

02-01: File Inputs

1 - Purpose

- Explaining the concept of a working directory
- Create a CSV file using R
- Get data from a CSV file and save it to a variable
- Introduce variable with multiple values

2 - Questions about the material...

If you have any questions about the material in this lesson *feel free to email them to the instructor here*.

3 - The R Root folder

The R Fundamentals class was originally done without using RStudio Projects. In the original class, I created a folder called **R Root** and set it as the root folder (or working directory) for all material in the class.

Functionally, **R Root is the same as the Root Folder** in your RStudio Project -- and all code in the lessons will work in your project.

4 - Data from files

Up until this point, all of the data used in our scripts have either been hardcoded (e.g., **myVal = 4**) or input from the user (e.g., **age = readline("How old are you?")**). However, you are often dealing with data sets that exists in a separate file, often a CSV (Comma-separated Values) file. The data is usually in the form of a data frame, which is data containing a series of related values (e.g., multiple types of weather measurements over a 15 day period). The data from the file is read in by the R script and saved to a variable -- similar to how R can use **readline()** to assign user input values to a variable.

Extension: Microsoft Excel and CSV files

5 - Data files

The first step is to create a data file that will be read in by an R Script. The data file will contain weather data from Lansing, Michigan (where yours truly is located).

The data file we will create is called a comma-separated value (CSV) file. A CSV file is pure text file and any text editor (e.g., Notepad, TextEdit) can be used to create, read, or modify the file. RStudio can also create, read, and modify a CSV files so, for simplicity, I am going to use RStudio to create the file.

5.1 - Information for the data file

The data file will contain weather information for Lansing, Michigan over a 14 day period (3/27/17 to 4/9/17). Each line will contain a high temperature and a low temperature. On the first line I put the labels **highTemp** and **lowTemp**, which R will treat as the headers for the data (*Fig.1*).

```
1 highTemp, lowTemp
2 57,45
3 50,43
4 54,42
5 40,38
6 39,37
7 58,45
8 60,46
9 53,50
10 55,48
11 44,40
12 39,36
13 53,43
14 61,45
15 75,63
```

Fig 1: Temperature data to be saved to a csv file.

5.2 - Creating a data file in R

1. In RStudio, create a new text file (**File -> New File -> Text File**)
2. Copy the 14 high temperatures and 14 low temperatures (directly from [Fig.1](#)) along with the **highTemp** and **lowTemp** headers to the new text file you just created.
3. Save the file as **LansingWeather.csv** (**File -> Save As...**) in the **data** folder *inside the Root Folder* ([Fig.2](#)).

Note: **LansingWeather.csv** is a data file -- not an R script file. Data files are read in by scripts files -- they are not executed on their own.

