
Importation and analysis of MATLAB produced dataset

Final project for COMP598 – Summer 2020

Zaida Escila Martinez Moreno

Integrated Program in Neuroscience, McGill University

McGill ID: 260807309

Introduction

Aging is associated with the decline of cognitive and sensory functions. Therefore, it is important to understand the differences between young and aged persons, and the effect the decline has in their brains. Cisneros-Franco et al. [1] demonstrated that the aged brain is more plastic and sensitive to changes than the young adult one. The group of Zatorre [2] previously described that learning can generate changes in the brain structure. Together, they showed that it was possible to elicit changes in the aged brain through training. Prior to assess if these changes can occur, it is necessary to understand the impact that age and other sociodemographic information could have in the performance at a cognitive training task. To approach this, the experimental design chosen was a set of young and aged subjects that will undergo a daily training for twelve days (Figure 1). Magnetoencephalography (MEG) imaging was acquired to compare the activity of their brains throughout the training. To evaluate whether age and other sociodemographic factors are important, our group created a spectral discrimination task and presented it to a group of 16 young adults (mean=22.72, max=35, min=19) and 8 older adults (mean=61, max=77, min=42). Each of them performed the task once daily for twelve days on a tablet (7 or 8 days) and inside a MEG scanner (4 or 5 days).

The aim of this project is to create scripts that organize, convert, import and graph the data stored in files that were produced daily for 12 days per subject to create a dataset that will facilitate further analysis.


Days	1	2	3	4	5	6
	<ul style="list-style-type: none"> ➤ Cognitive battery ➤ Training in MEG ➤ Long MEG recording 	Training at home	Young: <ul style="list-style-type: none"> • Short MEG • Training in MEG Aged: 	Training at home	<ul style="list-style-type: none"> ➤ Short MEG ➤ Training in MEG 	Training at home
	7	8	9	10	11	12
	Training at home	Training at home	<ul style="list-style-type: none"> ➤ Short MEG ➤ Training in MEG 	Training at home	Training at home	<ul style="list-style-type: none"> ➤ Cognitive battery ➤ Training in MEG ➤ Long MEG recording

Figure 1. Experimental design for the acquisition of the data.

Datasets

The data used for the creation of the code, depicted in Supplemental figure 1, consisted in a file per day in format *.csv* when the training was performed in a tablet (8 days) and in *.mat* when the training was done inside a MEG scanner, and a *.csv* file containing the sociodemographic data from each subject (Supplemental table 1).

Methodology

Due to the nature of the files, three different programming languages were used:

- Bash: organize the data, copy it to another folder to avoid modification of original data and obtain list of folders and *.csv* files
- MATLAB: conversion of *.mat* files to *.csv* files
- Python: importation of each *.csv* files, concatenation of data per day per subject into a single DataFrame, visualization of data in graphs and exportation of graphs to *.png* files

Bash script algorithm (Organizing_files.sh)

1. Go to directory where the original data is stored
2. For each subject's directory, create a directory in destination path and add the name of the folder to a text file (*Subjects.txt*)
3. Enter subject's folder
4. If there are .csv or .mat files, copy them to the corresponding subject's directory in destination path
5. Go to destination path
6. For each subject's directory, enter subject's folder
7. Add name of all .csv files to a text file (*filenames.txt*)

MATLAB script algorithm (Export_csv_files.m)

1. Go to directory where data is stored (equivalent to destination path in bash script)
2. Produce a cell with each subject's folder's name
3. For each subject's directory, enter the directory and produce a cell with .mat files
4. Import each .mat file
5. Modify the imported structure to resembles the .csv files produced directly by the tablet and erase all data that is not useful for the project
6. Convert the structure to a table and save it into new variable
7. Export the table to a .csv file that preserves the original name

Python script algorithm (Import_Analysis_Behaviour_files.py)

As this script is the most complex one, the algorithm will be divided in the main parts of the code:

