
Wei's menu for neuroimage

Release 0.0.1

wei

Jan 13, 2022

APPETIZER 1 LINUX_SYSTEM

1	Menu	3
2	“Is what each begins and perfects on himself”	267



This is documentation to give some basic instructions about how to run the volumetric, surface and fMRI analysis with different software packages. I hope this e-book can help you in the beginning of running image analysis

We have:

4 appetizers; Linux system, Neurodata, GitHub and basic statistics for neuroimage

5 main courses; FreeSurfer, FSL, AFNI, SPM and CONN

1 dessert, running analysis on SuperComputer

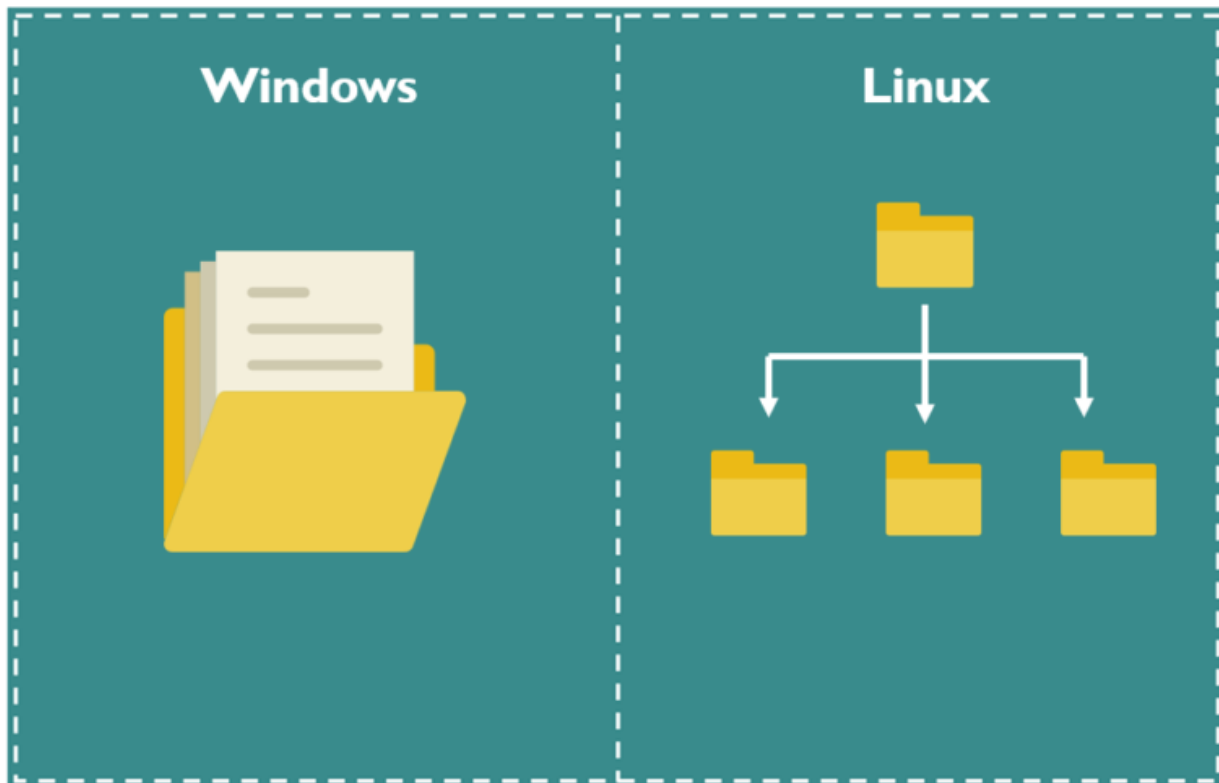
1.1 Linux system

1.1.1 Set up

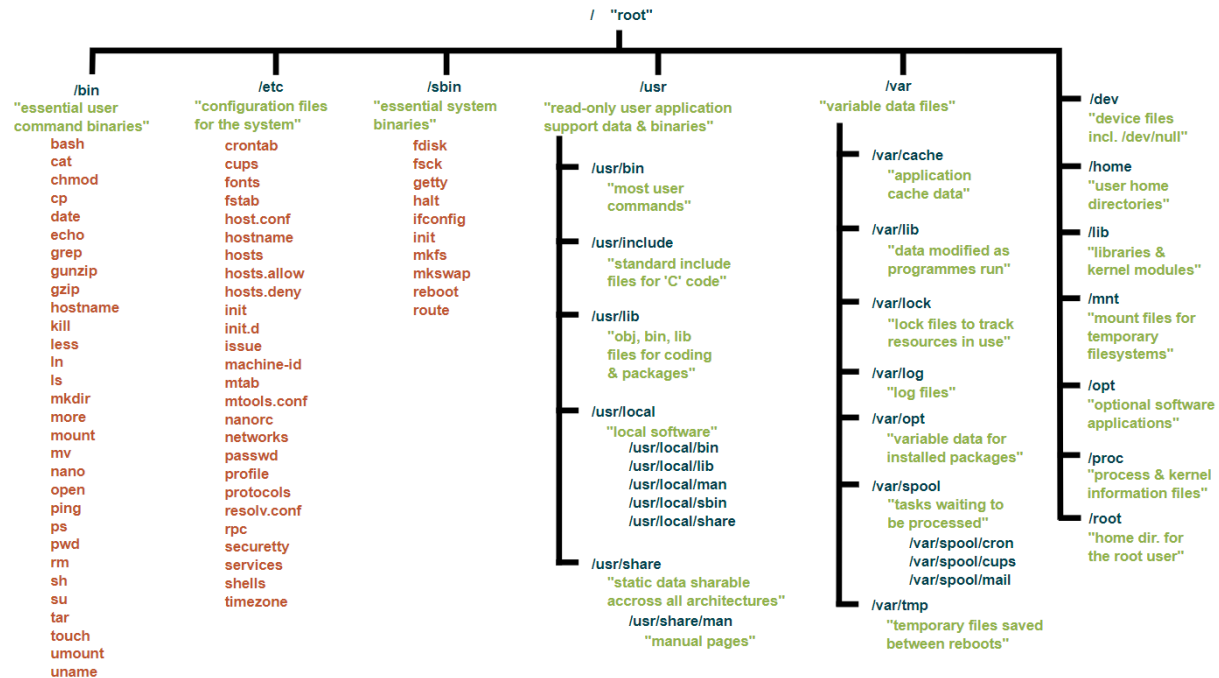
The first appetizer would be the Linux system. So, what is Linux system?

Linux is an open-source operating system created by Linus Torvald, it functions for programming user interface as well as Graphical user interface. More importantly, it is free. There are many free software and libraries developed for Linux system, especially, many neuroimage software packages.

Compared with the Windows file system, the file system of Linux is quite different.



So, it would be beneficial for us to become familiar with the file system of Linux.



If you use Macintosh developed by Apple such as macOS, no change needed. However, if you can't let Windows go, like me, you might need some extra works for analyzing the brain image. This documentation I will focus on the Linux system only.

Linux distribution

A Linux distribution is an operating system made from a software collection. It is the operating system that we use directly, which are available for a wide variety of systems ranging from embedded devices and personal computers to powerful supercomputers. There are two popular Linux distributions such as Ubuntu or Centos. If you are a Windows user, before you use the Linux distribution, make sure that you have set up for WSL because it is necessary to ensure that you can run a Linux distribution under Windows OS.

You can download [Ubuntu](#) and install it on Windows 10 with WSL. Or you can get latest CentOS from [here](#) and install it with WSL. These two Linux distributions are available from the Windows App store as well, you can purchase them and install with WSL.

WSL

For Windows users, we need use the **Windows Subsystem for Linux (WSL)** to run Linux. The Windows Subsystem for Linux (WSL) is a feature of Windows 10 that enables you to run native Linux command-line tools directly on Windows. Unlike virtual machine, WSL requires fewer resources (CPU, memory, and storage), it also allows you to run Linux command-line tools and apps alongside your Windows command-line, desktop and store apps, and to access your Windows files from within Linux. This enables you to use Windows apps and Linux command-line tools on the same set of files. To set up WSL, you need to use PowerShell to enable WSL. Then a Linux distribution needs to be installed in the WSL environment. Subsequently, additional software may need to be installed to run on the Windows side and/or in the Linux distribution running under WSL depending upon what you want to run under Linux. Now, enable the Windows subsystem for [Linux feature](#) for WSL.

Xming

Xming allows the Windows machines to display a graphical linux program which is basically running on a remote Linux server. It can enable to open graphics tools like freeview or FSL GUI. An X server needs to be installed on the Windows side (since the Windows OS is driving the graphics hardware).you can download [Xming](#) and follow the instruction to [set up](#). After installation, you can put xeyes or xclock to check the Xming.



1.1.2 Shell

The Shell is an interactive interface that allows users to execute commands and utilities in Linux. When you log in to the operating system, the standard shell would be displayed. There are many shells such as bash, Tcsh, Ksh, Zsh. bash is the default shell for many linux systems. We are going to use it for this documentation, you can check the current shell by typing:

```
echo $SHELL
```

```
[shao@WeiS ~]$ echo $SHELL
/bin/bash
```

if you see this, you are good to go, you can change the shell to bash by typing:

```
chsh -s /bin/bash
```

```
[shao@WeiS ~]$ chsh -s /bin/bash
Changing shell for shao.
Password:
```

1.1.3 Basic commands in Linux

Just like some spices are good all the time, some basic commands are necessary and useful for every time when you use them in Linux:

```
ls  #ls command lists files and directories within the file system, and shows detailed
    ↪ information about them

cd  #cd ("change directory") is used to change the current working directory in Linux

rm  #rm is a basic command to remove objects such as files, directories and symbolic
    ↪ links
```

(continues on next page)

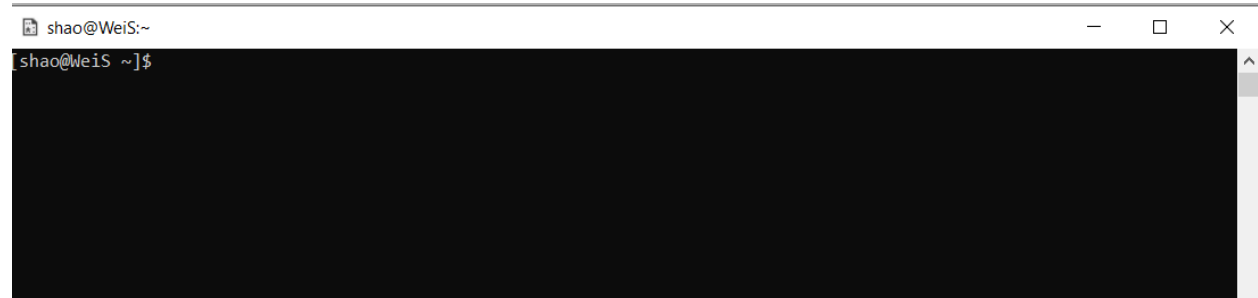
(continued from previous page)

```
mv #mv is a command that moves one or more files or directories from one place to
→ another

mkdir #mkdir (make directory) command allows you to create or make new directories

cp #cp is for copying files and directories
```

Open a terminal either by Centos or Ubuntu, you can see this.



```
shao@WeiS:~
[shao@WeiS ~]$
```

use ls, you will see all the files and sub-directories in the current directory

```
[shao@WeiS ~]$ ls
anaconda3          echo8.sh          matlab            neuroimage-book
Anaconda3-2020.11-Linux-x86_64.sh happy_birthday    MCRv84           Parallel_computing_recon_all
conf.py            license.txt       m1cn             README.md
docs              MatLab           m1cn_second     weishao_desserts
```

you also can find the files in sub-directories directly:

```
ls happy_birthday
```

```
[shao@WeiS ~]$ ls happy_birthday
docs
[shao@WeiS ~]$
```

cd to happy_birthday

Make cakes and party directory

```
[shao@WeiS happy_birthday]$ mkdir party cakes
[shao@WeiS happy_birthday]$ ls
cakes docs party
[shao@WeiS happy_birthday]$
```

Next, mv cakes to the party

```
[shao@WeiS happy_birthday]$ ls
cakes docs party
[shao@WeiS happy_birthday]$ mv cakes party
[shao@WeiS happy_birthday]$ cd party/
[shao@WeiS party]$ ls
cakes
[shao@WeiS party]$
```

Of course, cakes are always not enough for the party, so copy the pudding.


```
[shao@WeiS happy_birthday]$ ls
docs party Pudding
[shao@WeiS happy_birthday]$ cp -r Pudding party
[shao@WeiS happy_birthday]$ cd party
[shao@WeiS party]$ ls
cakes Pudding
[shao@WeiS party]$
```

Now, let's how many desserts we have

```
[shao@WeiS party]$ ls
biscuit cakes Pudding
[shao@WeiS party]$
```

Wait a minute, where is the biscuit come from, I don't want that in the party. just remove it

```
[shao@WeiS party]$ rm -r biscuit
[shao@WeiS party]$ ls
cakes Pudding
[shao@WeiS party]$
```

Now, we have all the desserts for a birthday party

These are the 6 basic commands you will use in the future whether you use you own laptop or server for the analysis you also can type:

```
man ls/cd/mv/rm/cp/mkdir
```

to look further details

For more information in [Linux commands](#)

1.1.4 Text editor

A text editor is a type of computer program that edits plain text such as nano, vi, vim or gedit. the default of Centos is nano and vim, you can choose any one of them. For example, open a terminal and put nano to open nano text editor.

1.2 For loop and sed

1.2.1 For loop

For loop probably is the one of the most useful commands to automated the processes with neuroimage data

First, you can use a for loop in your terminal by typing:

```
for i in 1 2 3; do echo $i; done
```

The for-loop has three sections, separated by semicolons.

1 The first section is the Declaration: it begins by assigning the first item after “in” to the variable “i”; in this case, it would assign the value “1” to “i”. The numbers after “in” are called the “List”

2 The next section is the Body, which runs the commands written after “do,” replacing the replacing the variable with whichever value is currently assigned to the variable, for the first loop, this will be the\$

3 The last section, called the End, contains only the word “done”, meaning to exit the loop after all of the items in the list have been run through the Body of the loop

You can add more commands to the Body section, if they are separated with a semicolon. For example, we could change the loop to:

```
for i in 1 2 3; do echo $i; echo "You just printed the number $i"; done
```

The for loops would be really useful when we run multiple subjects for the analysis, for example, you can use:

```
for subj in sub-01 sub-02 .... sub-99; do echo $subj; done
```

To add the subjects and prolong the “list”. More importantly, task would be quite simple if you can combine with other you can use for loop to help. for example:

```
ls . | grep ^sub- > subjList.txt
```

ls means we want to know something . indicates that what looking at the current directory, | means that whatever we did, keep continue, grep help us to collect a string of characters, ^sub means that we want to collect the file start with sub-, > to create a file, subjList.txt is the new txt file name.

In order to analyze multiple subjects, take recon-all as an example:

```
ls | grep ^sub- > subjlist.txt

for sub in `cat subjList.txt`; do

recon-all -all -i ${sub}_T1w.nii.gz -s ${sub}
```

This will create a txt file which contains all the subject information and run the recon-all command for each subject one by one.

1.2.2 sed command

The commands you have learned so far will allow you to create flexible scripts that can be adapted to many different scenarios. There is one more command you will learn to round out your toolkit of Unix commands: The sed command.

Sed is an abbreviation for “stream editor”, in that the input to sed is a stream of text - the same concept as the input and output streams that were discussed in a previous chapter. Our goal is to take a stream of input text and replace one string with another. Sed’s advantage over doing a similar procedure with for-loops is that sed can edit a file and only change certain words while overwriting the file and leaving the rest of the text intact.

Sed introduction

To see how sed works, create a text file called diner.sh that contains the following line:

```
#!/bin/bash

echo "Hello, we are are going to cook Masala Chicken, Again, we will eat Masala Chicken,
↳as dinner."
```

#!/bin/bah tells your laptop that this is a bash script.

This is simply a text file that runs one line of code. If you wanted to swap the name Masala Chicken with Tender beef, you need to type the following:

```
sed "s|Masala Chicken|Tender beef|g" diner.sh
```

There are 3 basic sections:

- 1 Declaring the sed command
- 2 A pattern to match and replace with another pattern, enclosed in quotes
- 3 The file to be read into sed (in this case, diner.sh)

The first part of this section is an “s”, which means to “swap” the following pair of strings. The first of the pair (Masala Chicken) is what is being searched for in the text file, and the second of the pair (Tender beef) is what it will be replaced with. The “g” stands for “global”, which means to replace every instance of the first word with the second word.

If you run this command by `bash diner.sh`, you should see the following output:

```
#!/bin/bash
echo "Hello, we are are going to cook Tender beef, Again, we will eat Tender beef as_
↪dinner."
```

If you wanted to redirect this output into a new text file, you would use this code:

```
sed "s|Masala Chicken|Tender beef|g" diner.sh > diner_Beef.sh
```

Of course, you can call the output file whatever you like.

Editing Files In Place

If you want to edit the file and overwrite it instead of redirecting the output into a new file, you can use the `-i` options:

```
sed -i "s|Masala Chicken|Tender beef|g" diner.sh
```

The `-i` option stands for “in-place”, and signifies that the text file should be overwritten after the words have been swapped.

Using sed with for-loops

Actually, sed can be combined with for-loops and conditional statements to write more sophisticated code that can make your life easier. For example, let’s say that we want to create several copies of a template file, and only change one word of it over a list of names. Let’s start by creating a file called `food.sh` which contains the following:

```
#!/bin/bash
echo "what we eat is FOOD."
```

In here, `FOOD` is a placeholder. Now we can use a for-loop to create several copies of this file, replacing `FOOD` with whichever you like in the loop:

```
for fo in Masala_Chicken Tender_beef Pancake; do
  sed -i "s|FOOD|${fo}|g" food.sh > ${name}_food.sh
done
```

You can write a bash script to run this command as well.

Although it seems irrelevant to learn for loop and sed command, you will find these two have become really useful when you want to automate the process of neuroimage, which we will discuss later.

1.3 Useful tricks

There are some tricks that can make your Linux life easier.

let's take `bash_profile` as an example, you can edit this profile by type:

```
nano ~/.bash_profile
```

`bash_profile` is a configuration file for bash shell, whenever you invoke bash with a login, it will search for and load `~/.bash_profile` and all of the code contained within. In other words, `bash_profile` is a script that is executed each time you start a terminal.

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

export FREESURFER_HOME=/usr/local/freesurfer/7.1.1-1
source $FREESURFER_HOME/SetUpFreeSurfer.sh

alisa 9="cd /home/shao/neuroimage-book/"
```

For example, In here, I have set Freesurfer by two commands:

```
export FREESURFER_HOME=/usr/local/freesurfer/7.1.1-1 source $FREESURFER_HOME/SetUpFreeSurfer.sh
```

By using `export` I have pointed out the freesurfer home directory where I install the freesurfer. `source` to activate freesurfer whenever I log in terminal.

`alisa 9` helps me set a hot key that I press 9 to run the command `cd /home/shao/neuroimage-book`

It is not hard to find files in the current directory by `ls`, However, if I want to find/delete files located at subdirectory, I have use different command such as `ls -d $PWD/sub-??/run*` as well as `rm -r $PWD/sub-??/run*`.

The `-d` option means to only list directories, and `$PWD` expands to an absolute path pointing to the current working directory. Within the current directory, any directory starting with `sub-` and ending with two digits (represented by the `??`) is added to the path. Finally, within each subdirectory, any directory beginning with the string `run` will be appended to the path name (e.g., `run1.feet` and `run2.feet`). `rm` is the remove command, `-r` means recursive, it tells the computer to remove directories and their contents recursively

Also find `find . -name "file_name"`. can help you to find the `file_name` related information from the subdirectory

1.4 Jargon for Neuroimage data

1.4.1 Basic terminology

Before we start our journey, let's review some basic terms, this could cost you extra time, but it worth it.

Voxels

Each MRI image contains information about a three-dimensional (3D) volume of space. An MRI image is composed of a number of voxels; the voxel size is the spatial resolution of the image. A voxel is a 3D unit of the image with a single value, just as for digital photographs a pixel is a 2D unit of the image with a single value. The words image and volume (and sometimes scan) are often used interchangeably to mean “a single 3D representation of brain data”. The size of the voxel gives some indication as to the spatial resolution of the data, with smaller voxels giving a higher spatial resolution. A typical voxel size for a structural MRI is 1 mm x 1mm x 1mm; for fMRI, 3 mm x 3 mm x 3 mm. However, these can vary substantially from study to study.

4D files

It is also possible to have 4D image files—in other words, a series of 3D image files. Generally this is a timeseries; so, for a single continuous scanning run, the same data could be represented by a large number of 3D image files, or a single 4D file (containing the same number of 3D functional volumes). However, 4D files could also be used to store a single image per subject, for example, when doing group analyses. FSL makes frequent use of 4D images. SPM supports 4D images under certain circumstances, but it is possible to conduct all steps using multiple 3D images.

Isotropic

In the MRI sequences used for most fMRI studies, each volume of the brain is acquired in a series of slices. Thus, each slice is acquired at a different point in time within the TR. Knowing the order in which the slices are acquired (ascending, descending, interleaved) can be important for certain preprocessing steps. Within a slice, the voxel dimensions are almost always equal in size. If voxels are equal in size in all three dimensions (e.g., 3 mm x 3 mm x 3 mm), they are isotropic.

TA and TR

The time it takes to acquire an entire volume is the TA (acquisition time). The time between repeated volumes (i.e., between collecting a slice in one volume, and that same slice in the next) is the TR (repetition time). If one is collecting data as quickly as possible (as is usually the case), there will be no pause between volumes that are collected, and thus the TA and TR are equal. Because of this many people use the terms interchangeably. However, it is possible to have a TR that is longer than the TA, which would result in a pause in data collection. This is sometimes done in auditory studies to allow stimuli to be presented in quiet (in the absence of echoplanar scanner noise).

Run or Session

An fMRI experiment may last for an hour or more, but very rarely will scanning be continuous for this hour. Typically there will be a 5–20 minute sequence of scanning, and then a break to ensure the subject is comfortable, before resuming scanning. These breaks are important to note when performing analysis, as many stages of analysis make assumptions about how related one volume is to the next. A single continuous series of scans is typically called a “run”, “block”, or “session”. FSL refers to these as “run”, SPM refers to these as “sessions”, think of a session as “any continuous period of scanning”; it doesn’t matter whether these are separated by 30 seconds or a week.

First level and second level

you might hear “1st level” or “2nd level” analyses a lot. These refer generally to variations on a summary statistics approach in which data for each subject are analyzed independently, and then the results from these analysis (i.e., the parameter estimates) are entered in a group analysis. In fMRI, a 1st level analysis generally refers to analysis of a single subject’s timeseries. Results from this analysis would tell you whether, for this subject, a region showed significant activation for a given paradigm. A 2nd level analysis is equivalent to a group study, in which linear contrasts of parameter estimates are fit with a statistical model. Results from a 2nd level analysis would tell you whether a pattern of activation is significant across a group (and thus likely to generalize across a population).

Files format

The Brain Imaging Data Structure (BIDS) is a standard for organizing, annotating, and describing data collected during neuroimaging experiments. Neuroimaging Informatics Technology Initiative (NIFTI) is file format of brain images. It is transferred from DICOM format that comes from Siemens scanners. Nearly all modern software packages use the Nifti file format. Nifti files can be a single file (with an `.nii` or `.nii.gz` extension), or image/header pairs (`file1.img`, `file1.hdr`). In both cases, there are two sets of information about each image:

The header information, which contains information necessary to interpret the image. This includes things like how the data are encoded, and how the voxel space of the image translates into the physical space of the world (voxel-to-world mapping).

The image data themselves; for each voxel, a single value.

“Talairach” space

Talairach space is a three-dimensional cartesian coordinate system aligned over the brain as proposed by Talairach & Tournoux. It can also refer to this general system of alignment, within this system, the `[0,0,0]` point ($X=0$, $Y=0$, $Z=0$) is the anterior commissure:

Negative X values indicate the left hemisphere, positive X values the right.
Negative Y values towards the rear of the brain (relative to the anterior commissure),
→ positive toward the front.
Negative Z values towards the bottom of the brain (relative to the anterior commissure),
→ positive toward the top.

Anatomical descriptions

Towards the front of the brain (towards the eyeballs): **anterior**

Towards the back of the brain (opposite the eyeballs): **posterior**

Towards the top of the brain: **superior**

Slice orientations

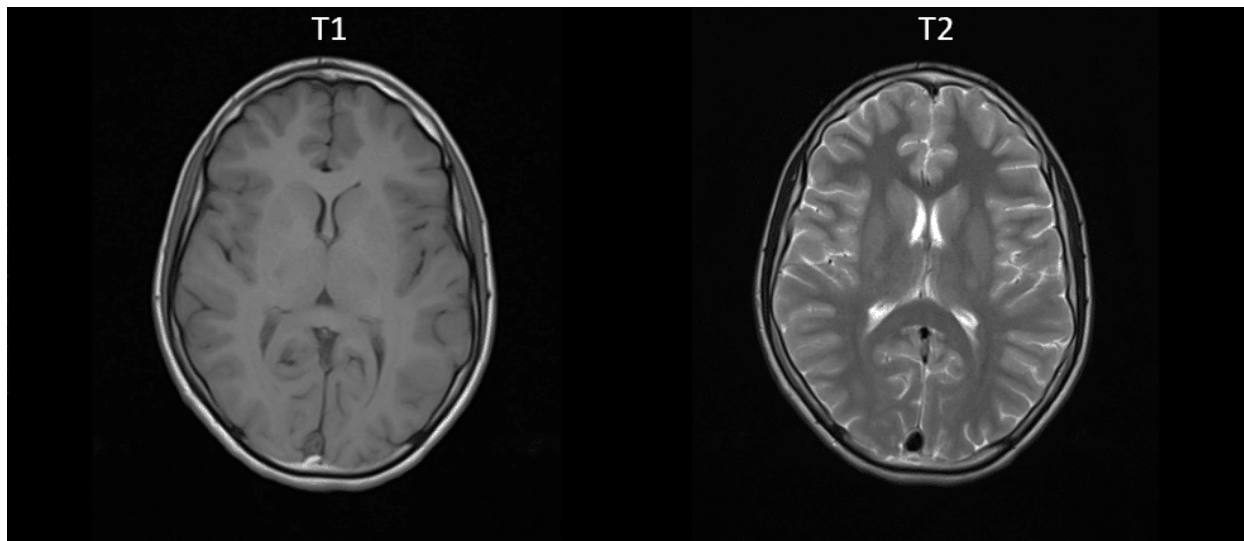
Axial are parallel to the top of the head and the floor, if you are standing straight.

Sagittal slices are parallel to the interhemispheric commissure (dividing the brain in two).

Coronal slices are parallel to the face, perpendicular to the floor.

Different structural MRI

Different types of MRI sequences provide sensitivity to various aspects of neuroanatomy. The two basic types of MRI images are T1-weighted and T2-weighted images, often referred to as T1 and T2 images. T1 images are probably the most common. T1 will have grey matter being darker than white matter. T2, in contrast, will have white matter being darker than grey matter.



FLAIR images

Fluid-attenuated inversion recovery (FLAIR) is an MRI sequence with an inversion recovery set to null fluids. It can be used in brain imaging to suppress cerebrospinal fluid (CSF) effects on the image, so as to bring out the periventricular hyperintense lesions, such as multiple sclerosis (MS) plaques. FLAIR can be used with both three-dimensional imaging (3D FLAIR) or two dimensional imaging.

Quantitative MRI(qMRI)

Quantitative MRI (qMRI) is a collection of methods aiming at generating parametric maps that can characterize underlying tissue properties. Unlike those of conventional MR images (such as T1w or T2w), intensity values of quantitative maps are not represented in an arbitrary range. Instead, these maps are represented either in absolute physical units (seconds for T1map), or within an application dependent range of arbitrary units (myelin water fraction MWFmap in brain)

1.5 OpenNeuro

As we know that a good chef always knows where he/she can find the best ingredients, so do we.

OpenNeuro (OpenfMRI) is an open-science neuroinformatics online database storing datasets from human brain imaging research studies. Neuroimaging researchers around the world can upload their data to this site so third-party researchers can download the data and analyze it.



OpenNEURO

A free and open platform for sharing MRI,
MEG, EEG, iEEG, and ECoG data



[Browse All Public Datasets](#)

500

Public Datasets

15883

Participants

Go to [OpenNeuro](#), click Browse All Public Datasets or search the dataset by name, pick one you are most interested in.

PUBLIC DATASETS									
PUBLIC DATASET RESULTS									
<div> <div>Sort By: Created</div> <div>Name</div> <div>Stars</div> <div>Downloads</div> <div>Subscriptions</div> </div>									
<div> <div>Flanker task (event-related)</div> <div> <div>5120</div> <div>70921</div> <div>3</div> <div>9</div> </div> </div>									
<div> <div>UPLOADED BY Chris Gorgolewski ON 2016-10-14 - OVER 4 YEARS AGO</div> <div> <div>FILES: 1664</div> <div>SIZE: 1.75GB</div> <div>SUBJECTS: 26</div> <div>SESSION: 1</div> <div>AVAILABLE TASKS: Flanker</div> <div>AVAILABLE MODALITIES: T1w, bold</div> </div> </div>									
<div> <div>Classification learning</div> <div> <div>2077</div> <div>107800</div> <div>2</div> <div>3</div> </div> </div>									
<div> <div>UPLOADED BY Chris Gorgolewski ON 2016-10-12 - OVER 4 YEARS AGO</div> <div> <div>FILES: 2789</div> <div>SIZE: 5.4GB</div> <div>SUBJECTS: 17</div> <div>SESSION: 1</div> <div>AVAILABLE TASKS: deterministic classification, mixed event-related probe, probabilistic classification</div> <div>AVAILABLE MODALITIES: /participants, T1w, inplaneT2, bold, events</div> </div> </div>									
<div> <div>UCLA Consortium for Neuropsychiatric Phenomics LA5c Study</div> <div> <div>1886</div> <div>2039717</div> <div>21</div> <div>32</div> </div> </div>									
<div> <div>UPLOADED BY Franklin Feingold ON 2018-03-19 - ALMOST 3 YEARS AGO</div> <div> <div>FILES: 9010</div> <div>SIZE: 79.28GB</div> <div>SUBJECTS: 272</div> <div>SESSION: 1</div> <div>AVAILABLE TASKS: bart, rest, stopsignal, bht, scap, taskswitch, pamenc, pamret</div> <div>AVAILABLE MODALITIES: T1w, beh, events, dwi, bold, physio</div> </div> </div>									
<div> <div>Forrest Gump</div> <div> <div>1845</div> <div>885187</div> <div>15</div> <div>22</div> </div> </div>									
<div> <div>UPLOADED BY Joseph Wexler ON 2018-08-12 - OVER 2 YEARS AGO</div> <div> <div>FILES: 45879</div> <div>SIZE: 18.72GB</div> <div>SUBJECTS: 37</div> <div>SESSIONS: 8</div> <div>AVAILABLE TASKS: Forrest Gump, objectcategories, movielocalizer, retmapccw, retmapcon, retmapexp, movie, retmapclw, coverage, orientation, auditory perception</div> <div>AVAILABLE MODALITIES: T1w, bold, physio</div> </div> </div>									

You can see a brief description about the dataset and take a look.

Flanker task (event-related)

uploaded by Chris Gorgolewski on 2016-10-14 - over 4 years ago

last modified on 2018-07-14 - over 2 years ago

authored by Kelly AMC, Uddin LQ, Biswal BB, Castellanos FX, Milham MP

4519 29629

Download

Analyze on brainlife.io

OpenNeuro Accession Number: ds000102

Files: 1664, Size: 1.75GB, Subjects: 26, Session: 1

Available Tasks: Flanker

Available Modalities: T1w, bold

README

This dataset was obtained from the OpenfMRI project (<http://www.openfmri.org>).

Accession #: ds102

Description: Flanker task (event-related)

The "NYU Slow Flanker" dataset comprises data collected from 26 healthy adults (age and sex included in Slow_Flanker_age_sex.txt) while they performed a slow event-related Eriksen Flanker task.

BIDS Validation

Valid 1 WARNING

Dataset File Tree

- Flanker task (event-related)
 - CHANGES
 - DOWNLOAD
 - VIEW
 - dataset_description.json
 - DOWNLOAD
 - VIEW
 - participants.tsv
 - DOWNLOAD
 - VIEW
 - README
 - DOWNLOAD
 - VIEW
 - T1w.json
 - DOWNLOAD
 - VIEW
 - task-flanker_bold.json
 - DOWNLOAD
 - VIEW

Download the data.

How to Download

Download with your browser

This method is convenient and allows you to select a local directory to download the dataset to.

Steps

1. Select a local directory to save the dataset and grant permission to OpenNeuro to read and write into this directory.
2. Download will run in the background, please leave the site open while downloading.
3. A notification will appear when complete.

DOWNLOAD

Download with Node.js

Using `openneuro-cli` you can download this dataset from the command line using Node.js. This method is good for larger datasets or unstable connections, but has known issues on Windows.

```
openneuro download --snapshot 00001 ds000102 ds000102-download/
```

This will download to `ds000102-download/` in the current directory. If your download is interrupted and you need to retry, rerun the command to resume the download.

Download from S3

The most recently published snapshot can be downloaded from S3. This method is best for larger datasets or unstable connections. This example uses AWS CLI.

```
aws s3 sync --no-sign-request s3://openneuro.org/ds000102 ds000102-download/
```

To download unpublished datasets or older snapshots, see advanced methods below.

Download with DataLad

Public datasets can be downloaded with DataLad or Git Annex from GitHub.