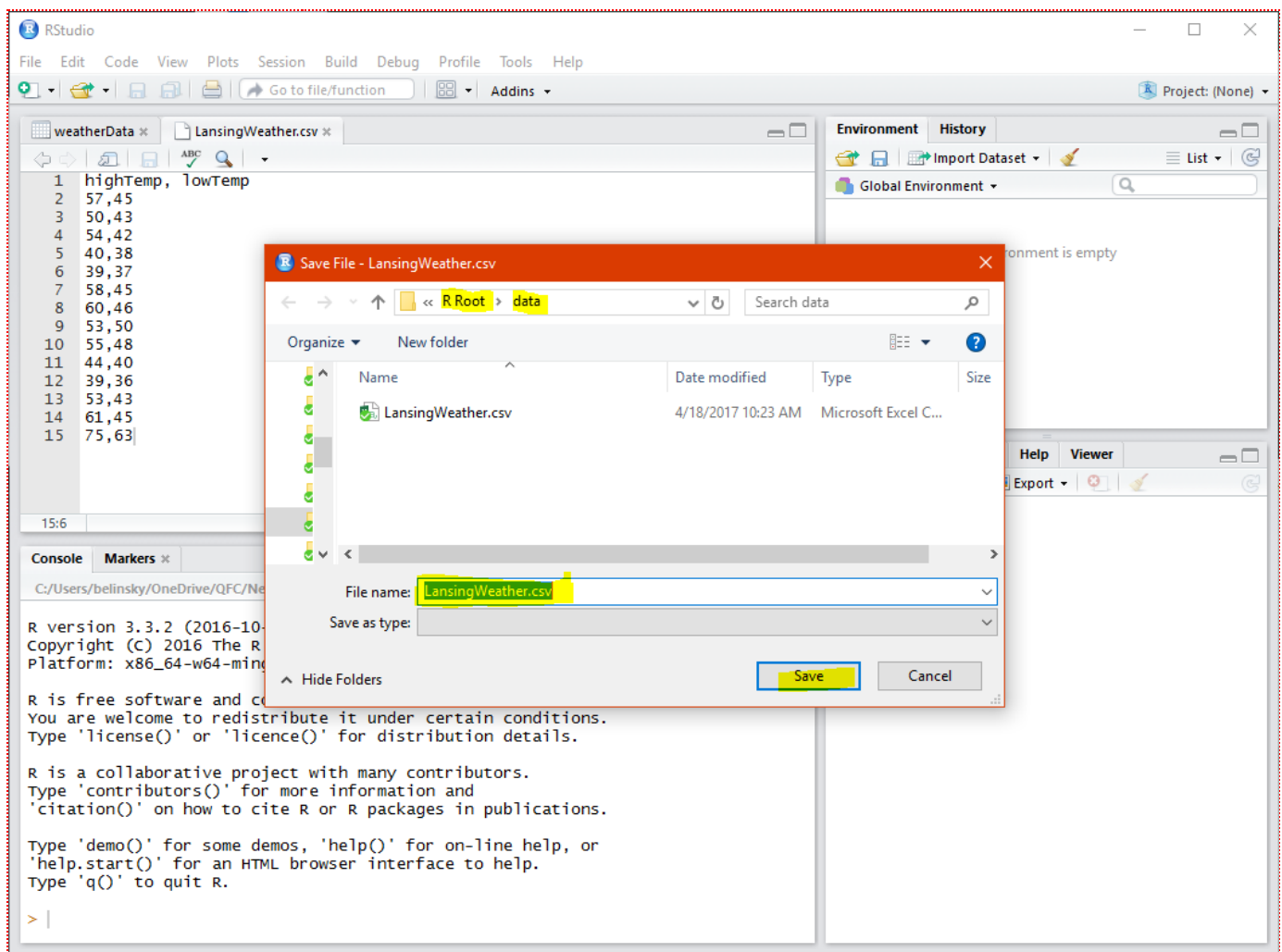


Fig 2: Save **LansingWeather.csv** file to data folder

To reopen a csv file in R go to **File -> Open File** and find the file in the **data** folder.

6 - Setting up RStudio to get data

For the first time in this class, we are working with multiple files (script files and data files). In this setup, a script file gets information from a data file. In order for this to work, *the script file needs to give the file location and name of the data file.*

6.1 - Working directory

In R, the location of a file is always given relative to the working directory. In an RStudio Project, the working directory is the same as the Root Folder of your project. So, whenever you need to reference a file in an RStudio PProject, you reference it relative to the Root Folder.

If a data file named **myData.csv** is in the **data** folder in your RStudio Project (this will most likely be the case), then the location of **myData.csv** is:

```
1 data/myData.csv
```

If there is another folder within **data** called **newStuff** and **myData.csv** is in **newStuff** then the location of **myData.csv** is:

```
1 data/newStuff/myData.csv
```

The *forward slash* (/) means go into this folder -- if **myData.csv** is four folders deep, you would have four slashes in the file path (/):

```
1 folder1/folder2/folder3/folder4/myData.csv
```

Note: the forward slash (/) is on the same key as the question mark (?) , the backward slash (\) is on the same key as the pipe character (|).

Extension: relative and absolute file paths.

6.2 - Input data from a file to script

Now that we have a data file in our data folder, the next step is to create a script that can read the data file.

The **R Root** folder is my Root Folder/default working directory. Data files get saved in the **data** folder and the **data** folder is inside the **R Root** folder. This is why I used **"data/LansingWeather.csv"**, to tell R to go inside the **data** folder. You will do the same for your project.

```
1 {  
2   rm(list=ls()); options(show.error.locations = TRUE);  
3  
4   # The line of code below will work for you as long as you put LansingWeather.csv  
5   # in the data folder inside your R Root folder.  
6  
7   # read data from LansingWeather.csv and save to the variable weatherData...  
8   weatherData = read.csv("data/LansingWeather.csv");  
9 }
```

Line 7 of the script opens the **LansingWeather.csv** file and saves the information inside the **LansingWeather.csv** file to the variable named **weatherData**.

Note: A common mistake is to include the Root Folder in the file path:

```
weatherData = read.csv("R Root/data/LansingWeather.csv"); # error
```

But this instruct R to look for a second folder **R Root** within the folder **R Root**

7 - A variable that holds multiple values

weatherData is a different kind of variable from any other variable we have worked with so far. Up until this point all variables held one value but **weatherData** is a variable that *holds multiple values*. In fact, **weatherData** holds all of the values in the CSV file.

Looking at the Environment Window, we see **weatherData** has "14 obs. of 2 variables". The 2 "variables" are **highTemp and lowTemp** (the labels we put on the first line) and the 14 observations are the 14 temperatures we entered for each day (*Fig.3*). In this case, *variable refers to the column headers of the data frame*. This is

not the same as the way we have used the term variable, so from now on we will refer to these data frame headers as *column headers*.