- Defined functions:
 - **read_simple_text**
 1. Open the .txt or .csv file
 2. If it has a header, read first line and assign to variable
 3. If it has a delimiter, split the string
 4. Read each line, split it if necessary and add it to a list
 5. Close file
 6. Return list of strings and header, if present
 - **find_duplicates**
 1. Iterate through each element
 2. Count each element in the list
 3. If more than 2, add it to variable without repeating it
 4. Return list of duplicates

- **convert_month2number**
 1. Declare list of months in 3-letter format (Jan, Feb, Mar, etc)
 2. Verify if input is present in that list
 3. Return the conversion to month in number (1, 2, 3, etc.)
- **n_days_dict**
 1. Takes a list of filenames sorted by date, a list of creation dates of each filename and number of days (*n_days*) as input
 2. Create a list [*date*, *date+1day*, *date+2days*, ...] with length *n_days*
 3. Create a dictionary with keys = list of filenames
 4. Compare the creation date of each filename with the list that includes all days
 5. Obtain the index of the day in the list that corresponds to the creation date of the filename
 6. Assign that index as value to the filename key
 7. Return the dictionary
- **sort_by_date**
 1. Take a list of filenames (imported *filenames.txt*) in the format: “*Subject03_1-2-2015_stims4humans.csv*” or “*Subject06_01_20-Jul-2015_stims4humans.csv*”
 2. Iterate through each filename
 3. Extract the date: “*1-2-2015*” or “*20-Jul-2015*”
 4. Separate the date in list [day, month, year]
 5. If month is not a digit, convert it to a number with function **convert_month2number**
 6. Convert the date to integers (it is a string)
 7. After extracting dates from all filenames, use library **datetime** to convert each string of day, month, year into a date
 8. Given the association of filename with the extracted date, sort the list of filenames by the date that was extracted using the Schwartzian transform or decorate-sort-undecorate idiom
 9. Use function **n_days_dict** to create a dictionary that associates each filename to the day it was created:

Key	:	Value
<i>Subject01_06-May-2020_description.csv</i>	:	01
<i>Subject01_10-5-2020_description.csv</i>	:	05
<i>Subject01_14-5-2020_description.csv</i>	:	09
<i>Subject01_17-May-2020_description.csv</i>	:	12

10. Return the sorted list of filenames and the dictionary

- **find_subject**
 1. Take the subject ID, the list of lists that contains the sociodemographic data and the header information from file *Subjects_info.csv*
 2. Find the subject ID in the list and assign the value in each important column to a variable: group, gender, age, musical training, laterality, language and education.
 3. Return the index and all the created variables

- Import *Subjects.txt* and sociodemographic data from *Subjects_info.csv*
 1. Go to main directory
 2. Import *Subjects.txt* and *Subjects_info.txt* files with function **read_simple_txt**
 3. Convert subjects' information to a DataFrame and calculate mean, max and min of some of the variables
- Create the full dataset (*Tablet_training_FULLresults.csv*)
 1. Iterate through each subject using list of subjects (from *Subjects.txt*)
 2. Import *filenames.txt* to a list with function **read_simple_txt**
 3. Use function **sort_by_date** on list of filenames
 4. Look for repetitions in dates' dictionary to correct them manually if needed
 5. Iterate through each file using sorted list of filenames
 6. Import each *.csv* file in subject's folder with library **pandas**
 7. Use function **find_subject** to extract the sociodemographic information corresponding to the subject in the current loop
 8. Assign that information in new columns in the imported dataset to be able to group data later
 9. Append the dataset of each day in a list of DataFrames
 10. Once the iteration on subjects is done, concatenate list of DataFrames in one
 11. Export full dataset to *.csv* file: *Tablet_training_FULLresults.csv*
- Organize dataset per groups using **pandas** DataFrame properties
 1. Sort DataFrame by different groups (day, age group, subject, age, gender, musical training, laterality, spoken languages and education level, imported from *Subjects_info.csv*).
 2. Get the maximum level reached per day per subject and assign this DataFrame to new variable.
- Plot using **seaborn.relplot** library
 1. Create a plot per each different group
 2. Save each figure

