**Documentation Automating SPM Multiple Regression Analysis of VBM-MRI Data**

By Stéphane Hess, 17th of July, 2017 in Wabern, Switzerland in collaboration with Prof. Sandra Báez, Universidad de los Andes in Bogotá, Colombia.

**Report of Work Done**

**Objective**

The goal of this project was to write an algorithm that allows for an automated execution of multiple regression analysis of VBM MRI data and different behavioural variables. The algorithm has to account for different missing values in the different behavioural variables.

**Carrying out the Analysis by Use of the GUI**

First the VBM MRI (voxel based morphometry magnetic resonance imaging) data was analysed by using the graphical user interface (GUI) of SPM (statistical parametrical mapping) in Matlab:

VBM MRI data showed volumes of grey and white brain matter of test persons: Healthy control persons, Alzheimer Disease patients and Fronto-Temporal Dementia patients. By dividing the grey and white matter volumes by the total intracranial volume (TIV) the proportion of grey and white matter in each brain was obtained. The SPM multiple regression procedure then searched for correlations between the relative volume of brain structures and behavioural variables (?). The age of the test persons was chosen as a covariate (i.e. the correlations were controlled for age). The procedure was carried out on two different regressors: Ojos\_17 and Tasit\_tot.

Image processing was performed according to the instructions given in the tutorial by John Ashburner (VBM Tutorial, March 2015). When designing the model for the multiple regression the settings were chosen according to this same tutorial.

When using the ‘results’ function of SPM no masking was applied, FWE was used for multiple testing correction (p-value = 0.05) and the extent threshold voxels was set to 0.

The entire procedure was stored by saving the batches and the Matlab scripts in the folders ‘Compare\_Ojos\_GUI’ and ‘Compare\_Tasit\_tot\_GUI’ (file names: RegMulti\_TasitTot\_GUI\_16.7.17\_verified\_job.m and RegMulti\_TasitTot\_GUI\_16.7.17\_verified.m; RegMulti\_Ojos\_GUI\_16.7.17\_verified\_job.m and RegMulti\_Ojos\_GUI\_16.7.17\_verified.m respectively).

Screenshots were used to document the settings of the GUI in general and the settings chosen when creating the batch.

**Automating the Procedure**

The first two steps of the SPM analysis procedure were automated: For each one of the two SPM functions ‘Basic models’ and ‘Estimate’ a script was written. Typing the filename without extension into the Matlab command window and pressing ‘enter’ runs the scripts (provided spm is activated). The third step (the ‘Results’ function) has not been automated and must still be carried out by using the GUI.

The SPM procedure stored in the job-script (obtained by saving the batch defined by using the GUI) has to be carried out multiple times for different behavioural variables. The code in the job-file should thus be incorporated in a loop. This alone, however, would not be sufficient since the code in the job-script is only executed by running a second script, which was also created when saving the GUI-procedure. This script contains the spm\_jobman command, which prompts SPM to execute the batch saved in the job-script. In order to integrate this procedure in a loop the instructions from both scripts must be integrated in one code, i.e. the instructions of the batch from the job-script and the spm\_jobman command must be integrated in one script. This is done according to the instructions on: <https://en.wikibooks.org/wiki/SPM/Batch>

Note that the type of data used must be specified in the code: In the present case this is PET (PET and VBM is the same option in SPM).

The first script automates the ‘Basic model’ function of SPM. The code first loads all the needed data from the working directory into the Matlab workspace.

The loop then performs one procedure per behavioural variable. For each variable it checks the behavioural data for missing values and subsequently eliminates all data points corresponding to the missing values from MRI data, from the age data, from the total intracranial volume data and from the behavioural data. In the same round the loop then carries out the Matlab batch from SPM. In each round an SPM.mat file and a new folder are created. The newly created SPM.mat file is moved to the newly created folder. This is necessary because the file would be overwritten by the next file (created in the next round of the loop) if it were to stay in the working directory.

The second script then loads the data created by the first script (the SPM.mat files) into Matlab and loops through them. For each one the batch from the ‘Estimate’ function is executed. The newly created files are stored in the same folder where the SPM.mat file is.

Finally each folder can be accessed in order to execute the ‘Results’ function by using the SPM GUI.

**Testing the Scripts**

The results obtained by using the automated procedure (running the scripts) for the variables Ojos\_17 and Tasit\_tot were compared to the results obtained for the same variables by using the GUI. The results matched which is an indication that the automated procedure is working as it is supposed to. The batches used in the GUI procedure were documented by saving the batch both as a file that can be loaded by the GUI and as scripts. Screenshots were used to document the settings of the GUI in general and the settings chosen when creating the batch.

The folders containing the results obtained by running the scripts are recognisable by the tag ‘RegMulti\_Age’ at the beginning of their names.

**Instructions for Using the Scripts**

The code in the first script (RegMulti\_FullLoopDataLoad.m) refers to different working directories or file paths several times. Make sure that all needed files are stored in the right directories and that you adapt the paths and folder names in the script accordingly (see instructions in the script).