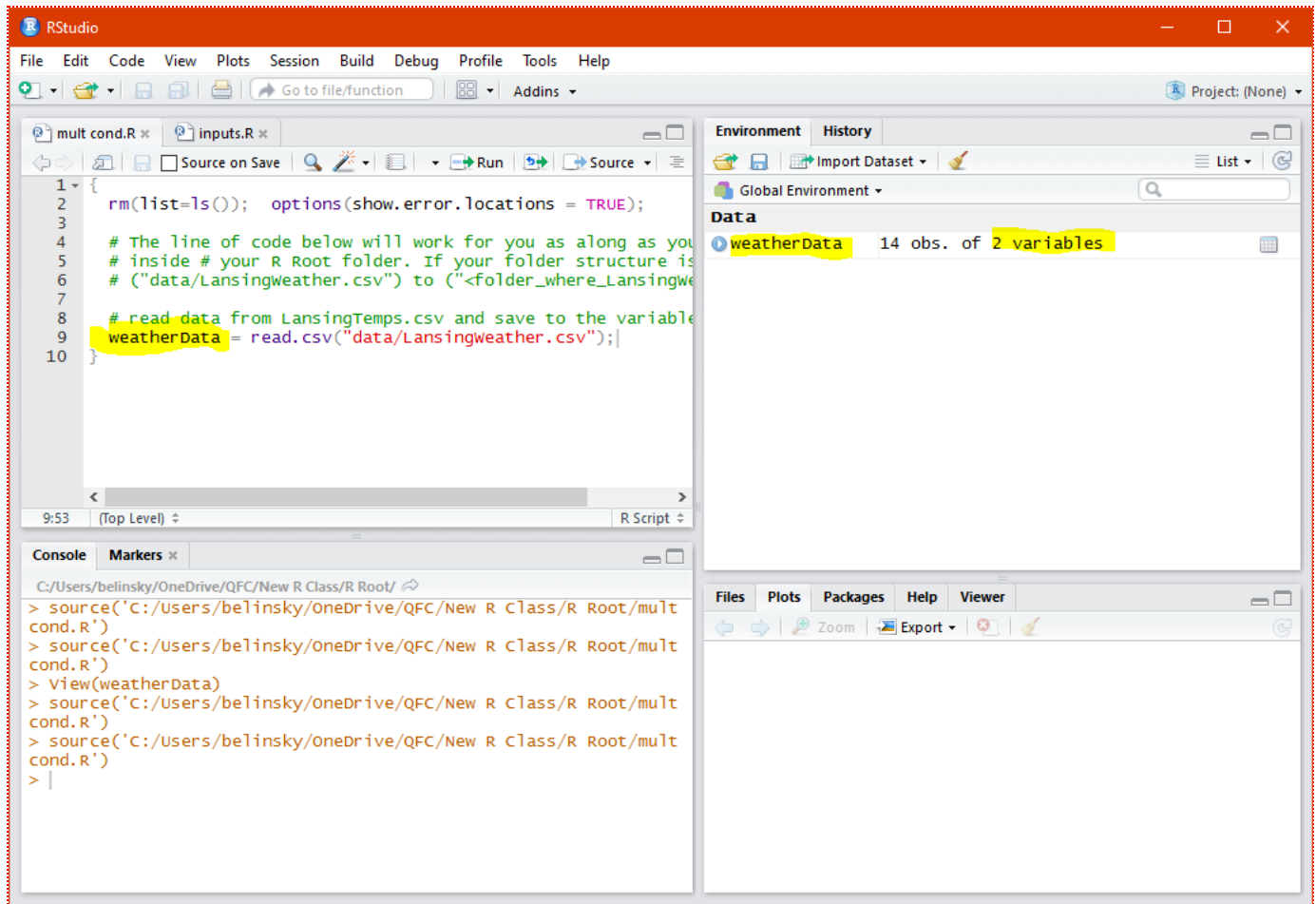


Fig 3: Saving the data from **LansingWeather.csv** to a variable

7.1 - Viewing the input data

If you click on the arrow to the left of **weatherData**, two more rows appear representing the column variables **highTemp** and **lowTemp** (Fig.4), the type of values stored in the column variable (*int*, which stands for integer) and a partial list of the values held in each column variable (57, 50, 54...)