Results

By creating a dataset that contained the data from all training days and all subjects, it was possible to analyze and graph easily any variable inside it. I deemed that the most interesting one was the highest level of difficulty reached per day per subject because it would give an estimate of how each subject was responding

to the training. As expected, the age of the subjects played an important role in their performance in the task, as younger subjects reached higher levels since the beginning of the training (Figure 2, left). Surprisingly, laterality also yielded differences in performance (Figure 2, right).

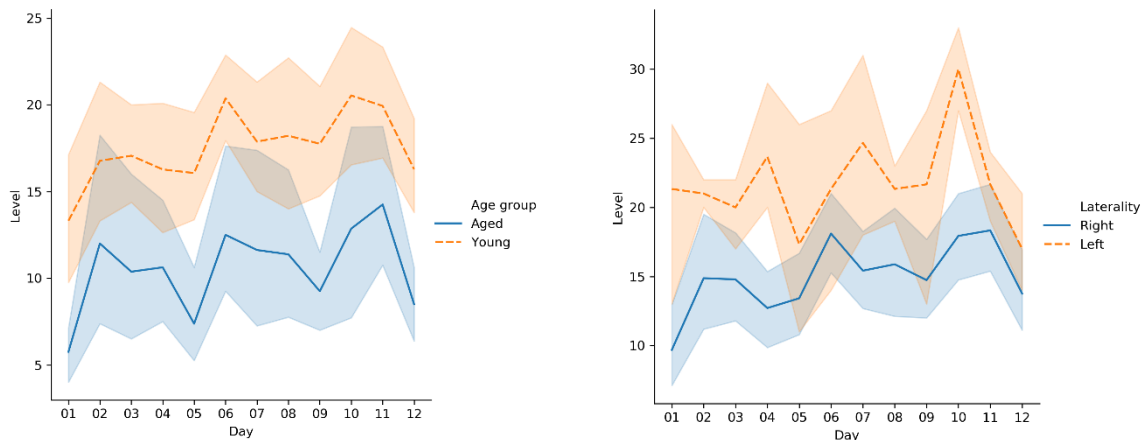


Figure 2. Maximum level of difficulty reached each day when grouped by age (left) or laterality (right). As expected, young adults have better performance than old adults. Surprisingly, left laterality improves the performance.

Discussion

Difficulties

The development of this code was a true challenge because with my current knowledge, I deemed necessary to use different languages. As each language has its own syntax and management of variable and complex datasets, the first roadblock was to be able to program in those three. To manage these difficulties, I resorted to the Internet to find the basic syntax for MATLAB and Bash and to keep it present while programming.

The next challenge occurred during the importation of .mat files to Python. The data structure of that file consists in a structure composed by cells. Both are data types that are exclusive of MATLAB and are not compatible for direct importation to Python. To solve this, I had to modify the structure in a way that resembled as much as possible the rest of the .csv files. Subsequently, I converted the data to a table and then to a .csv file. These multiple changes created a file identical that could be imported with the same code as the others.

Once all the data was importable to Python, the next issue was to sort the files by date as this is crucial for this dataset and for the analysis. The first approach that I took was to get the “creation date” from the metadata of the file; however, as I transferred the files from one folder to another in the bash script, the date had been modified. Then, I decided to use string manipulation to retrieve the date directly from the filenames; but not all of them were named the same, so I had to create a function that did the conversion of the months and to count the number of underscores to decide which part of the split string was the date. Once it was extracted and converted to numbers, I had to use a specific library (**datetime**) to convert those numbers into a date that could be sorted easily.

The next complication was to find a way to sort both the list of filenames and the dates extracted from them at the same time and in the same order using only the list of dates. I knew that it would require the use of zip function but after trying different things, I managed to find a code in StackOverflow that provided a solution using an example of decorate-sort-undecorate idiom. With the data sorted by date and after learning how to use **pandas**, the importation of the data was simple.