The script must be stored in the current working directory. Then type the file name of the script (without extension) into the Matlab command window. The script will then run in Matlab. Do not do anything with Matlab until the process is finished. When the prompt reappears in the Matlab command window the process is finished and Matlab is ready for new instructions.

All SPM.mat files have now been created and stored in newly created folders. Running the second script (EstimateAuto.m) will modify the SMP.mat files (?) and create new files that will be stored in the same folders. **Important:** Be sure to run the following command after running the first script and before running the second script:

clearvars

After running the first script the batch of the first script and a lot of variables are still stored in the Matlab workspace. This can confuse SPM (probably because spm\_jobman will try to execute the previously stored batch?). The clearvars command clears the Matlab workspace and you are ready to go. Again you run the script by typing its file name without extension into the Matlab command window.

After running the script, again make sure to run the clearvars command.

To obtain the results, access each folder by setting it as your working directory (use the cd command followed by the path to the folder in question). Once you are in the correct folder carry out the ‘Results’ procedure by using the SPM GUI.

**Pitfalls to Watch out for**

While working on my code I encountered a series of technical difficulties. In order to avoid similar problems when using or adapting my scripts it may be important to pay attention to the following points:

**Define SPM settings**

When running a Matlab batch from a script, the following commands have to precede the Matlab batch:

spm('defaults','fmri'); spm\_jobman('initcfg');

At the end of the Matlab batch, Matlab needs to be told to actually execute the batch. This is done by the following command:

spm\_jobman('run',matlabbatch);

Hence, the part of the code where the Matlab batch is specified should have the following structure (according to <https://en.wikibooks.org/wiki/SPM/Batch>, 17th of July, 2017):

spm('defaults','fmri');

spm\_jobman('initcfg');

matlabbatch{1}.spm... = ...;

spm\_jobman('run',matlabbatch);

Note the specification in the first line of code above: ‘fmri’ This specification must, of course, be adapted depending on the kind of data you are working with. In the present case ‘PET’ is the correct specification. In the beginning of my work I did not pay attention to this and did not make the adaption (stupid!). This first lead to different results when running the scripts as opposed to using the GUI. Importantly, running the script with the wrong specification changed this setting in the GUI as well. Hence, when using the GUI subsequently I obtained different results than when using the GUI in the beginning. Figuring out why this was happening led to a lot of headache and loss of time!

Note that if you save a batch (set up by using the GUI) as a script, the batch will be saved in two different scripts (as mentioned above). The specifications of the settings along with the spm\_jobman command will not be saved in the job-file but in the other file. When automating a procedure that was previously executed by the GUI, it is therefore important to pay attention to this script file as well.

**Accidentally Cutting off Decimals when Importing Data**

As I experienced the way data are imported into Matlab can affect how many decimals (digits after the decimal point) are imported. The exact reasons why this happened are beyond me since I am not a computer scientist. Getting to the bottom of this would be too time consuming. But based on my experience I can speculate about where the causes of the problem are. For practical purposes, in absence of certain knowledge about the cause, it is best to avoid all possible causes in order to avoid problems.

* *Working on data in the Matlab command window and then copy paste them to the script:*

When trying to figure out the reasons for the problems with my code I experimented with a lot of different versions of my code, which involved a lot of copy-pasting. When writing one of my code versions I imported data into the Matlab workspace and printed out the data in the Matlab command window, where I selected the needed data points. I then copy pasted the selected data into my script. In this way I only got the decimals that were printed out in the Matlab command window. So, copy-pasting data from the command window into the script should probably be avoided.

* *Corrupted Data Files?*

When experimenting with my codes I created many new folders to store new versions of my code. I usually copy-pasted the necessary data files (the files containing the behavioural data, the covariates and the total intracranial volumes) to the new folders as well. While experimenting, some attempts to run my scripts produced errors. I am therefore asking myself if some of my file copies could have been corrupted in the process. In order to be sure to avoid such problems it might be a good idea to recreate the data files from the original excel file.

**Running a Script without Clearing the Matlab Workspace First**

When running a script, the variables, data and of course the Matlab batch specified in that script are imported into the Matlab workspace. If those objects are not removed before running a second script they might confuse Matlab. I obtained different results when running a script without clearing the workspace first than I did when first clearing the workspace. I am not sure as to what the reason is, but I am guessing that the spm\_jobman command in the second script might try to execute the batch specified in the previous script since it is still present in the workspace.

I therefore inserted the clearvars command in the beginning of each one of my scripts. Nevertheless, it might be a good idea to get into the habit of running the clearvars command every time before running a new script (except if that script for some reason needs to refer to previously introduced objects, of course).

**Too Much Copy-Pasting May Lead to Confusion and Mistakes**

When modifying or further developing code, it may be a good idea to keep the original version of code in order to be able go back in case there are problems with the new versions. The different versions of code should, of course, be saved either with different file names or in different folders. Yet, if working on a larger project one may end up with a multitude of different folders, subfolders and file versions. This situation is prone to errors such as accidentally saving a new file in the wrong folder. In order to avoid these problems it might be a good idea to work with a version control system such as Git or SourceTree.