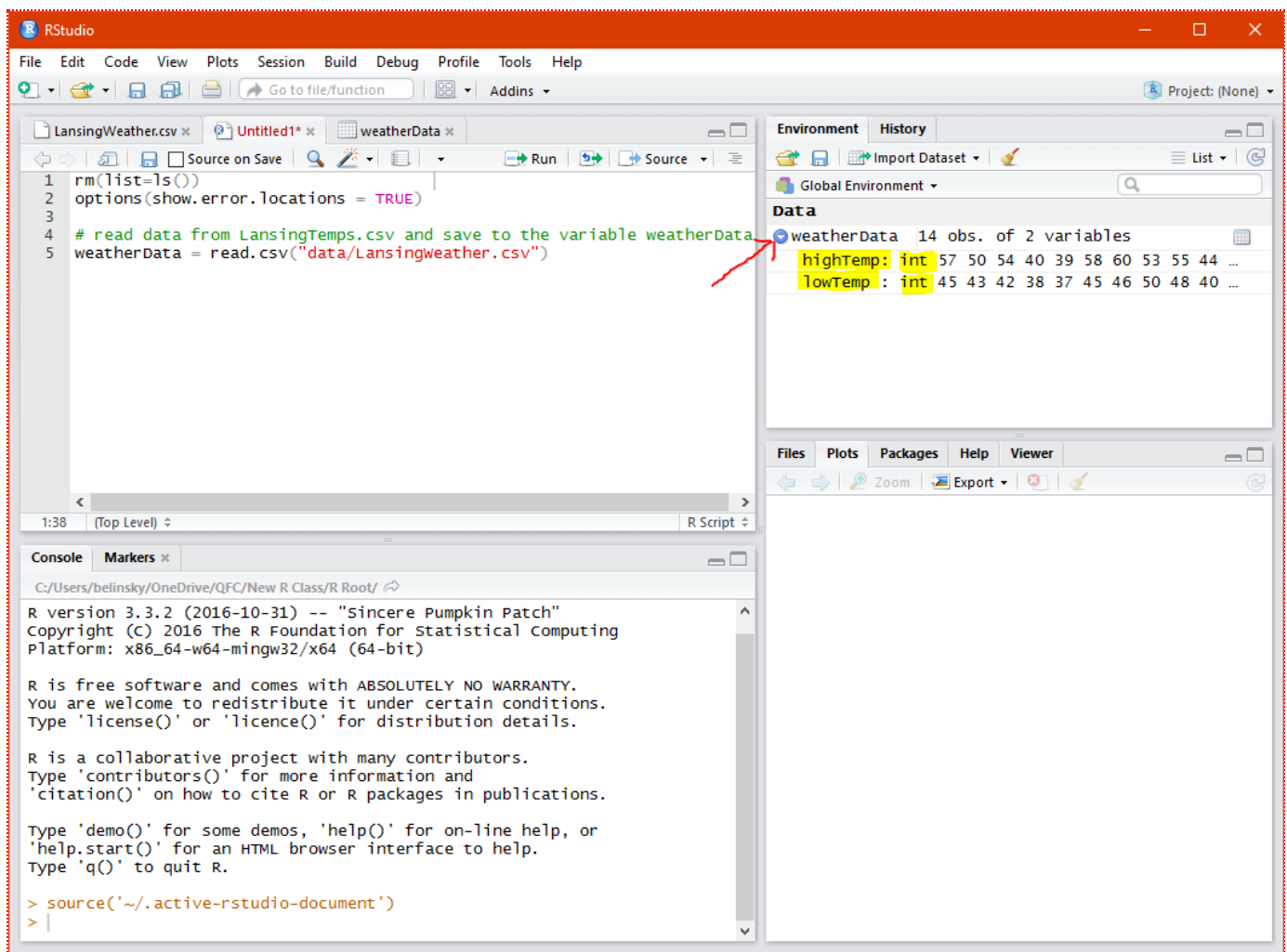


Fig 4: Using the Environment Window to show the values within the variables

If you want to see all the values associated with **weatherData** -- you can double-click on the **weatherData** line and a new tab will open in the main window that gives an indexed table of all of the values in **weatherData** (Fig.5).

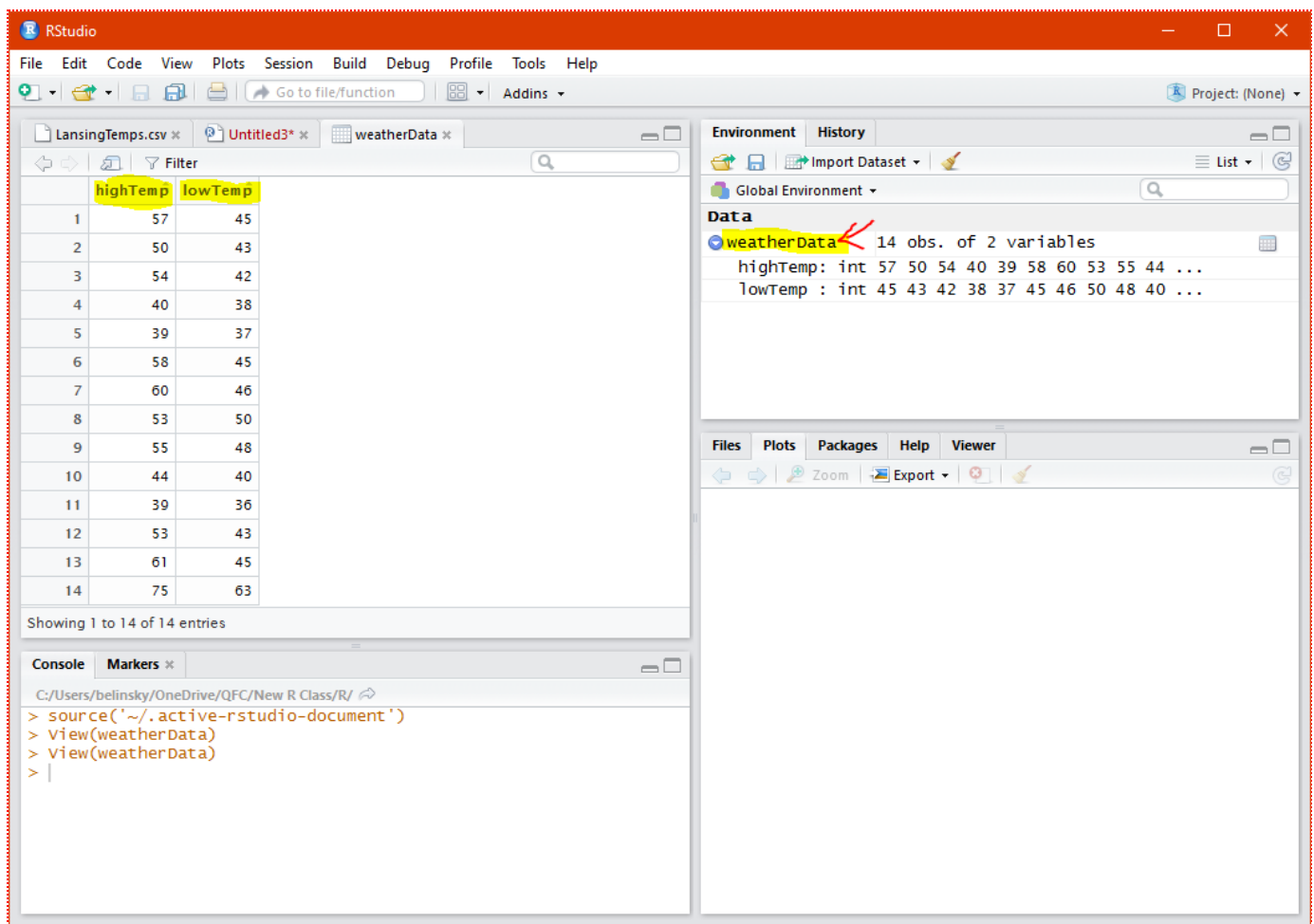


Fig 5: Showing data in tabular form

8 - Application

If you have any questions regarding this application, feel free to email them to the instructor here.

