Day Six: Error Resolution

SDS 192: Introduction to Data Science

Lindsay Poirier Statistical & Data Sciences, Smith College

Spring 2022

Introduction

The goals of this lab are to provide you with practice in problem-solving and error resolution in `R`. Specifically, you will be practicing.

1. How to interpret error messages in `R`
2. How to generate reproducible examples
3. How to read `R` cheatsheets
4. How to access `R` help pages
5. How to reference Stack Overflow and other online resources for help

Pre-requisites:

Up until this point we have been working with what is called base `R`. This is comprised of the functions that come pre-packaged in the `R` programming languages. Many functions that we will use in this course are not in base `R`, however.

The R community has developed a series of *packages* or collections of functions that we can use when coding in `R`. In the coming weeks, we will be leveraging a series of packages known as the *Tidyverse*, which are designed to help users clean, transform, analyze, and visualize data. The Tidyverse is actually comprised of a number of packages, including `dplyr`, `ggplot`, `stringr`, and `reprex` - all of which we have access to once we install the package.

You will need to install the *Tidyverse* in your `R` environment. To do so, you should copy and paste the following line into your Console, removing the backticks before and after the function call.

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

> Note: We only need to install packages once to have their functions available for us in our R environments. In other words, we do not need to run `install.packages("tidyverse")` every time we want to reference tidyverse functions in our code. However, occasionally the creators of packages will update and improve the functions. To get these improvements, we may wish to update packages we have installed on our computers. To see a list of packages installed in your R environment, you can turn your attention to the Files Pane in RStudio, and click on the Packages tab. This lists the packages installed on your computer and their version number. You can update packages to their newest version directly from this tab.

Once packages are installed, their functions are not readily available for us to use until we load them into our workspace. To load the package, we call the function `library()`, which let's `R` know that we will be

referencing the functions encoded for this package in our code. You can think of an R package like a book, stored in a computer library. When you call, `library()`, it is as if we are checking that book out of the library so that we may reference its functions in our code.

Run the line of code below to load the functions in the tidyverse library into our workspace.

```
library(tidyverse)
```

Interpreting Error Messages

Throughout this week, we have taken a look at different error messages that R presents when it can't evaluate our code. Today, we will consider these in more detail. First, it's important to make some distinctions between the kinds of messages that R presents to us when attempting to run code:

- *Errors* terminate a process that we are trying to run in R. They arise when it is not possible for R to continue evaluating a function.
- *Warnings* don't terminate a process but are meant to warn us that there may be an issue with our code and its output. They arise when R recognizes potential problems with the code we've supplied.
- *Messages* also don't terminate a process and don't necessarily indicate a problem but simply provide us with more potentially helpful information about the code we've supplied.

Exercise 1: Check out the differences between an error, a warning, and a message in R by reviewing the output in the Console when you run the following code chunks.

Error in R

```
sum("3", "4")
```

```
## Error in sum("3", "4"): invalid 'type' (character) of argument
```

Warning in R

```
Inf/Inf
```

```
## [1] NaN
```

Exercise 2: Run the codes below and check out the error messages. Review the code to fix each of the errors.

```
# Create a vector
a <- 1, 2, 3, 4, 5
```

```
## Error: <text>:2:7: unexpected ','
## 1: # Create a vector
## 2: a <- 1,
##          ^
```

```r
global_landslide <- read.csv("https://data.nasa.gov/api/views/dd9e-wu2v/rows.csv")
```

re