

# The Functionnectome User-guide

Version 0.0.2

## Content

<b>Introduction.....</b>	<b>2</b>
Preliminary notes .....	2
Requirements (Read this before starting).....	2
General framework .....	3
Quick-start .....	3
<b>Comprehensive guide .....</b>	<b>4</b>
Installation.....	4
Loading the input 4D volume .....	4
Voxelwise and regionwise analysis .....	7
Number of processors to use .....	8
White matter anatomical priors.....	9
Output options.....	10
Save, Load, Launch & Exit .....	11
<b>Frequently asked question (probably) and advanced functions .....</b>	<b>12</b>
How do I use the Functionnectome with command lines only? .....	12
Can I manually change / create a setting file? .....	12
How do I specify different paths to alternate priors? .....	12
How do I make my own anatomical priors? .....	13
How do I create my own HDF5 file for my priors? .....	14
How do I use the Functionnectome on a distant computational grid/server? .....	14
Will this work on my OS? .....	14
Will parallelizing the analysis use more RAM? .....	14
I think you are doing something wrong / poorly optimized... ..	14

## Introduction

The Functionnectome is a Python-based tool developed to apply the Functionnectome method, projecting functional signal (or more generally gray matter information) onto the white matter for further analysis. As such, it accepts **NifTI 4D volume (usually fMRI files) as input**, uses white matter structural **priors** for the computation, and **output functionnectome 4D volumes**. It also includes a GUI (Graphical User Interface) for an improved ease of use.

### Preliminary notes

- “**Functionnectome**” (with an uppercase “F”) refers to the method, or to the program applying the method; whereas “**functionnectome**” (with a lowercase “f”) refers to the 4D volume, resulting from the application of the Functionnectome.
- “**Voxelwise**” analysis refers to the projection of functional signal to the white matter done at the voxel level, and thus requires hundreds of thousands of processing steps. “**Regionwise**” analysis runs on predefined parcels, grouping voxels together and reducing the number of steps to the order of hundreds at most, though at the cost of precision.
- Using the GUI should be quite straightforward for the most part. Don’t hesitate to have a look before diving into the user guide.

### Requirements (Read this before starting)

1. The Functionnectome runs with **Python 3.6** (or superior). It does not use any fancy module, so basic Python distributions should be enough.
2. Depending on the size of the input data, RAM usage can vary. 4GB of RAM is probably a bare minimum for small input volumes. Prefer **at least 8GB or 12GB**. More is better, especially for big input files.
3. Expected input files are **NifTI (.nii or .nii.gz) 4D volumes**.
4. Input files should either be all in the **same folder, or be in different folders but with “similar paths”**, i.e. with the same number of subfolders and the folder with the data ID at the same depth (e.g. `“/myHome/subject01/data/mydata.nii”` and `“/myHome/subject02/data/mydata.nii”`).
5. For voxelwise analysis, a **mask** (selecting the voxels with “interesting” functional signal) is also necessary. It should be a **NifTI 3D volume**, with the same spatial dimensions as the main 4D volume input.
6. Don’t forget to download the **white matter anatomical priors**. (In the future, should be automated)

7. The provided priors require input data to be in the **MNI152 space, with 2mm<sup>3</sup> isotropic voxels**.
8. We strongly recommend to run the Functionnectome on a machine equipped with multiple CPUs (8 cores or more). At least for the voxelwise analysis. Regionwise analysis is less computationally hungry. Note that process parallelization should not increase the RAM load.

## General framework

With the GUI, you can set the analysis parameters, define the input and the output, and save or load those settings.

- Input: 4D volume, NifTI.
- (opt.) Input mask: 3D volume, NifTI.
- Output1: the **functionnectome**, 4D volume, NifTI.
  - User defines the **general output folder**
  - A **subfolder** is generated for **voxelwise** or **regionwise** analysis
  - A second **subfolder** is generated with the **unique ID\*** of the input data
  - The output 4D data is saved there and named **"functionnectome.nii.gz"**
  - Example:  
`"/myHome/myOutput/voxelwise_analysis/subject01/functionnectome.nii.gz"`
- Output2: 3D volume, NifTI.
  - **sum\_probaMaps\_voxel.nii.gz** or **sum\_probaMaps\_region.nii.gz**
  - In the same directory as Output1
  - Sum of all probability maps used for a given subject (depends on the mask).
  - Used during the processing, maybe useful to someone else (?)