You can attach files to the email above or send the whole Root Folder as a zipped file. [Instructions for zipping the Root Folder are here.](#)

A) In the **LansingWeather.csv** file, add a third set of values called **precipitation** and use the following values for the 14 days (the values are in inches)

0.01, 0.005, 0.04, 1.11, 0.12, 0, 0.005, 0.49, 0.45, 0.30, 1.13, 0.004, 0, 0

B) In your R Script, save the data in **LansingWeather.csv** to a variable called **newWeatherData**.

C) Open **newWeatherData** in the Main Window and make sure **newWeatherData** has 3 filled columns of weather data (**highTemp**, **lowTemp**, and **precipitation**)

Verify, by double clicking on the data frame **newWeatherData** in the Environment Window, that **newWeatherData** looks like this:

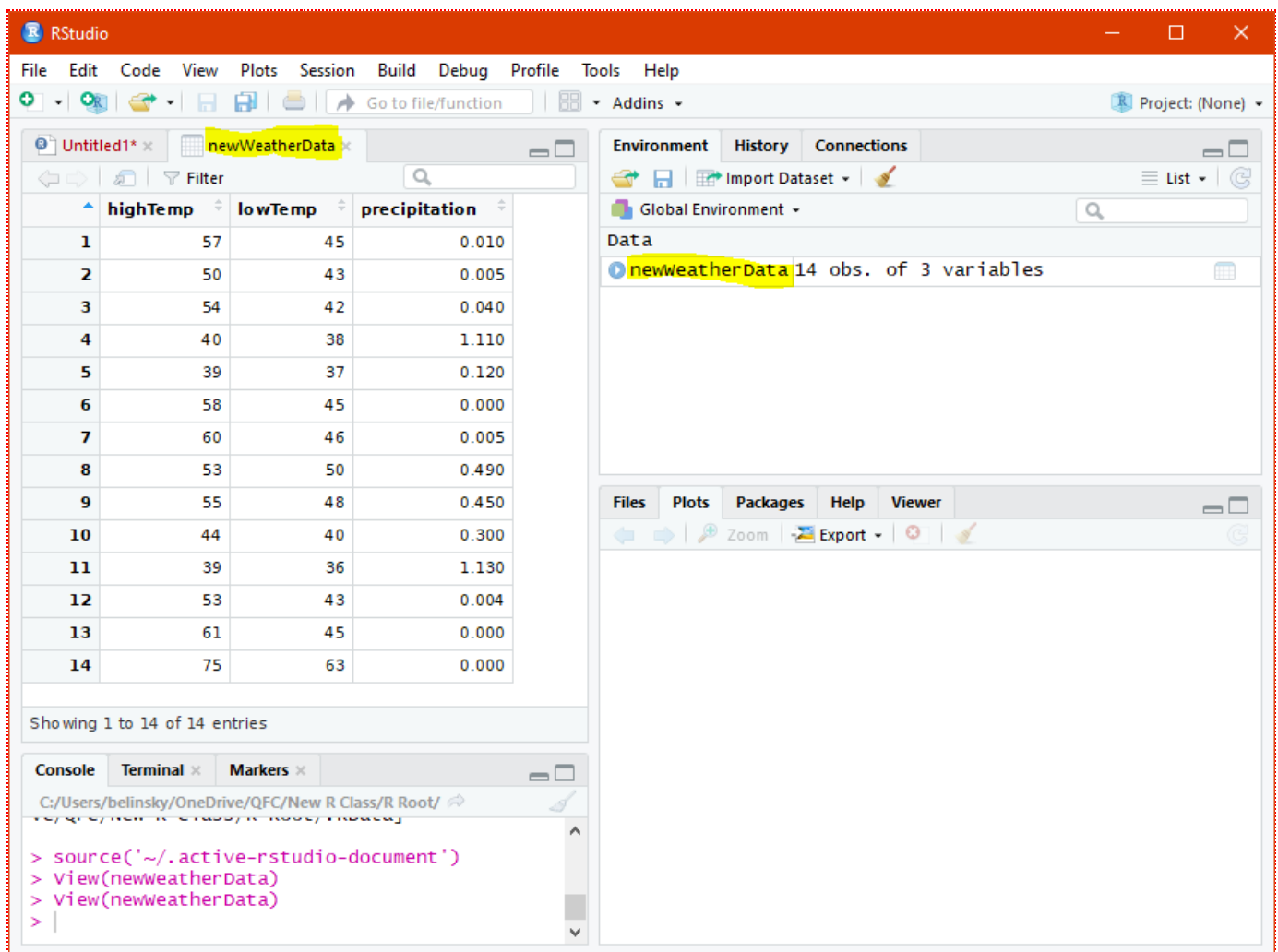


Fig 6: `newWeatherData` data frame with `precipitation` values added.

Save your script file as **app2-1.r** in the **scripts** folder of your RStudio Project for the class.

