**Naming Protocols:**

One of the hardest tasks in data management is conforming different names for variables, entries, files etc. A successful databank is built when some naming protocols are established and followed. There are tons of different protocols out there, but here are a few that I have found useful when working with Python, R and postgress.

**A.File Names:**

* The name should contain information about the author, the task and the date completed. This is especially useful in collaborative efforts when multiple authors are contributing to the same files.
* One of the easiest ways to keep things clean is the following convention. Initials of the author (with a number at the end, most of the times 1) followed by the name of the file, and then the date as month day year. In between the different parts use underscore “\_” instead of spaces.
* Avoid using special characters, various other symbols and prefer capital letters for the initials and capitalize the first letter of every word in the name. Do not break the name of the file with extra underscores.

**Example:** This document is named TG\_NamingProtocols\_01\_07\_2019. This immediately tells us that it was created by Thanos Gentimis and if there was another person with TG as initials, they would establish the correct number for each person. We also know that this was created January 13th of 2019. Finally, the document has something to do with Naming protocols. This way if in the future we create a “parser” basically a little code that reads names and breaks them down to parts, we will know that “\_” is the “divider” amongst the different parts of the name and the convention is:

Initials \_ NameOfFile\_Date with Date being Month\_Day\_Year.

Notice also the “padding” with zeroes in front of the day. Instead of day 7 we put day 07, knowing that this can go all the way up to day 31 and similarly for the month. The reason for that is to be consisted with alphabetical ordering. For example if we did not do the padding, we would have the following bad sorting:

1,11,12,13,…19,2,20,21,3,4 … etc which would be an absolute mess.

**B.Data matrices:**

When dealing with datasets it is harder to have a global naming convention for all the entries. Still here are a few things that will make your life easier if you are planning to utilize programs to analyze them:

* Don’t use special characters, or spaces in each entry. Instead of space use underscore or, in case underscore is used as a divider just concatenate and capitalize the first letter of each word.
* Don’t use different symbols, or different alphabets if possible.
* Don’t use dollar signs or measurement units in general in variables that are numeric.
* Don’t use commas! Most of the times we will be using what is called a comma separated value file (CSV) so commas are reserved for separating our variables.
* Variables go vertically, datapoints (observations) go horizontally.
* The first line of the table contains the variable names. It is strongly suggested that those names are written with capital letters. Make sure all variables have a name.
* Although it looks great in excel, empty columns and rows are to be avoided. Yes, the human eye loves a break from numbers but the mechanical one gets confused.
* Avoid having both numbers and letters in variables if possible. This is a soft requirement, one that I break mostly, but useful if you can have it.
* Avoid formulas created in excel. One idea is to copy the sheets to new excel sheets and keep numbers only in the copy option.
* Make sure there are no comments or graphs in the excel sheets and avoid having metadata.

**C.Codes:**

One very important thing you will learn soon is that “version control” is very important in coding. Most of the times you don’t want to “throw away” your old code, but you want to update it and keep both the old and the new. Also, other times, multiple collaborators will be working on the same code at the same time, making shared repositories a mess in keeping in check. Again, some suggestions to make things easier:

* Start again with the initials of the coder. This will keep the different coder’s codes in different places in common repositories if they are sorted alphabetically.
* Continue with the name of the code. Make it simple this time.
* Follow it instead of the date by a numbering convention as follows: Major versions followed by tweak number.
* Use underscores to separate the different parts.
* Avoid other special characters especially dots and spaces.

**Example:** Let’s say we want to do an analysis of a dataset. The code for it should be TG\_Analysis\_1\_1 when it starts. This means that Thanos Gentimis is writing a code to do some sort of analysis. This is the first version without any revisions. Let’s now assume that later I went in and added 2 more graphs and a summary statistic. The code now could be named TG\_Analysis\_1\_2 since some revisions were implemented. If instead I removed 1/3 of the code and replaced it with something new and extended practically all aspects of it (perhaps I changed libraries or packages) then perhaps it is better to name it TG\_Analysis\_2\_1, the 2 meaning that a whole new version was established. If a collaborator now worked on it and fixed some little things of version 2, she can go in and name the new output MB\_Analysis\_2\_2. This way we have some sort of “progression” that we can keep track of, and we never throw away old code.