**\*Unique ID:** If all input files are in the same folder, the program uses the name of the file (without the extension). If the files are in different folders with similar paths (cf. bullet 4 in **Requirements**), the ID is the name of the folder branching between subjects.

## Quick-start

1. Select the input 4D files by clicking on either "Choose files" buttons.
2. Select whether you want a regionwise or voxelwise analysis
3. If you chose the voxelwise approach, select the masks that will filter out unwanted voxels. You can either chose one mask per input file, of one unique mask applied to all input files.
4. Select the number of parallel processes you want to launch concurrently. The more, the faster, but it is obviously limited by the actual number of available CPU (core).
5. Chose the prior type. Usually, we recommend using the HDF5 (this is also what we provide you with).

6. Chose the output folder. If it doesn't exist, it will be automatically created. The output filenames are automatically generated.
7. Choose whether or not you want to mask the output to remove voxels outside of the brain (as defined by a template).
8. Launch the analysis. Or save the settings to launch it later or with a command line.
  - If the priors have not yet been downloaded (or the path leading to the files not set), you will be offered to automatically download it (or asked to set the path). This should only happen once.

## Comprehensive guide

### Installation

Download the project from GitHub: <https://github.com/NotaCS/Functionnectome>

I will also create a package that can be installed with pip in the near future.

Download the priors (one HDF5 file):

[https://www.dropbox.com/s/22vix4krs2zgtnt/functionnectome\\_7TpriorsH5.zip?dl=0](https://www.dropbox.com/s/22vix4krs2zgtnt/functionnectome_7TpriorsH5.zip?dl=0)

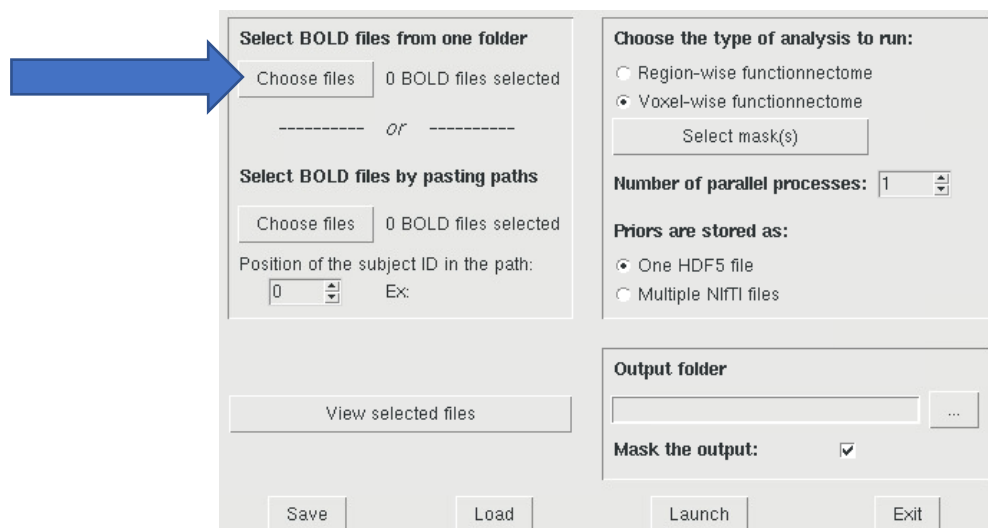
Unzip the priors and put the path to the file in the Functionnectome.py script: replace the placeholder `'/put/your/path/here/functionnectome_7Tpriors.h5'` in the “priors\_paths” dictionary (should be line 362) by your path.

In version 0.1.0 (soon), the download and setup of the paths will be automated.