9 - Extension: relative and absolute file paths

Let's take a look at the default working directory again:

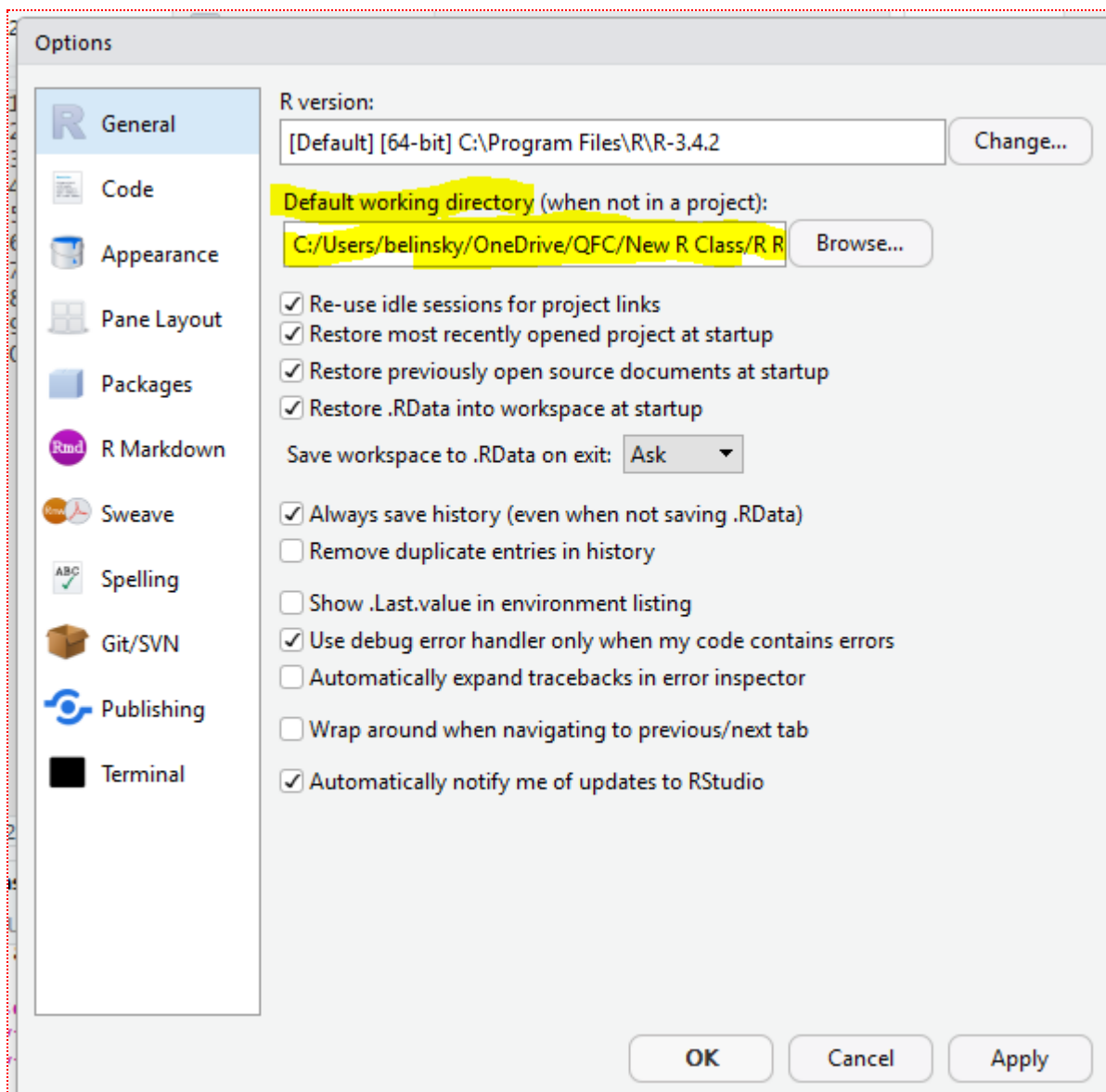


Fig 7: The absolute file path for the default working directory.

The path for me starts at the **C:** drive, moves to the **Users** folder, then **belinsky**, **OneDrive**, **QFC**, **New R Class**, and finally **R Root**.

This is called an **absolute file path**. Absolute file paths start at the base of the tree (your harddrive, or network folder) and move down the branches (folders) until it find the required directory.

The problem with absolute file paths is that they are usually unique. No one else reading this lesson will have the same file path (unless you are on a Windows 10 machine using OneDrive, have the last name belinsky, work at the QFC, and are developing a new R class). This means that I cannot use absolute file paths in my script and expect anyone reading this to be able to execute the script.

This is why we usually use relative file path naming in programming. Relative file paths start at whatever folder you are currently in (as opposed to the base-of-the-tree folder). With relative file paths, I can set up a folder structure that anyone can copy and use. In R, the working directory is the starting point for all relative file paths.