```
datalad install https://github.com/OpenNeuroDatasets/ds000102.git
```

Check out [getting started with DataLad](#) for more on how to use this download method.

or go to Dataset File Tree and view the image.

– README

[DOWNLOAD](#) [VIEW](#)

– T1w.json

[DOWNLOAD](#) [VIEW](#)

– task-flanker_bold.json

[DOWNLOAD](#) [VIEW](#)

– derivatives

– sub-O1

– anat

– func

– sub-O1_task-flanker_run-1_bold.nii.gz

[DOWNLOAD](#) [VIEW](#)

– sub-O1_task-flanker_run-1_events.tsv

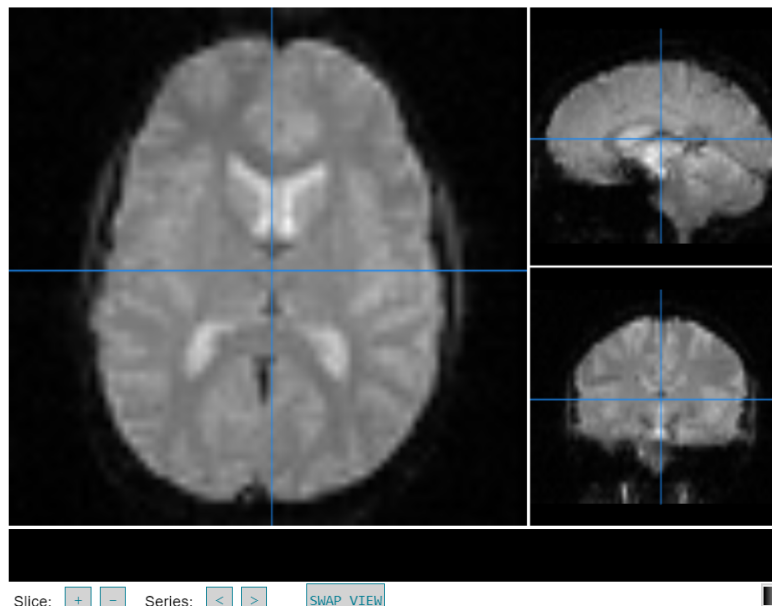
[DOWNLOAD](#) [VIEW](#)

– sub-O1_task-flanker_run-2_bold.nii.gz

[DOWNLOAD](#) [VIEW](#)

– sub-O1_task-flanker_run-2_events.tsv

[DOWNLOAD](#) [VIEW](#)



You can also download the file by `git clone`, which we will discuss later.

1.6 GitHub introduction

What are the most 3 important steps to produce a cuisine? the answer is collaboration, collaboration and collaboration!

So, What is GitHub?

In Short, GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on a project from anywhere.

You can go to [here](#) GitHub create an account and sign in

Here we will learn the 5 basic ideas for GitHub:



- 1.repositories
- 2.branches
- 3.commits
- 4.pull Requests
- 5.merge branches

1.6.1 Step 1 Create a repository

A repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs

After registration, you can sign in from Github, and go to the left-right corner where you can see a tab named **Repositories**. Click **new** to create a new repository and name the repository **Hello-World** you can write a short description about this repository, choose **Public** repository and initialize this repository with a **README** file. Lastly, click **Create repository**


Owner * Repository name *


 WeiShaoD ▾ / Hello-World 

Great repository names are short and memorable. Need inspiration? How about [bug-free-invention?](#)

Description (optional)

This is open Hello-World

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.


☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

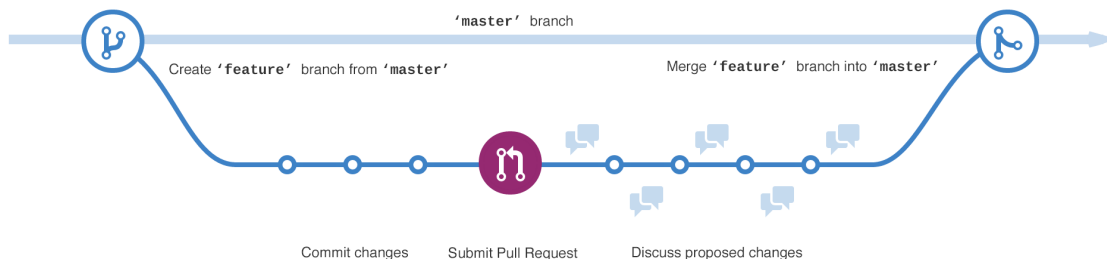
☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

This will set  `main` as the default branch. Change the default name in your [settings](#).

1.6.2 Step 2 Create a Branch

Branching is the way to work on different versions of a repository at one time, By default, your repository has one branch named `main` which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to `main`

When you create a branch off the `main` branch, you're making a copy, or snapshot, of `main` as it was at that point in time. If someone else made changes to the `main` branch while you were working on your branch, you could pull in those updates.

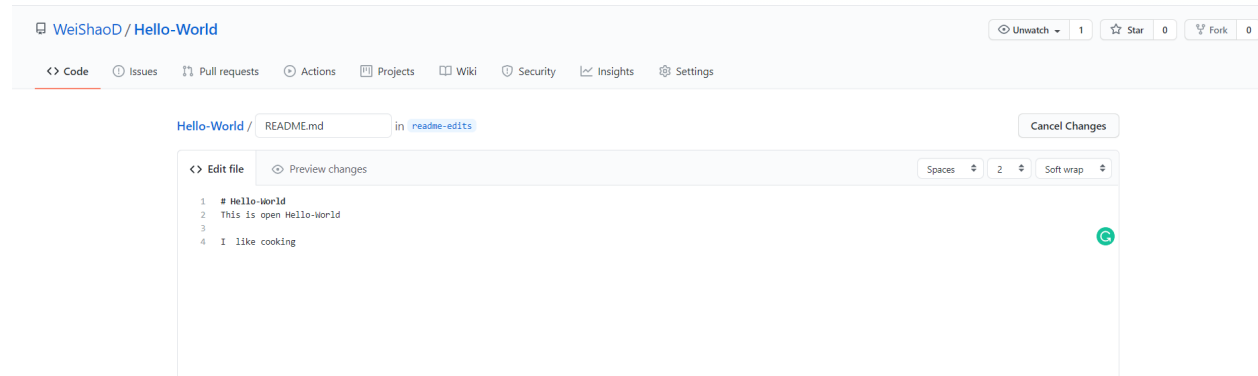


There is a `main` branch and another branch called `feature`. a journey that `feature` takes before it's merged into `main` has shown in the figure below.

let's create a new branch! Go to our repository `hello-world`, choose branch: `main` tab, type a branch name such as `readme-edits`, and click the `Create branch` textbox. Now you have two branches, `main` and `readme-edits`

1.6.3 Step 3 Make and commit changes

Choose the new branch we just created, click the pencil icon in the upper right corner of the file view to edit the file **README.md**. You're on the code view for your `readme-edits` branch, which is a copy of `main`. Let's make some edits in this file. You can type **I like cooking** in this text file, and write a commit message in the textbox `Commit changes` to describe your changes. Then, click `Commit changes` button.



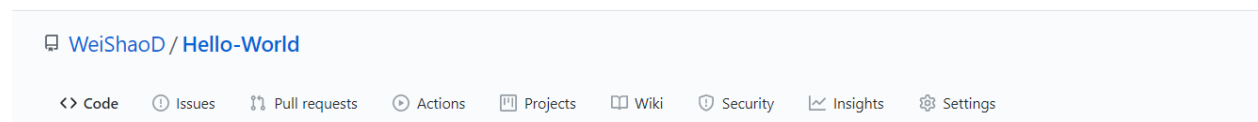
These changes will be made to just the `README` file on your `readme-edits` branch, so this branch contains content that's different from `main`.

1.6.4 Step 4 Open a Pull Request

So far, we have successfully made some changes in a branch off of the `main` branch, we need to continue with a pull request.

Pull Requests are the heart of collaboration on GitHub. When you open a pull request, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red by GitHub.

After the commit of changes in the `README` file, go back to the `readme-edits` branch.



You will see a new textbox. By click the `Compare & pull request`, jump to the **Open a pull request** page. Leave a comment in the main textbox, scroll down to look over your changes in the diffs on the `Compare` page, make sure they're what you want to submit. Click the green button `Create pull request`

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for comparing changes between the 'main' branch and the 'readme-edits' branch. At the top, there's a header with the branch names and a message: 'Can't automatically merge. Don't worry, you can still create the pull request.' Below this is a yellow bar with the text 'Discuss and review the changes in this comparison with others. [Learn about pull requests](#)' and a green 'Create pull request' button. A summary bar shows '1 commit', '1 file changed', '0 comments', and '1 contributor'. The commit history shows a commit on Feb 26, 2021, titled 'Update README.md' by user 'WeiShaoD'. Below the commit, it says 'Showing 1 changed file with 2 additions and 0 deletions.' The file 'README.md' is shown with a diff view. The diff shows two lines added: '# Hello-World' and 'This is open Hello-World'. The file is marked as 'Verified' with a green checkmark and the commit hash '623619e'.

1.6.5 Step 5 Merge your Pull Request

In this final step, let's bring all the changes together – merging branches. From the Comparing changes page, you can see that direction of the merge, we are going to merge `readme-edits` branch into the `main` branch. Click the `View pull request`, `Merge pull request` and `Confirm merge`

The screenshot shows the GitHub 'Merge pull request' page. At the top, there's a comment from 'WeiShaoD' saying 'Job is done'. Below the comment, it says 'Merge branch 'main' into readme-edits' with a green checkmark and the commit hash 'a352c3b'. A message below the merge says 'Add more commits by pushing to the `readme-edits` branch on `WeiShaoD/Hello-World`.' Below this is a green box with a 'Continuous integration has not been set up' message, followed by a green checkmark and the text 'This branch has no conflicts with the base branch' and 'Merging can be performed automatically.' At the bottom, there's a green 'Merge pull request' button and a link to 'open this in GitHub Desktop or view command line instructions'.

Now, go ahead and delete the `readme-edits` branch since its changes have been incorporated.

1.7 More GitHub commands

After the basic content, let's have more fun by knowing how to download the content from GitHub repository/ or upload an existing local file to a GitHub repository with the 4 command lines:

```
git clone
git add
git commit
git push
```

1.7.1 GitHub configuration

First, sign in with a GitHub account and create a new repository named dinner_menu

The screenshot shows the GitHub 'Create new repository' form. At the top, there are two fields: 'Owner' with a dropdown menu showing 'WeiShaoD' and a green checkmark, and 'Repository name' with the text 'dinner_menu' and a green checkmark. Below these fields is a text prompt: 'Great repository names are short and memorable. Need inspiration? How about [urban-octo-couscous?](#)'. Underneath is a 'Description (optional)' text box. Further down, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a description: 'You choose who can see and commit to this repository.' Below these options is a section titled 'Initialize this repository with:' followed by the instruction 'Skip this step if you're importing an existing repository.' There are three checkboxes: 'Add a README file' (checked), 'Add .gitignore', and 'Choose a license'. Each checkbox has a brief description and a 'Learn more' link. At the bottom, it says 'This will set `main` as the default branch. Change the default name in your [settings](#).'

Now, open a terminal and go to the directory that you want to work on from there.

set up the GitHub configuration:

```
git config --global user.name "your name"
git config --global user.email "your GitHub_Emailaddress@.com"
```

Then, use `git clone` to get the remote directory content from your GitHub repository

1.7.2 git clone

`git clone` is a Git command line that targets an existing repository and creates a clone in your directory. It is primarily used to point to an existing repo and make a clone or copy of that repository in a new directory.

Type `git clone git@github.com:WeiShaoD/dinner_menu.git` from your work directory.

```
[shao@WeiS ~]$ git clone git@github.com:WeiShaoD/dinner_menu.git
Cloning into 'dinner_menu'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
[shao@WeiS ~]$
```

1.7.3 Git add/commit/push

The `git add` adds a change in the working directory. It tells Git that you want to include updates to a particular file in the next commit. However, `git add` doesn't really affect the repository until you run `git commit`.

The `git commit` captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as "safe" versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of `git commit`, the `git add` command is used to promote or 'stage' changes to the project that will be stored in a commit.

The `git push` uploads local repository content to a remote repository. Pushing exports commits to remote branches therefore Pushing has the potential to overwrite changes, caution should be taken when pushing.

cd to the `dinner_menu`, add a new `menu.txt` file by typing `nano menu.txt`, use `git add` and `git commit -m "menu for dinner"`

```
[shao@WeiS dinner_menu]$ git add menu.txt
[shao@WeiS dinner_menu]$ git commit -m "menu for dinner"
[main e670bf2] menu for dinner
1 file changed, 3 insertions(+)
create mode 100644 menu.txt
```

Use `git push -u origin main` to synchronize your local and GitHub remote repository.

```
[shao@WeiS dinner_menu]$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 324.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:WeiShaoD/dinner_menu.git
ff9c16b..e670bf2 main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Now you can see the new menu from GitHub webpage.

main
 1 branch
 0 tags
 Go to file
Add file
Code

WeiShaoD menu for dinner
 e670bf2 3 minutes ago
2 commits

README.md	Initial commit	6 minutes ago
menu.txt	menu for dinner	3 minutes ago

README.md

dinner_menu

It's time for you to practice these commands by editing your own dinner menu.

```
[shao@WeiS dinner_menu]$ nano menu.txt
[shao@WeiS dinner_menu]$ git add menu.txt
[shao@WeiS dinner_menu]$ git commit -m "add the first course"
[main 0612faf] add the first course
1 file changed, 2 insertions(+)
[shao@WeiS dinner_menu]$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 348 bytes | 174.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:WeiShaoD/dinner_menu.git
e670bf2..0612faf  main -> main
```

More details form [here](#)

1.8 Statistics for neuroimage

1.8.1 Power statistics

Power analysis is directly related to tests of hypotheses. While we conducting tests based on different hypotheses, there are two types of errors researchers can make: Type I error and Type II error. Statistical power mainly deals with Type II errors. In general, it is worth to note that the higher the sample size, the easier it will be to obtain the 0.05 level of significance. However, if the sample size is too small, the investigator may make a Type II error due to insufficient power.

Data collection is usually preceded by a power analysis. The principal goal of power analysis is to assist researchers in determining the minimum sample size necessary to detect the effect of a given test at the desired level of significance. The reason for doing power analysis is that we would prefer a lower sample size because larger samples are frequently more expensive. The significance testing is also improved with smaller samples.

1.8.2 Multivariate data

Multivariate data contains, at each sample point, multiple scalar values that represent different simulated or measured quantities. Multivariate data can come from numerical simulations that calculate a list of quantities at each time step, or from medical scanning modalities such as MRI, which can measure a variety of tissue characteristics, or from a combination of different scanning modalities, such as MRI, CT, and PET. Multidimensional transfer functions are an obvious choice for volume visualization of multivariate data, since we can assign different data values to the different axes of the transfer function. It is often the case that a feature of interest in these datasets cannot be properly classified using any single variable by itself. In addition, we can compute a kind of first derivative in the multivariate data in order to create more information about local structure. As with scalar data, the use of a first derivative measure as one axis of the multi-dimensional transfer function can increase the specificity with which we can isolate and visualize different features in the data.

One example of data that benefits from multi-dimensional transfer functions is volumetric color data. A number of volumetric color datasets are available, such as the Visible Human Project's RGB data. The process of acquiring color data by cryosection is becoming common for the investigation of anatomy and histology. In these datasets, the differences in materials are expressed by their unique spectral signatures. A multidimensional transfer function is a natural choice for visualizing this type of data. Opacity can be assigned to different positions in the 3D RGB color space.

1.8.3 mean squared error

let's compute the mean squared error for a set of inputs x , measurements y , and slope estimate \hat{y} . Here, x and y are vectors of data points. We will then compute and print the mean squared error for 3 different choices of θ .

As a reminder, the equation for computing the estimated y for a single data point is:

$$\hat{y}_i = x_i$$

and for mean squared error is:

$$\min \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

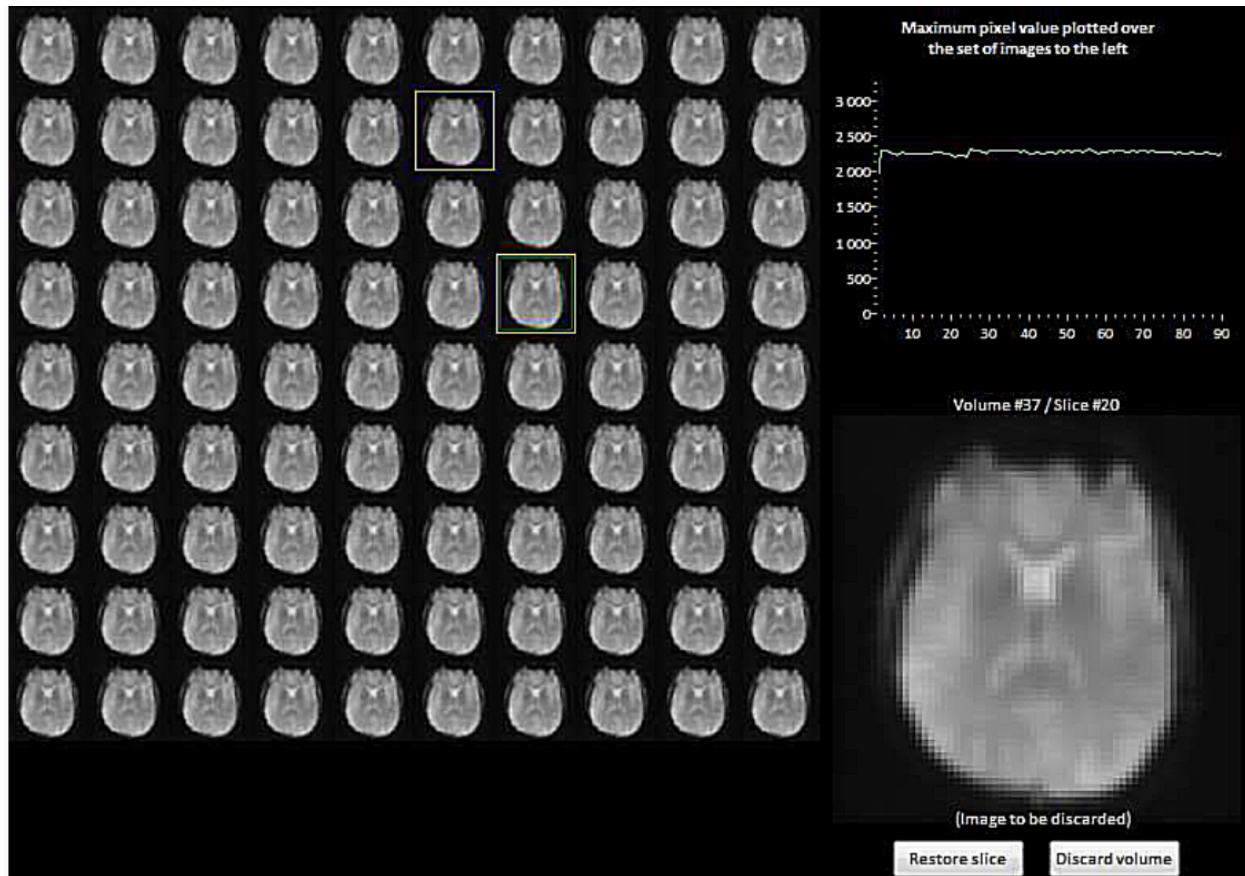
1.9 Image quality and preprocessing

How can the wine is absent for a nice meal, after appetizer, we need go through the drinks menu.

1.9.1 1 Inspect the image quality

When you go to a fancy restaurant, sit down and finish the appetizers, the next for you would be the drink menu and you will ask for the quality of wines. The most important ingredient for a wine is the grapes, they are the secret to the aroma and taste of wine. Unfortunately, there are several factors that could impact the quality of grapes such as the temperature, humidity, or latitude and longitude, so do the images. So, what could be better guarantee than inspect them closely by yourself?

Individual slices in an fMRI acquisition commonly suffer from random variations in average signal intensity, noise spikes, ghosts, and data glitches. These may result from physiological sources (participant motion, respiration, cardiac pulsations, anxiety, drowsiness, drugs) or from the scanner itself (field inhomogeneities, eddy currents, gradient heating, electronics). If unrecognized and included in the data analysis, these may spoil the entire experiment. A quick visual review of all source images together in montage mode is highly recommended to search for and exclude ("scrub") aberrant slices that appear too bright, too dark, or contain artifacts.



1.9.2 2 Slice correction

There are no two identical wine in the world. we have to look at each bottle.

Most fMRI studies acquire one slice at a time, meaning that the signal recorded from one slice may be offset in time by up to several seconds when compared to another. Each of these slices takes tens to hundreds of milliseconds. The two most commonly used methods for creating volumes are sequential and interleaved slice acquisition. sequential acquisition slices have been acquired in sequential (1,2,3,4,5,6...), Interleaved slice acquisition (1,3,5,...2,4,6...) acquires every other slice and then fills in the gaps on the second pass o. Although slice timing differences may not be important for simple block design experiments, they can impact considerable errors in rapid, event-related fMRI studies if not accounted for. In addition, the statistics model requires the time-series for each slice needs to be shifted back in time by the duration it took to acquire that slice.

Fig. 1: figure created by Andrew Jahn

1.9.3 3 Motion correction

Another influential factor for the quality is the move, even a small motion would make a difference to the wine, just like they did to the scan.

Head motion is the largest source of error in fMRI studies, If the subject in the scanner is moving, the images will look blurry. If the subject moves a lot, we also risk measuring signal from a voxel that moves. We are then in danger of measuring signal from the voxel from a different region or tissue type. It is the false positive. In addition, motion can introduce confounds into the imaging data because motion generates signal. If the subject moves every time in response to a stimulus - for example, if he jerks his head every time he feels an electrical shock, then it can become impossible to determine whether the signal we are measuring is in response to the stimulus, or because of the movement. Therefore, a variety of strategies have been developed to cope with this problem such as immobilization of the head using padding and straps, bite bars, masks during the scanning. What's more, the standard motion correction method of neuroimage packages considers the head as a rigid body with three directions of translation (displacement) and three axes of rotation. A single functional volume of a run is chosen as the **reference** to which runs in all other volumes are aligned. Procedures are performed in which each volume is rotated and aligned with the reference, with the goal to minimize a cost function. This iterative adjustment terminates once no further improvement can be achieved..

Fig. 2: figure created by Andrew Jahn

1.9.4 4 Registration

The easiest and quickest way to know the quality of a wine is comparison, we need to take different wines to know which one is better and we also need a standard for a reference, we can compare each wine to each other and a standard wine for a quality assessment. So, here is the registration and normalization.

Although most people's brains are similar - everyone has 4 lobes and subcortical, there are also differences in brain size and shape. As a consequence, if we want to do a group analysis we need to ensure that each voxel for each subject corresponds to the same part of the brain. If we are measuring a voxel in the visual cortex, make sure that every subject's visual cortex is in alignment with each other.

This alignment between the functional and anatomical images is called Registration. Most registration algorithms use the following steps:

- 1 Assume that the functional and anatomical images are in roughly the same location. If they are not, align the outlines of the images.
- 2 Take advantage of the fact that the anatomical and functional images have different contrast weightings - that is, areas where the image is dark on the anatomical image (such as cerebrospinal fluid) will appear bright on the functional image, and vice versa. This is called mutual information. The registration algorithm moves the images around to test different overlays of the anatomical and functional images, matching the bright voxels on one image with the dark voxels of another image, and the dark with the bright, until it finds a match that cannot be improved upon.
- 3 Once the best match has been found, then the same transformations that were used to warp the anatomical image to the template are applied to the functional images.

1.9.5 5 Normalization

Each brain needs to be transformed to have the same size, shape, and dimensions to be compared. We do this by normalizing (or warping) to a template. A template is a brain that has standard dimensions and coordinates. Each subjects' functional images will be transformed to match the general shape and large anatomical features of the template. Most researchers have agreed to use them when reporting their results. The dimensions and coordinates of the template brain are also referred to as standardized space.

1.9.6 6 Smoothing

If we have the best wine, why don't we mix it with other drinks in order to achieve the best taste? It is common for neuroimage software to smooth the functional data, or replace the signal at each voxel with a weighted average of that voxel's neighbors. This may seem strange at first - why would we want to make the images blurrier than they already are?

Spatial smoothing is the averaging of signals from adjacent voxels. This improves the signal-to-noise ratio (SNR) but decreases spatial resolution, blurs the image, and smears activated areas into adjacent voxels. The process can be justified because closely neighboring brain voxels are usually inherently correlated in their function and blood supply. The standard method is to convolve ("multiply") the fMRI data with a 3D Gaussian kernel ("filter") that averages signals from neighboring voxels with weights that decrease with increasing distance from the target voxel. In practice, the full width half maximum (FWHM) value of the Gaussian spatial filter is typically set to about 4-6 mm for single subject studies and to about 6-8 mm for multi-subject analyses. The benefits of smoothing can outweigh the drawbacks:

1 We know that fMRI data contain a lot of noise, and that the noise is frequently greater than the signal. By averaging over nearby voxels we can cancel out the noise and enhance the signal.

2 Smoothing can be good for group analyses in which all of the subjects' images have been normalized to a template. If the images are smoothed, there will be more overlap between clusters of signal, and therefore greater likelihood of detecting a significant effect.

Fig. 3: figure created by Andrew Jahn

If you are interested in data quality and preprocessing and want to know more, please go [here](#)

1.10 FreeSurfer

Now, it is time for our first main course, FreeSurfer.

What is FreeSurfer?

FreeSurfer is a brain imaging software package that focuses on analyzing magnetic resonance imaging (MRI) and functional scans of brain tissue from cross-sectional or longitudinal research, it can help researchers to conduct both volume-based and surface-based analyses. It is developed by the Laboratory for Computational Neuroimaging at the Athinoula A. Martinos Center for Biomedical Imaging. FreeSurfer includes tools for the reconstruction of topologically correct and geometrically accurate models of both the gray/white and pial surfaces, for measuring cortical thickness, surface area and folding, and for computing inter-subject registration based on the pattern of cortical folds



1.10.1 Installation

You can download and install FreeSurfer from [Here](#) and follow the [Video](#) to set up FreeSurfer

There are different versions of FreeSurfer, it is recommended to use the latest one but FreeSurfer6.0.0 and above would be appropriate. After the installation, type:

```
source $FREESURFER_HOME/SetUpFreeSurfer.sh
```

Then, you are supposed to see this


```
[shao@WeiS ~]$ source $FREESURFER_HOME/SetUpFreeSurfer.sh
----- freesurfer-linux-centos7_x86_64-7.1.0-20200511-813297b -----
Setting up environment for FreeSurfer/FS-FAST (and FSL)
FREESURFER_HOME   /usr/local/freesurfer/7.1.0-1
FSFAST_HOME       /usr/local/freesurfer/7.1.0-1/fsfast
FSF_OUTPUT_FORMAT nii.gz
SUBJECTS_DIR      /usr/local/freesurfer/7.1.0-1/subjects
MNI_DIR           /usr/local/freesurfer/7.1.0-1/mni
FSL_DIR           /usr/local
```

We can check whether FreeSurfer has been installed, there are two important messages for FreeSurfer, 1 FREESURFER_HOME is the home directory where your FreeSurfer has been installed, 2 SUBJECTS_DIR is the subject directory where you point for the FreeSurfer output results.

Test your FreeSurfer

You also want to test the freesurfer performance before it run the actual data.

Go to freesurfer subject directory:

```
cd $FREESURFER_HOME/subjects/
```

put `mri_convert sample-001.mgz sample-001.nii.gz` to test FreeSurfer. Due to different requirements of different versions of FreeSurfer, you might meet a license problem.

```
[shao@WeiS subjects]$ mri_convert sample-001.mgz sample-001.nii.gz
mri_convert sample-001.mgz sample-001.nii.gz
reading from sample-001.mgz...
TR=7.25, TE=3.22, TI=600.00, flip angle=7.00
i_ras = (-0, -1, -0)
j_ras = (-0, 0, -1)
k_ras = (-1, 0, 0)
writing to sample-001.nii.gz...
```

1.10.2 Recon-all

The most useful function of FreeSurfer is the recon-all command, it will tell FreeSurfer to performs all, or any part of if you specify, the FreeSurfer cortical reconstruction process. the basic format:

```
recon-all -all -i subjname_T1w.nii.gz -s subjname
```

-all tells FreeSurfer to do everything, including subcortical segmentation

-i stands for input

-s subject id

Normally, the input file would be T1 file, you also can improve the quality of surfaces by feed the T2 such as `recon-all -subject subjectname -i /path/to/input_volume -T2 /path/to/T2_volume -T2pial -all` FreeSurfer provided a tutorial [dataset](#) for you to play around it. You can follow this [link](#) to play with recon-all command.

Tutorial data practice

Once you have download the tutorial dataset, cd to `tutorial_data_20190918_1558/practice_with_data` and put `export SUBJECTS_DIR=$PWD` to specify the subject output directory to the current directory , then, go to `dicoms` and input `recon-all -all -i I50 -s Subj001` to take the I50 and create the Subj001 output. Of course, make sure you have activated the Freesurfer by source `$FREESURFER_HOME/SetUpFreeSurfer.sh` before you run the command. Here is a detailed instruction and a function list for `recon-all` . `recon-all` usually will cost 6-8 hours, depends on the computing power you have. Fortunately, there is a way to speed up this process.

```
[weishao@gra-login3 dicoms]$ recon-all -all -i I50 -s Subj001
Subject Stamp: freesurfer-linux-centos7_x86_64-7.1.1-20200723-8b40551
Current Stamp: freesurfer-linux-centos7_x86_64-7.1.1-20200723-8b40551
INFO: SUBJECTS DIR is /home/weishao/practice_with_data
Actual FREESURFER_HOME /cvmfs/soft.computecanada.ca/easybuild/software/2020/Core/freesurfer/7.1.1
Linux gra-login3 3.10.0-1127.8.2.el7.x86_64 #1 SMP Tue May 12 16:57:42 UTC 2020 x86_64 GNU/Linux
'/cvmfs/soft.computecanada.ca/easybuild/software/2020/Core/freesurfer/7.1.1/bin/recon-all' -> '/home/weishao/practice_with_data/Subj001/scripts/recon-all.local-copy'
/home/weishao/practice_with_data/Subj001

mri_convert /home/weishao/practice_with_data/DICOM/I50 /home/weishao/practice_with_data/Subj001/mri/orig/001.mgz

mri_convert /home/weishao/practice_with_data/DICOM/I50 /home/weishao/practice_with_data/Subj001/mri/orig/001.mgz
reading from /home/weishao/practice_with_data/DICOM/I50...
Getting Series No
INFO: Found 779 files in /home/weishao/practice_with_data/DICOM
INFO: Scanning for Series Number 13
Scanning Directory
INFO: found 176 files in series
INFO: loading series header info.

RunNo = 12
INFO: sorting.
sdfiSameSlicePos() eps = 0.000001
INFO: (256 256 176), nframes = 1, ismosaic=0
sdfi->UseSliceScaleFactor 0
INFO: rescale not needed
datatype = 4, short=4, float=3
PE Dir ROW ROW
AutoAlign matrix detected
AutoAlign Matrix -----
1.00000 0.00000 0.00000 0.00000;
0.00000 1.00000 0.00000 0.00000;
0.00000 0.00000 1.00000 0.00000;
0.00000 0.00000 0.00000 1.00000;
```

This will cost a few hours but we will come up with a solution to save our time in the next chapter. you will see a new directory has been created in the freesurfer subject directory

```
label mri scripts stats surf tmp touch trash
```

There are 3 directories you need to pay attention to, **mri** has all the volume file, **surf** contains all the surface outcome and scripts keep all the log information.

1.10.3 Freeview

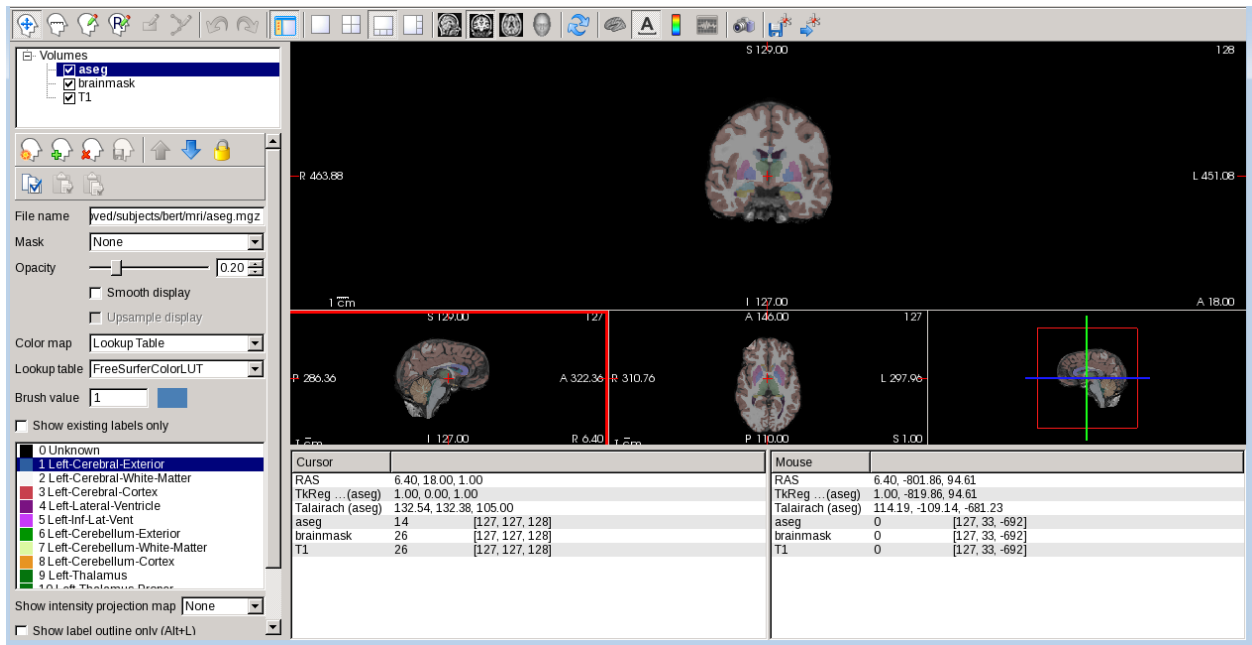
volume freeview

Freeview is a visualization tool comes with Freesurfer, you could test the freeview as well. `cd $SUBJECTS_DIR`, and type:

```
freeview -v \
bert/mri/T1.mgz \
bert/mri/brainmask.mgz \
bert/mri/aseg.mgz:colormap=lut:opacity=0.2
```

freeview will invoke the freeview of freesurfer

The flag -v is used to indicated that we are open volumes, T1.mgz: T1 anatomical image brainmask.mgz: skull-stripped volume primarily used for troubleshooting aseg.mgz : subcortical segmentation loaded with its corresponding color table and at a opacity=0.2



Freeview window will appear and load the data, you can play around with different functions. For example, you can use



buttons at the top to change which orthogonal view appears in the main viewing window. We can also use the



buttons to change the organization of the viewing panes.

To change which brain slice you are viewing, use the 'Page Up' or 'Page Down' keys on your keyboard or the up and down arrows. (Mac users: press the fn key while using the up and down arrows.) While Freeview can load many volumes at once, we cannot necessarily see them all at once. we are able to see whichever volume is at the top of the list in the menu on the left. An exception to this are volumes such as the wm.mgz and aseg.mgz which can be made translucent, allowing you to view the information they contain simultaneously with the volume directly below it on the list. For example, you are currently seeing information from both the aseg (labeled structures) and the brainmask (voxel intensities).

We can hide or turn off a layer by unchecking the check box next to the layer name. You can also use the up and down arrows (located below the menu on the left) to move the aseg down on the list, below the brainmask (try it!).

Note: It is important to ensure that you have install the Xming if you use WSL.