### Loading the input 4D volume

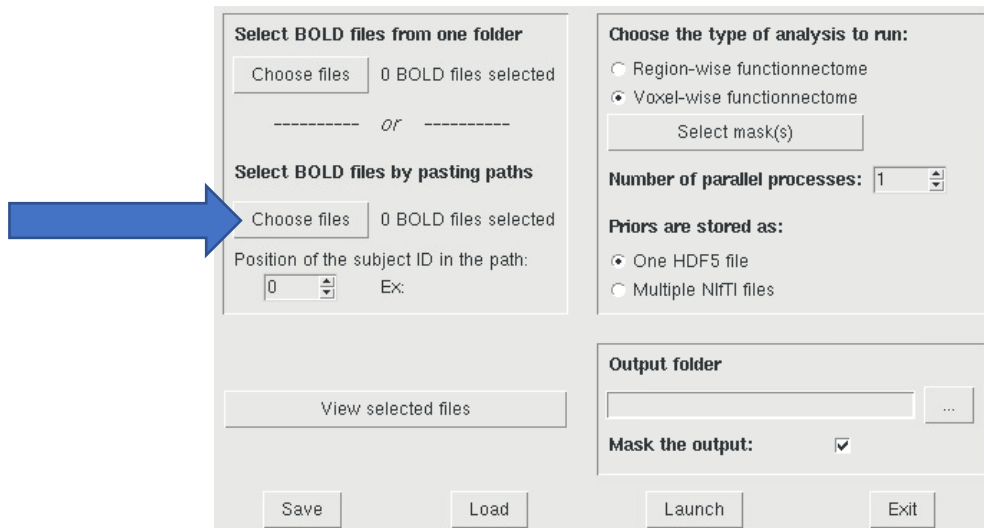
Once you have downloaded the package, launch the GUI. There, you can specify the input data (assumed to be BOLD data, but any 4D nifty files are accepted). Two ways of selecting the files are possible depending on how they are stored on your computer:

1. If they are all in the same folder, click on the “Choose files” button just under the “Select BOLD files from one folder” label.



An generic interactive window will open to allow you to chose the files from a folder. You can manually select those you want to keep/discard. Only nifti files are accepted (extension “.nii” or “.nii.gz”).

2. If the files are in different folders (but properly organized with folders and sub-folders following the same pattern), click on the “Choose files” button under the “Select BOLD files by pasting paths” label.



A new window will open. In the central box, you can paste the paths of each file. An easy way to obtain the paths of all the files in one go requires the use of a command window (a.k.a a terminal).

**On Unix/Linux**, use the “ls” command.

For example, let’s say that all my files are store in paths following this naming convention:

“/myhome/current\_project/subject????/functionalFiles/ fMRI\_acquisition.nii.gz” with “????” being unique ID (e.g a number), different for every subject. Use as many “?” as there are missing characters.

To display all the file-paths, simply enter the command (in the terminal):

“ls -1 /myhome/current\_project/subject????/functionalFiles/fMRI\_acquisition.nii.gz”

Here, “ls” will list all the paths, and the “-1” option will display one path per line.

**On Windows**, use the “dir” command, similarly to the above-mentioned “ls”. For example: “dir C:\myHome\project\subject????\fMRI\_acquisition.nii.gz”