For example, say I want to get the file "weather.csv" from the **data** folder in the working directory:

```
1 | weatherData = read.csv("data/weather.csv");
```

Perhaps "weather.csv" is in the folder **Lansing** inside the folder **data** in the working directory:

```
1 | weatherData = read.csv("data/Lansing/weather.csv");
```

Or, "weather.csv" is in the parent folder of your working directory (i.e., you need to go back a folder):

```
1 | weatherData = read.csv("../weather.csv");
```

*Note: ../ means "go back a folder",
../../ means "go back two folders"*

Lastly, sometimes you need to move backwards and forward through directories. In other words, go back to find a common parent folder, and then move forward. In this case, we move back a folder and then into the **Lansing** folder:

```
1 | weatherData = read.csv("../Lansing/weather.csv");
```

10 - Extension: Default working directory (DFD) is a session variable

The DFD is a variable like the other variables you have used in your scripts. The difference is that the DFD is a variable for *the whole RStudio session*, not just your one script file. This means that if you have multiple scripts open in RStudio and one script changes the DFD, then the DFD changes for all the scripts opened in RStudio. This is called a *session variable*.

When you open RStudio directly (i.e., not by double-clicking on a .r script file), it sets the DFD to the file path you set it to in **Tools -> Global Options**. When you open RStudio through a file association by double-clicking a script file on your hard drive or network folder, the DFD is set to the file path of the script file.

Each time RStudio opens, it sets the DFD. So, the behavior of your script can change based on how RStudio is opened and your script may fail if it depends on the DFD.

11 - Extension: Microsoft Excel and CSV files

Quite often you will have data in Excel format (*.xls or *.xlsx). There are functions in R that can read in Excel file but it is best to first convert the Excel file to a CSV file. This can be done easily in Microsoft Excel:

1. Open your Excel file
2. Click **File -> Save As**
3. Choose "CSV" in **Save as type** dropdown

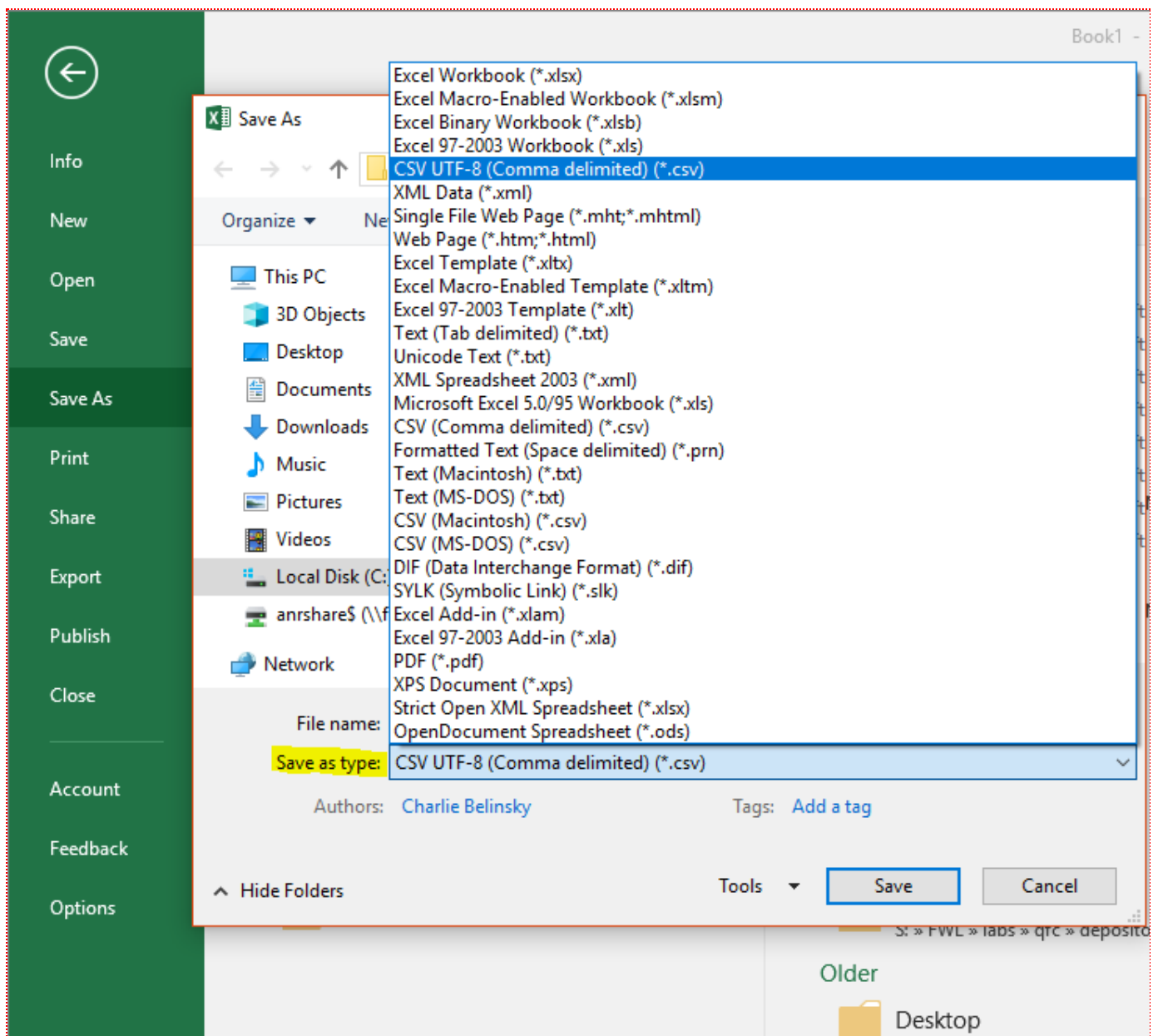


Fig 8: Converting an Excel file to CSV

Note: There are multiple CSV options and different Excel version will have different options. You probably will see no difference between the options unless your data has strange characters in it. A good rule of thumb is to always choose the top-most CSV option.