You also can see the volume files located in the mri directory.

```

antsdn.brain.mgz          brain.mgz                ribbon.mgz
aparc.a2009s+aseg.mgz    ctrl_pts.mgz            segment.dat
aparc+aseg.mgz           filled.mgz              surface.defects.mgz
aparc.DKTatlas+aseg.mgz  lh.ribbon.mgz           T1.mgz
aseg.auto.mgz            mri_nu_correct.mni.log  talairach.label_intensities.txt
aseg.auto_noCCseg.label_intensities.txt mri_nu_correct.mni.log.bak talairach.log
aseg.auto_noCCseg.mgz    norm.mgz                talairach_with_skull.log
aseg.mgz                 nu.mgz                  transforms
aseg.presurf.hypox.mgz   orig                     wm.asegedit.mgz
aseg.presurf.mgz         orig.mgz                wm.mgz
brain.finalsurfs.mgz     orig_nu.mgz             wmparc.mgz
brainmask.auto.mgz       rawavg.mgz              wm_seg.mgz
brainmask.mgz            rh.ribbon.mgz

```

Surface freeview

Now, we have know the basics of volume, let verify the surface, which means we need check two things:

- 1 whether the surface accurately follow the gray matter and white matter boundaries
- 2 whether the aseg accurately follows the subcortical intensity boundaries

In order to do that, let's start with "brainmask". From \$SUBJECTS_DIR, put the code:

```

freeview -v bert/mri/T1.mgz bert/mri/wm.mgz bert/mri/brainmask.mgz bert/mri/aseg.mgz:colormap=lut:opacity=0.2
-f      bert/surf/lh.white:edgecolor=blue      bert/surf/lh.pial:edgecolor=red      bert/surf/rh.white:edgecolor=blue
bert/surf/rh.pial:edgecolor=red

```

Double click on 'brainmask' in the left menu to bring it to the top of the volume list. The white surface (blue line) is used to calculate total white matter volume and should accurately follow the boundary between white matter and gray matter. The pial surface (red line) is used to calculate cortical gray matter volume and should accurately follow the boundary between the gray matter and the CSF.



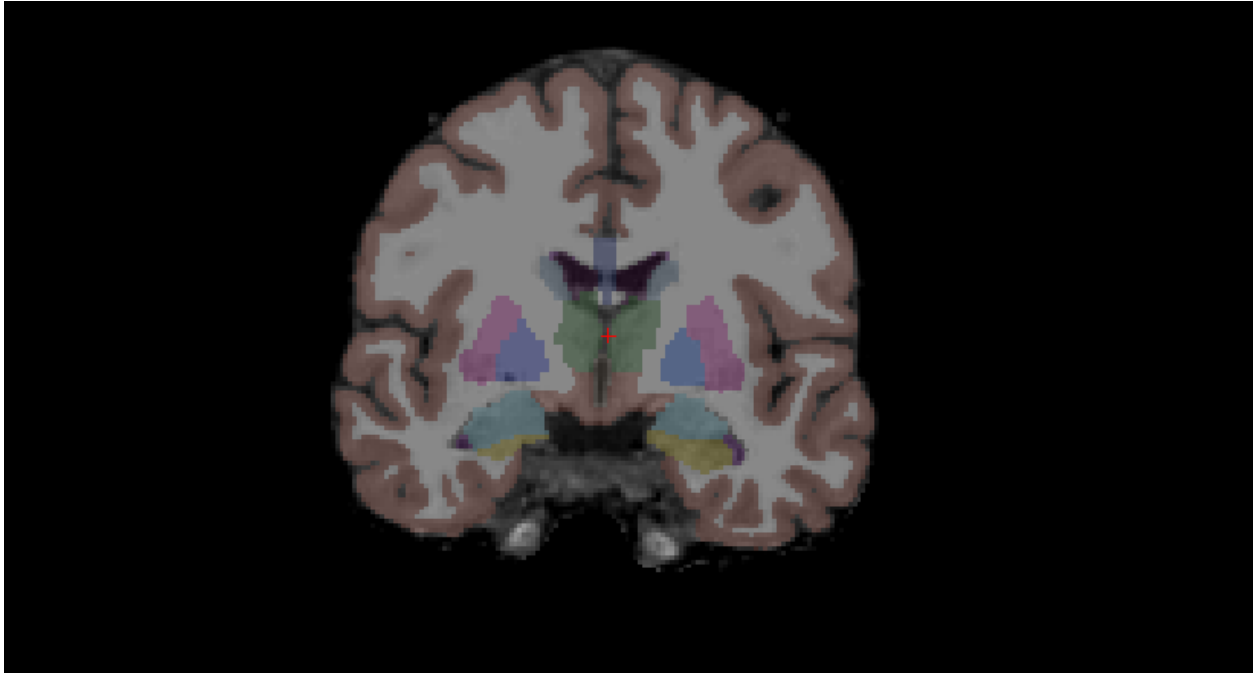
Note: There are regions where the surfaces are not intended to be accurate that we should be aware of: Areas around

the hippocampus and amygdala. The surfaces will not completely include or exclude certain

subcortical regions. These inaccuracies can be ignored as subcortical regions are excluded from the cortical measures and subcortical volume is measured by the aseg, not the surfaces.

Subcortical Segmentation

Uncheck all of the surfaces. Then check the box next to the aseg volume and double click it. The aseg volume will jump to the top of the left menu, above the brainmask volume. This will show the complete segmentation of the subcortical structures.



Each structure is labeled with a unique color/number distinction. If you click on a voxel the structure's name and number label will be shown in the 'Cursor' section under the viewing window next to the word.

1.10.4 3D Freeview in FreeSurfer

As we have view the volumes and surface above, there is 3D usage from freeview, we are only use the left hemisphere for simplicity.

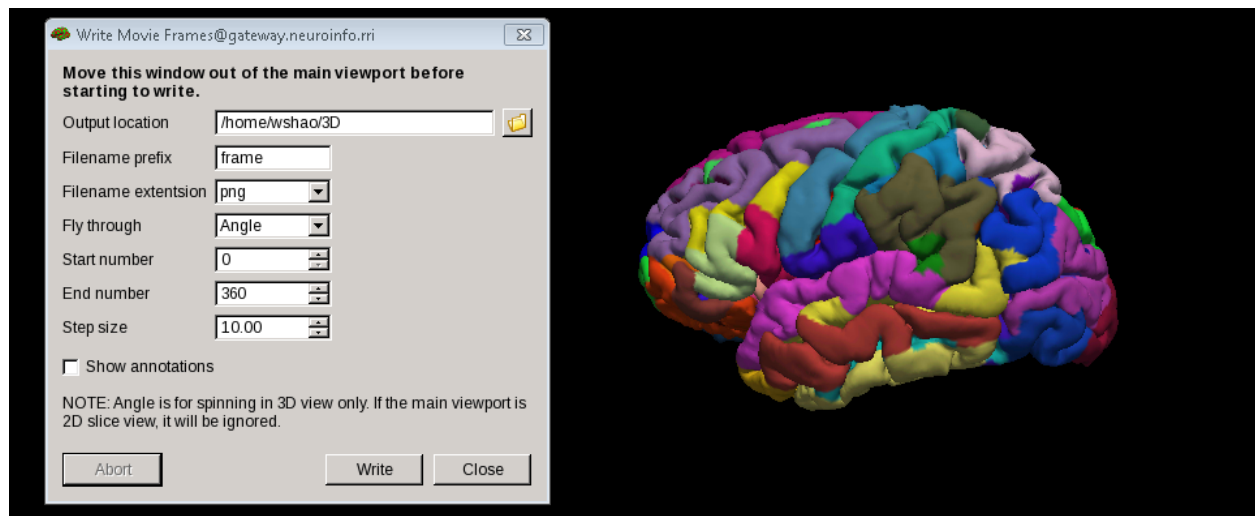
1 Open up any surface from the data as did above. Type the following commands from \$SUBJECTS_DIR:

```
freeview -f bert/surf/lh.pial:annot=aparc.a2009s.annot:name=pial_aparc_des:visible=0
```

2 Set the viewport to 3d view, right click in the viewport and select Hide All Slices

3 In the File menu, select Save Movie Frames

4 Set up the options as in the following picture to save the output - as we create a new directory named 3D in our home directory already..



5 From terminal, navigate to the new directory our output located -3D, Run this command:

```
convert -delay .1 *.png brainanim.gif
```

Note: `convert` is from the ImageMagick library, you can find the files from [here](#)

To view your GIF, open it with firefox browser by type `firefox brainanim.gif` from your terminal

More details from [freeview](#)

1.10.5 Segmentation of hippocampal subfields

One of important function of FreeSurfer is the subfield segmentation of Hippocampus and amygdala

After `recon-all` has been completed, you can take the output from `recon-all` and the pipeline:

```
segmentHA_T1.sh subject_name [SUBJECTS_DIR]
```

[SUBJECTS_DIR] is optional, the output files will be appear in the mri directory of SUBJECT_DIR (\$SUBJECTS_DIR/subjects_name/mri/)

You can check the outfiles with freeview:

```
freeview -v nu.mgz -v lh.hippoAmygLabels-T1.v21.mgz:colormap=lut -v rh.hippoAmygLabels-
↪T1.v21.mgz:colormap=lut
freeview -v nu.mgz -v lh.hippoAmygLabels-T1.v21.HBT.mgz:colormap=lut -v rh.
↪hippoAmygLabels-T1.v21.HBT.mgz:colormap=lut
freeview -v nu.mgz -v lh.hippoAmygLabels-T1.v21.FS60.mgz:colormap=lut -v rh.
↪hippoAmygLabels-T1.v21.FS60.mgz:colormap=lut
freeview -v nu.mgz -v lh.hippoAmygLabels-T1.v21.CA.mgz:colormap=lut -v rh.
↪hippoAmygLabels-T1.v21.CA.mgz:colormap=lut
```

[lr]h.hippoSfVolumes-T1.v21.txt: these text files store the estimated volumes of the hippocampal substructures and of the whole hippocampus..

[lr]h.amyNucVolumes-T1.v21.txt: these text files store the estimated volumes of the nuclei of the amygdala and of the whole amygdala.

[lr]h.hippoAmygLabels-T1.v21.mgz: they store the discrete segmentation volumes at subvoxel resolution (0.333 mm).

[lr]h.hippoAmygLabels-T1.v21.FSvoxelSpace.mgz: they store the discrete segmentation volume in the FreeSurfer voxel space (normally 1mm isotropic, unless higher resolution data was used in recon-all with the flag -cm).

[lr]h.hippoAmygLabels-T1.v21.[hierarchy].mgz: they store the segmentations with the different hierarchy levels.

[lr]h.hippoAmygLabels-T1.v21.[hierarchy].FSvoxelSpace.mgz: same as above, but in FreeSurfer voxel space.

In addition T1 scan, you can also use T2 scan as an additional scan:

```
segmentHA_T2.sh subjects_name FILE_ADDITIONAL_SCAN ANALYSIS_ID USE_T1 [SUBJECTS_
↳DIR]
```

FILE_ADDITIONAL_SCAN is the additional scan to use in the segmentation

ANALYSIS_ID is a user defined identifier that makes it possible to run different analysis with different types of additional scans

USE_T1 is a flag that indicates whether the intensities of the main T1 scan should be used (multispectral segmentation). The words USE_T1 must be replaced with a 0 or 1 on the command line

SUBJECTS_DIR is optional, and overrides the FreeSurfer subject directory when provided

For MacOC user, please follow this [video](#)

Go [HippocampalSubfieldsAndNucleiOfAmygdala](#) to see all the instructions

1.10.6 Extract the volume matrix from FreeSurfer

Once we use the freesurfer automated segmentation, we can also collect the volumes of the subregions of the hippocampus / amygdala of all subjects and write them to a single file, extracting the volume matrix:

```
quantifyHASubregions.sh hippoSf <T1> <output_file> <Output_file_directory>
```

The first argument `quantifyHASubregions.sh` specifies that we want to collect the volumes of the hippocampus (hippoSf). The second argument is the name of the analysis: for the first mode of operation (only main T1 scans), it is simply type T1, and don't forget to name the output_file.

After a few seconds, you will see the output files in the current directory, open it with `less`

```
Subject left Hippocampal tail left subiculum-body left CA1-body left subiculum-head left hippocampal-fissure left presubiculum-head left CA1-head left presubiculum-body left parasubicu$
01 547.573823 269.542726 144.093222 148.469652 135.527776 147.425482 529.001421 145.024610 78.105639 331.925440 220.223649 165.706227 87.826016 126.384699 134.100140 109.684497 68.1620$
02 643.483724 266.623763 116.937474 210.294662 134.383683 159.031199 525.572406 184.410845 76.308266 349.818053 230.566687 169.610246 76.378866 138.467592 141.273438 119.575313 100.039$
03 623.166315 211.448107 138.495856 182.848024 135.010585 131.076089 544.053013 142.740832 56.280756 343.088025 223.425565 165.264843 84.750354 128.297465 137.263200 114.400725 70.4941$
04 476.421163 250.717484 121.351679 174.002416 138.036521 127.005116 483.231803 211.622547 66.286364 311.086096 225.118592 149.176093 70.061044 126.088168 122.652961 99.567051 79.0864$
05 580.782241 250.363893 130.472972 206.988375 160.402111 162.543548 612.121695 165.616802 77.586157 374.606426 227.484194 180.877607 83.414294 132.519104 147.090380 115.317537 88.1001$
06 619.373194 206.269883 135.483975 181.230045 107.799823 140.453036 550.304392 133.587514 75.060548 346.100894 212.853179 165.009713 87.707794 123.940052 136.079871 109.566584 63.1895$
07 645.009051 273.342507 153.228970 217.878808 210.362747 164.714948 654.090685 188.373291 99.138325 401.843649 259.086937 192.581498 96.300901 146.117337 157.270968 127.668455 77.4098$
08 549.926740 278.823575 182.824411 235.747355 163.001548 165.351464 519.922876 216.075799 92.685494 346.766125 228.946183 152.782585 71.260679 134.917947 127.022956 113.688670 120.214$
09 660.075883 238.418035 129.894161 205.120150 141.492711 149.861974 577.268735 183.274153 63.153491 359.939061 233.315498 165.524363 84.367276 145.640898 136.293361 126.488178 75.1390$
10 644.673062 242.258491 162.383968 190.263558 151.354168 138.220184 532.430664 146.653622 64.401202 347.161314 237.254692 167.040067 93.239340 121.424996 143.532546 107.361719 75.4610$
11 679.078133 276.708942 129.705647 166.154022 134.705884 144.290036 590.315591 193.701684 94.374987 350.645978 238.486835 191.076477 70.585447 146.318916 155.958150 125.297670 108.232$
12 537.566880 259.365762 100.609740 174.888533 133.000565 141.957930 470.801665 219.097839 67.536279 308.727745 221.845983 150.451368 58.106508 124.252932 125.211188 103.961447 86.8718$
13 541.136052 211.532362 135.739442 185.899783 115.228640 124.882499 522.269577 115.823071 59.038112 328.291552 208.418433 152.346662 81.976287 121.332008 123.619968 109.575931 64.5638$
14 515.654660 209.202155 136.806623 172.410320 137.617395 122.586516 445.036107 120.760821 65.042888 293.795429 223.439569 134.753923 97.930552 129.727009 115.623403 118.517661 55.6226$
15 454.317635 187.364309 102.616111 205.329865 156.984056 134.991305 521.367057 167.239887 74.780479 324.963557 185.768531 140.907183 53.413519 108.095484 115.669291 92.795409 98.61098$
16 700.051882 240.932210 152.720727 203.052663 147.789478 162.442532 606.773363 167.490228 72.109264 389.942954 255.969952 195.054296 111.369154 142.238659 163.417512 131.020134 80.393$
```

What a mess! Fortunately, there is a solution.

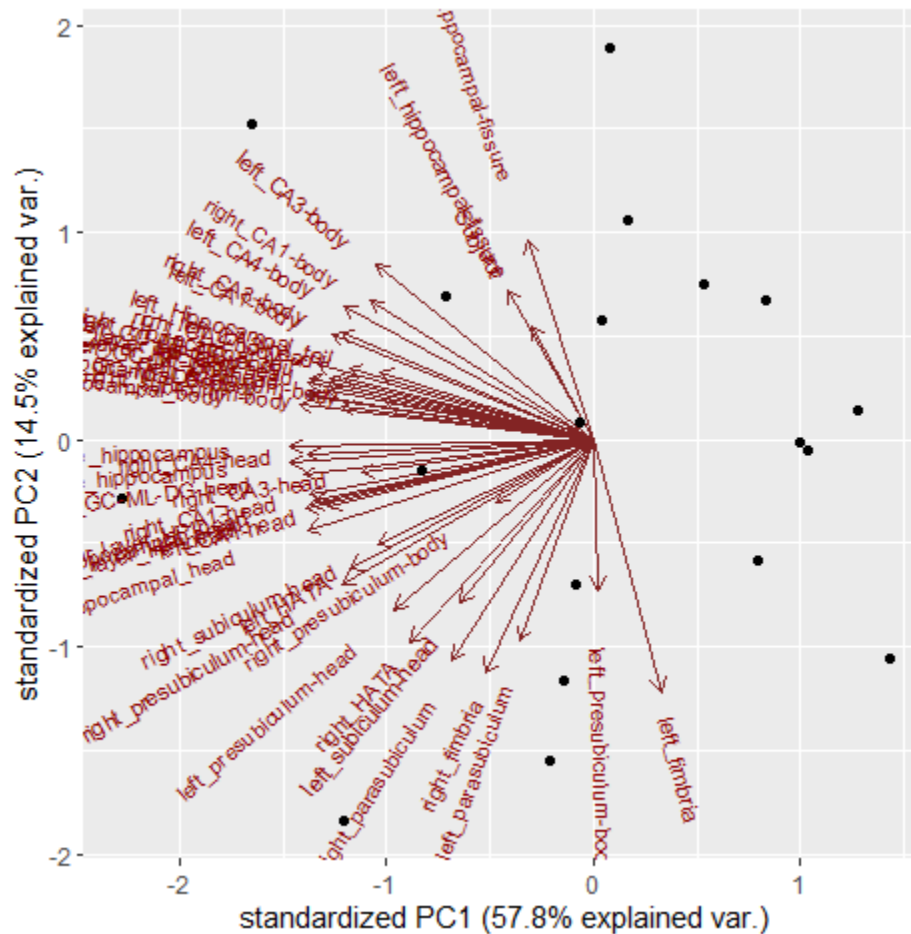
- 1 Open the file with Excel 2016.
- 2 Look for “Data” tab and “Text in/to column” button.
- 3 In the step 1, select “Delimited”.
- 4 In the step 2, select first “space”, and then choose “string classifier” as “.”

Change format in step 3. "Finish". And save as csv file

Then, You will get a file like this

[illegible]

and have fun by play it around like PCA analysis.



1.10.7 FreeSurfer commands

There are commands that could be quite useful.

Like 3dinfo from AFNI, there is a useful command that can help we know the basic information of the image, for example, we can `mri_info` to find the basic information from the NIFTI files

```

type: nii
dimensions: 192 x 256 x 160
voxel sizes: 1.000000, 1.000000, 1.000000
type: SHORT (4)
fov: 192.000
dof: 1
xstart: -96.0, xend: 96.0
ystart: -128.0, yend: 128.0
zstart: -80.0, zend: 80.0
TR: 0.00 msec, TE: 0.00 msec, TI: 0.00 msec, flip angle: 0.00 degrees
nframes: 1
PhEncDir: UNKNOWN
FieldStrength: 0.000000
ras xform present
xform info: x_r = -1.0000, y_r = -0.0000, z_r = -0.0000, c_r = -2.5132
           : x_a = -0.0000, y_a = 1.0000, z_a = -0.0000, c_a = 30.6014
           : x_s = 0.0000, y_s = 0.0000, z_s = 1.0000, c_s = -55.5551
Orientation : LAS
Primary Slice Direction: axial

voxel to ras transform:
-1.0000 -0.0000 -0.0000 93.4868
-0.0000 1.0000 -0.0000 -97.3986
0.0000 0.0000 1.0000 -135.5551
0.0000 0.0000 0.0000 1.0000

voxel-to-ras determinant -1

ras to voxel transform:
-1.0000 -0.0000 -0.0000 93.4868
-0.0000 1.0000 -0.0000 97.3986
-0.0000 -0.0000 1.0000 135.5551
-0.0000 -0.0000 -0.0000 1.0000

```

Note: It is worth to note that there are 4 aspects that need to know such as dimensions , voxel sizes , TR , Orientation .

Dimension can tell you how many slices from each dimension of voxels. The size of the voxel gives some indication as to the spatial resolution of the data, with smaller voxels giving a higher spatial resolution. A typical voxel size for a structural MRI is 1 mm x 1mm x 1mm; for fMRI, 3 mm x 3 mm x 3 mm. However, these can vary substantially from study to study, The time between repeated volumes (i.e., between collecting a slice in one volume, and that same slice in the next) is the TR, which we will need know to the time of TR in order to do further fMRI analysis. Orientation can tell you the slice orientation, which is needed in FSL anf FreeSurfer.

1.11 Longitudinal processing

All data for this tutorial has already been processed for you since processing can take up to 24h. However, to continue you need to understand the three processing steps (cross, base, long) and associated directory names.

Let's say we have a subject name sub with two time points: time_point_1 and time_point_2. Here are 3 major parts of data FreeSurfer processed:

[CROSS]: There are two images independently first (cross in crosssectional analysis/processing) come from recon-all:

```
recon-all -subjid input_T1_point1 -all recon-all -subjid input_T1_point2 -all
```

After FreeSurfer processed the same subject with two timepoints, There are two new directories with the name Timepoint_1 and Timepoint_2 appeared in the SUBJECTS_DIR directory. Then, we are going to continue the second step.

[BASE]: The aim of this processing is to create a within-subject template (also called base) and end up with a new directory. There is only one base directory for per subject. FreeSurfer decides to name the base: base_subj. Inside this directory are the results for the average anatomy of the subject across time. we can use this data for quality checking or editing:

```
recon-all -base base_subj -tp Timepoint_1 -tp Timepoint_2 -all
```

[LONG]: Finally, we would need to create the two longitudinal runs for our subject (these are the ones we are actually interested in). There are new two more directories (FreeSurfer will automatically assign the name: Timepoint_1.long.base_subj and Timepoint_2.long.base_subj. They are the final, most reliable and accurate processing results

1.11.1 Longitudinal processing in hippocampus

Longitudinal analysis greatly reduces the confounding effect of inter-individual variability by using each subject as his or her own control. While one could analyze each time independently with segmentHA_T1.sh, such a cross-sectional analysis disregards the key fact that the scans are of the same subject. Instead, the scans corresponding to the different time points can be segmented jointly. FreeSurfer method relies on a subject-specific atlas, and treats all time points the same way in order to avoid processing bias. It is important to remark that this method does not assume any specific trajectory for the segmentations or corresponding volumes. Essentially, this means this method does not assume that the hippocampus continuously shrinks or grows.

Let's say that <baseID> is the ID of the base subject (template from mainstream). Then, we can produce the longitudinal hippocampal subregion segmentation with the following command:

```
segmentHA_T1_long.sh <baseID>
```

The output of this script can be found under the corresponding mri directories of the longitudinally processed subjects, which located at Timepoint_1.long.base_subj/mri The output files will contain the suffix .long such as:

[lr]lh.hippoAmygLabels-T1.long.v21.[hierarchy].<FSvoxelSpace>.mgz: segmentations:

lh.hippoAmygLabels-T1.long.v21.CA.FSvoxelSpace.mgz

lh.hippoAmygLabels-T1.long.v21.CA.mgz

lh.hippoAmygLabels-T1.long.v21.FS60.FSvoxelSpace.mgz

lh.hippoAmygLabels-T1.long.v21.FS60.mgz

lh.hippoAmygLabels-T1.long.v21.FSvoxelSpace.mgz

lh.hippoAmygLabels-T1.long.v21.HBT.FSvoxelSpace.mgz


```
lh.hippoAmygLabels-T1.long.v21.HBT.mgz
```

```
[lr]h.hippoSfVolumes-T1.long.v21.txt: volumes of the hippocampal substructures:
```

```
lh.hippoSfVolumes-T1.long.v21.txt
```

```
[lr]h.amygNucVolumes-T1.long.v21.txt: volumes of the nuclei of the amygdala:
```

```
lh.amygNucVolumes-T1.long.v21.txt
```

1.11.2 Visualization

Bear in mind that the longitudinally processed time points are coregistered, so you can overlay their images and segmentations on top of each other. if you have two longitudinally processed subjects `timepoint1.long.baseID` and `timepoint2.long.baseID`, From `SUBJECTS_DIR`, you could run:

```
freeview -v timepoint1.long.baseID/mri/nu.mgz -v timepoint1.long.baseID/mri/lh.
↪hippoAmygLabels-T1.long.v21.mgz:colormap=lut -v timepoint1.long.baseID/mri/rh.
↪hippoAmygLabels-T1.long.v21.mgz:colormap=lut \
    -v timepoint2.long.baseID/mri/nu.mgz -v timepoint2.long.baseID/mri/lh.
↪hippoAmygLabels-T1.long.v21.mgz:colormap=lut -v timepoint2.long.baseID/mri/rh.
↪hippoAmygLabels-T1.long.v21.mgz:colormap=lut
```

1.11.3 Volume extraction

We can extract the volume matrix from the longitudinal result FreeSurfer has processed. We can also collect the volumes of the subregions of the hippocampus/amygdala of all subjects and write them to a single file, extracting the volume matrix:

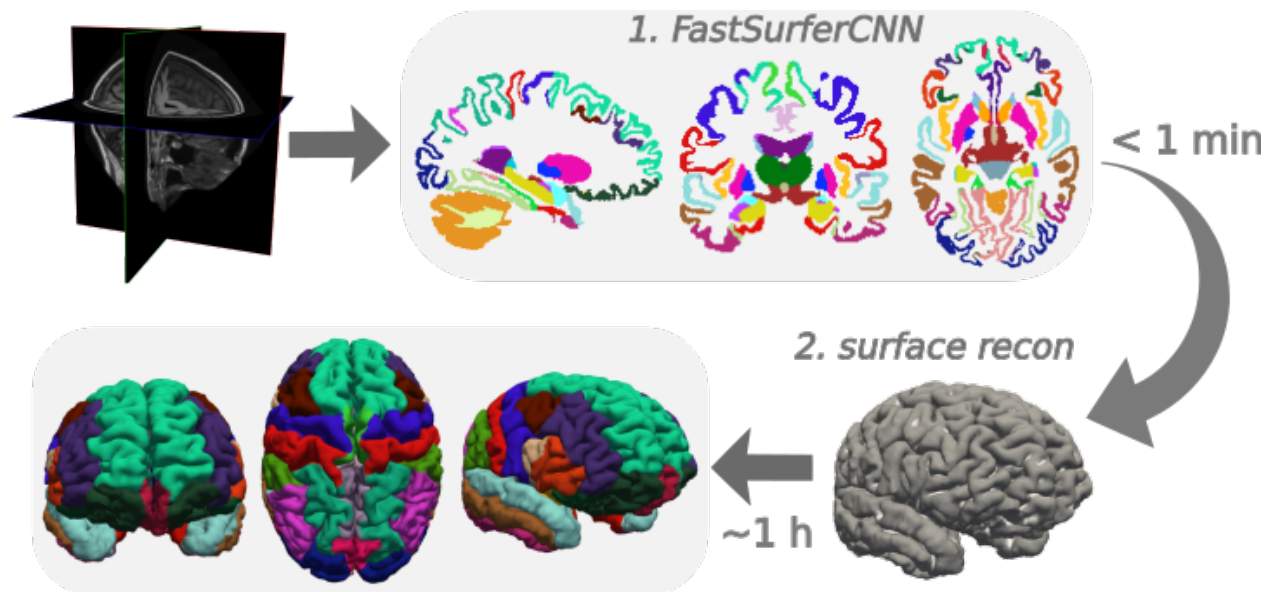
```
quantifyHASubregions.sh hippoSf T1 <output_file directory>
```

After a few seconds, you will see the output files in the current directory, open it with `less` or any text editor.

1.12 FastSurfer

After you run the recon-all with a tutorial data, I bet you might wonder how come I can skip this process time. The first solution would be FastSurfer.

FastSurfer is a fast and deep-learning pipeline for the fully automated processing of structural human brain MRIs. It provides conform outputs\$



FastSurfer consists of two main parts:

FastSurferCNN Volumetric Segmentation.

FastSurferCNN is an advanced deep learning pipeline for whole brain segmentation into 95 classes in under 1 minute, mimicking FreeSurfer's anatomical segmentation and cortical parcellation.

go to [Here](#) either use `git clone` from you home directory to ge the file or download the file and put it in your home directory.

1.12.1 Set up and run FastSurfer

Set the path `export FREESURFER_HOME=/usr(username)/local/freesurfer/6.0.0`

`source $FREESURFER_HOME/SetUpFreeSurfer.sh` to activate the Freesurfer

`datadir=/home/user/mri_data_directory`

`fastsurferdir=/home/user/fastsurfer_analysis_directory`

Run FastSurfer:

```
./run_fastsurfer.sh --t1 $datadir/subject1/orig.mgz \
                    --sid subject1 --sd $fastsurferdir \
                    --parallel --threads 4 --py $(which python)(optional)
```

`--sd` Output directory \$SUBJECTS_DIR

`--sid` Subject ID for directory inside \$SUBJECTS_DIR to be created

`--t1` T1 full head input. The network was trained with confirmed images, these specifications are checked in the `eval.py` script and the image is automatically conformed if it does not comply.

`parallel` means we use parallel computing to run 4 threads indicated by `threads 4`

`--py $(which python)` specify the python Fastsurfer use

Before you run the script, just ensure you check all the required packages

`sed -i "s/==/>=/g" requirements.txt` and `pip install --no-index -r requirements.txt` might help

```
~/FastSurfer/FastSurferCNN ~/FastSurfer
/home/weishao/FastSurfer/env/bin/python eval.py --in_name /home/weishao/projects/rrg-brad/aim_raw_data/sub-1005/anat//su
ri/aparc.DKTatlas+aseg.deep.mgz --order 1 --network_sagittal_path ../checkpoints/Sagittal_Weights_FastSurferCNN/ckpts/Ep
al_Weights_FastSurferCNN/ckpts/Epoch_30_training_state.pkl --network_coronal_path ../checkpoints/Coronal_Weights_FastSur
run
Reading volume /home/weishao/projects/rrg-brad/aim_raw_data/sub-1005/anat//sub-1005_T1w.nii.gz
Conforming image to UCHAR, RAS orientation, and 1mm isotropic voxels

Input:    min: 0  max: 1348
rescale:  min: 0.0 max: 928.772  scale: 0.27455608050199615
Output:   min: 0.0 max: 255.0
Cuda available: False, # Available GPUS: 0, Cuda user disabled (--no_cuda flag): False, --> Using device: cpu
Loading Axial
Successfully loaded Image from /home/weishao/projects/rrg-brad/aim_raw_data/sub-1005/anat//sub-1005_T1w.nii.gz
Loading Axial Net from ../checkpoints/Axial_Weights_FastSurferCNN/ckpts/Epoch_30_training_state.pkl
Axial model loaded.
-->Batch 0 Axial Testing Done.
-->Batch 1 Axial Testing Done.
-->Batch 2 Axial Testing Done.
-->Batch 3 Axial Testing Done.
-->Batch 4 Axial Testing Done.
-->Batch 5 Axial Testing Done.
-->Batch 6 Axial Testing Done.
-->Batch 7 Axial Testing Done.
-->Batch 8 Axial Testing Done.
-->Batch 9 Axial Testing Done.
-->Batch 10 Axial Testing Done.
-->Batch 11 Axial Testing Done.
-->Batch 12 Axial Testing Done.
-->Batch 13 Axial Testing Done.
-->Batch 14 Axial Testing Done.
-->Batch 15 Axial Testing Done.
-->Batch 16 Axial Testing Done.
```

It is worth to notice that FastSufer needs the support from FreeSurfer and it is works for FreeSurfer6.0.0 now

1.13 Parallel computing for recon-all

The second solution to speed up recon-all is parallel computing!

Thanks to the development of technology, we have more much powerful computers than before. For example, a physical core is an actual physical processor core in your CPU. each physical core has its own circuitry to read and execute instructions separately. A normal laptop usually has 4-8 physical cores.

Freesufer supports the OpenMP code, which means that recon-all can be processed by multiple-cores/threads, you can recon-all one subject with multiple cores to speed with the process time. since the default setting for recon-all is one core, we have to tell our computer what we want. The command to tell freesurfer to run recon-all with multiple cores is `-openmp X`, X indicates how many cores you want to run.