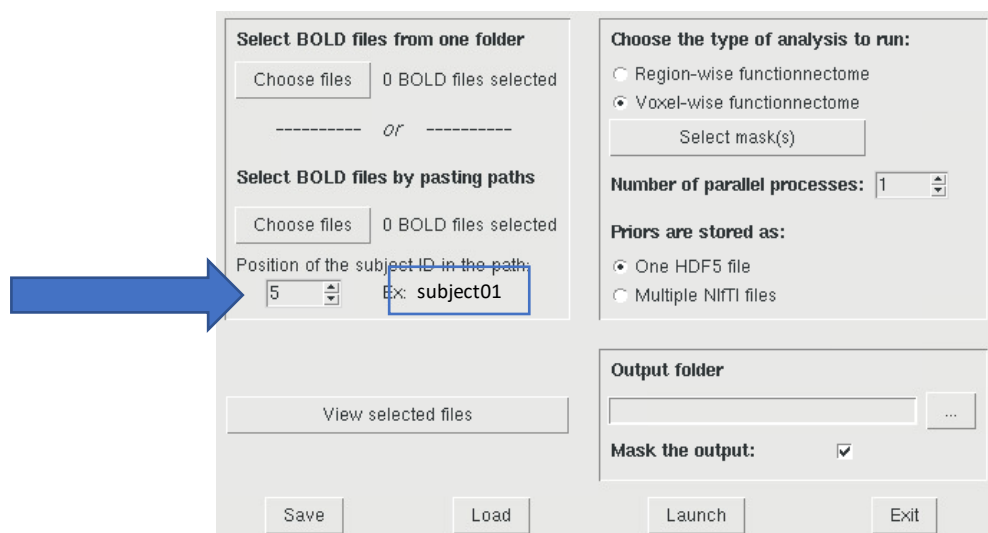
Copy the displayed result and paste it in the GUI.

If for some reason you enter or paste something wrong in the GUI text box, you can easily correct it manually or clear the whole box by clicking on “Clear”.

When you are done, click on “OK” to accept the file-paths. Or “Cancel” to return to the main window without keeping the paths or the changes made to existing paths.

**Advanced tip:** Once you have files selected, you can manually edit them (add/remove) by opening the “pasting” window and adding ore removing a line, even if you selected the all from one folder. Note that the added paths should still keep the same naming pattern as the other already there. Likewise, if you “Clear” the window and click on “OK”, it will delete every path (not the files of course, just the paths stored by the GUI). If you accidentally modify or clear the paths, simply click on “Cancel”.

To differentiate the input files and name the output folders, a **unique ID** is attributed to each of them. When selecting all inputs from the same folder (option 1), the **unique ID** is the name of the file (without the extension). When selecting from multiple similar folders (option 2), the program will automatically select the first divergence in the paths as the ID. The position of the **ID** in the path is displayed, and an **example of ID** (from the first selected path) is given. Subsequently, the GUI gives you the option to manually change where in the paths should the program look for an ID, using the little arrows near the position number:



List of the paths given:

/myHome/workspace/myProject/myAcquisition/MRI/**subject01**/fMRI/readyData.nii.gz

/myHome/workspace/myProject/myAcquisition/MRI/**subject02**/fMRI/readyData.nii.gz

/myHome/workspace/myProject/myAcquisition/MRI/**subject03**/fMRI/readyData.nii.gz

Position of the subject ID in the path (index of the folder in the path, **starting at “0”**):

/myHome/workspace/myProject/myAcquisition/MRI/**subject01**/fMRI/readyData.nii.gz

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Here, “**subject01**” has been selected automatically selected as it is the first (and only in the example) divergence between the paths.

Sometimes, the first divergence is not the one you want to keep. You can then manually modify the position. For example, if we have the paths:

/myHome/myProject/session1/**subject01**/readyData.nii.gz

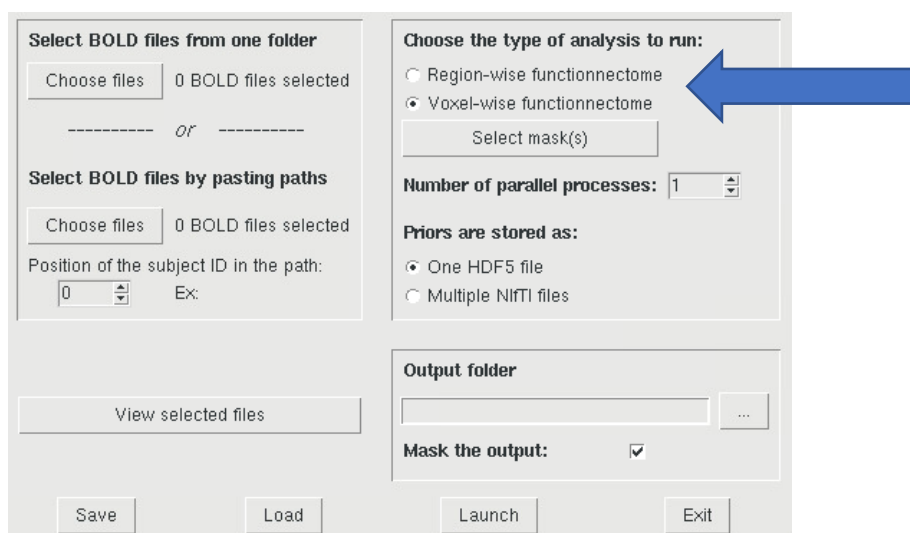
/myHome/myProject/session1/**subject02**/readyData.nii.gz

