SOP 85: Using NOAA Climate Records

Marc Los Huertos

August 9, 2017

1 Introduction

Raw data sets often come with untidy/non-useful formats or information that must first be cleaned or processed before an accurate and useful analysis of the contents can be done. After obtaining a data set there are some preliminary steps you must follow in order to get your data file into working order for your analysis.

1.1 Purpose

This document is intended a resource and guide to help you:

- upload your data file into the R environment using the Rstudio server online
- clean, organize and reformat the data to prepare it for analysis

The program that we will be using to run an analysis of our data is R, and we will access it via the Rstudio server using your computer's web browser. Essentially, we will be using cloud based computing software to do our analysis so the first thing you will want to do is know how to access the server.

Note: Before you attempt to upload your data file into an appropriate directory in Rstudio you should already know how to access your Pomona Rstudio account as well as have connected to Marc's Github repository. Click on the following link to download a guide on how to get started with Rstudio and Github:

An Introduction to R, Rstudio, Github

2 Preparing CSV file(s)

2.1 Preprocessing CSV files

In most cases, we don't need to preprocess the csv files. However, Mac users have been confronted with a host of problems that has something to do with how Macs format CSV files.[1]

[1] I will update this when I try using a Mac for this.

2.2 Upload CSV Files into Appropriate Rstudio Directory

The first step to getting your data into R is to upload it into the the Rstudio server online. Once you are in Rstudio you will want to locate a good folder/location in the Rstudio directory to which to upload your data file. We suggest uploading

3 Reading CSV Files into R

Although the csv file might be present in an Rstudio directory, it is not in the R environment. One way to confirm this is to look at the Rstudio window tab 'Environment', where the file listed.

To have the file in the R environment, we read the file using the `read.csv()` function. The function is expecting the path and name of the file as an argument inside the perentecies, which is tough to type without making errors.

3.1 Assigning the File Path and Name

Okay, now we know what the file name and path look like from the eyes of R. In this example, the path and filename is:

We could paste the whole mess into the `read.csv()` function. Okay, let's keep moving. Although R was able to see the file, we still don't have an object we can actually do something with. Instead, we merely printed it to the screen. However, you have not created an object yet. To do this you need to assign it a name. So, what do we need to make an object? Yes, the name of the object and the use of the assignment symbol. Let's create an object names filename instead that has the whole path and file name in it. Start with the name you want to use before the `read.csv()` function like this:

First, we will import the CSV file. In most cases, we don't need to preprocess the csv files. Use "file.choose" to find the folder path name.

We will use a pop-up window to select the file the first time and then we'll assign the file path and name to an object.

```
> choose.file()
```

Need an impage of the pop-up window...

3.2 Reading a CSV File into a Dataframe

3.3 Importing CSV Files

Importing Scripts

import = read.csv(file)

```
# Read CSV

file = "/home/CAMPUS/kaj41925/Climate_Change_Narratives/Khalil/NewOrleansNOAAdata.csv"

import = read.csv(file)

## Warning in file(file, "rt"):  cannot open file '/home/CAMPUS/kaj41925/Climate_Change_Narr
Permission denied
## Error in file(file, "rt"):  cannot open the connection
```

3.4 Confirming the Dataframe

3.5 Checking the Data

Now we will check the data by plotting it.

3.6 Checking the Data

What's going on? What is the deal with the -9999? These are used for missing data. We need to remove them!

Okay, now we'll check again, for dates less then Dec. 31, 1913:

Yikes! What's wrong? As it turns out the problem is that with how the dates are specified. In particular, the Dec 31 to Jan 1 transition:

19131231 -¿ 19140101, if you use these as numbers it should be 19131232, 19131232, etc. You see we are missing lots numbers!

3.7 Converting Dates

3.8 Re-assigning Missing Values

3.9 Fixing the Dates

First, we convert the date to a string of character values. Next, we'll convert the strings to a data format.

3.10 Checking the New Dates

3.11 Subset Sites

Olivia has two sites in here data, so, we need to subset it.

Let's choose the FAIRPLEX NY US because the record is longer than the airport.

Okay, you know have created a data frame. To confirm this, type `str(maunaloa)` and you should see some strange text that describes the data frame. This function allow you to peer into the data frame structure. You you see it is a data frame and it has several variables and each one has certain characteristics and

R even shows you some of the observations. This is a good thing to get into the habitat of check, for you want to ensure the data have been imported in a way that you expect.

Remember, a data frame is a list of vectors. To access the data inside the data frame, you can use the following command

```
#maunaloafaverage
```

to dump the average CO_2 concentrations readings onto your screen as a vector. You should see some 627 observations, depending on how recent the data have been uploaded. So, the dollar symbol is used to drill into the data frame vectors. And when you look at the `str()` function again, you will see these dollar signs again.

4 Preparing Records for Analysis