First, you need to check the CPU information, type the "lscpu" in linux system environment or press Ctrl + Shift + Esc to open Task Manager and select the Performance tab to see how many cores and logical processors under the Windows OS.

```
[shao@WeiS FreeSurfer]$ nano Parallel_computing.rst
[shao@WeiS FreeSurfer]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                 8
On-line CPU(s) list:   0-7
Thread(s) per core:    2
Core(s) per socket:    4
Socket(s):              1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                  142
Model name:             Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
Stepping:               11
CPU MHz:                1992.000
CPU max MHz:            1992.0000
BogoMIPS:               3984.00
Virtualization:         VT-x
Hypervisor vendor:      Windows Subsystem for Linux
Virtualization type:    container
```

As you can see, I have 8 cores on my laptop. Then i can add no more than the actual CPUs I have by this magic line:

```
recon-all -all -i input_T1 -s subjname -openmp 8
```

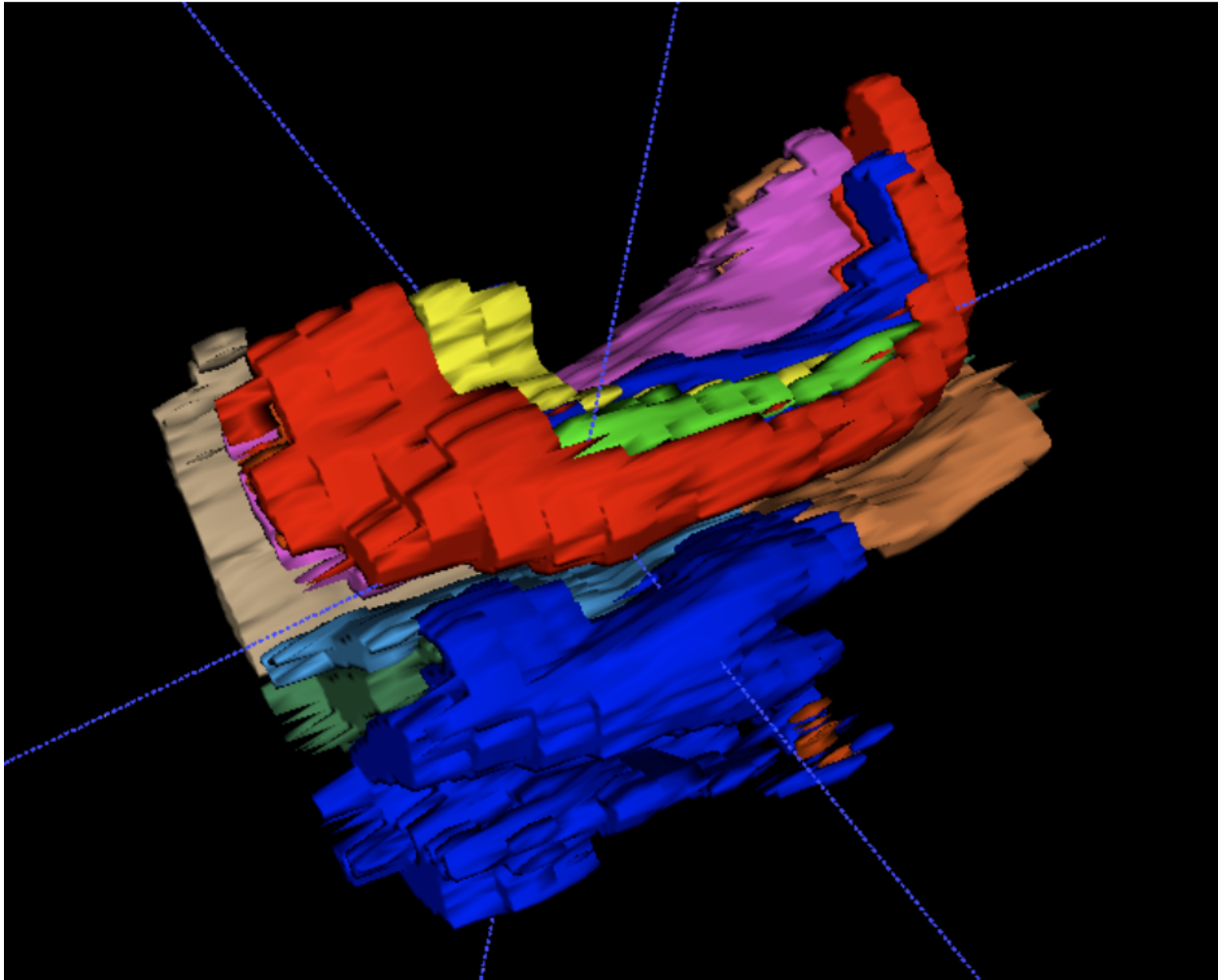
This pipeline works for your personal computer and Supercomputer both

1.14 Volume analysis with FreeSurfer

This chapter is in construction

What I should to do in here is incredible

1.15 ASHS



Segmentation of hippocampal subfields and MTL subregions by ASHS
(a 3D rendering) from T2-weighted MRI scans

1.15.1 What is ASHS?

ASHS is software for automatic segmentation of the medial temporal lobe (MTL) substructures from brain MRI scans. These structures are involved in early Alzheimer's disease and in a number of other neurological conditions. They are also important in research on memory and cognition. For more information, please check [here](#).

ASHS Installation

ASHS installation is very user friendly. All you need to do is go [here](#) and download the most recent version of fast-ashs(v2.0.0 (July 2018)) and put the file into your ASHS directory. Next, unzip the file *ashs-fastashs_2.0.0_07202018.zip* by type:

```
unzip ashs-fastashs_2.0.0_07202018.zip
```

Now, you need to set up the environment, let's start with the ASHS_ROOT:

```
export ASHS_ROOT=/home/ASHS/ashs-fastashs
```

To avoid use this command everytime when you log in, you can type this in the bashrc (Linux system) or .cshrc(Mac) file, please go [there](#) for more details.

After set up, you can verify the the environment by:

```
$ASHS_ROOT/bin/ashs_main.sh -h
```

If you see the output like this:

```
ashs_main: automatic segmentation of hippocampal subfields
usage:
ashs_main [options]
...
```

You are good to go.

Run ASHS

Once you set up, you can run ASHS by a very simple command.

For running ASHS on a single core:

```
bash $ASHS_ROOT/bin/ashs_main.sh \
-I subj001 \
-a $HOME/ASHS/atlas \
-g subj001_mprage.nii.gz \
-f subj001_tse.nii.gz \
-w $HOME/ASHS/output
```

The -I option provides the ID for the subject, which will be the ID in the output directory. The -g and -f options require the T1 and T2-weighted MRI scans, respectively. If you are only using the ASHS-PMC-T1 atlas, simply supply the T1-weighted MRI scan to both -g and -f arguments. -a points to the location of the atlas file, -w points to the directory where the ASHS output will be.

You also can run ASHS with multiple cores with add one more argument:

```
bash $ASHS_ROOT/bin/ashs_main.sh -P \
-I subj001 \
-a $HOME/ASHS/atlas \
-g subj001_mprage.nii.gz \
-f subj001_tse.nii.gz \
-w $HOME/ASHS/output
```

Building the Atlas

One of the advantages of ASHS is the customized atlas. You can build the unique segmentation protocol with ASHS. Normally, it will need 20-40 subjects to build the atlas. the files are:

Data manifest file (required) Label description file (required) Configuration file (optional) Slice heuristics file (optional) Cross-validation file (optional)

Let's start from the simplest one, assume you have all the required files in your ASHS directory and put the file in the right place. All you need to do is create a bash script:

```
#!/bin/bash
export ASHS_ROOT=/home/ASHS/fastashs

$ASHS_ROOT/bin/ashs_train.sh \
-D $HOME/ASHS/filedata.txt \
-L $HOME/ASHS/snaplabels.txt \
-w $HOME/ASHS/output
```

1.16 Comprehensive MTL Segmentation Protocol

1.16.1 Overview in Medial temporal lobe segmentation guide

In addition to segmentate hippocampus with softwares, there is better way— human protocol. In this chapter, we will see one of them. **Olsen-Amaral-Palombo Protocol**, created by **Dr. Rosanna Olsen** with her colleagues and students. For more information, please visit [here](#).

1 Getting Started

GETTING STARTED WITH SEGMENTATION

In this section, we will start off by surveying the goals of medial temporal lobe segmentation and introducing the necessary tools for manual segmentation.

GOALS OF SEGMENTATION

The medial temporal lobes (MTL) are composed of several regions of interest (ROI) for perception and memory research, including the rhinal cortex, hippocampus, and parahippocampal cortex. To understand the relationship between grey matter volume in these MTL regions and specific cognitive processes, we need to determine the boundaries of these regions. This is achieved through the manual segmentation, or tracing, of MTL regions. Here, we will introduce segmentation based on the Olsen-Amaral-Palombo (OAP) protocol.

TOOLS REQUIRED FOR SEGMENTATION

ITK-SNAP (Yushkevich et al., 2006) is used as the primary segmentation software in this guide. We use version 3.8.0 in this guide (download link [here](#)).

We recommend using a drawing pad, tablet or a stylus pen for segmentation. A mouse can also be used, however. You can use this [pad](#).

Anatomical plane is used for segmentation

Segmentation is done on the coronal view, which allows for the best visualization of the ROIs. The axial and sagittal views are useful when differentiating between sulci in the brain.

scan to do segment under this protocol

The OAP protocol segments on T2-weighted images. T1 scans are used for anatomical reference and should be co-registered for alignment prior to starting segmentation. To learn more about high-resolution co-registration, we recommend this [tutorial](#), which uses ANTs registration (Advanced Normalization Tools; to learn more see [here](#))

Other tools needed for segmentation

You will also need a spreadsheet file for segmentation notes. This will include all your ROIs for each subject along with helpful notes about landmarks in the brain.

2 Using ITK-SNAP for segmentation

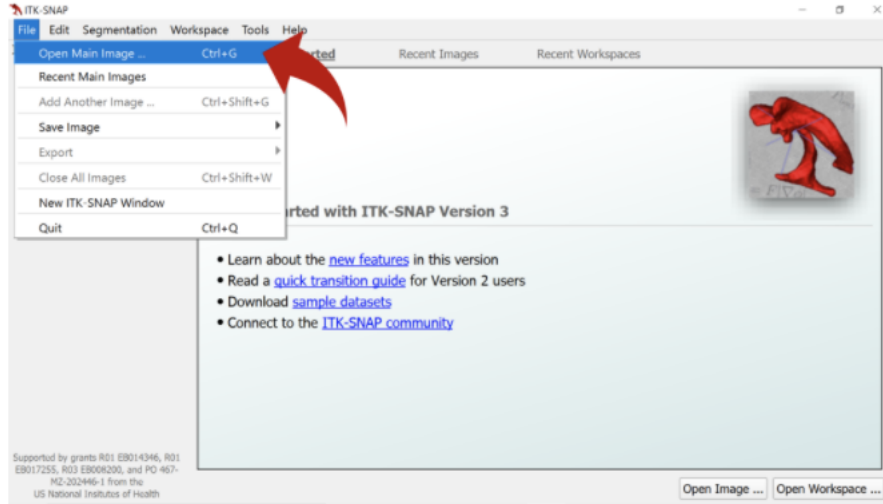
In this section, we review how to open images for segmentation and set up your workspace

STEP 1: DOWNLOADING ITK-SNAP

First, download ITK-SNAP 3.8.0 (click [here](#) here for download link). It is strongly recommended that you install the latest version of ITK-SNAP for segmentation.

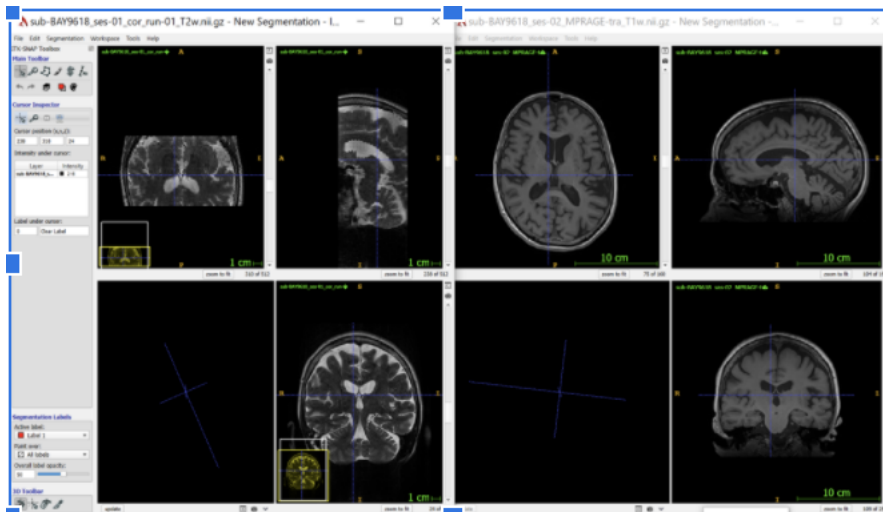
STEP 2: OPENING T2-WEIGHTED IMAGES

In ITK-SNAP, select File from the top menu ribbon, then Open Main Image to open the T2 image for one subject. After selecting the file (.nii/.nii.gz), a pop-up window will open. Select Next and then Finish to open the T2. You should see four panels, with a sagittal, axial, and coronal view, plus an empty panel for 3D rendering.



STEP 3: OPENING T1 IMAGES FOR STRUCTURAL REFERENCE

Next, open the T1 for the same subject as a reference. Select File, then New ITK-Snap Window. In the new window, click File and then Open Main Image to select the T1 file (.nii/.nii.gz). If the T1 and T2 are properly co-registered, the scans should automatically align in the same space.



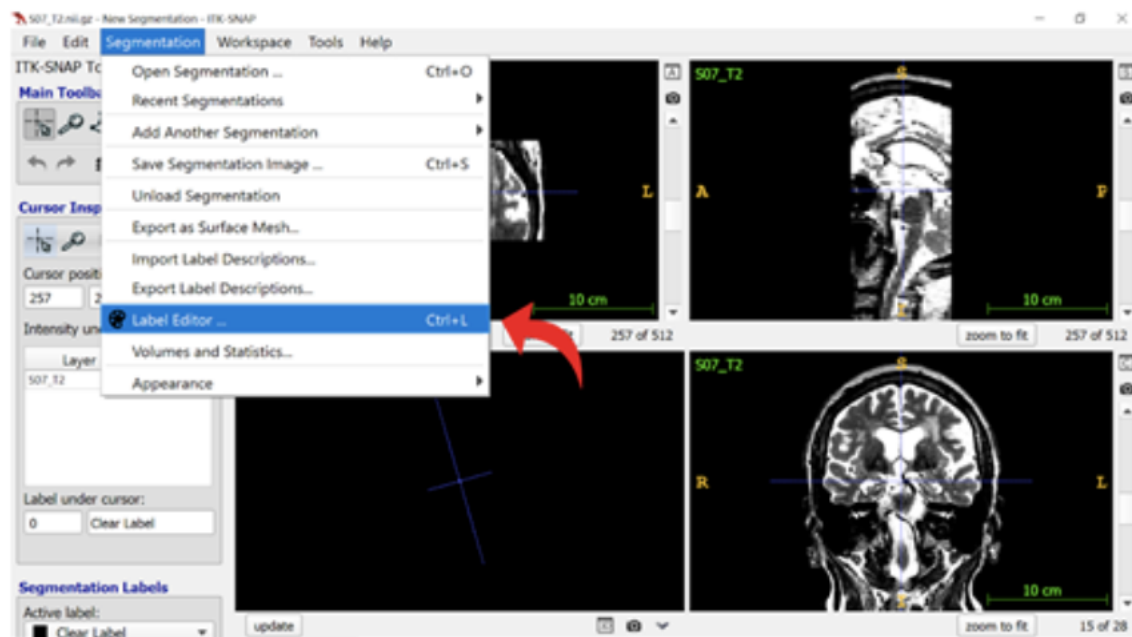
STEP 4: SWITCHING BETWEEN VIEWS

Since segmentation is done primarily on the coronal view of the T2-weighted image, you can easily switch between views (coronal, axial, and sagittal) by clicking Edit in the top menu ribbon and then Views in the dropdown menu to select Next Display Layout (for shortcuts in ITK-Snap, see [here](#)).



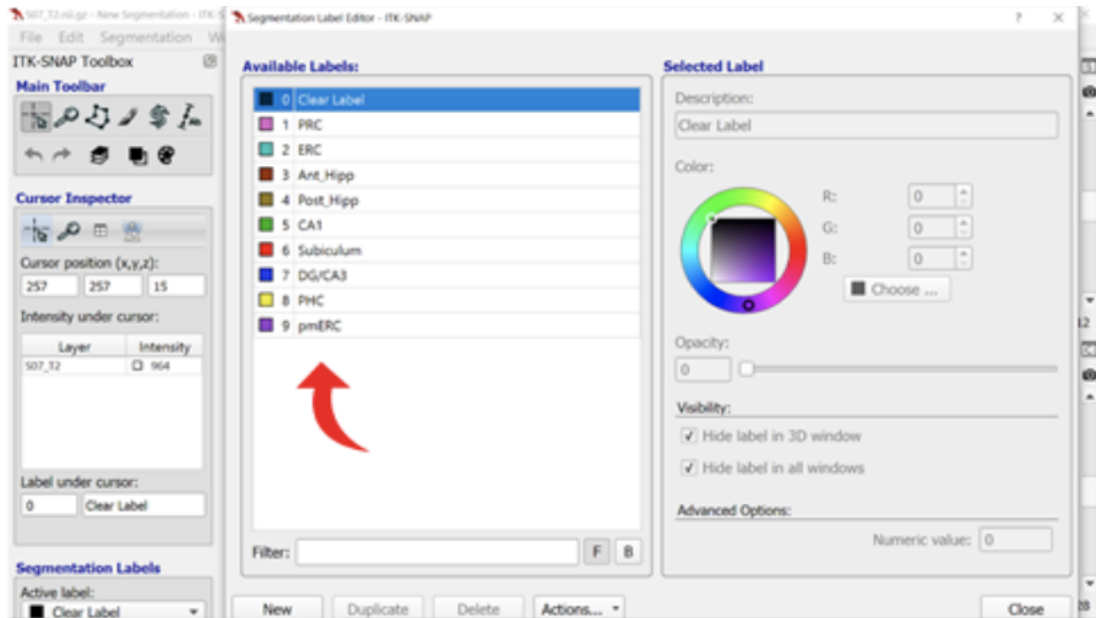
STEP 5: OPENING LABEL EDITOR

On the T2-weighted image ITK-Snap window, select Segmentation on the top menu ribbon and click on Label Editor in the dropdown menu to change your segmentation labels. A pop-up window will open called Segmentation Label Editor.



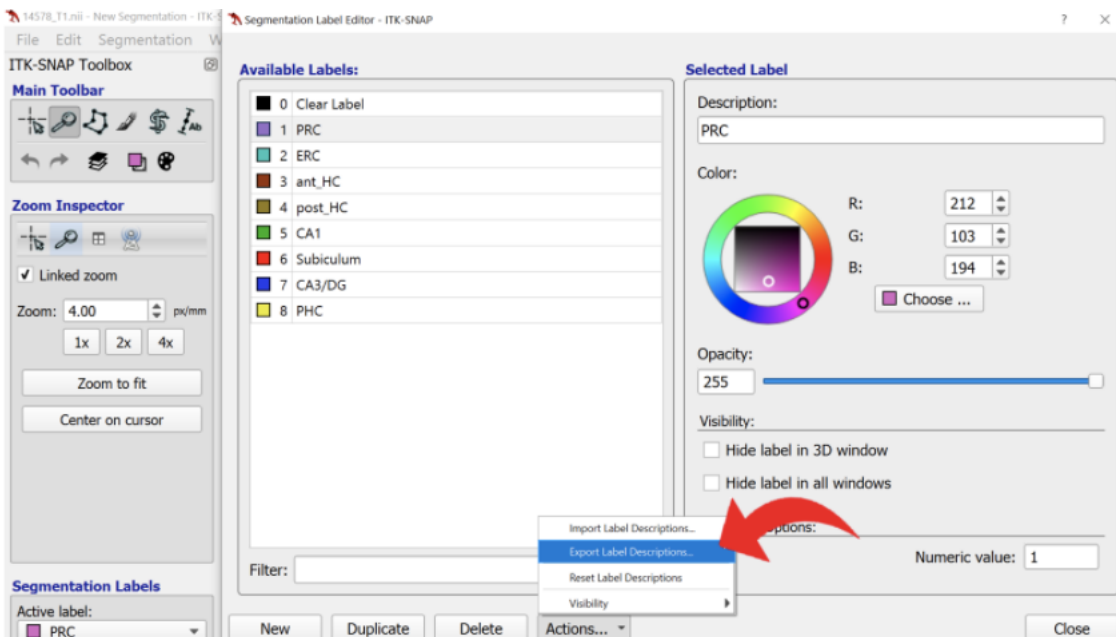
STEP 6: EDITING LABELS

Add the appropriate labels for MTL regions by typing each region's name in the Description field, then selecting the colour for that region. You can learn more about naming conventions and the standard segmentation colours in the OAP protocol in **Labels, Naming Conventions, and Contrasts**.



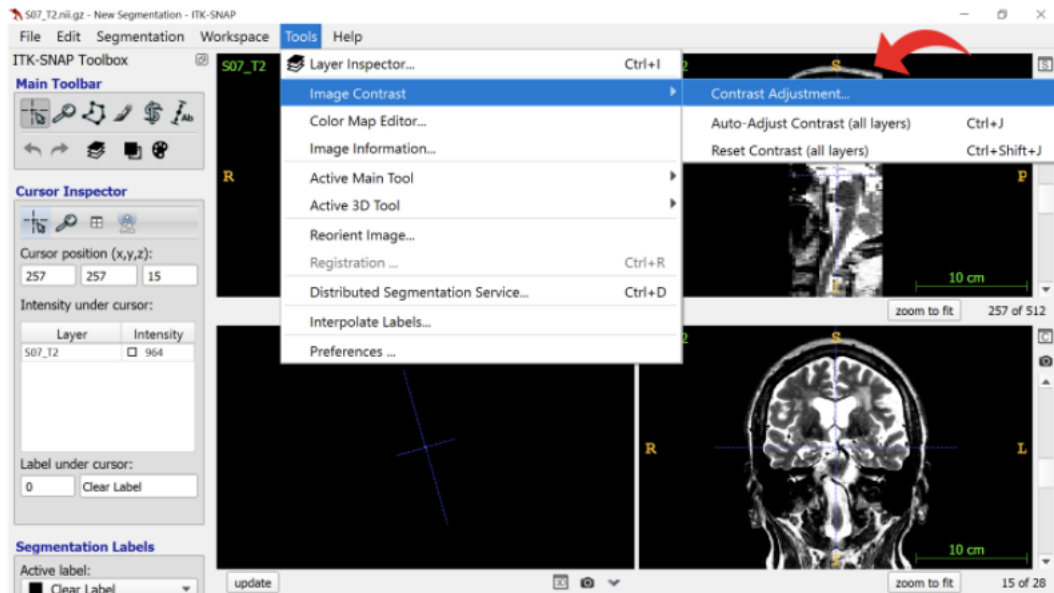
STEP 7: EXPORTING LABELS

To make segmentation easier each time you re-open ITK-SNAP, you can export and save your custom labels, which you can then later re-import. Select Actions... at the bottom of the pop-up window and then choose Export Label Descriptions from the dropdown menu. Save these labels as a text file (.txt). In future sessions using ITK-SNAP, you can simply select Segmentation from the top menu ribbon and then Import Label Descriptions... to re-import this file.



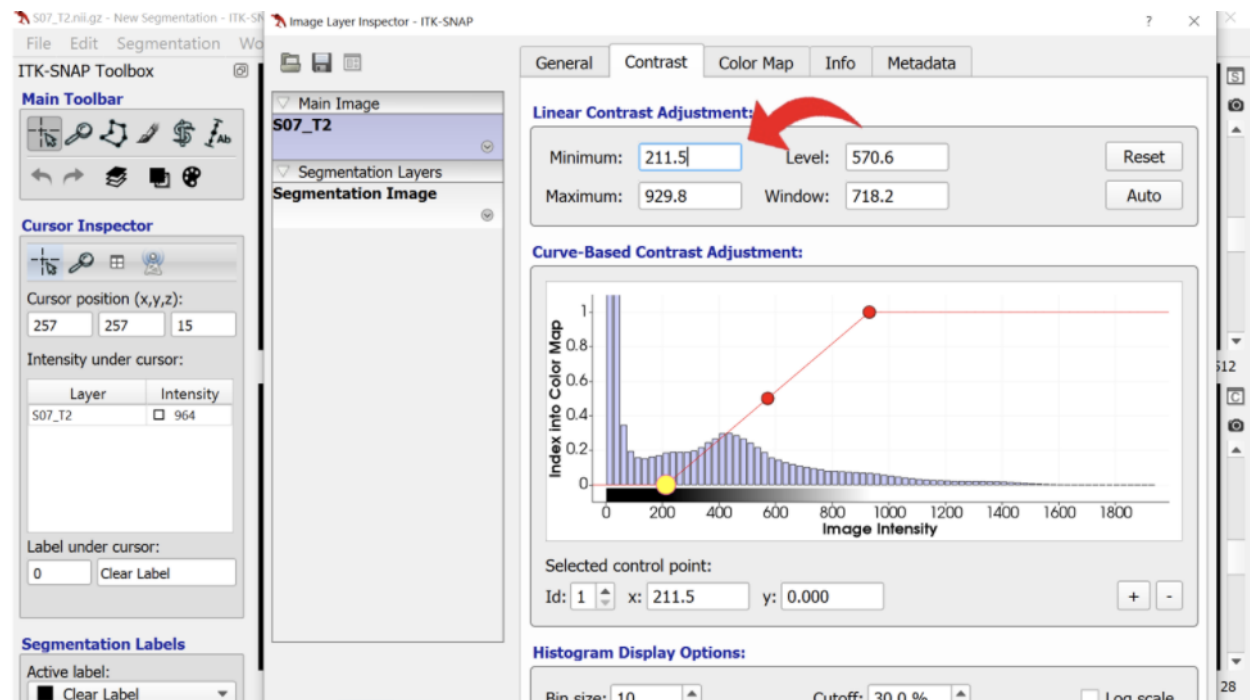
STEP 8: CHANGING IMAGE CONTRAST

To optimize the differentiation of gray matter from white matter in the T2-weighted scan, we can change the contrast levels. Select Tools in the top menu ribbon and click Image Contrast, then Contrast Adjustment.



STEP 9: CHOOSING IMAGE CONTRASTS

In the pop-up window called Image Layer Inspector, you may need to adjust your image contrast to better see certain boundaries. We recommend a minimum contrast at approximately 200 and a maximum contrast at approximately 800 to 900. Be sure to record your contrast selection for each brain in your segmentation notes.

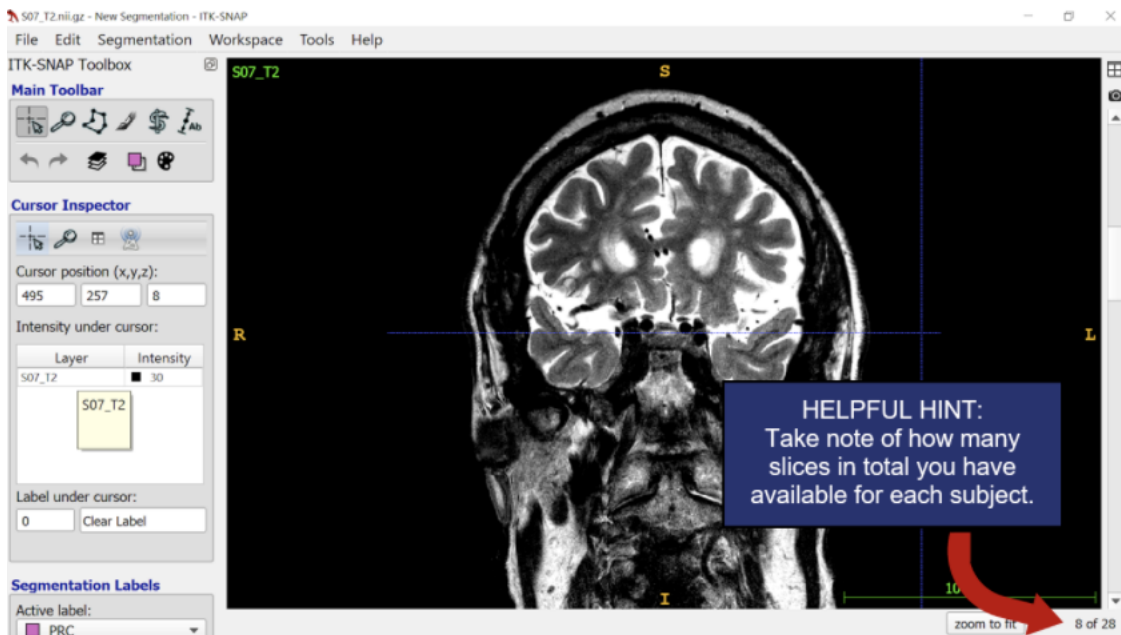


STEP 10: ORIENTATION IN ITK-SNAP

Lastly, before starting segmentation, you should get familiar with the layout of ITK-SNAP. The left side of the screen shows the right hemisphere of the brain, and vice versa. Small letters R (right) and L (left) represent the hemispheres on each side of the coronal view.

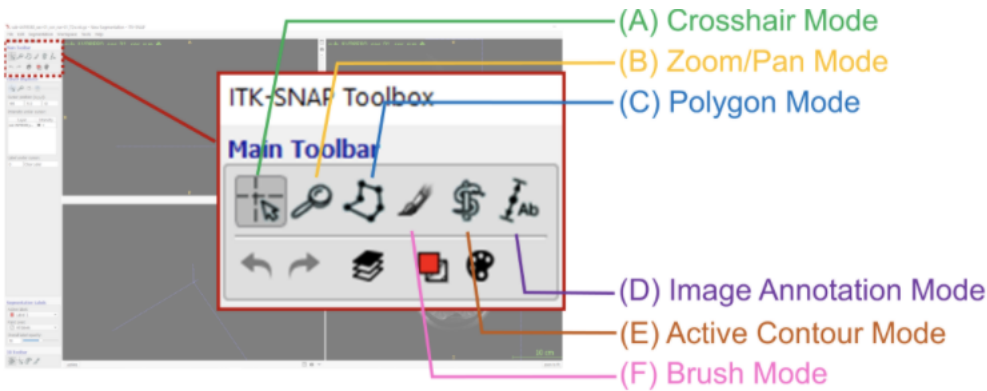


At the bottom right-hand corner of the ITK-SNAP window, you can see the total number of slices, as well as which slice you are currently on. The slice number should increase as you scroll from anterior to posterior in the brain. Make sure to record the total number of slices in your segmentation notes spreadsheet (see Getting Started with Segmentation for more information on the spreadsheet).



STEP 11: TRACING TOOLS AND SAVING SEGMENTATIONS

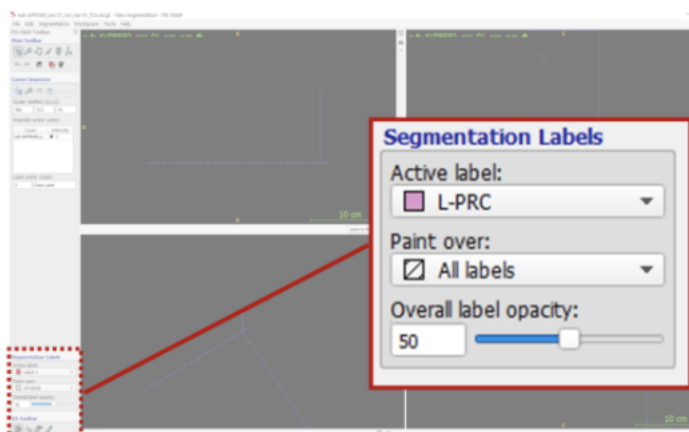
As a final step, let's review the toolbox on the left-hand side panel. See the figure below (Fig. 2.11) for information on how to use the main toolbar



Menu options:

TOOL MODE	DESCRIPTION
(A) Crosshair	See cursor coordinates and move crosshairs to a specific position
(B) Zoom/Pan	Adjust zoom on display and centre zoom on crosshair position
(C) Polygon	Useful outline tool to create shape outlines for tracing
(D) Image Annotation	Measure a region for segmentation depth rules or tag a slice
(E) Active Contour	Used for automatic segmentation (not featured in this manual)
(F) Brush	Used frequently for tracing, allowing freehand drawing on voxels

At the bottom of the toolbox, you can adjust the active segmentation label, its opacity, and whether you want to trace over all voxels or only specific labels.



Finally, to save a segmentation, select Segmentation from the top menu ribbon and then Save Segmentation Image... in the dropdown menu.

3 Conventions_and_Contrasts

ORDER OF TRACING

Draw one ROI in all slices at a time for one hemisphere of the MTL, after you understand the lay of the land (see Getting Started with Segmentation). Move to the other hemisphere only after you have completed all ROIs in the first side. If you are only segmenting hippocampal subfields, then you can skip the PRC and ERC steps to proceed right to the hippocampal head. Otherwise, for the whole MTL, we recommend the following order per hemisphere.

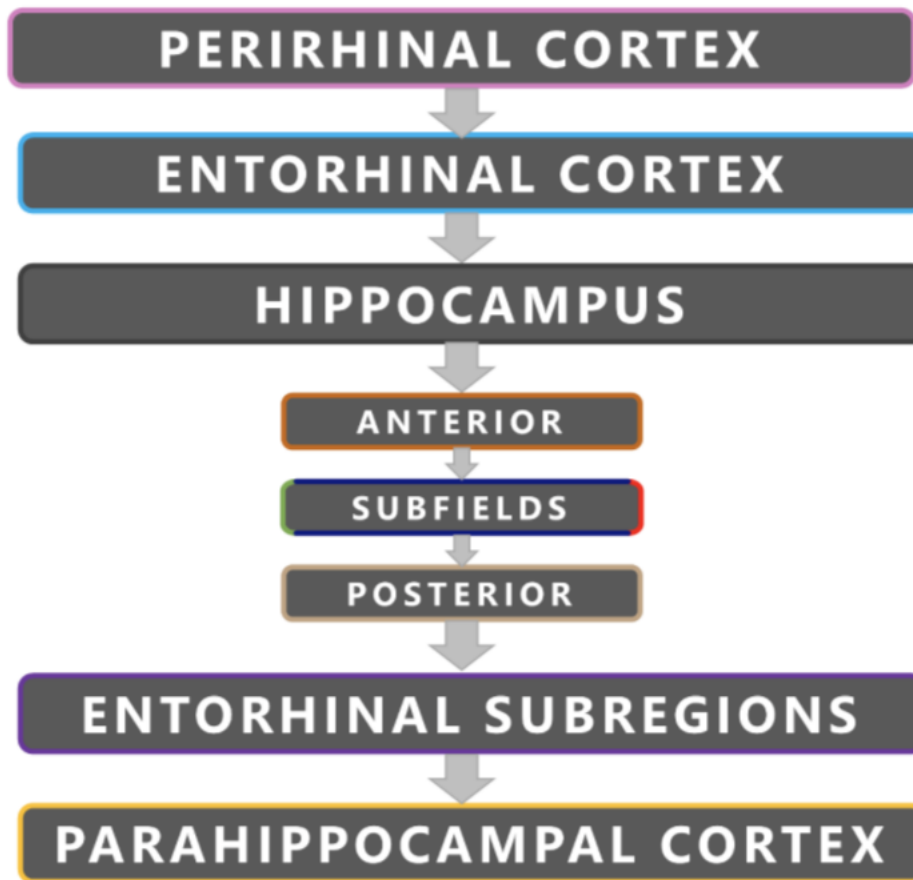


Figure 3.1:Order of segmentation in the medial temporal lobes.

NAMING CONVENTIONS AND LABELS

Naming conventions should include the prefix L- or R- for the hemisphere. If you have more than one segmenter or rater per subject, make sure that these labels and colour values are consistent. See the table below for our recommended labels:

REGION OF INTEREST	LABEL	COLOUR	HEX
Perirhinal Cortex	PRC	Pink	#f791d8
Entorhinal Cortex	ERC	Cyan	#00ccfc
Anterior Head	Ant_Hipp	Orange	#e26213
Posterior Hippocampus	Post_Hipp	Copper	#c3a37f

(continues on next page)

(continued from previous page)

Hippocampal CA1	CA1	Green	#7fc92d
Hippocampal Subiculum	Sub	Red	#ff0000
Hippocampal CA3 + Dentate Gyrus	CA3/DG	Navy Blue	#3915e9
Parahippocampal Cortex	PHC	Yellow	#fff900
Posteromedial Entorhinal Cortex	pmERC	Purple	#994cd3

Table 3: Labelling and colour conventions for the OAP protocol.