/myHome/myProject/session2/**subject03**/readyData.nii.gz

/myHome/myProject/session2/**subject04**/readyData.nii.gz

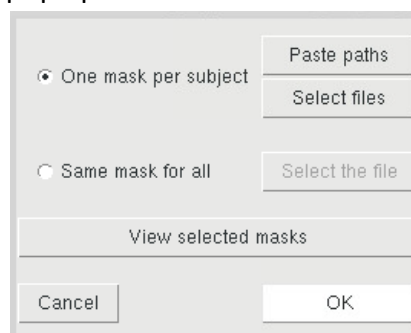
The number of the session is the first divergence between the paths, so the position will be automatically set at “2”, and the example given will be “session1”. The user will then have to change the position to “3” (which will change the example to “subject01”).

## Voxelwise and regionwise analysis



Select whether you want to run a **voxelwise** or a **regionwise** analysis. If you have enough computational power, we recommend using the voxelwise analysis.

If you select the voxelwise analysis (i.e. “voxel-wise functionnectome”), you will need to specify the mask(s) that will filter out unwanted voxels from the brain. Click on “Select mask(s)”. A new window will pop open:



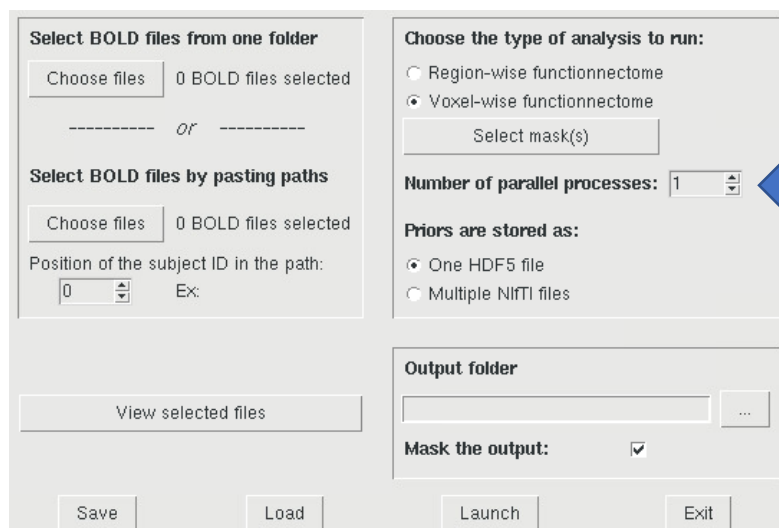
You can either attribute one mask per input file (“One mask per subject”), usually useful if the masks are created depending on the individual data; or select one unique mask that will be similarly applied to all inputs, for example a mask delimiting the gray matter.

Selecting one mask per input uses the same procedure than the selection of the input 4D volumes (Manual selection in one folder, or pasting the paths).

The paths of **the 4D inputs and of the masks** are automatically and systematically **sorted**. Consequently, these **inputs and the masks** must have the **same naming convention** because the pairing between a mask and an input 4D volume is done using the position on the paths in the corresponding list. Using the “View selected files” and “View selected masks” buttons, check the paths to verify that this is correct.