However, once all the data was in a DataFrame, the next problem was to sort it in a way that allowed me to compare the data per group, as it is necessary for this project. To do so, I had to add columns in the DataFrame that described each entry with the different groups it belonged to. Next, the DataFrame had to be sorted by those groups to be able to graph it, especially because the library **seaborn** required the data in a special format to be able to make the graphs I thought could represent the data in the best way. I resorted to the function **.index()** of **pandas**, and then to calculate the maximum level reached per day per subject. The next obstacle occurred when I tried to plot the data because as the grouping columns were indexed, **seaborn** couldn’t access them. It was then that I unindexed them and saved them in a new variable that saved only the data that I was going to use to plot.

Finally, using the function **seaborn.relplot** and experimenting with each one of the arguments inside it, I was able to get Figure 1, left. After it, it was only a matter of choosing different groups to get to the final results.

Future work

The final script, although useful, is far from perfect. There are many aspects of it that could be improved to have better results and a more efficient code with lower computing times. In the current status of the code, it is necessary to run twice *Organizing_files.sh* to update the *filenames.txt* files so that they include the converted .mat files by *Export_csv_files.m*. To avoid this would be one of the first steps to make the code more effective.

Another inconvenience of the current code is to use three different languages to get the results. Although difficult, it would be possible to integrate the conversion of .mat files into *Import_Analysis_Behaviour_files.py* through libraries such as **scipy.io.loadmat()**. It is also possible to substitute the bash script with the **global module** in python, and therefore, to avoid the use of three different programs to reach the same result.

Finally, the dataset is complex enough to do more complicated analysis. The results presented in this project are just a small example of what can be achieved with it.

References

- [1] Cisneros-Franco, J. M., Ouellet, L., Kamal, B., & de Villers-Sidani, E. (2018). A brain without brakes: reduced inhibition is associated with enhanced but dysregulated plasticity in the aged rat auditory cortex. *eNeuro*, 5(4). doi: 10.1523/ENEURO.0051-18.2018
- [2] Zatorre, R. J., Fields, R. D., & Johansen-Berg, H. (2012). Plasticity in gray and white: neuroimaging changes in brain structure during learning. *Nature Neuroscience*, 15(4): 528-536. doi: 10.1038/nn.3045
- [3] Senderle (<https://stackoverflow.com/users/577088/senderle>). How to sort two lists (which reference each other) in the exact same way. URL (version: 2018-01-16): <https://stackoverflow.com/a/9764364>

Appendix

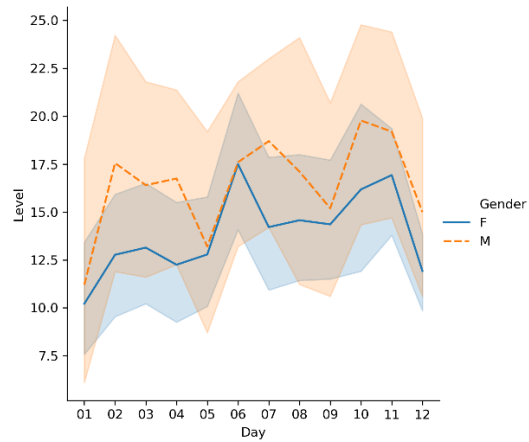
```
Main folder
├── Subjects_info.csv
├── SubjectNM01
│   ├── SubjectNM01_18-7-2016_stims4humans.csv
│   ├── ... (~12 files per folder)
│   └── SubjectNM01_29-7-2016_stims4humans.csv
├── SubjectNM02
│   └── ...
├── ...
├── SubjectNM08
│   └── ...
├── Training_1
│   ├── Subject01_02_14-Jan-2015_stims4humans.mat
│   ├── Subject01_03_16-Jan-2015_stims4humans.mat
│   ├── Subject01_04_20-Jan-2015_stims4humans.mat
│   ├── Subject01_05_23-Jan-2015_stims4humans.mat
│   ├── Subject01_13-1-2015_stims4humans.csv
│   ├── Subject01_15-1-2015_stims4humans.csv
│   ├── Subject01_17-1-2015_stims4humans.csv
│   ├── Subject01_18-1-2015_stims4humans.csv
│   ├── Subject01_19-1-2015_stims4humans.csv
│   ├── Subject01_21-1-2015_stims4humans.csv
│   └── Subject01_22-1-2015_stims4humans.csv
├── Training_2
│   └── ...
├── ...
├── Training_8
│   └── ...
├── Training_A1
│   └── ... (Same as Training1)
├── ...
└── Training_A8
    └── ...
```

