

The Functionnectome User-guide

Version 0.0.2

Content

Introduction.....	2
Preliminary notes	2
Requirements (Read this before starting)	2
Overview	3
Comprehensive guide.....	4
Installation.....	4
Loading the input 4D volume	4
Voxel-wise and region-wise analysis	7
Number of processors to use	8
White matter anatomical priors.....	9
Output options.....	10
Save, Load, Launch & Exit	11
Frequently Asked Questions and advanced functions.....	12
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 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.
- “**Voxel-wise**” 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. “**Region-wise**” 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 voxel-wise 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 2mm³ isotropic voxels**.
8. We strongly recommend running the Functionnectome on a machine equipped with multiple CPUs (8 cores or more). At least for the voxel-wise analysis. The region-wise

analysis is much less computationally hungry. Note that process parallelization should not increase the RAM load.

Overview

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. (required for voxel-wise analysis)
- Output1: the **functionnectome**, 4D volume, NifTI.
 - The user defines the **general output folder**
 - A **subfolder** is generated for **voxel-wise** or **region-wise** 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**
 - Stored in the **same directory as Output1** (for voxel-wise analysis), or in the **general output folder** (for region-wise analysis).
 - Sum of all probability maps used for a given subject (depends on the mask).
 - Used during the processing, might be useful for alternative processing.

***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.

Basic steps for running an analysis:

1. Select the input 4D files by clicking on either "Choose files" buttons.
2. Select whether you want a region-wise or voxel-wise analysis
3. If you chose the voxel-wise approach, select the masks that will filter out unwanted voxels. You can either chose one mask per input file, or 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 necessarily limited by the actual number of available CPU (cores).
5. Chose the file type of the priors. 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.
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. *(not yet implemented)*

Comprehensive guide

Installation

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

A Python package that can be installed with pip will also be created shortly.

Download the priors (one HDF5 file):

https://www.dropbox.com/s/22vix4krs2zgtnt/functionnectome_7TpriorsH5.zip?dl=0

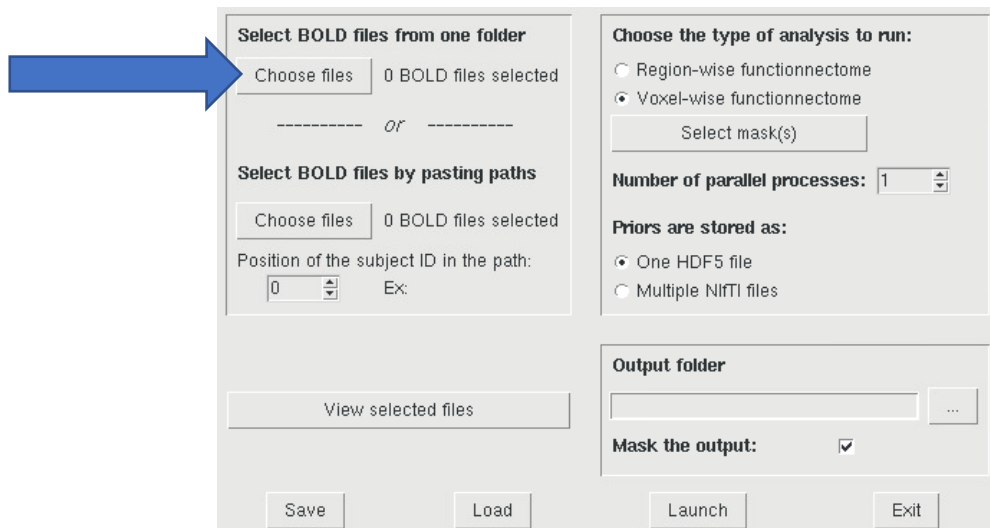
Unzip the priors and put the path to the file in the Functionnectome.py script: replace the placeholder `'/put/your/path/here/functionnectome_7Tpriors.h5'` in the “priors_paths” dictionary (should be line 362) by your path.

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:

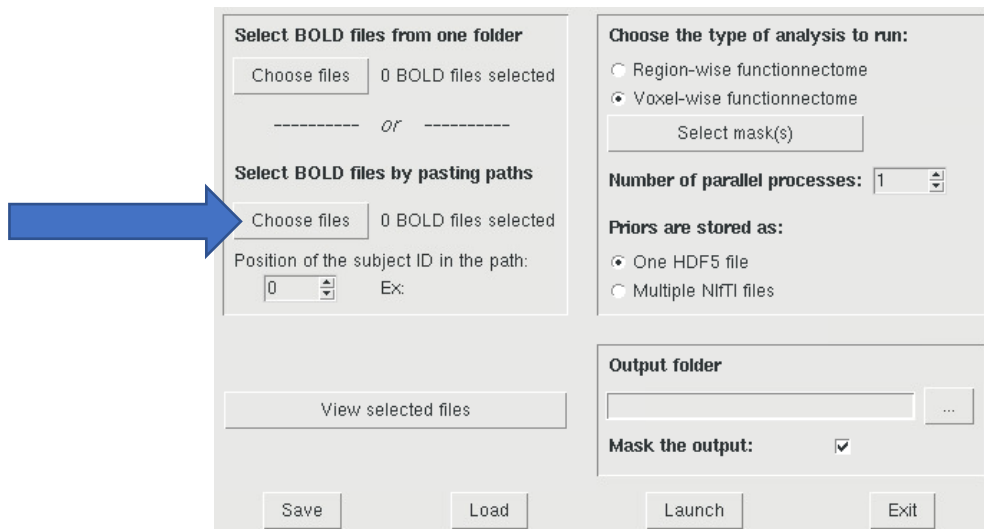
1. If they are all in the same folder, click on the “Choose files” button just under the



“Select BOLD files from one folder” label.

A generic interactive window will open to allow you to choose 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 a 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.

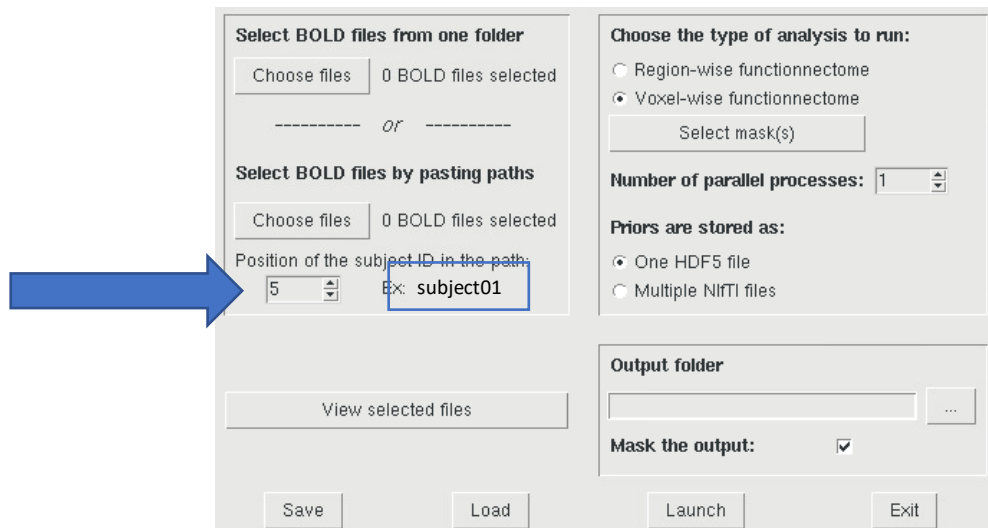
On Windows, use the “dir” command, similarly to the above-mentioned “ls”. For example: `“dir C:\myHome\project\subject????\fMRI_acquisition.nii.gz”`

Copy the displayed result and paste it into 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 the location in the paths where the program finds the IDs, 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

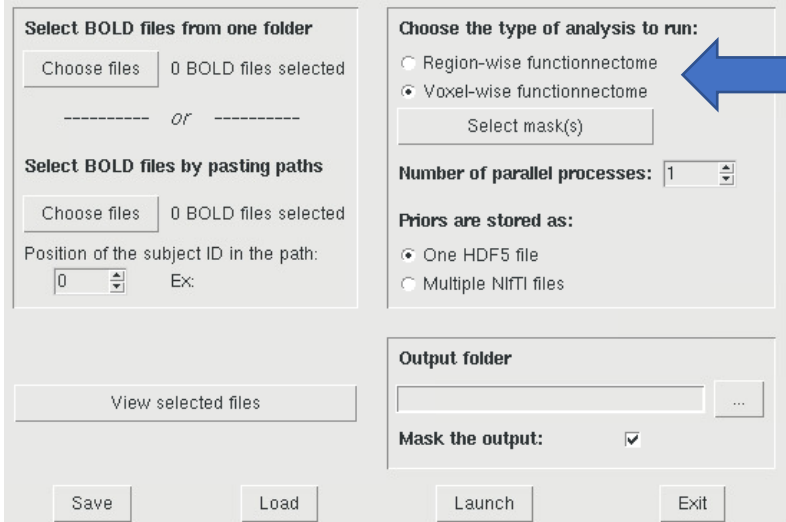
/myHome/myProject/session1/subject02/readyData.nii.gz

/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”).