The regionwise analysis uses priors derived from a brain parcellation. In this case, both white matter probability maps masks are necessarily linked. In the provided default priors, we used the HCP multi-modal parcellation (MMP). The functional signal of a region is defined as the median of the voxel values at a given time point.

#### Number of processors to use



The screenshot shows the Functionnectome software interface. On the left, there are two sections for selecting BOLD files: "Select BOLD files from one folder" and "Select BOLD files by pasting paths". Both sections have a "Choose files" button and a "0 BOLD files selected" status. Below these is a "View selected files" button. On the right, the "Choose the type of analysis to run:" section has two radio buttons: "Region-wise functionnectome" and "Voxel-wise functionnectome" (which is selected). Below this is a "Select mask(s)" button. The "Number of parallel processes:" is set to "1" in a dropdown menu, which is highlighted by a blue arrow. Below this, the "Priors are stored as:" section has two radio buttons: "One HDF5 file" (selected) and "Multiple Nifti files". At the bottom right, there is an "Output folder" field with a browse button "...". Below that, the "Mask the output:" checkbox is checked. At the very bottom, there are four buttons: "Save", "Load", "Launch", and "Exit".

The Functionnectome processing time can be significantly cut down by parallelizing it. You can choose the number of processes (i.e. the number of CPU or core used at the same time). The maximum number of parallel processes is automatically detected and set as a top limit. The parallelization happens at the subject level, so only one subject is run at a time, which limit the RAM load.

The Functionnectome parallel processing system does not support the spread of the computation across multiple machines / nodes. All the CPUs must be mounted on the same machine.



As an example, using 16 processes in parallel, the processing time of a typical fMRI scan with the voxelwise analysis can be around 4 hours (but it depends a lot on the size of the input and how many voxels are kept with the mask).

### White matter anatomical priors

The priors are necessary for the Functionnectome to run. They consist of the white matter probability maps, but also a brain template. The template is a classical T1 brain template and is used to define the images dimensions and to delimit the brain. The program can access them from two different type of storage:

- A collection of NifTI files, with a folder containing all the probability maps (voxelwise or regionwise), and a brain template (same dimensions as the probability maps). This option might be better if you need to easily change (some of) the priors.
- An HDF5 file (containing all the required images). This format allows for a rapid access to the files, fast transfer of the file from one location to another (contrary to a folder containing hundreds of thousands of files...), and because it is not easy to change, it reduces the risk of deleting stuff by accident. Check the FAQ *“How do I specify different paths to alternate priors?”* if you want to create your own.

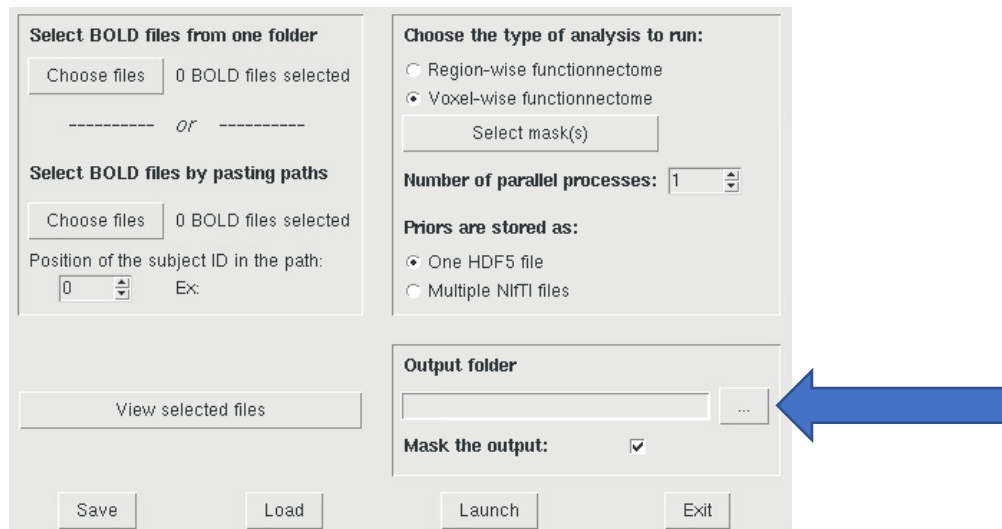
Paths to the priors are saved in a file and is purposely not readily available. Ideally, the paths to the priors should be set once (when downloading them), and left alone. It is possible to specify alternative paths directly in the settings file (see FAQ *“How do I specify different paths to alternate priors?”*). This way is to ease traceability, since the settings file is saved

