Data wrangling workshop I for Neuroscience Master students

David Munoz Tord

31 March, 2021

Part I

Make sure you have read the ReadMe file.

If you are struggling I suggest you look at the "cheat sheet" documents and/or you just look it up on the internet.

A huge part of being good at programing is knowing where/what to look for. I definitely encourage each of you to familiarize yourself with how to use search terms in google (https://www.dailyinfographic.com/how-to-become-a-google-power-user) or any other search engine and to look for answers in stack overflow.

If you are still struggling after that just try posting an issue on this github repo so I or other people following can help you.

A) Git basics

Fork this repository https://github.com/munoztd0/Data_Wrangling_NeuroMaster onto your github page. You can also "star" it by clicking the little star icon so you will get informations about this workshop and comments from others.

Clone YOUR fork (e.g. git clone https://github.com/YOURUSERNAME/Data_Wrangling_NeuroMaster inside a terminal) Open the test.txt file, add whatever you want e.g. "doneso" or whatever) and save it.

Open up you terminal within the "Data_Wrangling_NeuroMaster" directory, and type :

```
git add -A
git commit -m "test" #or whatever
git push
```

You will have to give you github username and password.

Now go to your repository (again https://github.com/YOURUSERNAME/Data_Wrangling_NeuroMaster) and check thhat the modifications are there.

That's it first task done!

Additional tip: If you are as lazy as I am sometimes (remember laziness WILL get you far in programing) you can add this function to your ~/.bash_profile or whatever your OS uses ro replace this three lines by one (e.g. lazygit).

```
function lazygit() {
            git add .
            git commit -a -m "$1"
            git push
            }
```

B) Exercices

Now that we are on point. Let's actually start. First thing wil be to create a file that will contain the responses to this workshop's exercises. The extension will of coursese depend on which languages you want to use (e.g. exercise.py, exercise.R, exercise.m, exercise.rmd, exercise.ipynb)

I defintely recommend that you start familiarising yourself with Rmarkdown and Jupyter notebooks if you are not.

1. Make you code work on other machines An important thing that will make you and your coworkers save a ton of time and hassle is to make your code flexible and reproducible.

This will take you a tiny bit more work but you will thank yourself later down the line.

So the first thing will be to make the current path of your "exercise.**" file flexible. There are different ways to do that depending on which program you are using and how much flexibility you want but here a some cues on how to start you journey.

For python:

```
import os
print(os.getcwd().strip('"\''))
```

```
## /home/davidM/Desktop/SwitchDrive/Data_Wrangling_NeuroMaster
```

For R:

```
homepath = getwd()
cat(homepath)
```

```
## /home/davidM/Desktop/SwitchDrive/Data_Wrangling_NeuroMaster
```

For Matlab/Octave:

```
homepath = pwd;
disp(homepath)
```

```
## /home/davidM/Desktop/SwitchDrive/Data_Wrangling_NeuroMaster
```

Now try to load the dataset "data.csv" using the "homepath" dynamic variable !

In this vibe try to make ALL your code work seemlessly on another computer with the same operating system you have (you can try to do that on your own machine but on a different user for example). Especially make sure you scripts imports/downloads the functions/packages/libraries/modules you use in your script.

For R (put that a the beginning of your script):

```
if(!require(pacman)) {
  install.packages("pacman")
  library(pacman)
}
pacman::p_load(tidyr, afex)
```

For Python (do that when you finished your scripts): It will automatically save module requirements in a ./requirements.txt file

```
pip install pipreqs # install pipreqs on your machine
pipreqs .  #run that in the directory where you have you script
```

Then someone can run "pip install -r requirements.txt" before running your script

Facultative:

Try to make ALL your code work seemlessly on another computer with the another operating system (OS) than yours (if you are on mac/linux try on windows and vice versa since mac/linux are oten really similar).

To make your life easier (and impress you colleagues and supervisors) you might want to take a look at repoducible workflows such as https://github.com/aaronpeikert/repro for R or https://swcarpentry.github.io/2014-01-31-ucsb/lessons/jk-python/reproducible_workflow.html for Python.

If you can't get your hands on another OS you are encouraged to post an issue on this workshop's github page so you guys can find a "partner" than can try your code and vice-versa.

Alternatively, you can also create tiny virtual machines to try you code on different OS (more on that on https://www.virtualbox.org/). Tip: windows is actually "free" for trying software so you can "skip" the license in the installation and it will still run.

2. Rearrange the data First let's just describe the dataset. This is a real dataset that I used but the measurment variables names have been removed and some of them have been "tweaked" for this workshop purpose to create real word issues. The experiment can be summarized by a 2x2 mixed design.

1 repeated (within) factorial variable with 2 levels ("pre" and "post") and 1 group (between) factorial variable with 2 levels ("placebo" and "treatment"). Then there is a bunch of continuous variables those are biomedical variables of interest.

However as you can see, this dataset is in "wide" format (var1_ses1, var1_ses2, ...) instead of a "long" format, which is something that I often encounter. If you are not familiar with the terms I suggest you take a look here (https://discuss.analyticsvidhya.com/t/difference-between-wide-and-long-data-format/8110/2).

Anyway here you will have first to transform this dataset from wide to long, this means thats we want 1 row per measurment (i.e. var1, var2,..., with a "session" column that either has a "pre" or "post" value).

You could of course do it by hand, but that of course not why we are here.. So try to implement a way of doing so automatically.

3. Plot the data But right now, our variables are in all sort of different scales and it's impossible to compare apple to oranges. Plus since you don't know what the variables are supposed to measure anyway, we will scale all variables (x - mean /sd). Like always try to implement a way of doing so for all avriables at once !

Now that you have you nice long format standardized data you will want to visualize it. Visualizing data has always a subjective part but there are plenty of people that have created guidelines and rules to make it clearer for everybody, so take a look when you can (e.g. https://www.data-to-viz.com/ or https://www.sigmacomputing.com/blog/7-best-practices-for-using-color-in-data-visualizations/) and try to think about it.

Anyway, for now we will want to visualize the overall density of each variable (no matter which session or group) and the correlation between each pair of variable (pair plots). Again try to look for a way of doing so efficiently this might require a similar skim than just before for gathering variables per keys (this is a clue).

What do you see ? Anything weird (e.g. outliers, correlations, densities) ? Write it down and we will discuss it.

4. Clean the data hh