REGION OF INTEREST	LABEL	COLOUR	HEX
Perirhinal Cortex	PRC	Pink	#f791d8
Entorhinal Cortex	ERC	Cyan	#00ccfc
Anterior Head	Ant_Hipp	Orange	#e26213
Posterior Hippocampus	Post_Hipp	Copper	#c3a37f
Hippocampal CA1	CA1	Green	#7fc92d
Hippocampal Subiculum	Sub	Red	#ff0000
Hippocampal CA3 + Dentate Gyrus	CA3/DG	Navy Blue	#3915e9
Parahippocampal Cortex	PHC	Yellow	#fff900
Posteromedial Entorhinal Cortex	pmERC	Purple	#994cd3

CONTRASTS

The contrast is set to optimize the differentiation of gray matter from white matter. We recommend a minimum contrast of 200, and a maximum of 800 to 900 on the T2-weighted image. The contrast should be consistent across raters. Generally, you can keep the same contrast throughout all slices of a brain, though you may adjust to better see a structure. For all structures segmented, make sure you record the minimum and maximum contrast in the segmentation notes spreadsheet.

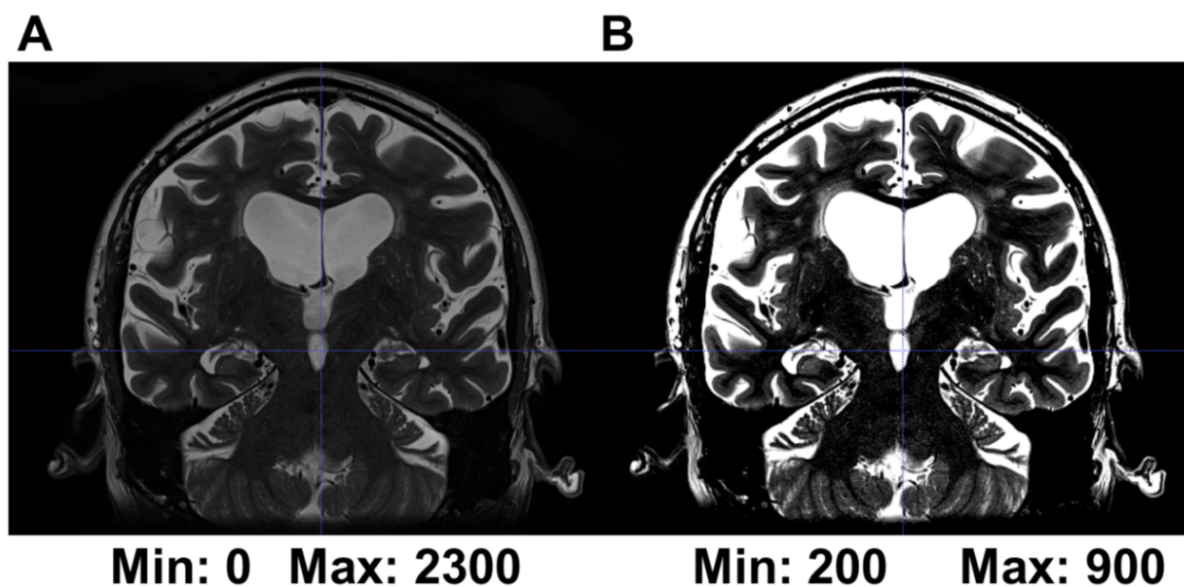


Figure 3.2: (A) Structures are harder to see in a T2-weighted image with default contrast. (B) Structures are clearer in a T2-weighted image when the minimum and maximum contrasts have been adjusted to 200 and 900, respectively.

VOXEL RULES

Cerebral spinal fluid (CSF) will appear on the T2-weighted scan as white voxels. When CSF in the collateral sulcus is greater than 1 voxel, draw around it. When CSF is 0 or 1 voxels wide, include it into the collateral sulcus structure. The voxel rule should also be followed when considering including CSF regions in other regions in the hippocampus. Furthermore, when considering whether to include the lateral border of the ERC (where the ERC climbs up the bank of the CS to meet the PRC), you should also follow the voxel rule and only include the border if it is 1 voxel thick or less.

4 Lay of the land: medial temporal lobes landmarks

With your segmentation notes spreadsheet open in a separate window, start by identifying the landmarks outlined in the following section. This process will allow you to get a “feel” for your particular subject’s anatomy before you actually start segmenting. You will make notes about these landmarks and which slice you identify them in your spreadsheet. It is recommended that you move in an anterior-posterior direction when identifying these landmarks. It is critical that you follow the order outlined in this section, as certain earlier decisions on landmarks will inform later ones. Screenshots with examples are included to help you. Please note, however, that you will need to read the rules for each landmark carefully as your T2 images will certainly vary greatly (see **Variability in Landmarks** for more information).

LANDMARK 1: FIRST SLICE CONTAINING THE COLLATERAL SULCUS

The first slice of the MTL is the first slice in your image set where you can clearly see grey matter ribbon only consists of the perirhinal cortex. The depth of the CS determines is important to identify the CS first. In your spreadsheet, note down the first and most anter example below.

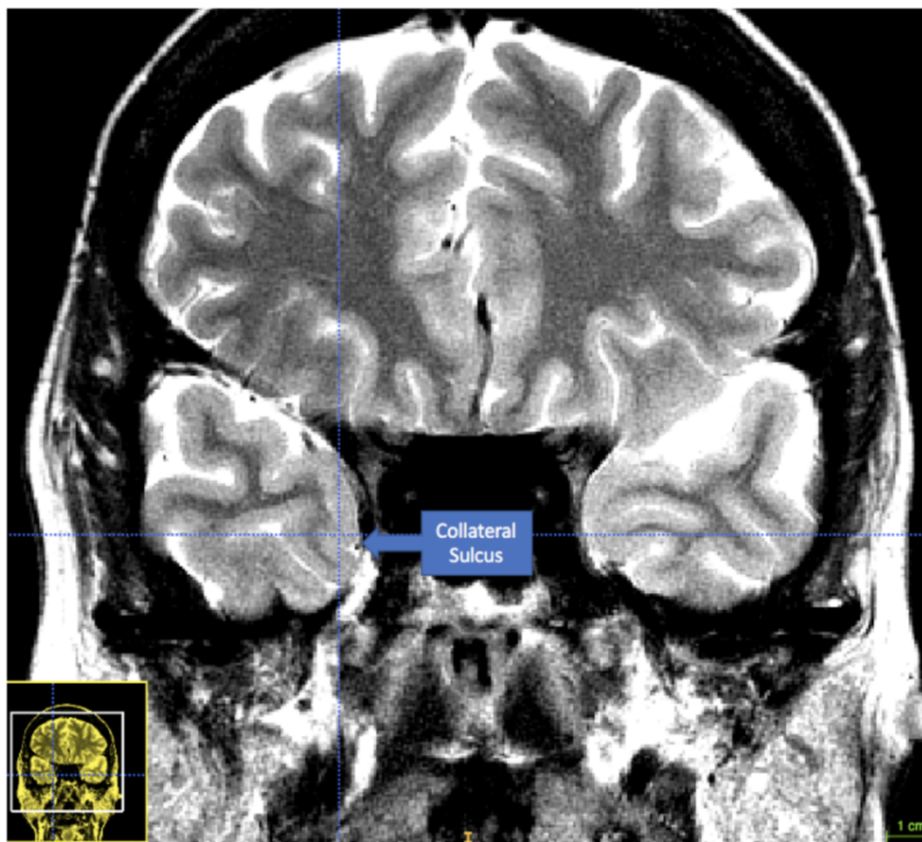


Figure 4.1: The first appearance of the collateral sulcus (CS) in a T2-weighted MR image.

Be careful - in some brains it is easy to confuse a prominent rhinal sulcus (RS) with the Sulcus in Segmenting Regions of Interest in the Medial Temporal Lobes**

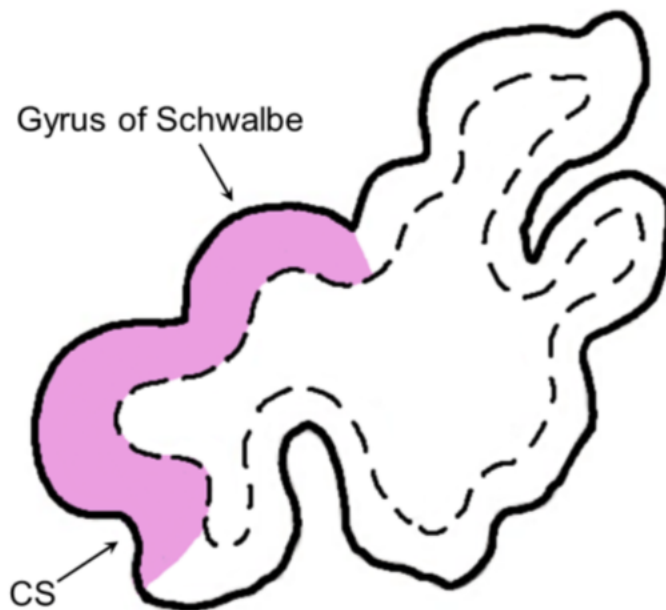


Figure 4.2: According to Insausti et al. (1998), the first appearance of the CS marks the transition between the temporopolar cortex (not included in the OAP protocol) and the perirhinal cortex (shown here in pink). Image adapted from Insausti et al. (1998). CS = collateral sulcus

LANDMARK 2: THE FRONTAL-TEMPORAL JUNCTION/LIMEN INSULAE.

This key landmark will determine where you start drawing the entorhinal cortex (ERC). To find this landmark, look for the frontal-temporal junction (FTJ)/limen insulae. The slice in which there is a clear band of white matter that joins the frontal lobe to the temporal lobe is the slice in which the FTJ/limen insulae is indicated. It is sometimes easier to visualize this landmark on a T1-weighted scan.

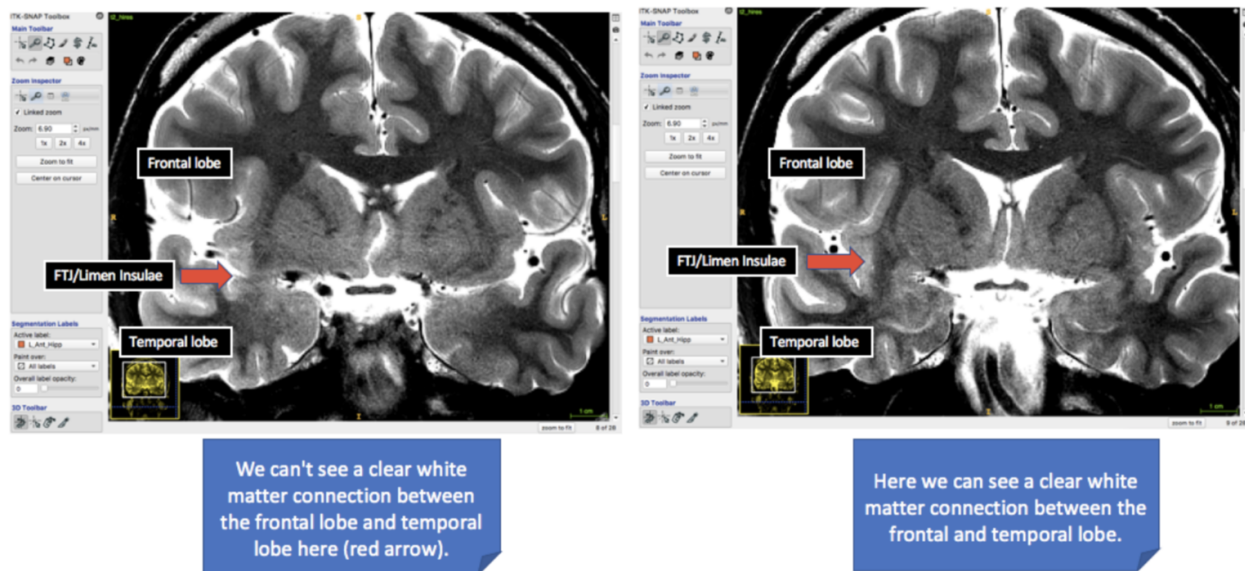


Figure 4.3: The two examples here compare a slice (T2-weighted) in which the limen insulae grey matter is visible in the left image; however, there is no clear white matter connection between the temporal and frontal lobe yet, versus in right image there is a clear connection between the white matter of the frontal and temporal lobe. This is a clear indication of the presence of the FTJ/limen insula, which will determine the delineation of the entorhinal cortex. See the next image below for another example.

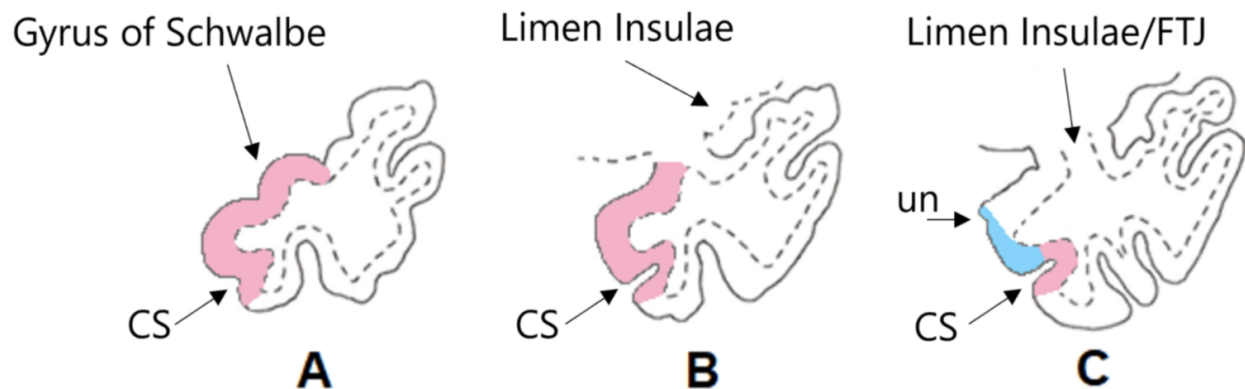


Figure 4.4: Adapted from Insausti et al. (1998). (A) The emergence of CS and gyrus of Schwalbe (with PRC depicted in pink), (B) Moving posteriorly, the limen insula is now present but the white matter connection between the frontal and temporal lobe is not clear, so we only continue to trace the PRC. (C) There is a clear connection between the white matter of the frontal lobe and temporal lobe. This is the slice in which you draw the ERC (depicted in blue) from the PRC up to the uncus notch (un). This is also the slice in which you make a note in your spreadsheet for the presence of the FTJ/limen insula. Boundaries based on Kivisaari et al. (2013), see **Helpful Additional Resources for Further Reading**.

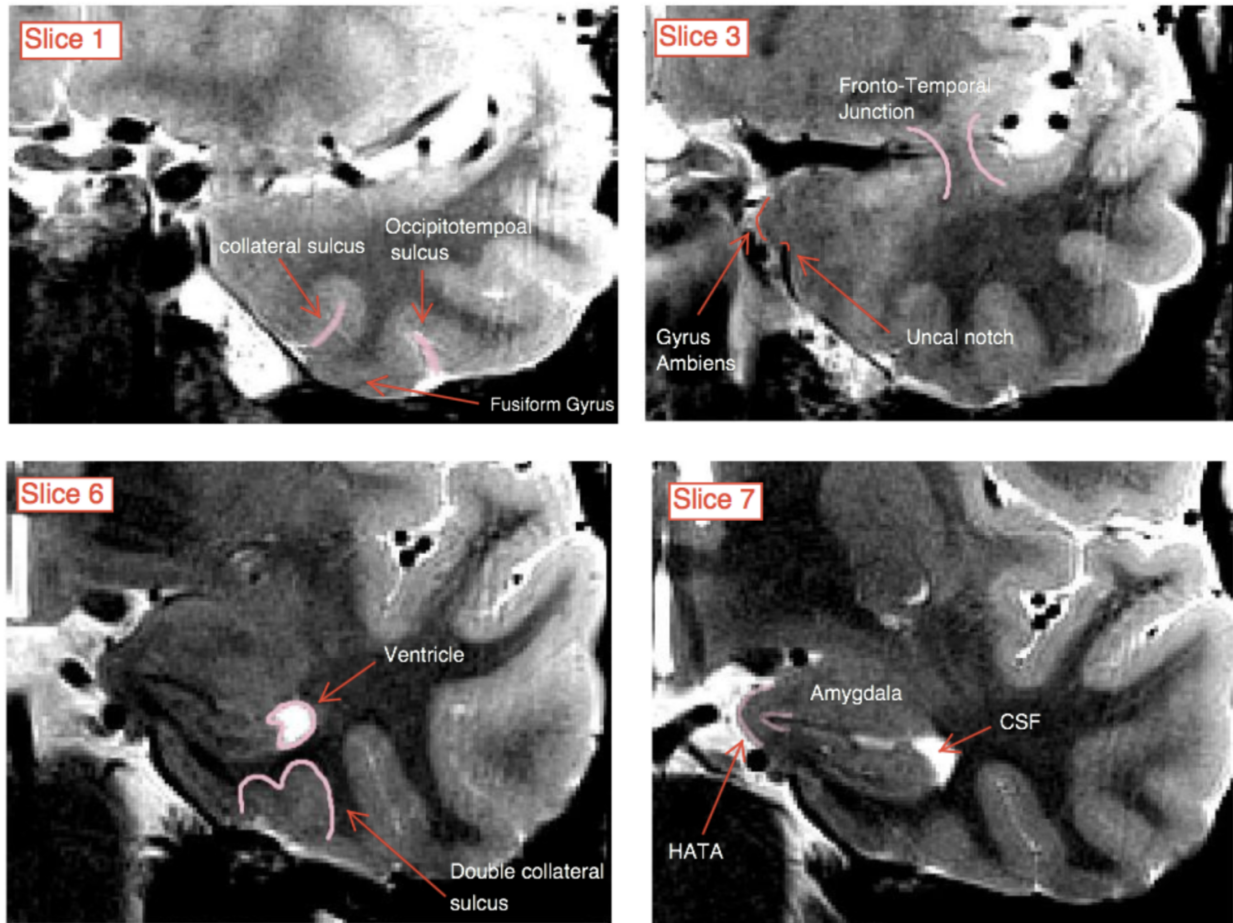


Figure 4.5: This image brings together the first two landmarks (CS and FTJ/limen insulae) as described above. As you move from anterior to posterior, the CS may change from a single to double CS.

LANDMARK 3: THE FIRST SLICE CONTAINING VISIBLE HIPPOCAMPAL HEAD

Next, you will need to look for the hippocampal head. To find this landmark:

- A In its first appearance, the hippocampal head will probably look like a “bean” shape
- B The amygdala is located superior and the ventricle is lateral to the hippocampal head
- C Ambient gyrus appears in the same slice as the appearance of the hippocampal head

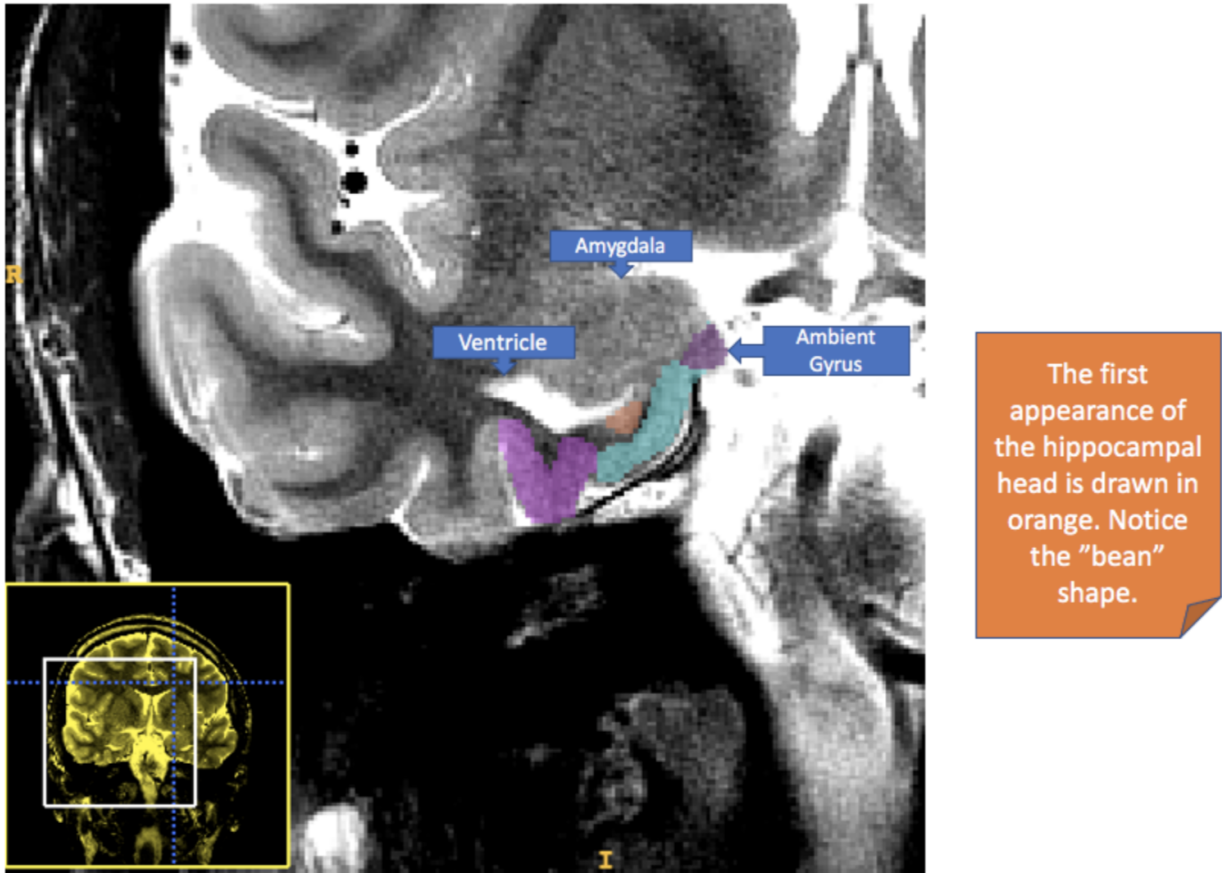


Figure 4.6: This image depicts the first appearance of the hippocampal head (shown in orange). Notice the ventricle laterally, and the ambient gyrus medially.

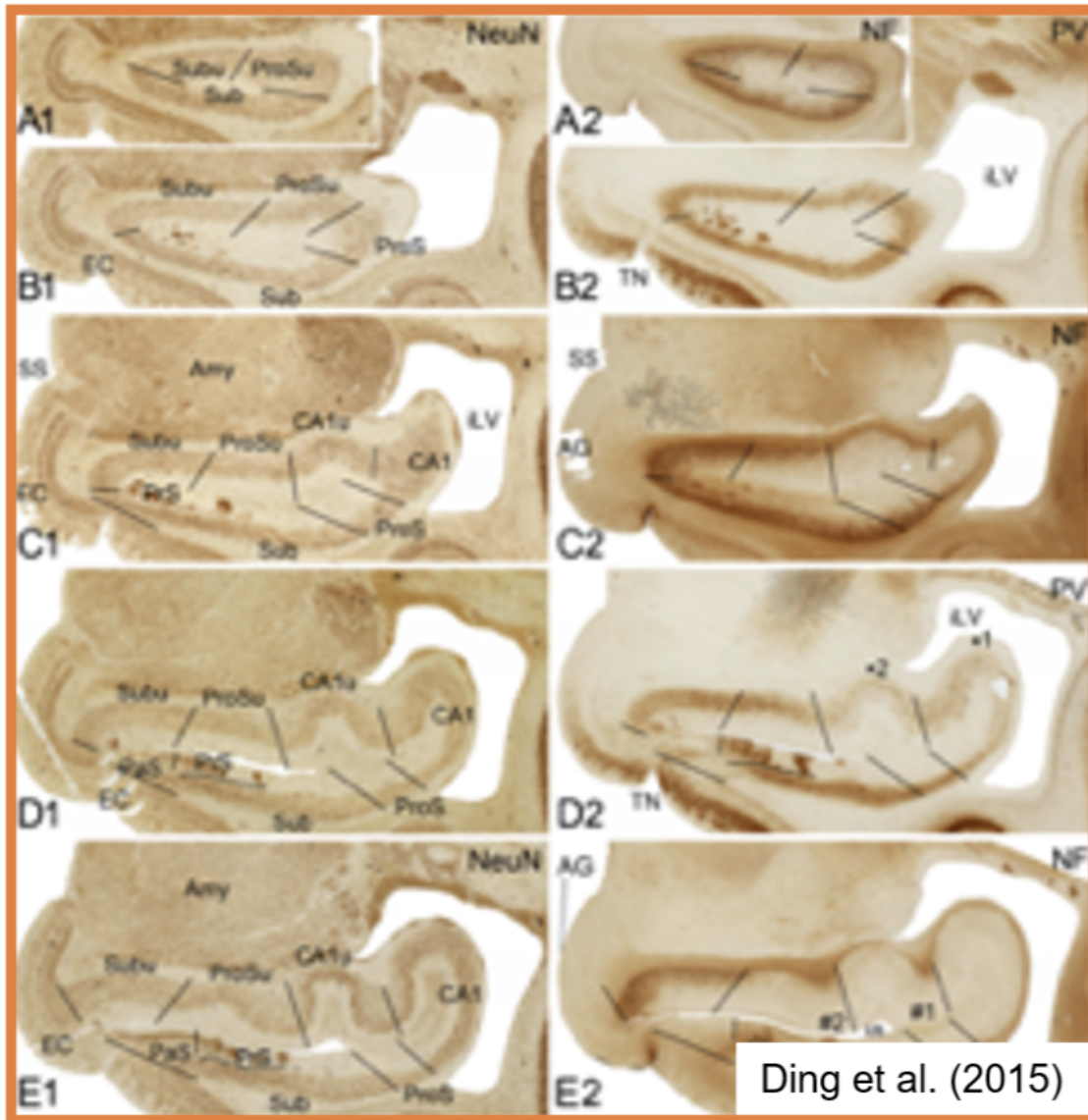


Figure 4.7: This image will help you determine the shape of the hippocampal head in the brain you are segmenting. The example shown is adapted from Ding et al. (2015). Notice how the head shape can resemble a “bean” (A1, A2, B1, B2) or more like the hippocampal body (C1, C2).

LANDMARK 4: THE FIRST SLICE CONTAINING DENTATE GYRUS

After identifying the hippocampal head on 2-3 slices (depending on the brain you are segmenting and the quality of the T2 scan) you will start to see subfields of the hippocampus. At this point, the hippocampus will look thicker than previous slices and the superior digitations of the hippocampus will have smoothed out. This is the first slice of the dentate gyrus (DG) and, by extension, other subfields of the hippocampus. Finally, a darker C-shaped band should be visible, separating hippocampal cornu ammonis area 1 (CA1) from DG. Note that in the OAP protocol, we do not distinguish between DG and cornu ammonis area 3 (CA3).

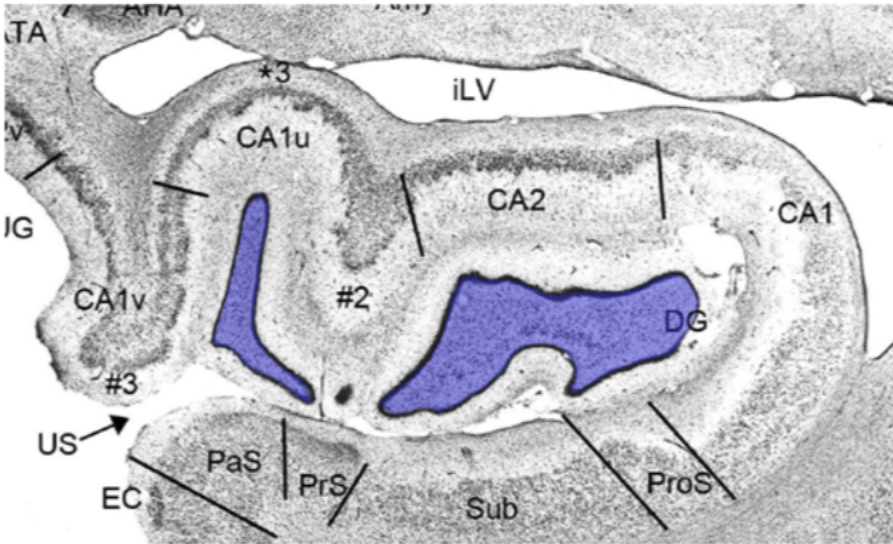


Figure 4.8: The image above, adapted from Ding et al. (2015), will help you with identifying dentate gyrus (highlighted in blue).

LANDMARK 5: THE LAST SLICE CONTAINING THE UNCUS

The last slice of the uncus in the image below would be the second box from the left. You should note here that this EC/PRC to PHC transition is valid for 2-3mm thick slices. For thinner slices, there will be more slices in between the uncus apex and the start of the PHC (Pruessner et al. (2000) suggests it starts 5mm posterior to the uncus apex).

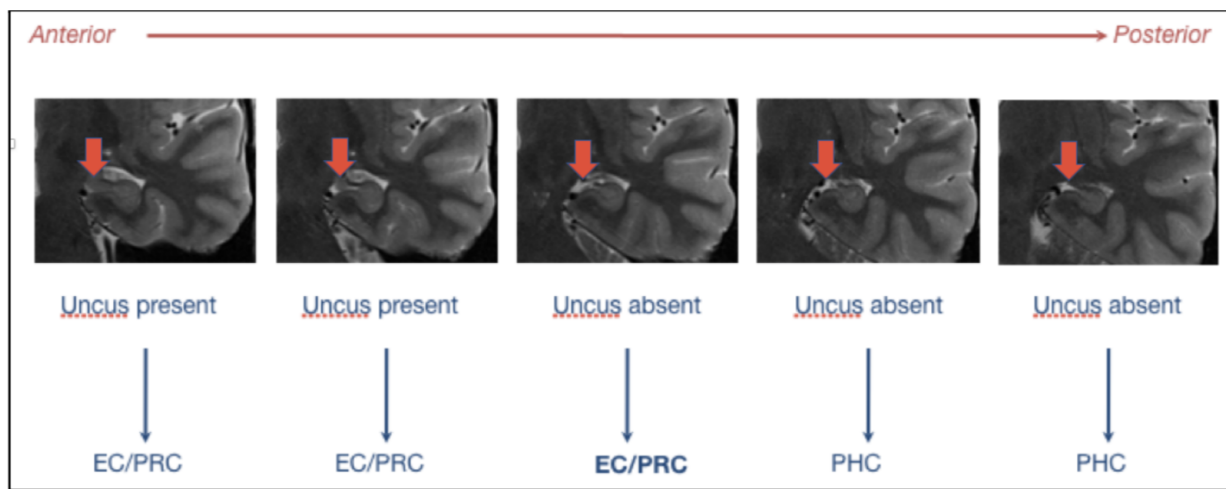


Figure 4.9: Anterior to posterior cortical transition showing the final slice containing the uncus. After one slice where

the uncus is absent, you can start tracing the PHC, and the ERC/PRC disappears. Image adapted from: Carr, V.A. (2013), Variability in collateral sulcus anatomy: The challenge of reliably segmenting medial temporal lobe cortices. Hippocampal Subfield Segmentation Summit, Davis: Oral presentation.

LANDMARK 6: THE LAST APPEARANCE OF THE COLLICULI

The last clear appearance of the colliculi is the final slice where we segment the hippocampal subfields. After this slice, the hippocampus transitions to the tail segment.



Figure 4.10: The final appearance of the colliculi, which resemble a “butterfly” shape in the centre of the brain.

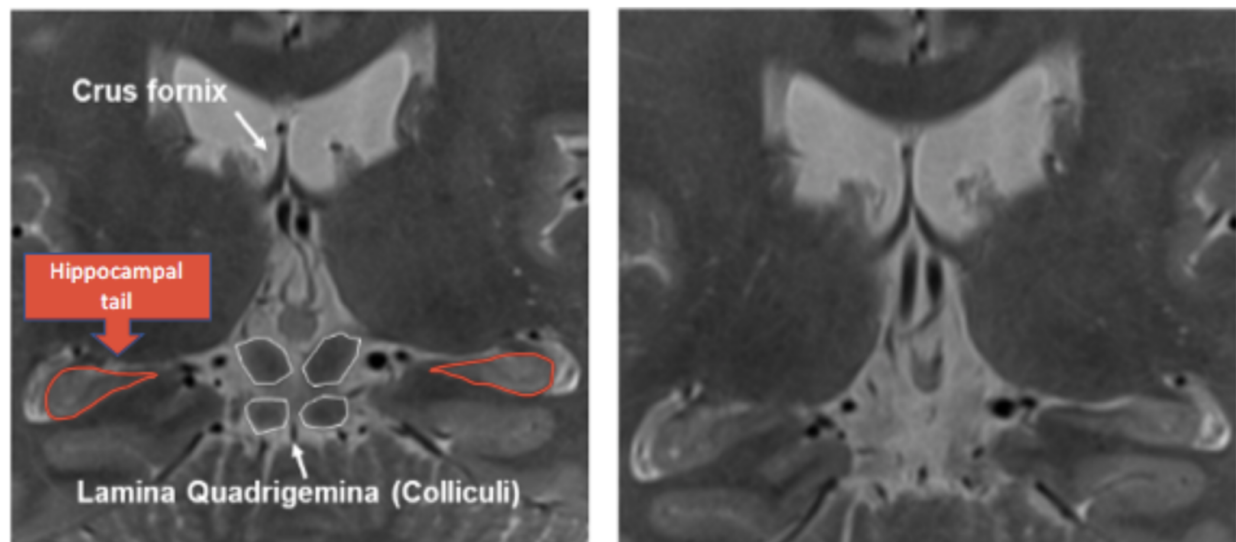


Figure 4.11: On the left, the final posterior slice of the hippocampal body, containing the colliculi, crus fornix, and the “tear drop” shape of the hippocampal body. On the right, the colliculi are no longer visible, making the first slice of the hippocampal tail.