The screenshot shows the Functionnectome settings window. The 'Priors are stored as:' section is highlighted with a blue arrow pointing to the 'Multiple NIfTI files' option. The 'Choose the type of analysis to run:' section shows 'Voxel-wise functionnectome' selected. The 'Output folder' section has a text box and a browse button (...). The 'Mask the output:' section has a checked checkbox. At the bottom are buttons for 'Save', 'Load', 'Launch', and 'Exit'.

with the results, so you'll always know what priors you used (default or manually changed).

## Output options

Chose the output folder by either clicking on the “...” button or by copy-and-pasting the path to the text-box.



If the folder does not exist, it will be automatically created. Inside of it will be stored the settings file (resulting from the parameters you entered in the GUI), and the sub-folders for the functionnectomes.

Typically, if you select “/myHome/myProject/myOutputFolder/” as the path to the output folder, the results will be stored in this structure (assuming you are doing a voxelwise analysis):

/myHome/myProject/**myOutputFolder**/

- settings.fcntm (text file containing the analysis parameters)
- voxelwise\_analysis/
  - subject01/
    - functionnectome.nii.gz
    - sum\_probaMaps\_voxel.nii.gz
  - subject02/
    - functionnectome.nii.gz
    - sum\_probaMaps\_voxel.nii.gz
  - subject03/
    - functionnectome.nii.gz
    - sum\_probaMaps\_voxel.nii.gz
  - subject04/
    - functionnectome.nii.gz
    - sum\_probaMaps\_voxel.nii.gz

The last parameter available is the “Mask the output” option. If checked (recommended and default behavior), the functionnectome will be masked to remove all voxels out of the brain (as defined by the brain template, which is part of the priors).

### Save, Load, Launch & Exit

The buttons at the bottom of the GUI are quite straightforward and control usual functions.

- **Save:** Saves the parameters entered on the GUI in a simple text file (with a “.*fcntm*” extension). It can be opened with any text editor. This settings file can be used manually launch the Functionnectome from a terminal. All the parameters must be properly entered in the GUI before saving is possible.
- **Load:** Loads a settings file (“.*fcntm*”) to fill the GUI parameters. Useful if you want to run a different analysis using the same input 4D files.
- **Launch:** Saves the parameters in the output folder, closes the GUI, and runs the analysis using the provided settings.
- **Exit:** Closes the GUI without doing anything.

## Frequently asked question (probably) and advanced functions

### How do I use the Functionnectome with command lines only?

You first need to create a settings file for your analysis (see below “Can I manually change / create a setting file?”). Let’s assume it is called “`settings.fcntm`” and saved in `/myPath/`

Then:

- If you manually downloaded the scripts and saved it in, e.g. `/myPath/`, you can run the script by simply calling:  
`python -u /myPath/Functionnectome/Functionnectome.py /myPath/settings.fcntm`
- If you installed the Functionnectome package using pip or conda, either you find where the `Functionnectome.py` script is stored (and use it as is shown above), or you need to create a simple script importing the package, then running the function. For example, something like that:

```
from Functionnectome.Functionnectome import run_functionnectome
fpath="/myPath/settings.fcntm"
run_functionnectome(fpath)
```

If it’s saved as `/myPath/runFunctionnectome.py`, you can then simply call (in the correct python environment):

```
python -u /myPath/runFunctionnectome.py
```

### Can I manually change / create a setting file?

Sure, it’s a simple text file on which I changed the extension for “`.fcntm`”. You can create your own file in a text editor (or copy a saved settings file from the GUI), you just need make sure that the internal architecture is the same as in a GUI generated file.

