Entering / cleaning data 1

```
## -- Attaching packages --------------------------------------
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## -- Conflicts -----------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

File pathnames

Relative versus absolute pathnames

When you want to reference a directory or file that is not in your working directory, you need to give R the directions for how to find the file. You can use one of two types of pathnames:

- *Relative pathname*: How to get to the file or directory from your current working directory
- *Absolute pathname*: How to get to the file or directory from anywhere on the computer

Relative versus absolute pathnames

Say your current working directory was `/Users/brookeanderson/RProgrammingForResearch` and you wanted to get into the subdirectory `data`. Here are examples of referencing that subdirectory using the two types of pathnames:

Absolute:

```
"/Users/brookeanderson/RProgrammingForResearch/data"
```

Relative:

```
"data"
```

Relative versus absolute pathnames

Both methods of writing filenames have their own advantages and disadvantages:

- *Relative pathname*: Which file you are indicating depends on which working directory you are in, which means that your code will break if you try to re-run it from a different working directory. However, relative pathways in your code make it easier for you to share a working version of a project with someone else. For most of this course, we will focus on using relative pathnames, especially when you start collaborating.
- *Absolute pathname*: No matter what working directory you're in, it is completely clear to your computer which file you mean when you use an absolute pathname. However, your code will not work on someone else's computer without modifications (because the structure of their computer's full directory will be different).

Relative versus absolute pathnames

I **strongly** recommend saving your data files somewhere in the directory structure of the R Project in which you're working and then using **relative pathnames** to reference that file when you need to read it in.

This practice establishes good habits for making your research computationally reproducible.

Getting around directories

There are a few abbreviations you can use to represent certain relative locations:

Shorthand	Meaning
`.`	Current working directory
`..`	One directory up from current working directory (parent directory)
`../..`	Two directories up from current working directory
`../data`	The 'data' subdirectory of the parent directory

One other useful abbreviation is `~`, which is shorthand for your **home directory**. On Windows, this will take you to your "Documents" folder.

Relative versus absolute pathnames

Here are some examples of relative pathnames that use these abbreviations:

If `data` is a subdirectory of your current parent directory (i.e., from your working directory, you need to go "up and over"):

```
"../data"
```

If `data` is a subdirectory of the subdirectory `Ex` of your current working directory:

```
"Ex/data"
```

Reading in an online flat file

Once you understand this idea of giving R directions to a file to read it, you shouldn't be too surprised that R can also do this for flat files that are hosted online.

In this case, the file isn't even on your computer, so you need to give R the directions to find it online. You do that by putting the file's online address as the file name. For example, to read in the shipwreck data directly from GitHub, you can run:

```
shipwreck_url <- paste0("https://raw.githubusercontent.com/",
                        "geanders/RProgrammingForResearch/",
                        "master/data/",
                        "AWOIS_Wrecks_KnownYear.tab")
shipwrecks <- read_tsv(shipwreck_url)
```

Taking advantage of `paste0`

You can create an object with your directory name using `paste0`, and then use that to set your directory. We'll take a lot of advantage of this for reading in files.

The convention for `paste0` is:

```
## Generic code
[object name] <- paste0("[first thing you want to paste]",
                        "[what you want to add to that]",
                        "[more you want to add]")
```

Taking advantage of `paste0`

Here's an example:

```
shipwreck_url <- paste0("https://raw.githubusercontent.com/",
                        "geanders/RProgrammingForResearch/",
                        "master/data/",
                        "AWOIS_Wrecks_KnownYear.tab")
shipwrecks <- read_tsv(shipwreck_url)
```