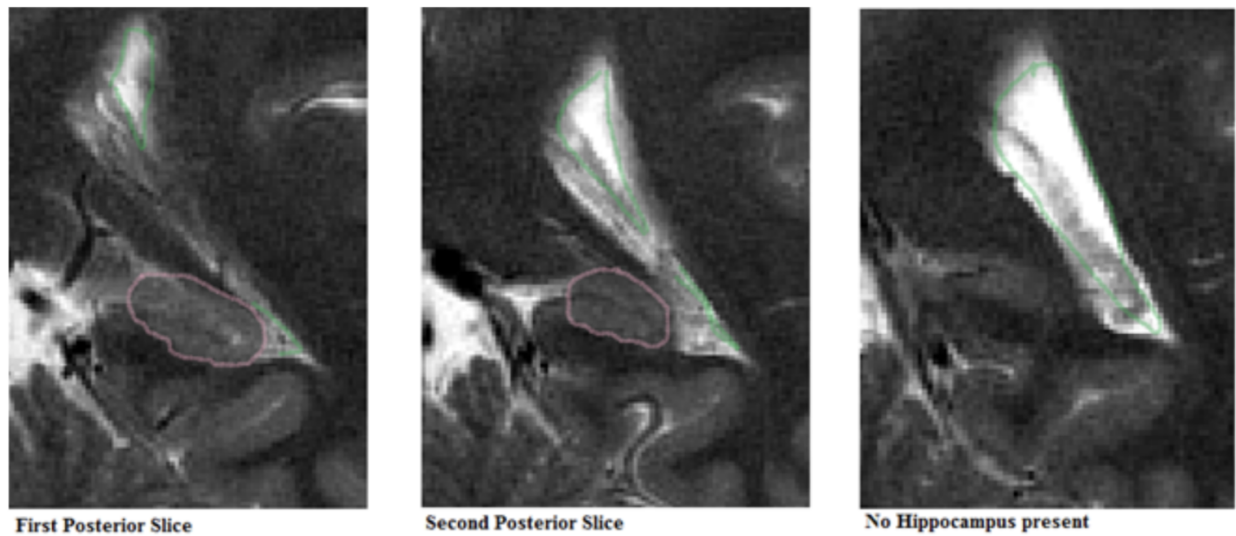
LANDMARK 7: THE LAST SLICE WHERE THE HIPPOCAMPAL TAIL IS VISIBLE

Figure 4.12: The “sweeping” of CSF towards the superior ventricle means that the hippocampal tail is no longer present in posterior slices.

The last slice of the MTL is the slice in your image set where you can clearly see the grey matter portion of the hippocampus tail. After the last slice of the MTL the bright CSF laterally to the hippocampus will clearly sweep up and meet up with the more superior ventricle.

5 Segmenting the Medial Temporal Lobes**Perirhinal Cortex**

The perirhinal cortex (PRC) is drawn in pink on all anterior slices. The first slice of PRC is drawn on the first slice in which the collateral sulcus (CS) is present. This is the most anterior slice of the MTL. In slices prior to the limen insulae/FTJ, the superior border is determined by the gyrus of Schwalbe (Figure 5.1). In slices after the limen insulae, the lateral border is determined by the depth of the CS (Figure 5.2).

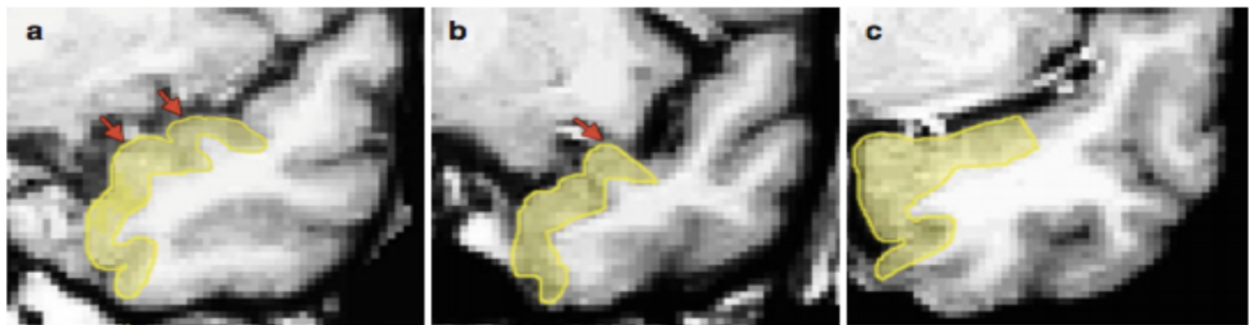


Figure 5.1: Image adapted from Kivisaari et al. (2013) showing the superior-lateral border of the PRC, based on the presence of the gyrus of Schwalbe. (a) Two gyri of Schwalbe are visible (red arrows) so the superior-lateral PRC border is drawn to the most lateral fundus. Do not include a bump that goes beyond the midpoint of the MTL as a gyrus of Schwalbe. (b) One gyrus of Schwalbe is visible, so the superior border is drawn to the fundus of the gyrus. (c) No gyri are visible, and the superior aspect of the temporal pole appears flat, so the superior-lateral border is drawn to the midpoint of the entire superior portion of the MTL.

If you are having trouble determining the midpoint of the MTL (i.e., the superior-lateral PRC border prior to the limen insulae/FTJ), you can flip forward through the slices until you find the FTJ/limen insulae. On slices before that one (i.e., more posteriorly), the superior-lateral border of the PRC should not extend past where the FTJ appears in more anterior slices.

After the limen insulae/FTJ appears, the medial border of the PRC is determined by the depth of the CS. See Figure 5.2 and Table 5A below for a visual of the CS depth rules.

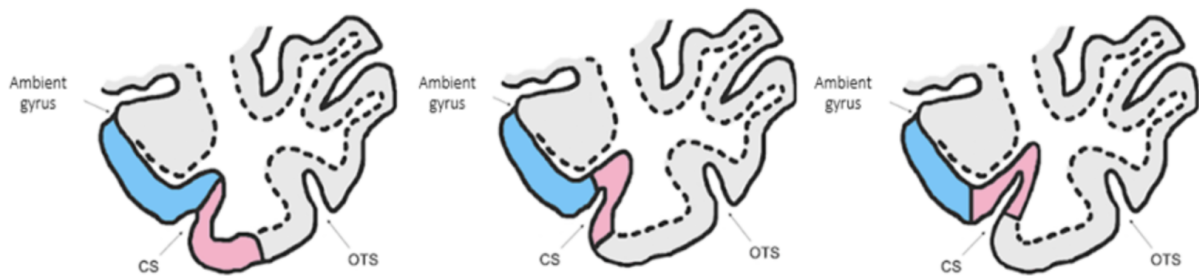


Figure 5.2: These examples of the collateral sulcus are adapted from Insausti et al. (1998): CS = collateral sulcus, OTS = occipitotemporal sulcus, blue = entorhinal cortex, pink = perirhinal cortex

Collateral Sulcus

The depth of the collateral sulcus (CS) determines where the medial and lateral border of your PRC. The depth rule only applies to PRC, as shown in Table 5A below (but does not apply to PHC). However, the double CS rule **does** apply to the PHC as well.

	SHALLOW	REGULAR	DEEP
MEASUREMENT	<1 cm	1 – 1.5 cm	>1.5 cm
FREQUENCY	~16%	~82%	~2%
LATERAL EXTENT	Half-way across fusiform gyrus	Lateral “elbow” of CS	Half-way up lateral side of CS
MEDIAL EXTENT	Fundus (tip) of CS	Half-way up the medial side CS	Medial “elbow” of CS
EXAMPLE			

Table 5A: The following table will help you determine the depth of the collateral sulcus in the brain you are segmenting. Note that the images above depict the left hemisphere of the brain. However, the same rules apply for the right hemisphere. Examples are adapted from Insausti et al. (1998; see **Helpful Additional Resources for Further Reading**).

In the case that there is a double CS, draw the medial edge to the fundus of the more medial CS and draw the lateral to the fundus of the more lateral CS. An estimated 25% to 35% of subjects will have a double CS (Insausti et al., 1998; Pruessner et al., 2002). Not all slices in one subject may be a double CS. In the figure below, B and C demonstrate an interrupted and a branched CS.



Figure 5.3: The variation seen in the collateral sulcus. The collateral sulcus can be uninterrupted, non-side-branched as in the first image, interrupted as in the second image and side branched as in the third image. (Perirhinal cortex = pink, Entorhinal cortex = blue).

Note that PRC bifurcation rules still apply before the limen insulae/FTJ - so if the CS bifurcates, draw to the fundus of the more lateral sulcus. However, the medial boundary rule (depth rule) would not apply for the medial sulcus at this point. So, if you are drawing the CS as a double before the limen insulae/FTJ, include the PRC all the way to the superior lateral extent of the gyrus of Schwalbe (but, see **Variability in Landmarks** for examples where the PRC may be interrupted by an irregular CS or deep rhinal sulcus before the limen insulae/FTJ).

Entorhinal Cortex

The entorhinal cortex (ERC) is drawn in **light blue** on all anterior slices. The first slice containing the ERC will be the first slice where the limen insulae/FTJ is present. There are two distinct ways to draw the ERC: in slices before the HPC anterior head, and in slices after the start of the HPC.

In the slices before the anterior head:

The medial extent is drawn along the grey matter ribbon to the natural taper point in the bisection of the apex of the most medial point of parahippocampal gyrus (see **Glossary of Key Terms**). The lateral edge is drawn to the edge of the PRC (see image below).

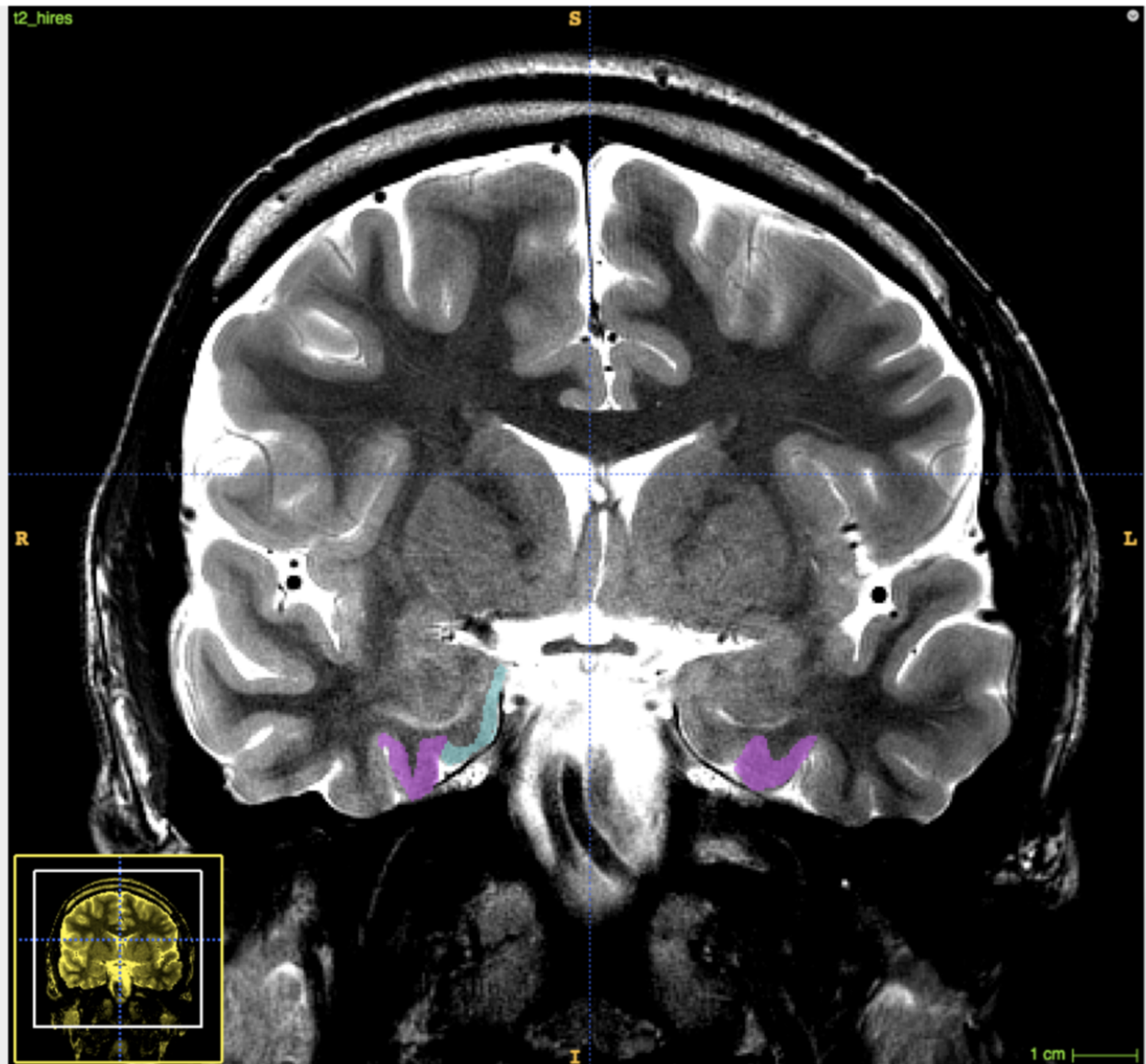


Figure 5.4: Segmentation of the entorhinal cortex (blue) in slices before the anterior head of the HPC, with the lateral edge drawn to the edge of the PRC (pink).

In the slices after the start of the hippocampus

The medial extent is drawn along the grey matter ribbon to the bisection of the hippocampal fissure. In more posterior slices, this will meet up with the subiculum. The lateral edge is drawn to the edge of PRC (see image below).



Figure 5.5: Segmentation of the entorhinal cortex (blue) in slices after the start of the hippocampus, with the lateral edge drawn to the edge of the PRC (pink)

Parahippocampal Cortex

The parahippocampal cortex (PHC) is drawn in **yellow**, on the posterior slices only. The PRC follows the grey matter ribbon connecting the subiculum to the collateral sulcus. Start drawing PHC on the second slice the uncus is absent (on the same slice where you start drawing CA1 phase 3). The medial extent is therefore the edge of the subiculum, and the lateral edge is drawn to the elbow of the collateral sulcus (collateral sulcus depth rules do not apply in the posterior hippocampus).

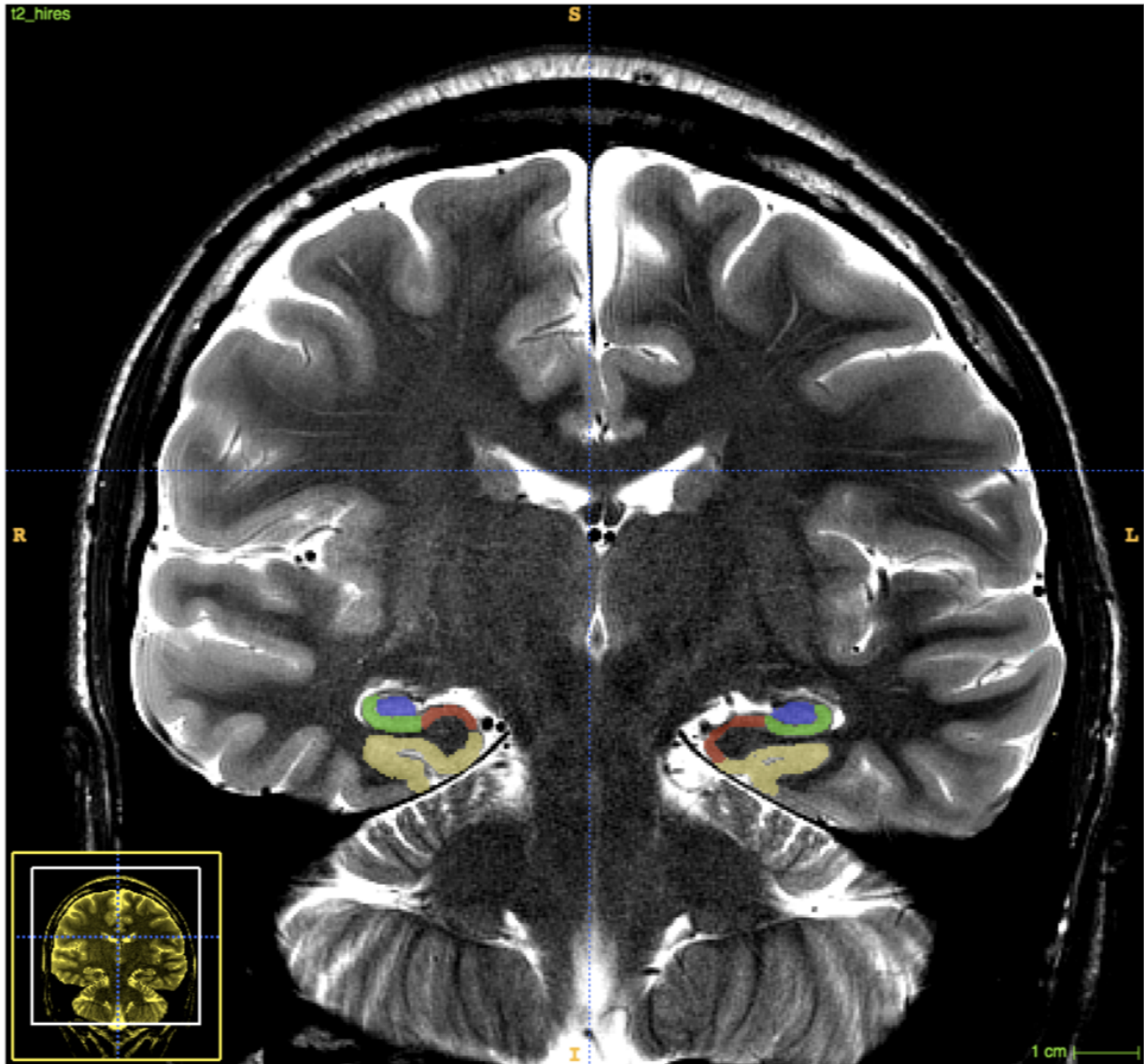


Figure 5.6: Segmentation of the parahippocampal cortex (yellow) in posterior slices of the MTL cortex.

If there is a double sulcus and the medial sulci and the lateral sulci originate from the same collateral sulcus, then follow the bifurcation rules. If the medial and lateral sulci do not come from the same collateral sulcus or do not merge in subsequent slices, then only draw the lateral sulci and do not include the medial sulci in the ROI at all. If a small medial sulcus appears (usually before the appearance of the calcarine sulcus) that later merges with the collateral sulcus (the lateral sulcus in this case), then include both in the ROI. Always draw the medial sulcus as a whole when the collateral bifurcates (unlike PRC bifurcation rules). In the case there is a calcarine sulcus in the posterior regions of the MTL, draw the subiculum to its natural taper point, and include only the collateral sulcus in the parahippocampal cortex.

6 Segmenting Hippocampal Subfields

In the OAP protocol, the hippocampus (HPC) is segmented into the anterior (head), the body (with subfields), and the posterior (tail) regions. The figure below should help you with visualizing the way that subfields (i.e., subiculum, CA1, and DG+CA3) are arranged along the horizontal long axis of the HPC.

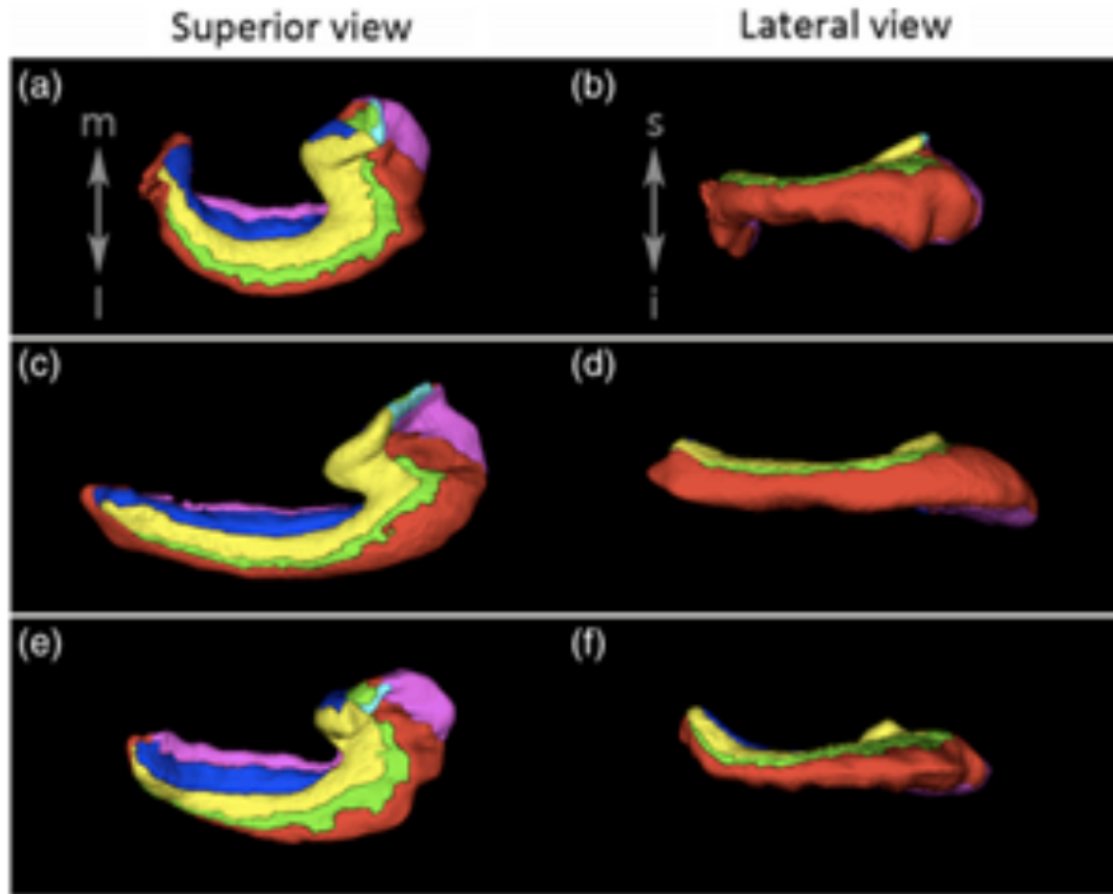


Figure 6.1: Illustration of the curvature of the tail in three hippocampi. In the top row (a, b) the tail shows a strong curve in the medial direction. In the middle row (c, d) the tail shows very little curving in either direction. In the third row (e, f) the tail shows limited curving in the medial direction, and stronger curving in the superior direction visible in the lateral view. Image adapted from de Flores et al. (2020).

Anterior Head

The anterior head (which is also sometimes referred to as the hippocampal head) is defined by the part of the HPC that falls anterior to the first appearance of the dentate gyrus. The OAP protocol does not differentiate this ROI into the various subfields. The most posterior part of this ROI resembles Figure 23.15.C in Insausti & Amaral's (2004) book, chapter 23, which we have reproduced here below. A key feature to look for is the “bumps” or “digitations” on the superior portion of the HPC without a very clear stratum radiatum lacunosum moleculare (SRLM). It is important to note that the digitations are not always completely visible.

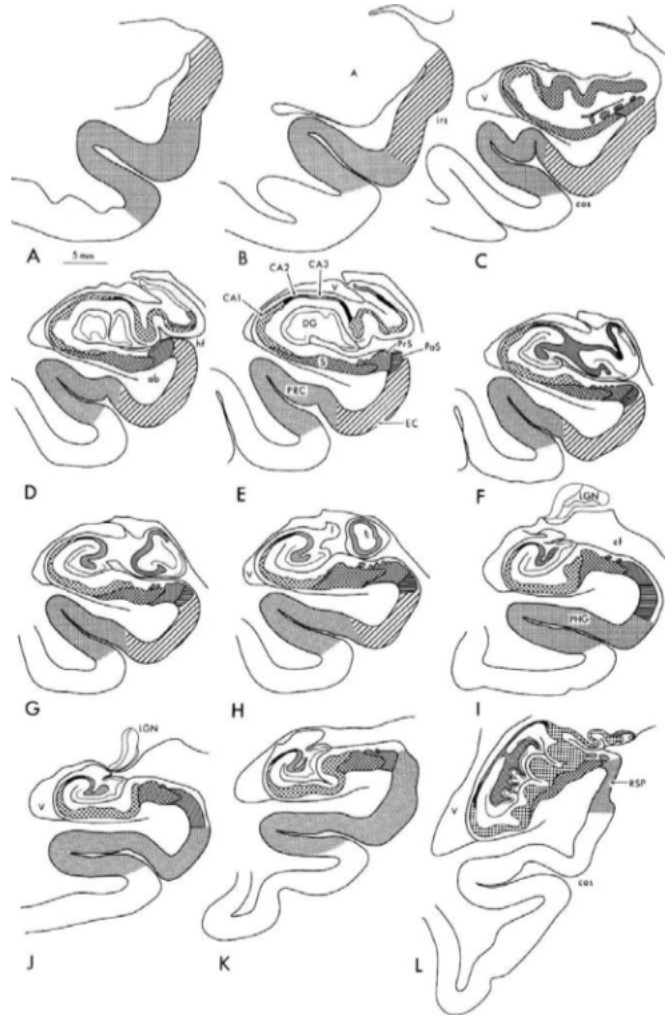


Figure 6.2: Reproduced image from Insausti & Amaral (2004; chapter 23) of the anterior head progressing to the body.

Keep in mind that in images with 2-3mm slice thickness the anterior head usually spans 3 coronal slices, sometimes 2 and rarely 1. In higher resolution images, there can be more than 3 slices of anterior head. The anterior head will fall in the predetermined lateral, superior, and inferior borders. If it is touching the amygdala, however, the medial extent will be drawn half way up the hippocampal-amygdala transition area (HATA). Only draw this ROI until morphology of the HPC includes evidence of the DG in the center of the HPC, and a visible C-shape on the lateral edge, at which point you can start segmenting subfields. It is highly recommended that you look at the Ding et al. (2015) paper (see **Helpful Additional Resources for Further Reading**) when learning how to draw the HPC head.



Figure 6.3: Detailed images of the hippocampal head (anterior head).

Figure 6.3: Detailed images of the hippocampal head (anterior head).

Posterior Hippocampus

The posterior HPC (sometimes referred to as the hippocampal tail) is where the posterior thalamus and caudate meet up with the HPC. At this point along the long-axis, the subfields can no longer be reliably segmented. In some brains, there will be “bumps” that can be visualized on the inferior portion of the body (compared to the hippocampal head that had them on the superior portion). The last slice containing the colliculi (butterfly shape in the centre of the brain) is the last slice the hippocampal subfields are drawn (see **Landmark 6 in Lay of the Land: Medial Temporal Lobes Landmarks** above). Start drawing the posterior HPC on the first slice that the colliculi are no longer present.

On 3mm thick slices, the posterior HPC usually spans 2 slices, sometimes 1 and rarely 3. However, on thinner slices you will likely have up to 5 slices with the posterior HPC present. The posterior slice will have a clear difference of the grey matter region between it and the slice one posterior to it (see **Landmark 7 in Lay of the Land: Medial Temporal Lobes Landmarks** above). This will also be the last slice of the MTL, where the bright CSF laterally to the HPC will “sweep up” to meet up with the more superior ventricle. This image below will help you find the final slice of the hippocampal body before you start segmenting the posterior HPC

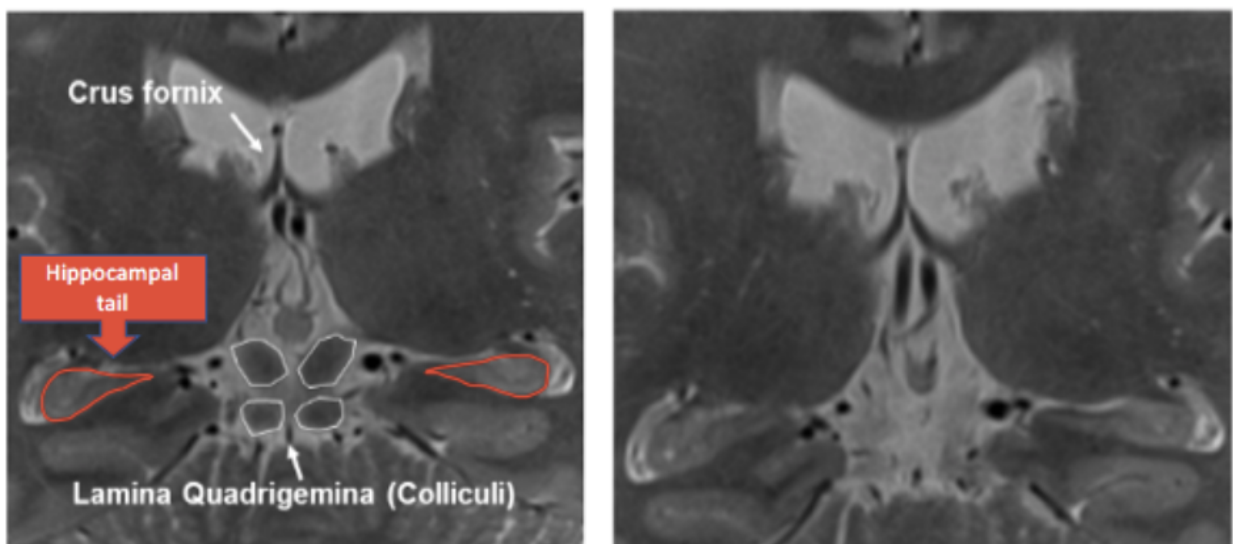


Figure 6.4: Left: Final posterior slice of the hippocampal body displaying the colliculi, crus fornix, and “tear drop”

shape of the hippocampal body. Right: The colliculi are no longer visible, and therefore the image is considered to be the first slice of the hippocampal tail (or posterior hippocampus).

The posterior HPC is drawn in **copper** (see **Labels, Naming Conventions, and Contrasts** above for more information).

Hippocampal CA1

After segmenting the anterior head (and identifying slices of the posterior HPC), move onto the subfields in the hippocampal body, starting with CA1. The most anterior sections of CA1 are drawn when the HPC starts to look like a “generic” HPC and there is a definitive C shape in the lateral portion of the body. This will coincide with the first slice of the appearance of the dentate gyrus (DG+CA3). The CA1 is drawn in **green** (see **Labels, Naming Conventions, and Contrasts** above). CA1 will always be guided by the border created by the SRLM and will include the SRLM. If the SRLM is not clear, extend the thickness of the SUB into the thickness of the CA1. There are 3 phases of CA1 that require different methods of drawing boundaries. See the 3 phases below:

CA1 Phase 1: The HPC is “open” and in anterior MTL (i.e., uncus is present):

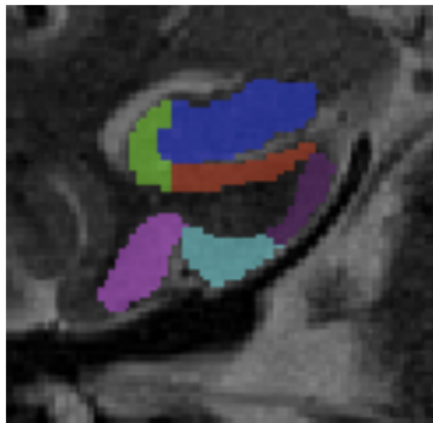


Figure 6.5: CA1 Phase 1, where the superior-medial extent of the CA1 is defined by going half way up the 1st bump, when there are 2 or fewer “bumps”/digitations. If there are 3+ digitations, use the “one third rule” for the superior-medial boundary (one third refers to the width of the HPC, so from the most medial part to the most lateral along the horizontal plane following the angle of the body). The inferior-medial extent is defined by the imaginary line drawn straight down from the superior border of CA1. This will give a resulting “C” shape.

CA1 Phase 2: The uncus is starting to disappear/disappeared and in anterior MTL:

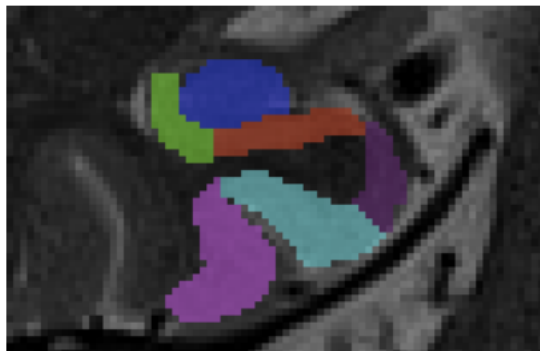


Figure 6.6: CA1 Phase 2, where the superior-medial extent of the CA1 is $\frac{3}{4}$ up the “C” shape. The inferior-medial extent is the bisection of the hippocampal body including the CA1 along with CA3+DG.

CA1 Phase 3: When the uncus has disappeared and the MTL is in the posterior slices:

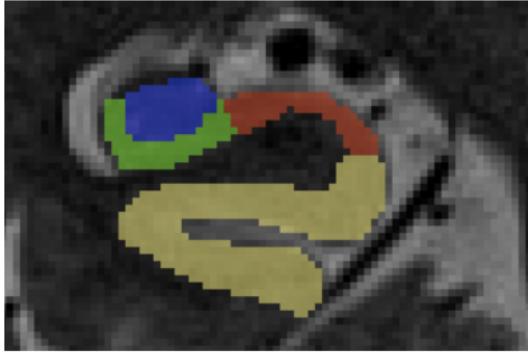


Figure 6.7: CA1 Phase 3, beginning on the second slice the uncus is absent. The superior-medial extent is $\frac{3}{4}$ up the C shape. The inferior medial-extent is drawn to the very medial tip of CA3+DG. It is a “teardrop” shape and the most medial portion might be easy to miss. At this phase, you should get a “3-point intersection” between the CA1, Sub, and CA3+DG at the medial edge of CA1.

Subiculum

The subiculum (Sub) is drawn on all slices in which the hippocampus can be divided into its subregions. The subiculum is drawn in **red** (see **Labels, Naming Conventions, and Contrasts**). The lateral extent of the Sub will always be marked by the inferior-medial portion of CA1. There are two different rules for drawing the subiculum and they depend on whether you are in anterior or posterior slices.

Anterior (CA1 is in phase 1)

The inferior-medial border will then always be drawn to the bisection of the elbow that connects to the ERC (to the hippocampal fissure).

Posterior (second slice the uncus is absent, CA1 is in phase 2-3)

The inferior-medial portion will always be drawn $\frac{1}{2}$ way down the isthmus (see below for helpful tips on how to find the halfway point). The isthmus is measured from the end of CA1 to the beginning of the collateral sulcus (CS).

