rows of a spreadsheet with the columns separated by commas, so you can view it in a text editor, like TextEdit for Macintosh, as well. Not surprisingly, Google Sheets easily converts CSV files into a Sheet, and also easily saves Sheets as CSV files.

For these reasons, we start with reading CSV files.

Loading CSV files

The easiest way to read a CSV file is with the function `read_csv()`. It is from a package, or an add-on to R, called `readr`, but we are going to install `readr` as well as other packages that work well together as part of a group of packages named the `tidyverse`. To install all of the packages in the tidyverse, use the following command:

```
install.packages("tidyverse")
```

You can also navigate to the Packages pane, and then click "Install", which will work the same as the line of code above. Note, here there is a way to install a package using code or part of the R Studio interface. Usually, using code is a bit quicker, but sometimes (as we will see in a moment) using the interface can be very useful and sometimes complimentary to use of code.

We have now installed the tidyverse. We only have to install a package once, but to use it, we have to load it each time we start a new R session. We will discuss what an R session is later on; for now, know that we have to load a package to use it. We do that with `library()`:

```
library(tidyverse)
```

And now we can load a file. We are going to call the data `student_responses`:

Since we loaded the data, we now want to look at it. We can just type its name, and a summary of the data will print:

This was a minor task, but if you loaded a file and printed it, give yourself a pat on the back. It is no joke to say that many times simply being able to load a file into new software. We are now well on our way to carrying out analysis of our data.

Loading Excel files

We will now do the same with an Excel file. You might be thinking that you can simply open the file in Excel and then save it as a CSV. This is generally a good idea. At the same time, sometimes you may need to directly read a file from Excel, and it is easy enough to do this.

The package that we use, `readxl`, is not a part of the tidyverse, so we will have to install it first (remember, we only need to do this once), and then load it using `library(readxl)`. Note that the command to install `readxl` is grayed-out below: The `#` symbol before `install.packages("readxl")` indicates that this line should be treated as a comment and not actually run, like the lines of code that are not grayed-out. It is here just as a reminder that the package needs to be installed if it is not already.

Once we have installed readxl, we have to load it (just like tidyverse):

```
# install.packages("readxl")
library(readxl)
```

We can then use `read_xlsx()` in the same way as `read_csv()`:

```
x <- read_xls()
x <- read_xlsx()
```

And we can print the data we loaded in the same way:

Loading SAV files

The same factors that apply to reading Excel files apply to reading `SAV` files (from SPSS). First, install the package `haven`, load it, and the use the function `read_sav()`:

```
# install.packages("haven")
library(haven)

x <- read_sav("")
```

Google Sheets

Finally, it can sometimes be useful to load a file directly from Google Sheets, and this can be done using the Google Sheets package.

```
# install.packages("googlesheets")
library(googlesheets)
```

When you run the command below, a link to authenticate with your Google account will open in your browser.

```
my_sheets <- gs_ls()
```

You can then simply use the `gs_title()` function in conjunction with the `gs_read()` function:

```
df <- gs_title('title')
df <- gs_read(df)
```

Conclusion

For more on reading files, we will discuss how to use functions to read every file in a folder (or, to write many different files to a folder).

Modeling data

Communicating / sharing results