Voxel-wise and region-wise analysis

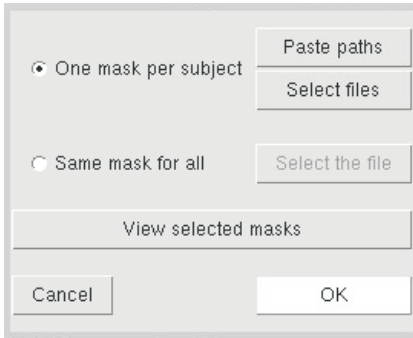


The screenshot shows a software interface for configuring an analysis. It is divided into several sections:

- Select BOLD files from one folder:** Includes a 'Choose files' button and a status '0 BOLD files selected'.
- Select BOLD files by pasting paths:** Includes a 'Choose files' button, a status '0 BOLD files selected', a dropdown for 'Position of the subject ID in the path:' set to '0', and an 'Ex:' label.
- Choose the type of analysis to run:** Contains two radio buttons: 'Region-wise functionnectome' and 'Voxel-wise functionnectome'. A blue arrow points to the 'Voxel-wise functionnectome' option. Below these is a 'Select mask(s)' button.
- Number of parallel processes:** A numeric input field set to '1'.
- Priors are stored as:** Contains two radio buttons: 'One HDF5 file' (selected) and 'Multiple NIFTI files'.
- Output folder:** A text input field with a browse button ('...').
- Mask the output:** A checkbox that is checked.
- Buttons at the bottom:** 'Save', 'Load', 'Launch', and 'Exit'.

Select whether you want to run a [voxel-wise](#) or a [region-wise](#) analysis. If you have enough computational power, we recommend using the voxel-wise analysis.

If you select the voxel-wise 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:



The 'Select mask(s)' dialog box has the following elements:

- Two radio buttons: 'One mask per subject' (selected) and 'Same mask for all'.
- Buttons on the right: 'Paste paths', 'Select files', and 'Select the file'.
- A 'View selected masks' button at the bottom.
- 'Cancel' and 'OK' buttons at the very bottom.

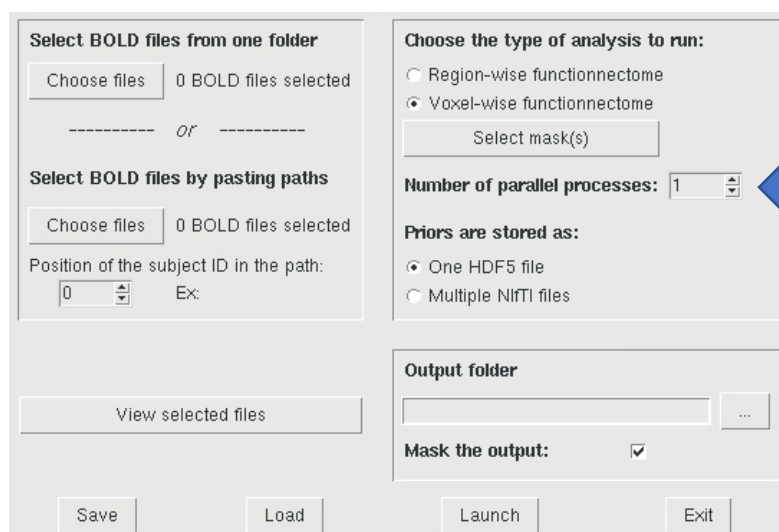
You can either attribute one mask per input file (“One mask per subject”), usually useful if the masks are created depending on the individual data; or select one unique mask that will be similarly applied to all inputs, for example, a mask delimiting the gray matter.

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

The paths of **the 4D inputs and 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 region-wise 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 configuration window. 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 show '0 BOLD files selected'. 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 a spinner set to '1', with a blue arrow pointing to it from the right. Below that, 'Priors are stored as:' 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 ('...') and a checked 'Mask the output:' checkbox. At the very bottom are 'Save', 'Load', 'Launch', and 'Exit' buttons.

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 limits 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.

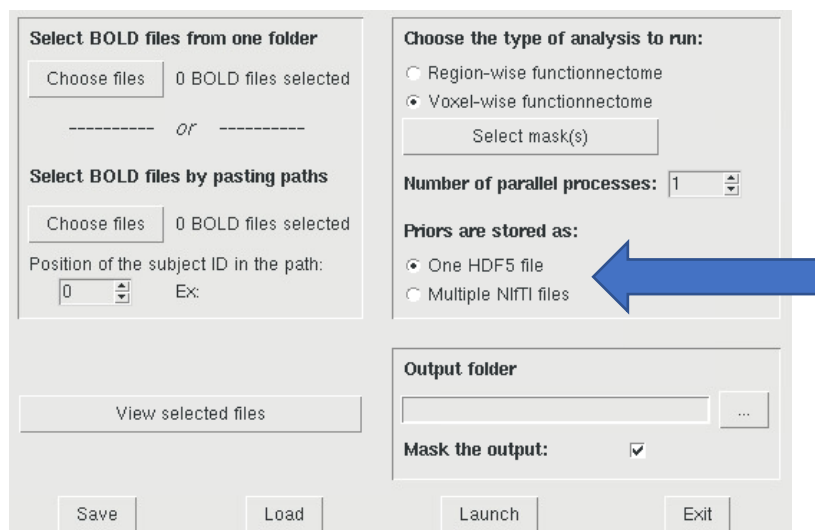
As an example, using 16 processes in parallel, the processing time of a typical fMRI scan with the voxel-wise analysis can be around 4 hours (but it depends a lot on the size of the input and how many voxels are kept with the mask).