24 directories, 277 files

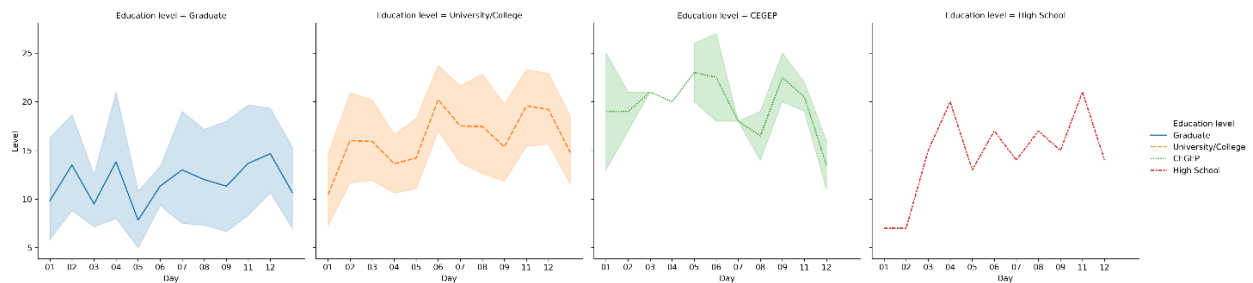
Supplemental figure 1. Data structure to import to Python for analysis.

Supplemental table 1. Sociodemographic data from all subjects.