Note 2: Excel will give you warning about not being able to preserve the original file in the new format. *Your data will not change*. However, if you have made calculations in Excel, those will go away.

12 - Extension: Working directories outside of RStudio Projects

RStudio uses the **default working directory** whenever it is launched directly. However, if RStudio is opened by clicking on script file with a **.R** extension (called a **file association link**), then RStudio will change the **default working directory** for that session to the directory the opened script is in.

Extension: The default working directory is a session variable

In other words, *how you open RStudio determines what the default working directory is*. The workarounds for this are:

1. Always open your script files in RStudio using **File -> Open File...**
2. Always open RStudio before double-clicking on a script file
3. Click on **Session -> Set Working Directory -> Choose Directory...** and navigate to your **R Root** folder
4. Add a line to your script file that sets the **default working directory**

The fourth option is the most robust but requires that you add the following line of code to the top of all your scripts:

```
1 | setwd("default working directory path");
```

The directory path highlighted below (*Fig.9*) can be copied and pasted into **setwd()** -- don't forget to put quotes around the file path.

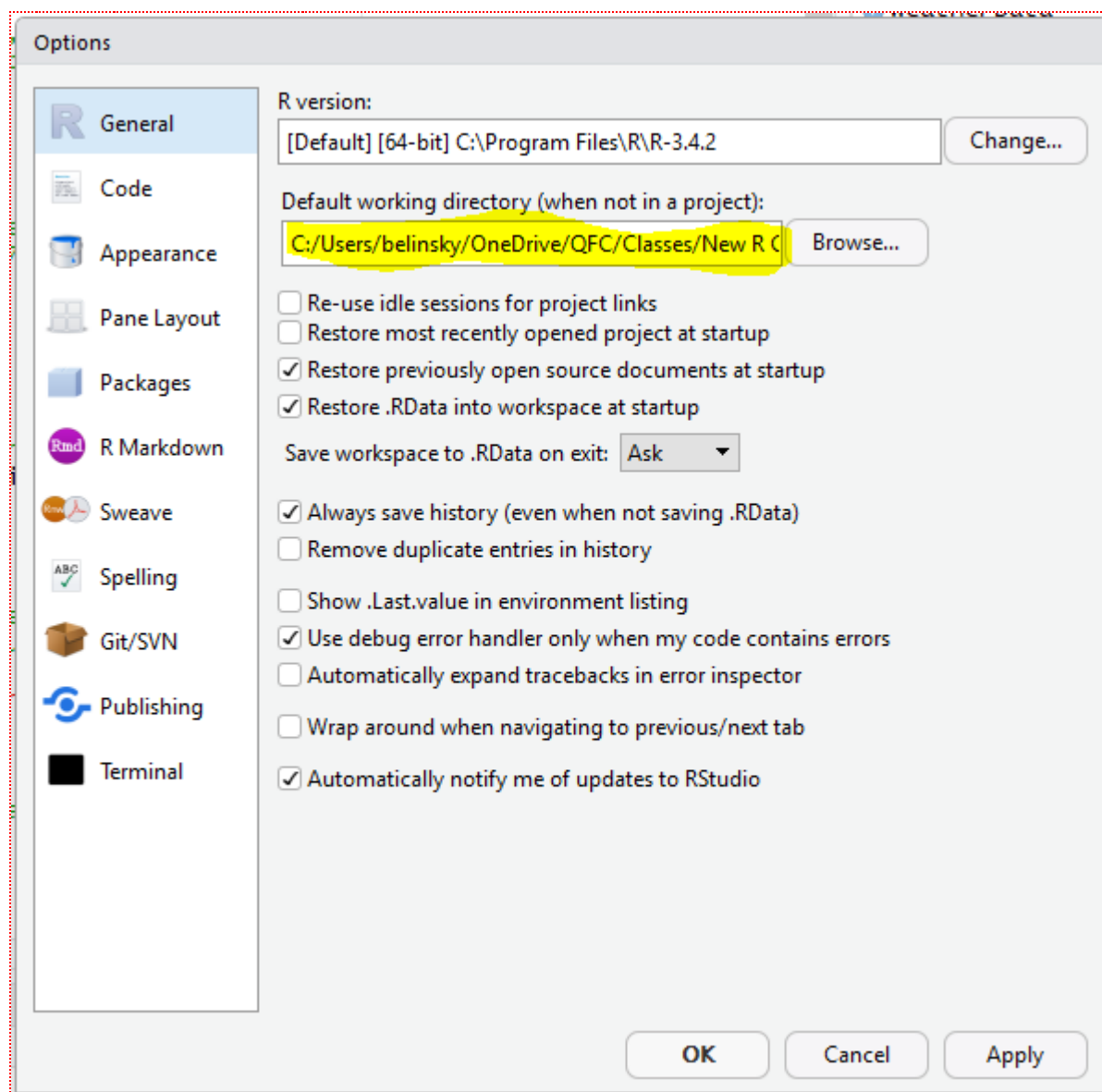


Fig 9: Default working directory to be copied/pasted into setwd()

The line of code for this author is (for you it will be different):

```
1 | setwd("C:/Users/belinsky/OneDrive/QFC/Classes/New R Class/R Root");
```

Note: R uses forward slashes (/) in its file path whereas Windows uses backward slashes (\)