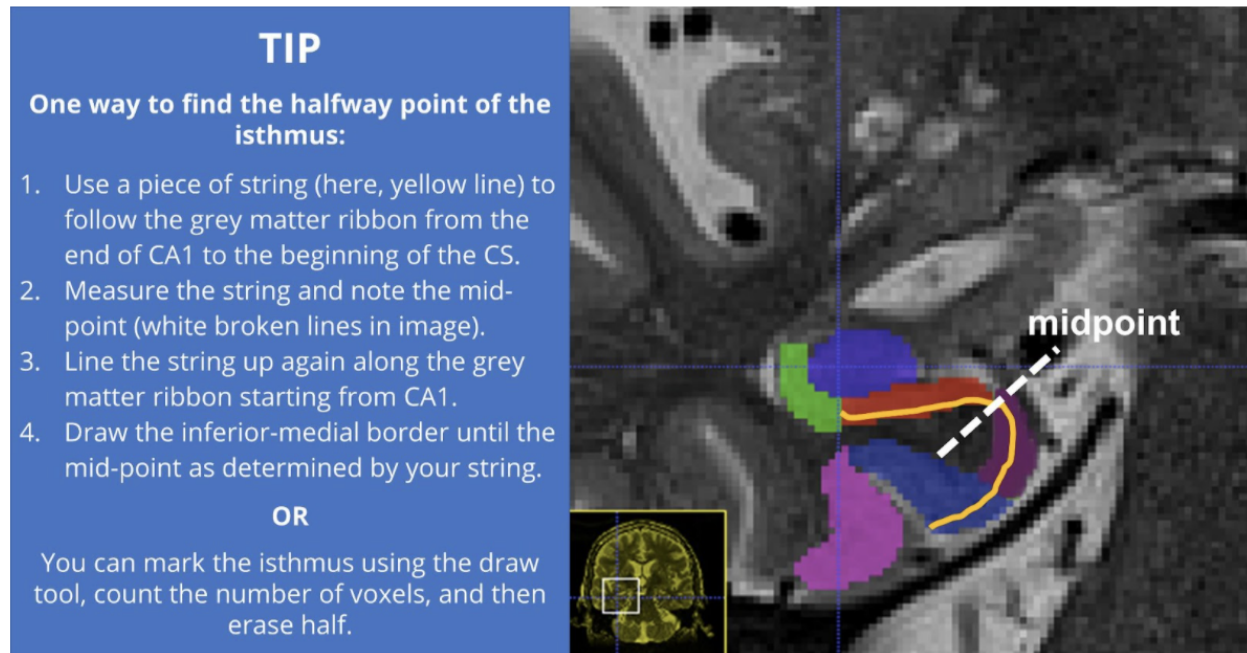


Figure 6.8: How to find the halfway point of the isthmus.

DG+CA3

The dentate gyrus (DG) and CA3 are drawn in as one ROI. DG+CA3 is drawn in **blue** (see **Labels, Naming Conventions, and Contrasts**). It is drawn on all slices that the hippocampal subfields are defined. Simply follow the grey matter region defined laterally by CA1 and superiorly by the Sub. Superiorly, CA3/DG will be determined by a strip of bright CSF, or in more anterior regions it will border the amygdala. This region will also be bordered by the temporal horn of the lateral ventricle.

When defining the region, it is important to not include the white matter (alveus and fimbria) on the superior edge. Keep in mind that this region also typically resembles a tear drop shape in the posterior slices. In the case where the uncus apex is present, trace out both the uncus apex and the lateral body of the HPC.

The medial portion is defined as the closing of the tear-drop shape of the HPC. In anterior slices where the HPC is “open” white matter will surround the medial portion.

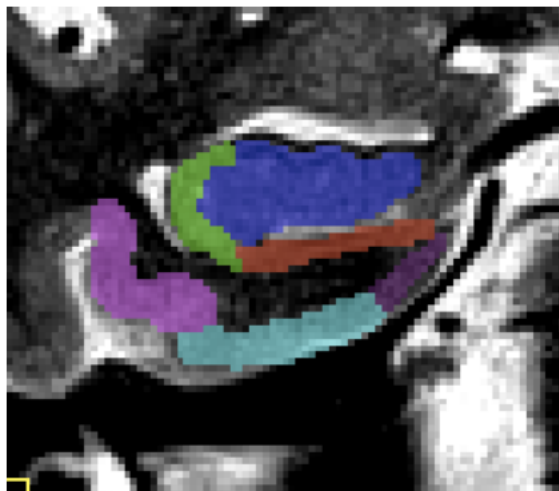


Figure 6.9: Do not include the white matter (alveus and fimbria) on the superior edge of DG and CA1.

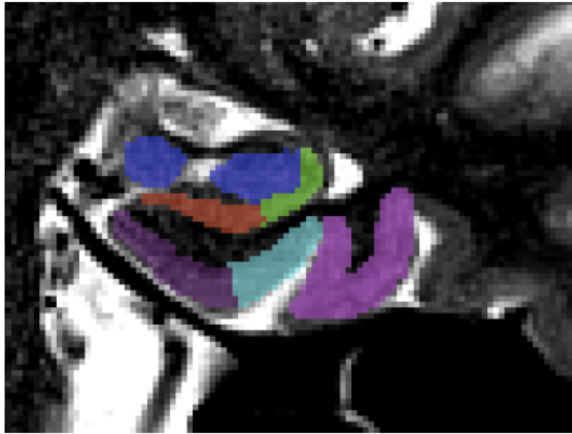


Figure 6.10: In the case where the uncus apex is present, trace out both the uncus apex and the lateral body of the HPC (depicted in navy blue).

7 Segmenting Subregions of the Entorhinal Cortex

Segmentation of subregions of the entorhinal cortex (ERC) is accomplished by tracing the posterior-medial ERC (pmERC) from the initial segmentation of the complete ERC. The pmERC is traced in **purple**.

ITK-Snap instructions for segmenting ERC into sub-regions

Click the Polygon tool in the Main Toolbar on the left panel of the ITK-SNAP window. Then, in the Segmentation Labels box in the left panel below, choose pmERC as your Active label and ERC as your Paint over label. Note you do not need another label called alERC as whatever is not pmERC is alERC. Now you can draw a circle around the area of ERC you want to be pmERC and click accept.

Rules for segmenting the ERC into sub-regions

The following rules will help you identify the alERC from the pmERC:

1 When the amygdala is present and the hippocampal head has not appeared yet, ERC is fully covered by alERC (i.e., do not label pmERC).

2 Approximately 2 mm after the appearance of the hippocampal head, draw the superior boundary of the pmERC to match the superior boundary of the ERC. The inferior boundary of the pmERC is at the uncus notch (see **Helpful Additional Resources for Further Reading**, Maass et al., 2015). This rule assumes that the hippocampus is still “enclosed” in white matter (i.e., the head looks like a digitized bean or “lazy boy” shape).

In the case that the pmERC starts on a slice where the uncus notch has not yet appeared (i.e., in cases where the hippocampal head starts before the limen insulae/FTJ), draw pmERC to the medial ¼ of the ERC (see Variability in Landmarks section for more information about this case).

3 Moving posteriorly (in subsequent slices), when the hippocampal sulcus is present, and lower bank of the anterior head (i.e. subiculum, which does not yet get its own label) is contiguous with ERC, draw the inferior pmERC boundary halfway between width of previous slice and next slice. Note that the hippocampal subfields are not yet segmented on this slice of MTL. More specifically, draw the inferior boundary of

pmERC up to about an average midway point from how far up it was drawn in the previous slice and how far up it will be drawn on the subsequent slice.

4 Moving more posteriorly, the pmERC covers $\frac{1}{2}$ of ERC when the first slice of DG appears.

5 Next, draw pmERC covering $\frac{1}{2}$ of ERC at the hippocampal slice that is $\frac{2}{3}$ away from the start of the hippocampal head to the last slice of the ERC.

6 For the last slice that contains uncus (i.e. last slice of hippocampal head), ERC is covered by $\frac{3}{4}$ pmERC. To review terminology of the hippocampus (e.g., hippocampal head), see **Figure 4.14 in Lay of the Land: Medial Temporal Lobe Landmarks**.

7 In the final slice where no uncus is present, ERC is fully covered by pmERC.

8 Variability in Landmarks

Rhinal Sulcus:

Variability in the depth of the rhinal sulcus (RS) within anterior slices of the temporal lobes has been noted previously (Hanke, 1997; Xie et al., 2017). However, most investigations into the structure of the perirhinal cortex (PRC, i.e., Brodmann areas 35 and 36) either do not describe slices before the

appearance of the hippocampal head (Insausti et al., 1998; Taylor & Probst, 2008), do not distinguish the RS from the CS (Bonisha et al., 2004), or do not

encounter any brains with significant variation of the RS before the appearance of the entorhinal cortex (Augustinack et al., 2013).

Ding and colleagues (2009) found that when a deep RS exists, anatomical area 35 is primarily found in the fundus and the lateral/anterior bank of the RS. In a follow-up study, Ding & Van Hoesen (2010) note that in brains with a deep RS which merges with the CS in anterior slices, area 35 is located in the medial bank of the CS, and area 36 occupies the fundus and lateral bank of the anterior CS.

From these findings reported in Ding and colleagues (2009; 2010) we recommend the following for segmenting early anterior PRC slices:

If a visible (deep) rhinal sulcus is present in early anterior slices such that it interrupts the extension of the PRC up to the Gyrus of Schwalbe from the CS, use the bifurcation rules. That is, when tracing the PRC before the limen insulae you should consider the rhinal sulcus and collateral sulcus as a double sulcus. Draw from the superior border of the Gyrus of Schwalbe to the fundus of the more lateral sulcus (i.e., CS).

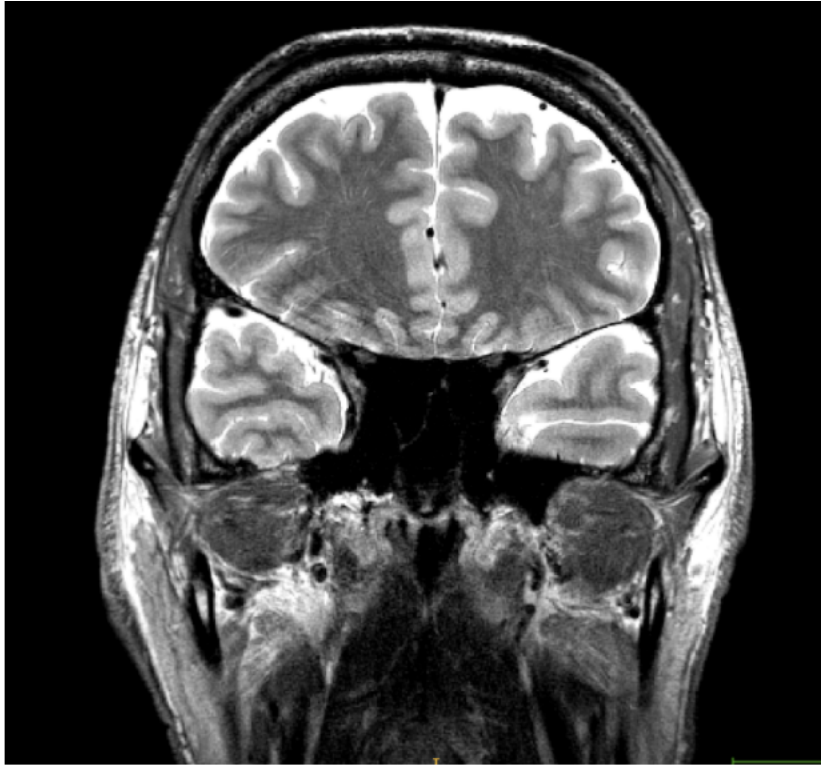
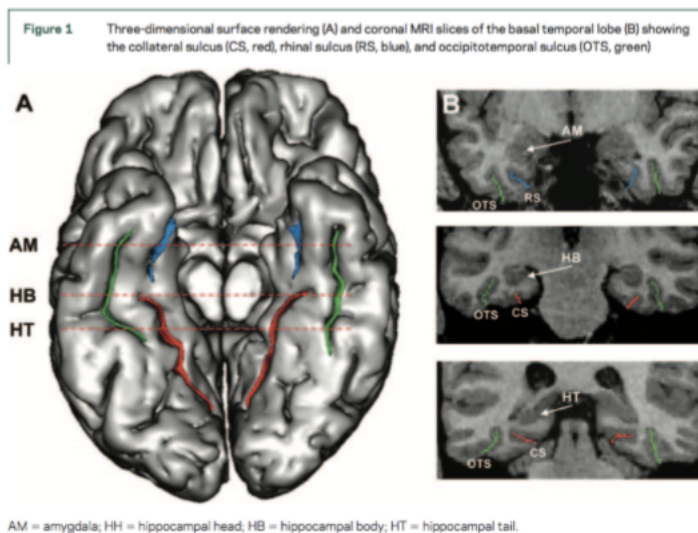


Fig. In rare cases, a very deep RS may join with the superior temporal sulcus and interrupt the extension of the PRC up to the Gyrus of Schwalbe. Here, the fundus of the RS is not visible. In these slices, only trace around the CS. In subsequent slices, if the RS has a visible fundus, continue to apply bifurcation rules (or, if the RS disappears, continue to trace the PRC as usual).

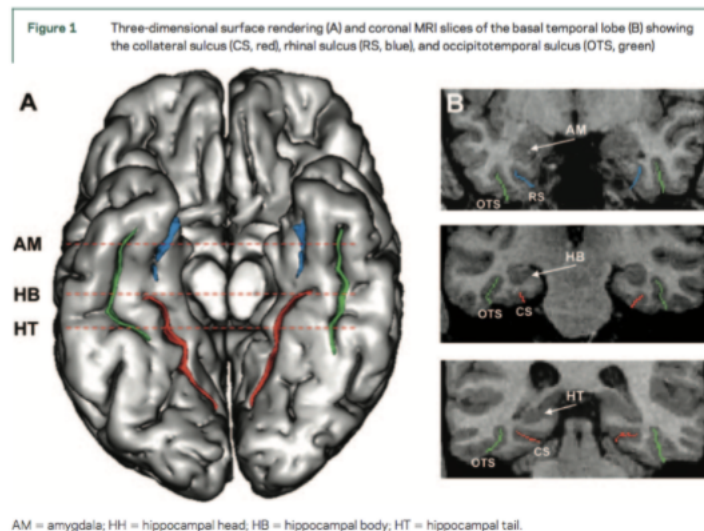
Other cases of variability:



Categorization of sulcal patterns of the basal temporal lobe. The ventral and medial surfaces of the temporal lobe are organized into strips by two prominent rostrocaudally oriented sulci. The more lateral of the two is the occipitotemporal sulcus (OTS), which is often broken forming small, transverse gyri. The more medial is the collateral sulcus (CS) and marks the border between the parahip-

- Type 1: single-branch, unbroken CS connected with the RS anteriorly
- Type 2: CS connected with the OTS, but separated from the RS
- Type 3: CS separated from the OTS and RS, which are connected
- Type 4: CS, OTS, and RS separated

Fig. This image depicts a coronal slice of the basal temporal lobe. In Boxes A and B, the collateral sulcus is shown in red, the rhinal sulcus is shown in blue, and the OTS is shown in green (Kim et al., 2008).



Categorization of sulcal patterns of the basal temporal lobe. The ventral and medial surfaces of the temporal lobe are organized into strips by two prominent rostrocaudally oriented sulci. The more lateral of the two is the occipitotemporal sulcus (OTS), which is often broken forming small, transverse gyri. The more medial is the collateral sulcus (CS) and marks the border between the parahip-

- Type 1: single-branch, unbroken CS connected with the RS anteriorly
- Type 2: CS connected with the OTS, but separated from the RS
- Type 3: CS separated from the OTS and RS, which are connected
- Type 4: CS, OTS, and RS separated

Fig. Different sulcal pattern classes (Kim et al., 2008):

- A) Type 1: one-branch CS connected with RS
- B) Type 2: two-branch CS connected with OTS in its posterior portion
- C) Type 3: two-branch CS having connection between RS and OTS in its anterior portion
- D) Type 4: three-branch CS with no connection between the sulci

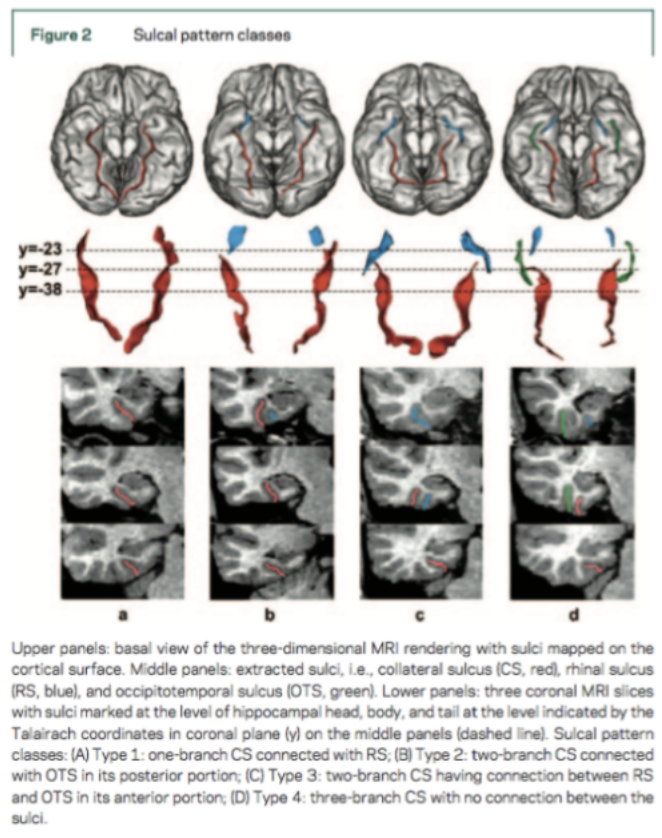


Fig. This is an example of a younger adult brain where the hippocampal head appears before the limen insulae.

9 Computing Volumes and Statistics

Once you have a segmentation, it is important to get some information about it. For this, there is a volumes and statistics option in ITK-Snap that allows you to see the volume corresponding to each label in your segmentation as well as intensity statistics for each label.

STEP 1

Open image and corresponding segmentation.

STEP 2

In the task bar, select **Segmentation -> Volumes & Statistics**

Here, you will see information about:

- Number of voxels that belong to the structure
- Volume of the structure (in cubic millimeters)
- Mean image intensity inside the structure
- Standard deviation of the image intensity in the structure

Volumes and Statistics - ITK-SNAP				
	Label Name	Voxel Count	Volume (mm3)	Intensity Mean ± SD (t2_hires)
0	Clear Label	7308447	4.048e+06	199.1416±255.3965
1	R_Ant_Hipp	1162	643.6	483.2074±70.6756
2	R_CA1	1155	639.7	506.4165±63.4851
3	R_Sub	847	469.1	443.3684±71.2210
4	R_CA23DG	1345	745	531.1442±58.6793
5	R_Post_Hipp	736	407.7	486.5231±51.3016
6	R_PRC	6246	3460	511.1068±69.0031
7	R_ERC	1547	856.9	456.4984±77.8576
8	R_PHC	3566	1975	460.4086±62.1593
9	R_pmERC	784	434.3	441.1186±69.1651
10	L_Ant_Hipp	555	307.4	479.7820±60.9753
11	L_CA1	1090	603.7	464.6853±60.0954
12	L_Sub	929	514.6	420.0452±75.5242
13	L_CA23DG	1476	817.5	487.8327±51.8246
14	L_PRC	4512	2499	460.4009±57.0501
15	L_ERC	809	448.1	443.3622±57.5962
16	L_PHC	3438	1904	469.4127±59.1624
17	L_Post_Hipp	765	423.7	483.9621±60.6131
18	L_pmERC	623	345.1	429.6196±74.1797

Update Copy Export... Close

STEP 3


Select **Export** and specify a filename with a **.txt** extension and click OK.

STEP 4

Import text file into a spreadsheet application for analysis. For analysis, it is important to note that you cannot use raw structural volumes in your models. This is because you need to account for total estimated intracranial volume (ICV).

To do this, refer to the methodology section of Raz et al. (2015). You may correct for ICV via a linear equation: $\text{Volume}_{\text{adj}} = \text{Volume}_{\text{raw}} - b(\text{ICV}_i - \text{Mean ICV})$, where $\text{Volume}_{\text{adj}}$ is the adjusted regional volume, $\text{Volume}_{\text{raw}}$ is the original volume for an individual, b is the slope of the ROI volume regressed on ICV, and Mean ICV is the sample mean of ICV (Raz et al., 2015).

10 Glossary of Key Terms

Term	Definition
Alveus	A thin layer of medullary nerve fibers on the ventricular surface of the hippocampus. OAP protocol does not include this region in any ROI.
Amygdala	Collection of nuclei involved in memory, decision making and emotional reactions. Not segmented in this protocol. Located anterior/superior to the hippocampal head. Can be visualized in the sagittal view to help identify the most anterior slice of the hippocampus.
Artifacts	Distortions in MR images that are not native to the actual structure. Can be due to various reasons (e.g., subject motion or normal cardiac function). 
Calcarine sulcus	Sulcus where the primary visual cortex is concentrated in the occipital lobe (posteriorly) and retrosplenial cortex is located more anteriorly. Not segmented in this protocol, but can be seen in the coronal view in posterior slices of the hippocampus. How anteriorly it lies depends on the individual's anatomy, but usually comes in inferior to the hippocampal tail in the most medial part of the parahippocampal gyrus.
Cerebral spinal fluid (CSF)	Clear colourless liquid found in the brain and spine. Shows up as bright white on T2 images and black in T1 images.

Collateral sulcus	Sulcus running anteriorly to posteriorly of MTL. Can disappear at times, or be shallow, regular or deep in size, and can also be a double sulcus. Best visualized in the axial view in T1 image.
Cornus ammonis (CA) 1-4	Subfields of the hippocampus. CA ₁ is drawn as a different section from CA ₂ , CA ₃ , and CA ₄ . CA ₄ is not recognized by all neuroanatomy labs as a distinct region from the "hilus" region of the dentate gyrus.
Dentate Gyrus (DG)	A subregion of the hippocampus involved in the formation of new episodic memories and neurogenesis. Drawn as one region of interest along with CA ₂ , CA ₃ and CA ₄ .
Elbow	Informal term used for the region where two structures meet at a definitive bend.
Fimbria	A prominent band of white matter along the medial edge of the hippocampus. OAP protocol does not include this region in any ROI.
Fornix	The fornix, named for its archlike configuration, is formed from the fimbria, which is the fringelike medial continuation of the alveus that sits on the superior surface of the hippocampus just below the ependymal lining on the floor of the temporal horn of the lateral ventricles. OAP protocol does not include this region in any ROI.
Hippocampal fissure	Also called, "Vestigial hippocampal fissure" or "Hippocampal sulcus". Area which lies in the "open" region (space in between different tissues) which lies in the medial hippocampus, superior to the subiculum and inferior to the uncus (in the head).
Entorhinal cortex (ERC)	Interface between the hippocampus and neocortex. Drawn only in the anterior portions of the MTL.
Fronto-Temporal Junction/Limen insulae	The region of grey matter that joins the frontal lobe to the temporal lobe. The first slice of the fronto-temporal junction is also the first slice that the ERC is also drawn.
Fusiform gyrus	Gyrus connecting the collateral sulcus and occipitotemporal/inferotemporal sulcus.

Gyrus ambiens	Most medial portion of the uncus. Gyrus ambiens is a landmark that demarcates the most superior aspect of the ERC.
Gyrus intralimbicus	Most posteromedial gyrus of the hippocampus and most posterior portion of the uncus. When the gyrus intralimbicus is present, it gives the hippocampal head the distinctive "open" or "island" shape.
Gyrus of Schwalbe	A gyrus that marks the superior lateral point of the PRC anterior to the limen insulae. Can exist as two gyri, one gyrus or not be present at all. When Gyrus of Schwalbe is either not present or unidentifiable, the midpoint of the superior grey matter ribbon is used as the superior-lateral boundary of the PRC (see Fig. 5.1).
Hippocampal-amygdala transition area (HATA)	The strip of grey matter that connects the amygdala and the hippocampus. Bisection of this region gives the superior boundary of the more anterior portion of the hippocampus. This region often has a mixture of cell types making it hard to classify as a particular subfield.
Hippocampus	Involved in declarative memory formation and storage of autobiographical memory. Includes the CA ₁₋₄ , dentate gyrus, , and subiculum.
Hippocampal sulcus/Uncal sulcus	Separates the uncus from the adjacent parahippocampal gyrus. Portion of hippocampal fissure lying ventral to uncus. (i.e., uncal notch). One of the boundaries for segmenting ERC. (see Duvernoy (2005) Chapter 7.1 p. 145-147)
Isthmus	Geographical term used here to describe the narrow strip of grey matter that connects the hippocampus to the MTL cortices. Contains subiculum, ERC, PRC, and PHC.
Medial Temporal Lobe (MTL)	Involved in episodic and semantic memory. Includes the amygdala, brainstem, hippocampus and hippocampal region (perirhinal, parahippocampal, entorhinal neocortical regions).
Occipitotemporal sulcus (OTS)	Also referred to as the Inferotemporal sulcus (ITS). Sulcus that is lateral to collateral sulcus. Can appear as a double and is not drawn as part of the MTL.
Parahippocampal	Posterior MTL cortex. Involved in memory encoding, retrieval,

cortex (PHC)	scene processing (e.g., "parahippocampal place area").
Rhinal sulcus	<p>From Augustinack et al. (2013): "we define the rhinal sulcus as completely separate from the collateral sulcus (Braak and Braak, 1992; Ono, 1990; Suzuki and Amaral, 1994a; Van Hoesen, 1995; Van Hoesen et al., 2000) and do not ascribe to the rhinal sulcus being the anterior part of the collateral (Hanke, 1997). The sulcal boundaries for the entorhinal and perirhinal cortices can be elaborate, but in the most simple terms, a rhinal sulcus borders anteriorly and the collateral sulcus borders laterally." See "Variability" section.</p> <p>https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3508349/</p>
Perirhinal cortex (PRC)	Involved in visual perception and memory. Drawn only in the anterior slices of the MTL.
ROI	Region of interest.
Stratum radiatum lacunosum moleculare (SRLM)	Stratum radiatum, stratum lacunosum, and stratum molecular layers. Represents the border between CA ₁ /subiculum and DG/CA2/3. The SRLM is included in the CA1 and subiculum ROIs and excluded from the DG/CA2/3 ROI.
Subiculum	Major output structure of the hippocampus. Bordered medially by the ERC and laterally by CA ₁ .
Sulcus semiannularis (ssa)	Sulcus that borders the gyrus ambiens and marks the superior-medial border of ERC. Hard to see on MR images, so is no longer used as a boundary in this segmentation protocol.
Uncus	Medial grey matter structure that is part of the hippocampus and contains a mixture of subfields. Used as an anatomical landmark to differentiate between the anterior and posterior hippocampus.
Ventricles	Cavities in the brain that are filled with CSF. Show up as bright white patches on T2 images.
Vestigial Hippocampal Fissure	With atrophy, can appear as white due to CSF, adjacent to the hypointense band (i.e. SRLM) separating DG from CA regions.

11 Helpful Additional Resources for Further Reading

Read the resources below to familiarize yourself with the anatomy, function and current segmentation procedures of the MTL.

A :Duvernoy, H. M. (2005). The human hippocampus: Functional anatomy, vascularization and serial sections with MRI (3rd ed.). New York: Springer-Verlag. This resource contains useful information about MR images and the anatomy of the hippocampus. Chapter 4 overviews hippocampal anatomy, while chapter 7 describes the sectional anatomy of the hippocampus and surrounding structures in MRI:

This resource contains useful information about MR images and the anatomy of the hippocampus. Chapter 4 overviews hippocampal anatomy, while chapter 7

describes the sectional anatomy of the hippocampus and surrounding structures in MRI.

B :Insausti, R., Juottonen, K., Soininen, H., Insausti, A. M., Partanen, K., Vainio, P., ... Pitkanen, A. (1998). MR Volumetric Analysis of the Human Entorhinal, Perirhinal and Temporopolar Cortices. American Journal of Neuroradiology, 19(4), 659-671. A good segmentation protocol. NB: We do not use the term temporopolar cortices in this current protocol. *Click on citation for full PDF:

A good segmentation protocol. NB: We do not use the term temporopolar cortices in this current protocol. *Click on citation for full PDF.

C :Amaral, R. & Insausti, R. (2004). Hippocampal formation. In J. K. Mai & G. Paxinos (Eds.), The Human Nervous System (3rd ed.). Chapter 24: Hippocampal Formation contains useful subfield figures for segmentation:

Chapter 24: Hippocampal Formation contains useful subfield figures for segmentation.

D :Mueller, S. G., Chao, L. L., Berman, B. & Weiner, M. W. (2011). Evidence for functional specialization of hippocampal subfields detected by MR subfield volumetry on high resolution at 4T. Neuroimage, 56(3), 851-857. This resource outlines functions of the medial temporal lobe ROIs. Figure 1a contains a segmentation example for an older adult participant. *Click on citation for full PDF:

This resource outlines functions of the medial temporal lobe ROIs. Figure 1a contains a segmentation example for an older adult participant. *Click on

citation for full PDF.

E :Kivisaari, S. L., Probst, A., & Taylor, K. L. (2013). The perirhinal, entorhinal and parahippocampal cortices and hippocampus: An overview of functional anatomy and protocol for their segmentation in MR images. In S. Ulmer & O. Jansen (Eds.), fMRI: Basics and Clinical Applications. Berlin: Springer. This resource provides a detailed description of the segmentation procedures of the MTL cortices. Section 19.4 describes the anatomy of the MTL overall, with section 19.4.2 describing a segmentation protocol for the MTL. Figure 19.3 highlights the Gyrus of Schwalbe (a border for the segmentation of perirhinal cortex), referred to in this protocol:

This resource provides a detailed description of the segmentation procedures of the MTL cortices. Section 19.4 describes the anatomy of the MTL overall,

with section 19.4.2 describing a segmentation protocol for the MTL. Figure 19.3 highlights the Gyrus of Schwalbe (a border for the segmentation of perirhinal cortex), referred to in this protocol.

F :Pruessner et al. (2000). Volumetry of hippocampus and amygdala using high-resolution MRI and three-dimensional analysis software: minimizing the discrepancies between laboratories. Cerebral Cortex, 10(4), 433-442:

This paper outlines a protocol to divide the whole hippocampus into the head, body and tail on T1 images. Useful as a reference when alternating between

T1 and T2-weighted images to make a decision on boundaries. *Click on citation for full PDF.

G :Pruessner et al. (2002). Volumetry of temporopolar, perirhinal, entorhinal and parahippocampal cortex from high-resolution MRI images: comparing the variability of the collateral sulcus. *Cerebral Cortex*, 12(12), 1342-1352:

This paper **is** a follow-up to the one listed above, **with** details on dividing the MTL on a **↪**T1 image. *Click on citation **for** full PDF.

H :Daugherty, A. M., Yu, Q., Flinn, R., & Ofen, N. (2015). A reliable and valid method for manual demarcation of hippocampal head, body, and tail. *International Journal of Developmental Neuroscience*, 41, 115-122.

I :de Flores, R., Berron, D., Ding, S. L., Ittyerah, R., Pluta, J. B., Xie, L., ... & Wisse, L. E. (2020). Characterization of hippocampal subfields using ex vivo MRI and histology data: Lessons for in vivo segmentation. *Hippocampus*, 30(6), 545-564.

J :Raz, N., Daugherty, A. M., Bender, A. R., Dahle, C. L., & Land, S. (2015). Volume of the hippocampal subfields in healthy adults: differential associations with age and a pro-inflammatory genetic variant. *Brain Structure and Function*, 220(5), 2663-2674.

K :Maass, A., Berron, D., Libby, L. A., Ranganath, C., & Düzel, E. (2015). Functional subregions of the human entorhinal cortex. *Elife*, 4, e06426.

L : Kim, H., Bernasconi, N., Bernhardt, B., Colliot, O., & Bernasconi, A. (2008). Basal temporal sulcal morphology in healthy controls and patients with temporal lobe epilepsy. *Neurology*, 70(22 Part 2), 2159-2165.

1.17 FSL



What is FSL? FSL (The FMRIB Software) is a comprehensive library of analysis tools for fMRI, MRI and DTI brain imaging data. It runs on Apple and PCs (both \$ can be run both from the command line and as GUIs (“point-and-click” graphical user interfaces))

1.17.1 FSL installation and set up

In order to install FSL, you need go to [FSL](#) find the right version of FSL for you to download a\$

```
[wei@WeiS ~]$ ls
fslinstaller.py
[wei@WeiS ~]$ python fslinstaller.py
--- FSL Installer - Version 3.0.21 ---
[Warning] Some operations of the installer require administrative rights,
for example installing into the default folder of /usr/local.
If your account is an 'Administrator' (you have 'sudo' rights)
then you will be prompted for your administrator password
when necessary.
When asked a question, the default answer is given in square brackets.
Hit the Enter key to accept this default answer.
Where would you like the FSL install to be (including the FSL folder name)? [/usr/local/fsl]:
Downloading...
396247040/4080459725 - 10%
```

```
[OK] Post installation setup complete
Setting up FSL software...
[OK] User profile updated with FSL settings, you will need to log out and back in to use the FSL tools.
To use FSLView please install the PNG12 and MNG libraries with: sudo yum install libpng12 libmng
[wei@WeiS ~]$
```

The goal of this chapter is to show you how to run fMRI analysis with FSL step by step. After successfully install FSL and set up, we are going to analyze a\$ first-hand experiences. The data is Ballon Analogue Risk Task. I hope after this chapter, you will be able to apply the knowledge of FSL in the future and p\$

We will start with the dataset downloading, preprocessing and end with 3rd level analysis as follow:

BART and data downloading

One of the interesting topics from psychology is risky behavior. In order to gain a better understanding of how the brain reacts when people make a decision, there are many many interesting experiment paradigms, Ballon analog risk task is one of them.

In Balloon analog risk task(BART), Participants can manipulate the Balloon to control the risk and reward. Tom Schonberg and his colleagues scanned participants using fMRI while they completed the Balloon Analog Risk Task. As the picture shows, escalating risk-taking occurs under uncertainty and might be experienced either as the accumulation of greater potential rewards, or as exposure to increasing possible losses and decreasing expected value.

In the BART, subjects inflate simulated balloons, and accrue monetary rewards for each successive “pump” during a particular trial. A trial is defined as a balloon that can be pumped a certain number of times and the trial can conclude in two different ways. On the one hand, the participant can “cash-out” at any point during the trial and secure the cumulative winnings up to that point for that balloon in their cumulative total “bank.” On the other hand, a balloon may explode. In this case, participants would lose the money accumulated on that trial alone (but not the total accumulated during previous cash-out trials). Each trial began with a balloon displaying a value of \$0.25 and the value of the balloon increased by \$0.25 for each successive pump. During each trial, participants were presented with one of three types of “reward” balloons, each having a different explosion probability and signified by a different color: red, green, or blue. But participants was not informed the odds of explosion probability.