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 dimensions of the images and to delimit the brain. The program can access them from two different types of storage:

- A collection of NifTI files, with a folder containing all the probability maps (voxel-wise or region-wise), 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 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.

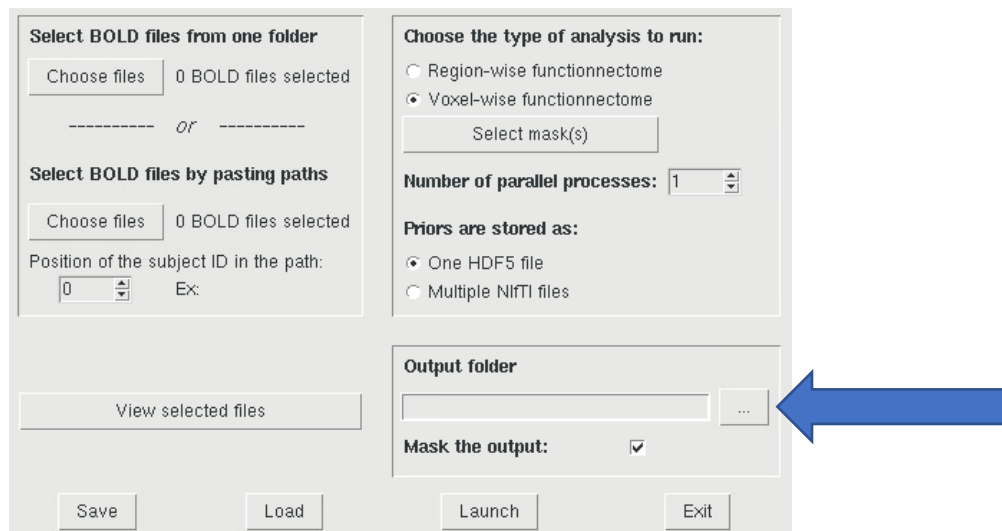
Paths to the priors are saved in a file and are purposely not readily available. Ideally, the paths to the priors should be set once (when downloading them) and left alone. It is possible to specify alternative paths directly in the settings file (see FAQ *“How do I specify different paths to alternate priors?”*). This way is to ease traceability since the settings file is saved with the results, so you’ll always know what priors you used (default or manually changed).



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 status '0 BOLD files selected'. Below the second section is a dropdown for 'Position of the subject ID in the path' set to '0' and an 'Ex:' label. A 'View selected files' button is at the bottom of this section. 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'. The 'Priors are stored as:' section has two radio buttons: 'One HDF5 file' (selected) and 'Multiple NIFTI files'. A large blue arrow points to the 'One HDF5 file' option. Below this is the 'Output folder' section with a text box and a browse button (...). The 'Mask the output:' checkbox is checked. At the bottom, there are four buttons: 'Save', 'Load', 'Launch', and 'Exit'.

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. It will serve as a “root” folder for the analysis, where the settings file (resulting from the parameters you entered in the GUI) and the additional sub-folders will be stored. Multiple analyses can use the same output folder, as long as the IDs of the input data are different.

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 voxel-wise 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 fonctionnectome 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 “.fctm” extension). It can be opened with any text editor. This settings file can be used to manually launch the Fonctionnectome from a terminal. All the parameters must be properly entered in the GUI before saving is possible.
- **Load:** Loads a settings file (“.fctm”) 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 Questions 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 we 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 to 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” starts 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 too often (for the sake of traceability), so 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 we plan to write a python script to do it more efficiently soon)

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 voxel-wise and region-wise priors must be given.

The script will detect all the NifTI files in the given folder, and import them into the HDF5 file. The voxel-wise probability maps files must all be named 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. Be aware that these are the “raw” coordinates, before applying the transformation matrix (quaternion) given in the NifTI header.

The region-wise 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 priors (in the HDF5 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 the 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. It hasn’t been tested on Windows yet, but was coded with portability in mind, so it might work. Proper tests on different OS are planned.

Will parallelizing the analysis use more RAM?

No. Or at least at least it shouldn’t. 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...

If you have any advice, as long as the discussion remains respectful, we are happy to hear it. Leave a comment on the GitHub page or send us an e-mail.