Don’t forget to have the “Number of subjects” and “Number of masks” values matching the respective numbers of paths you provide in the file.

The Functionnectome program possesses a simple system to check if the file is correctly organized, so if your custom settings file is completely off the mark, you’ll get a notification.

Note that the “Position of the subjects ID in their path” start at 0 (first folder). The last part of the path (i.e. the name of the file) can be designated with `-1`. This is actually what is used when loading all the 4D input files from a single directory.

You can find examples of settings files in the project directory.

### How do I specify different paths to alternate priors?

Since we strongly recommend not changing the default path to the priors file(s) too often (for the sake of traceability), we provide an alternative way to change the priors used for a

given analysis. This can be especially useful when working on a machine different from usual, as is the case when running analyses on a computer grid/cluster or a distant server.

To give non-default priors or override the absence of default priors, copy and paste the following lines at the end of the settings file (with the “.fcntm” extension) you are using as input for the Functionnectome program (add an empty line before the first “###” in the file):

```
###
HDF5 path:
    /myHome/myProject/priors/functionnectome_priors.h5
Template path:
    /myHome/myProject/anat/brainT1template.nii.gz
Probability maps (voxel) path:
    /myHome/myProject/priors/probaMaps_voxels
Probability maps (region) path:
    /myHome/myProject/priors/probaMaps_atlas
Region masks path:
    /myHome/myProject/priors/decomposed_atlas
###
```

The labels, written in green above, must always be present. Conversely, you only need to fill in the paths (in blue above) that are useful. For example, if you are using an HDF5 priors file, you only need to enter the first path. The lines where the unnecessary path would have been written must be kept though, with empty lines:

```
###
HDF5 path:
    /myHome/myProject/priors/functionnectome_priors.h5
Template path:

Probability maps (voxel) path:

Probability maps (region) path:

Region masks path:

###
```

An example file using this option can be found in the project directory:

- settings\_example2.fcntm

How do I make my own anatomical priors?

T.B.A

(You could use the “disconnectome” tool of the BCBtoolkit, with one-voxel masks for the “lesions”, and doing it for every brain voxel. This is what we did. But I plan to write a python script to do it more efficiently in the near future)

### How do I create my own HDF5 file for my priors?

We provide a script to import the priors in NifTI format into a single HDF5 file. You can find it in the project/package under the name `makeHDF5prior.py`.

It is not the final version yet. For now, you need to specify the paths to all folders directly in the script. And both voxelwise and regionwise priors must be given.

The script will detect all the NifTI files in the given folder, and import them in the HDF5 file. The voxelwise probability maps files must all be names with the following pattern:

`something_xx_yy_zz.nii.gz`

“something” can be any string of character, but must not contain a “\_”. “xx”, “yy”, and “zz” are the spatial coordinates of the specific voxel linked to the given probability map.

The regionwise probability maps and their associated region masks must have the same name.

### How do I use the Functionnectome on a distant computational grid/server?

The simplest way would be to transfer the Functionnectome.py and the HDF5 priors file to the machine you plan to use. Prepare manually the settings file (see “*Can I manually change / create a setting file?*”) and give the location of the HDF5 prior on the distant machine as an alternate priors file (see “*How do I specify different paths to alternate priors?*”). Then run the script with the settings file as argument (see “*How do I use the Functionnectome with command lines only?*”).

### Will this work on my OS?

The scripts have been tested on a Linux machine. It should work perfectly fine on a Mac. I worked hard to make it compatible with a Windows OS, but haven’t tested it yet...

### Will parallelizing the analysis use more RAM?

No. At least I don’t think so. The script uses shared memory between processes to avoid RAM overload, and more processes should only split the data in smaller chunks, not copy it multiple times.

### I think you are doing something wrong / poorly optimized...

Well, I tried my best, but I learned Python on the job. So, if you have any advice, as long as the discussion remains respectful, I’m happy to hear it. Leave a comment on the GitHub page and send me an e-mail.