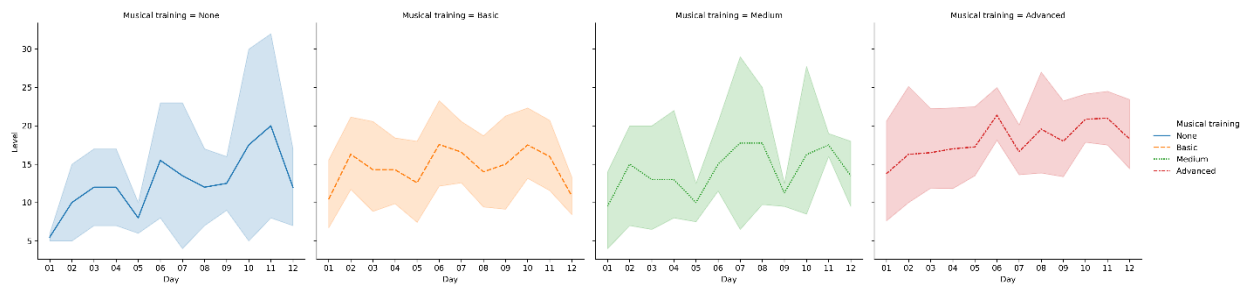
<i>Subjects</i>	<i>Group</i>	<i>Gender</i>	<i>Age</i>	<i>Musical training</i>	<i>Laterality</i>	<i>Language</i>	<i>Education</i>
<i>SubjectNM01</i>	Young	F	19	Basic	Right	4	University/College
<i>SubjectNM02</i>	Young	F	20	Advanced	Right	2	University/College
<i>SubjectNM03</i>	Young	M	24	Advanced	Left	3	Graduate
<i>SubjectNM04</i>	Young	F	22	Basic	Left	3	CEGEP
<i>SubjectNM05</i>	Young	F	21		Right		CEGEP
<i>SubjectNM06</i>	Young	F	20	Advanced	Right	2	High School
<i>SubjectNM07</i>	Young	M	28	Medium	Right	2	University/College
<i>SubjectNM08</i>	Young	M	19	Advanced	Right	2	University/College
<i>Training_1</i>	Young	F	23	Advanced	Right	3	University/College
<i>Training_2</i>	Young	F	20	Medium	Right	1	University/College
<i>Training_3</i>	Young	M	35	Basic	Right	2	Graduate
<i>Training_4</i>	Young	F	20	Advanced	Right	3	University/College
<i>Training_5</i>	Young	F	20	Advanced	Right	4	University/College
<i>Training_6</i>	Young	M	22	Advanced	Right	2	University/College
<i>Training_7</i>	Young	M	27	None	Right	3	University/College
<i>Training_8</i>	Young	F	20	Medium	Left	2	University/College
<i>Training_A1</i>	Aged	F	69	None		2	Graduate
<i>Training_A2</i>	Aged	M	76				Graduate
<i>Training_A3</i>	Aged	F	77	Basic	Right	4	University/College
<i>Training_A4</i>	Aged	F	54	Medium	Right	2	University/College
<i>Training_A5</i>	Aged	M	66	Basic	Right	1	University/College
<i>Training_A6</i>	Aged	M			Right	1	
<i>Training_A7</i>	Aged	F	42	Basic	Right	2	Graduate
<i>Training_A8</i>	Aged	M	43	Basic	Right	2	Graduate
<i>SubjectNT01</i>	Young	F	20	Advanced		4	University/College
<i>SubjectNT02</i>	Young	F	24	Medium	Right	3	University/College
<i>SubjectNT03</i>	Young	F	21	Medium	Right	3	High School
<i>SubjectNT05</i>	Young	F	23	Advanced		2	University/College
<i>SubjectNT06</i>	Young	F	21	Basic	Right	3	High School
<i>SubjectNT07</i>	Young	F	28	None	Right	2	University/College
<i>SubjectNT08</i>	Young	M	21	Basic	Right	3	University/College



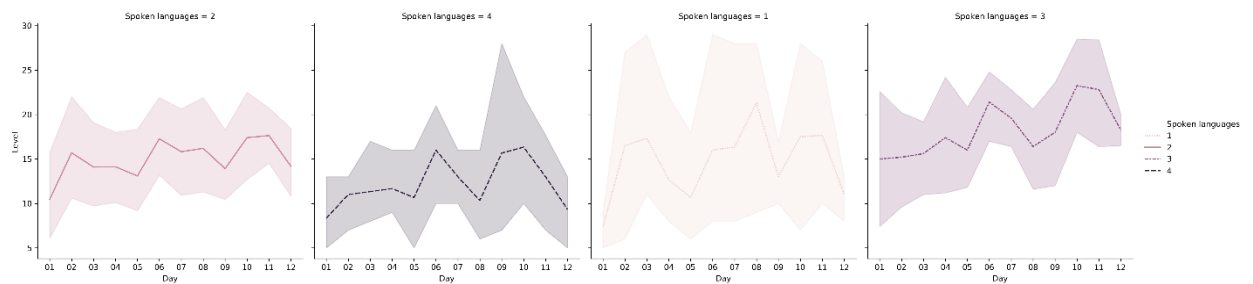
Supplemental figure 2. Maximum level of difficulty reached each day when grouped by gender. No difference was found.



Supplemental figure 3. Maximum level of difficulty reached each day when grouped by education level. No difference was found.



Supplemental figure 4. Maximum level of difficulty reached each day when grouped by musical training. No difference was found.



Supplemental figure 5. Maximum level of difficulty reached each day when grouped by number of spoken languages. No difference was found.