As a control task, participants intermittently inflated a gray “control” balloon (maximum 12 pumps) that did not explode and had no associated monetary value. The participants were instructed to inflate the control balloon until it disappeared

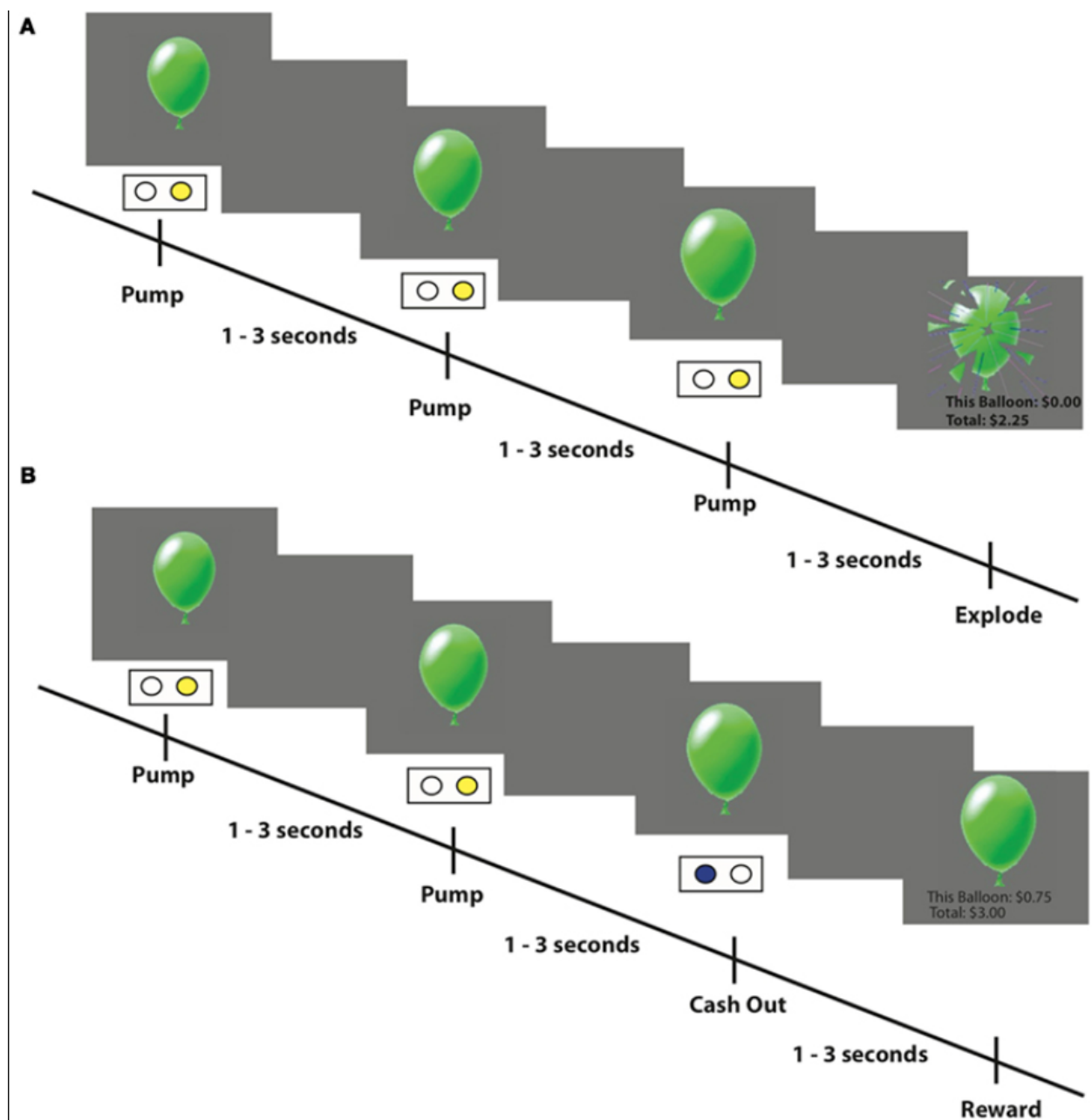


Fig. 4: (A) An example of an explosion trial: participants press one of two buttons to inflate puffs of air into a balloon presented on a computer screen. Every successful pump adds \$0.25 to their temporary bank for that trial. If the balloon explodes before the participant cashes out then nothing is won on that trial. However, an explosion does not affect the cumulative total winnings earned on prior trials. (B) An example of a cash-out trial where the participant decided to stop pumping the balloon and earn the amount accumulated up to that point.

from the screen, and the next trial began. Unlike with reward balloons, participants had no control over how many times they could inflate the control balloon before the trial ended.

for more information, please go [here](#)

Downloading the data

As an Open Source dataset, BART dataset has a standardized structure: Each subject folder contains an anatomical directory and a functional directory labeled anat and func, these contain the anatomical and functional images collected during the experiment. The func directory also contains onset times, or timestamps for when the subject underwent different trials. This format is known as Brain Imaging Data Structure, as we saw in previous chapter [BIDS](#). As an example of the BIDS format. The func directory of BART contains functional data, three runs of timestamps of which condition happened at what time. You can open these as a text file or as a spreadsheet.

Now, go to [there](#), download the data and save it as BART in our home directory.

```
[wshao@gateway BART]$ ls
CHANGES      participants.tsv  sub-01  sub-03  sub-05  sub-07  sub-09  sub-11  sub-13  sub-15  task-balloonanalogrisktask_bold.json
dataset description.json  README      sub-02  sub-04  sub-06  sub-08  sub-10  sub-12  sub-14  sub-16  timing.sh
```

As you can see, we have 16 subjects. **participants.tsv** will tell you the demographic information of subjects, **task-balloonanalogrisktask_bold.json** contains TR information. You can preview all of these information from the Open-Neuro data web.

Let's take a close look

```
anat func
```

cd to sub-01, you will find two directories, anat and func that correspond anatomical and functional. These are two important directories that we will visit most.

```
sub-01_inplaneT2.nii.gz  sub-01_T1w.nii.gz
```

anat includes all the anatomical images such as T1 and T2 (if possible).

```
sub-01_task-balloonanalogrisktask_run-01_bold.nii.gz  sub-01_task-balloonanalogrisktask_run-02_events.tsv
sub-01_task-balloonanalogrisktask_run-01_events.tsv  sub-01_task-balloonanalogrisktask_run-03_bold.nii.gz
sub-01_task-balloonanalogrisktask_run-02_bold.nii.gz  sub-01_task-balloonanalogrisktask_run-03_events.tsv
```

func has all the functional images, end with **bold.nii.gz** as well as trial-related files end with **events.tsv**.

Whenever you're ready, let's go to next

Preprocessing

Remember our drink menu? It is a little difference for what we should do in FSL but overall there are very similar. Anyway, let's have a couple of drinks before the meal:

- 1 Inspecting the data
- 2 Skull stripping
- 3 Motion correction
- 4 Slice-Timing Correction
- 5 Smoothing
- 6 coregistration and Normalization

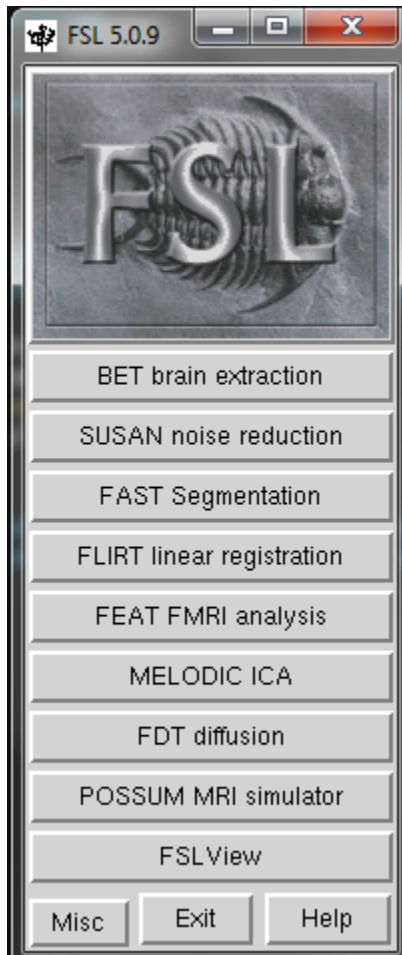
Inspecting the image

Before we are actually running the analysis, it is safe for us to check the data for any problems such as scanner spikes, incorrect orientation, or poor contrast, and so on. Although it might be unnecessary for the open neuroimaging data, it is really important for you to check the image before when it comes to your own data.

Open your Linux terminal and cd to the BART directory.

```
[wshao@gateway ~]$ cd BART
[wshao@gateway BART]$ fs1
```

From BART directory, type `fs1` to open the **FSL GUI tool**



Click **FSLView** and **File** to find the T1 image from anat directory - **sub-01_T1w.nii.gz** and functional image from func directory- **sub-01_task-balloonanalogrisktask_run-01_bold.nii.gz**

You might notice there is a black block on the face areas, it is because all the data from the open-source dataset need to be defaced for the purpose of privacy.

For more information about the clinical diagnosis of brain images, please go to [here](#)

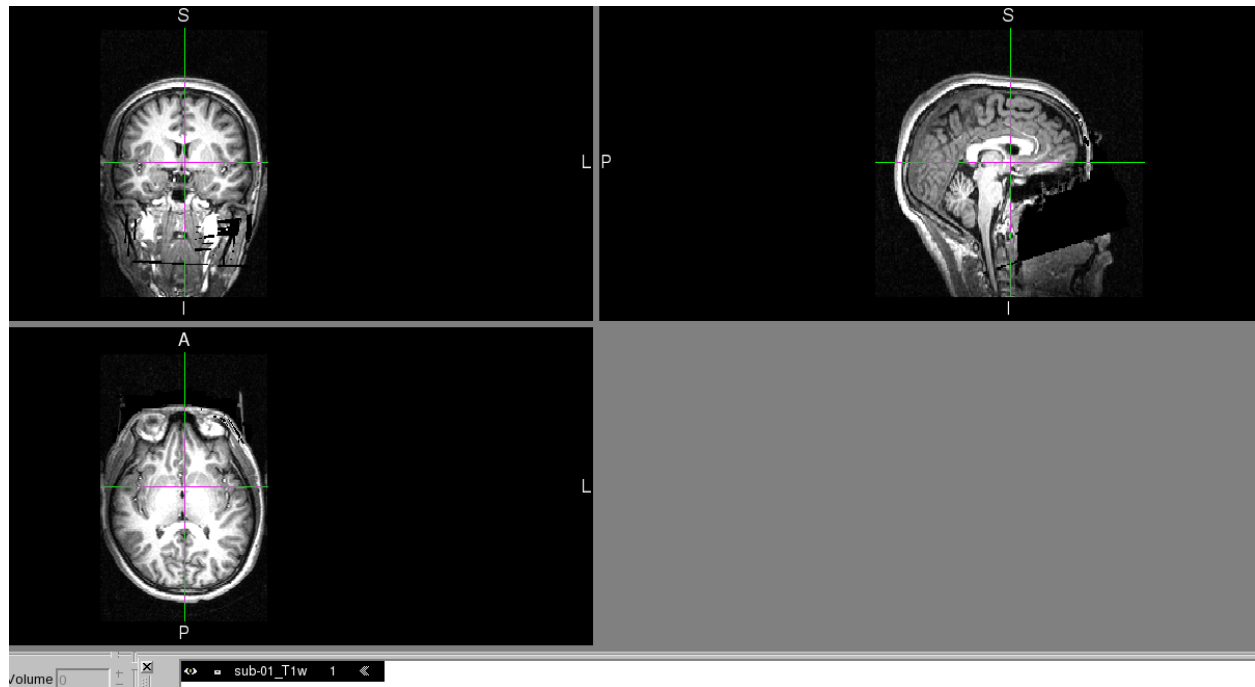


Fig. 5: T1 anatomical image

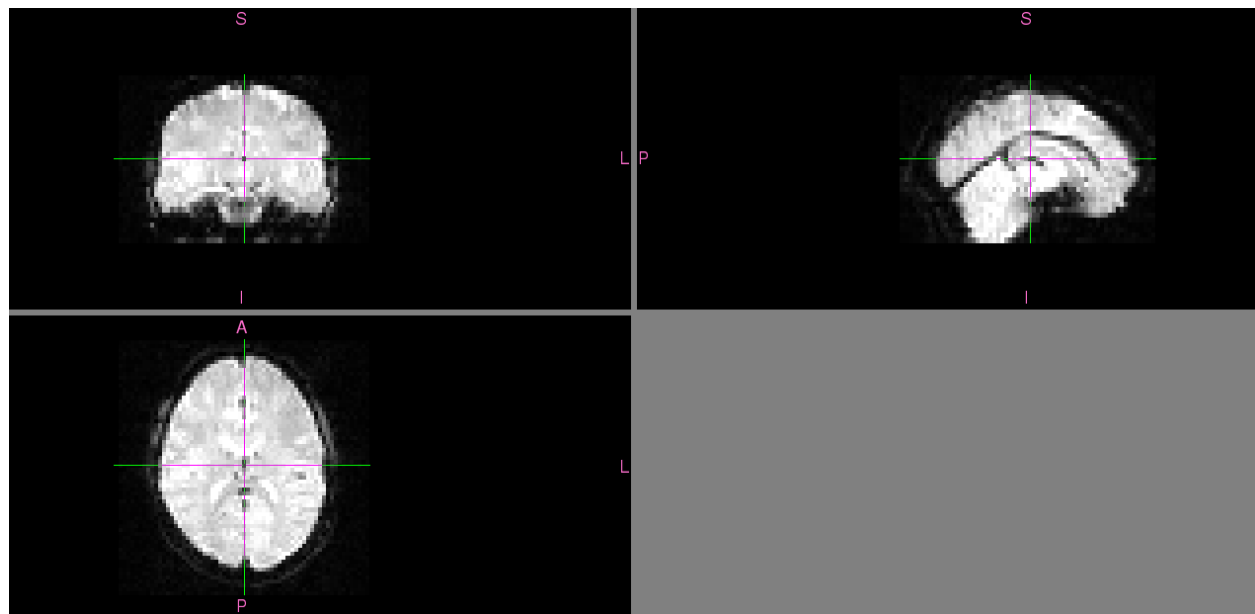
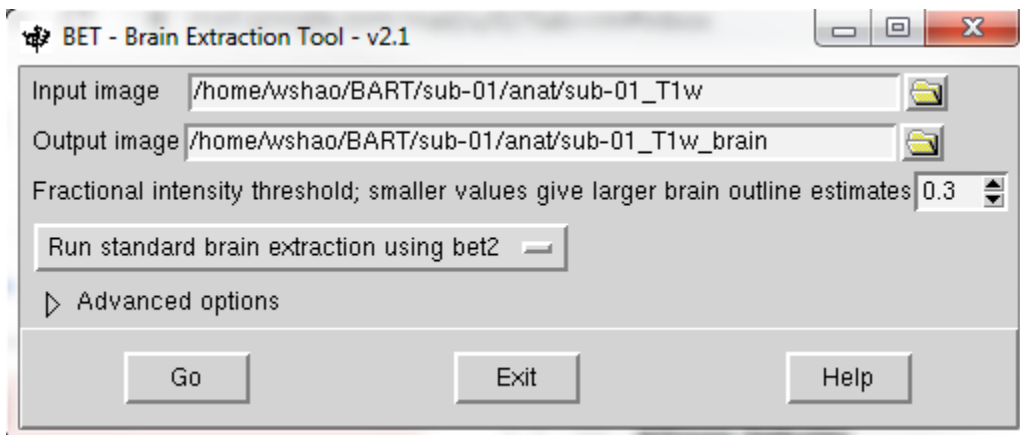
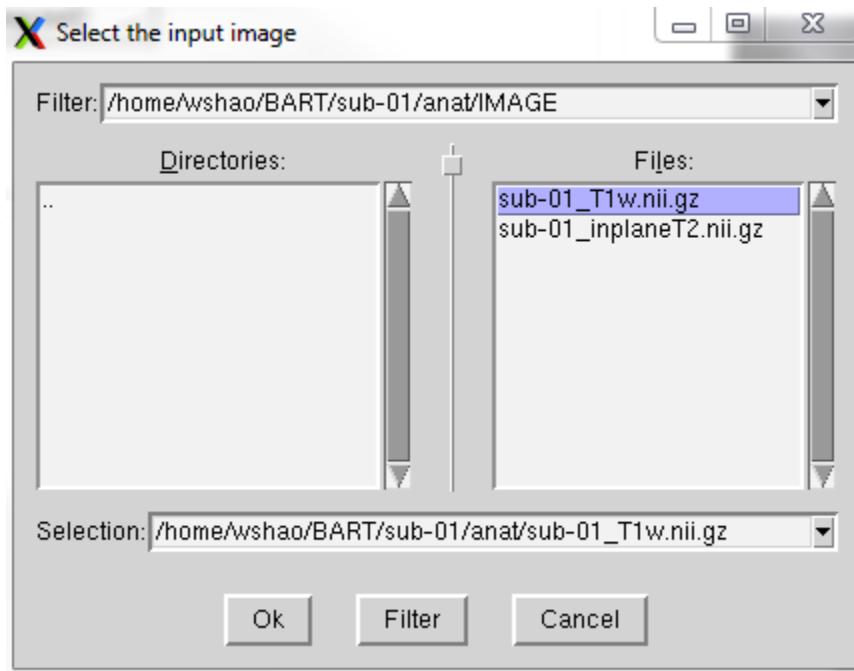


Fig. 6: Functional image

Skull stripping

Since brain tissue is the focus of fMRI studies, our first step is to separate the skull and non-brain areas from the brain tissue in the image. FSL provides this function with tool called **bet** that can help you achieve this goal.

Open **FSL GUI** from sub-01 directory and select the first tab **BET brain extraction** on the GUI list. A new window will appear. In the input image, click the file icon, select the sub-01 first, then anat and sub-01_T1w.nii.gz. The output image will be generated automatically.

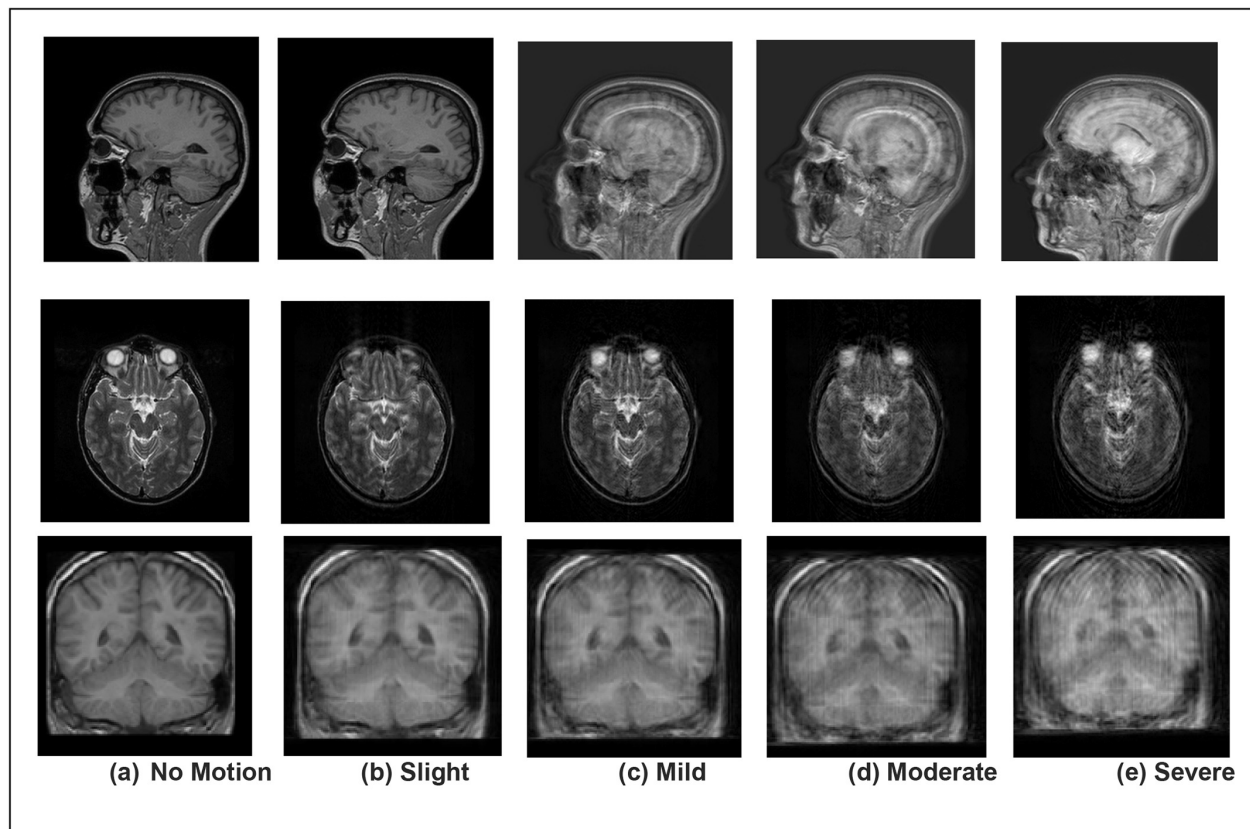


Adjust the stripping range

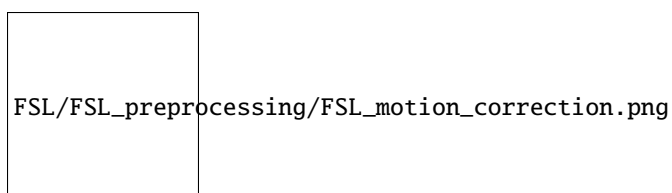
You might notice that there is a tab called **Fractional intensity threshold...**, as the description, you can adjust the number for the skull stripping range by changing the digit number. If too many brain tissues have been removed, you should set this to a smaller number, and vice versa if you think only a little skull has been removed. The default of FSL is 0.5.

Motion Correction

The concept is that when we take three-dimensional pictures of the brain, if the subject is moving, the images will look blurry; if the subject is still, the images will look more defined. In addition, if the subject moves a lot, we also risk measuring signals from a moving voxel. We are then in danger of measuring signal from the voxel for different parts of regions we have targeted before when the subject moves. In other words, motion can introduce confounds into the imaging data because motion generates signal. If the subject moves every time in response to a stimulus - for example, if he jerks his head every time he feels an electrical shock - then it can become impossible to determine whether the signal we are measuring is in response to the stimulus, or because of the movement.



Open the FEAT GUI, motion correction is specified in the Pre-stats tab. FEAT's default is to use FSL's MCFLIRT tool, which you can see in the dropdown menu. You have the option to turn off motion correction, but unless you have a really good reason to do that, otherwise, leave it as it is.



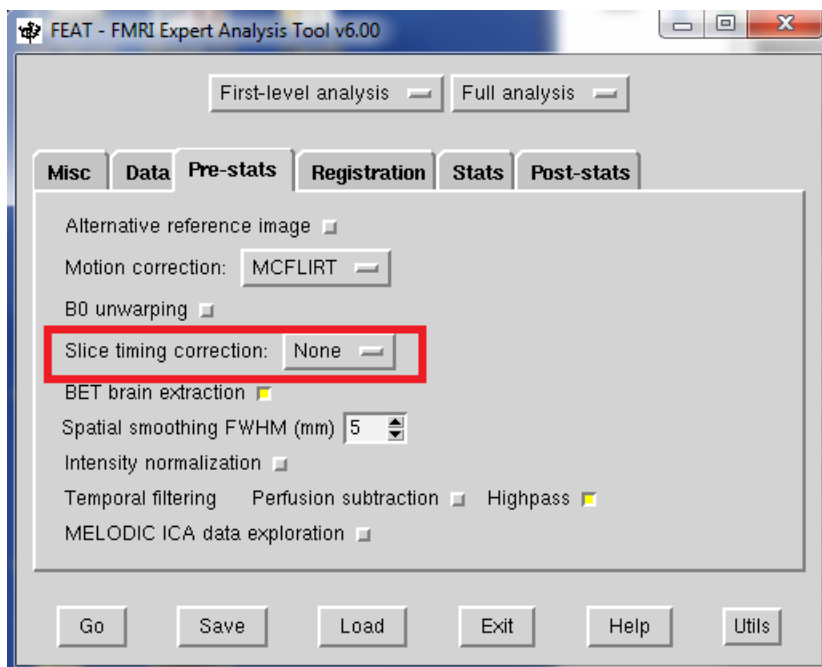
Slice-Timing Correction

An fMRI volume is acquired in slices. Each of these slices takes time to acquire - from tens to hundreds of milliseconds. The two most common methods for creating volumes are sequential and interleaved slice acquisition. Sequential slice acquisition acquires each adjacent slice consecutively, interleaved slice acquisition acquires every other slice, and then fills in the gaps on the second pass.

Fig. 7: figure created by Andrew Jahn

Later, when we use statistics models, we will assume that all of the slices were acquired simultaneously. To make this assumption valid, the time-series for each slice needs to be shifted back in time by the duration when it took to acquire that slice. FSL's default is to not do slice-timing correction, and to include a temporal derivative instead because of the 3 considerations:

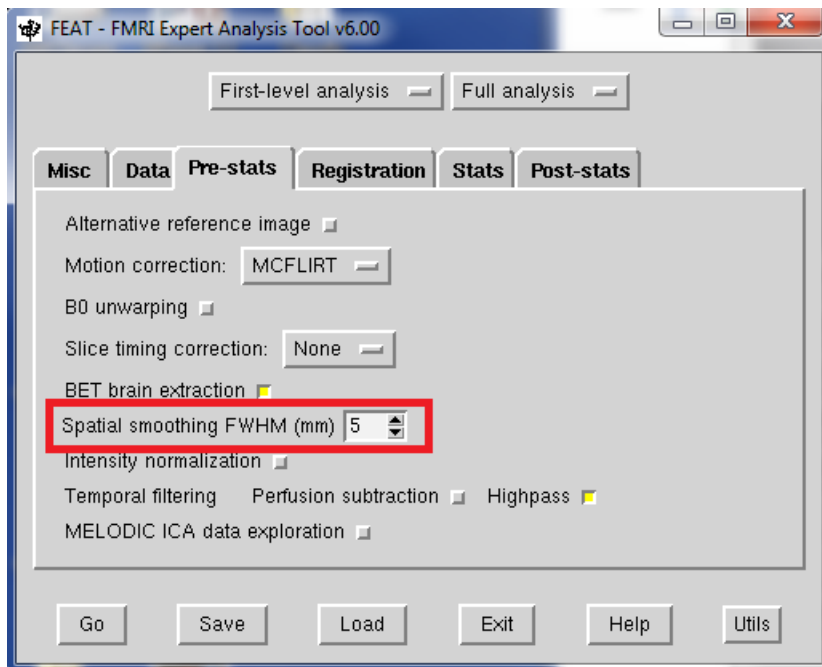
- 1 It is best to not interpolate the data unless you have a good reason to do it.
- 2 Slice-timing correction doesn't appear to lead to any significant gains in statistical power for short TRs - around 1 second or less.
- 3 There is an alternative solution for slice-timing correction problem, temporal derivative.



Smoothing

Although it sounds weird at the first time, people want to smooth the functional data, or replace the signal at each voxel with a weighted average of that voxel's neighbors. Why would we want to make the images blurrier than they already are? It is true that smoothing does decrease the spatial resolution. but there are two benefits that outweigh the disadvantages at least:

- 1 As fMRI data contains a lot of noise, and that the noise is frequently greater than the signal. By averaging over nearby voxels we can cancel out the noise and enhance the signal.
- 2 Smoothing data also can be useful when it comes to Normalization, which the goal is to normalize every subject's brain for a standardized template brain.



The default of Smoothing in FSL is 5mm, again, unless you have a good reason, otherwise, you can skip this.

Registration and Normalization

As human beings, Most of us have very similar brains - everyone has 4 lobes, hippocampus or cerebellum. However, there are also differences in terms of brain size and shape. Therefore, if we want to do a group analysis, it is reasonable to ensure that each voxel for each brain in the subjects corresponds to the same part of the brain. If we are measuring a voxel in the hippocampus, we need to make sure that every subject's hippocampus is in alignment with each other.

In order to do that, we need **Registration** and **Normalization** in FSL. Just as you would fit the material into the baking molds, each brain has to be transformed into the same size, shape, and dimensions. We do this by normalizing them to a template (standard space). A template is a standardized brain that has standard dimensions and coordinates, and most researchers have agreed to use them to report their results. So, if someone has a breakthrough finding, other researchers can check it accordingly.

We have both anatomical and functional images in our dataset. and our goal in here is to organize the functional images to the template so that we can do a group-level analysis across all of our subjects. It seems easy to just simply arrange the functional images directly to the template. However, it doesn't work in reality. functional images are low-resolution, and therefore there are less likely to match up with the anatomical details of the template. In other words, start with the anatomical image is a better option. Warping the anatomical image can be very helpful for filling the functional images into the template because the anatomical and functional scans are typically acquired in the same session. As long as we have normalized the anatomical image to a template and recorded what kind of transformations were done, we can apply the same transformations to the functional images as well.

This alignment between the functional and anatomical images is called **Registration**. Most registration use the following steps:

- 1 Assume that the functional and anatomical images are in roughly the same location. If they are not, align the outlines of the images.
- 2 Take advantage of the fact that the anatomical and functional images have different contrast weightings - that is, areas where the image is dark on the anatomical image (such as cerebrospinal fluid) will appear bright on the functional image, and vice versa. This is called mutual information. The registration algorithm moves the images around to test

different overlays of the anatomical and functional images, matching the bright voxels on one image with the dark voxels of another image, and the dark with the bright, until it finds a match that cannot be improved upon.

3 Once the best match has been found, then the same transformations that were used to warp the anatomical image to the template are applied to the functional images.

Fig. 8: figure created by Andrew Jahn

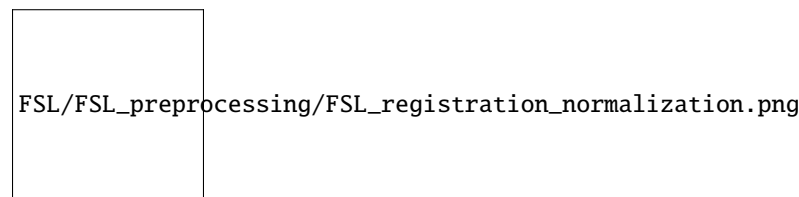
In general:

anatomical image → template

functional image → anatomical image

functional image → template

Registration tab



In FSL, the registration includes all the functions you need. There two options you need to pay attention:

1 Main structural image, you need select the skullstripping anatomical image.

2 Standard spance

for the option 2, select the standard template that already installed in your FSL library, MNI152 would be the most common choice.

In the search window below, there are three options:

1 No search

2 Normal search

3 Full search

This signifies to FSL how much to search for a good initial alignment between the functional and anatomical image(for registration) and between the anatomical and template images (for normalization). The Full search option takes longer, but you will find that worth it because this is more likely to produce better registration and normalization.

In the Degrees of Freedom window, which is the right tab after the search window, you can use 3, 6, or 12 degrees of freedom to transform the images. Registration has an additional option, BBR, which stands for Brain-Boundary Registration. This is a more advanced registration technique that uses the tissue boundaries to fine-tune the alignment between the functional and anatomical images. Similar to the Full search option above, it takes longer, but often gives a better alignment. For simplicity, we will use 12 degrees in this case.

Now, if you have loaded the data, and checked the Motion correction, Slice-Timing, and Smoothing tabs. Click Go and good to go. This could require a few minutes to process and a HTML webpage will jump up and show you the results

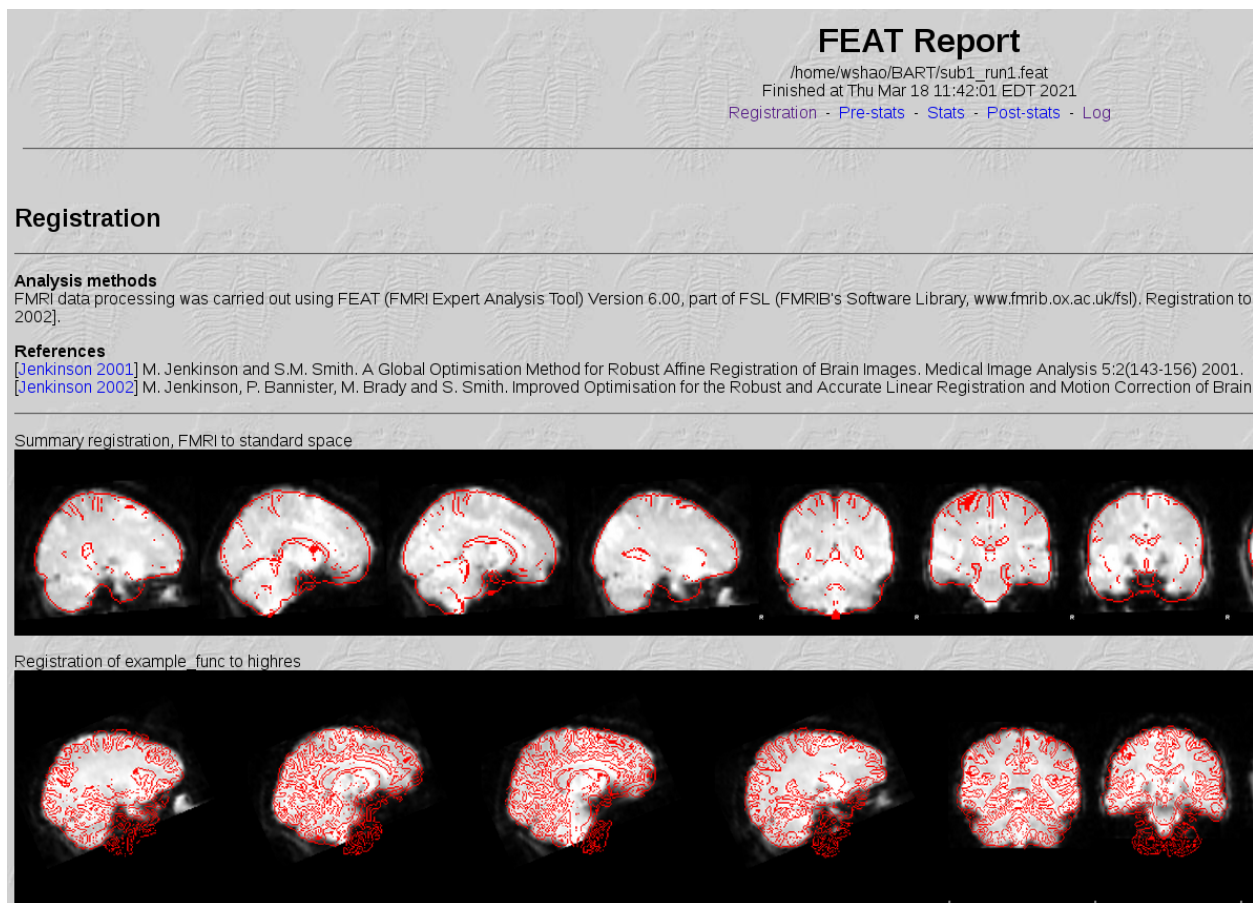
Check the Preprocessed Data

When FSL finish data preprocessing, the next goal for us is to check the preprocessed data manually. For example, We are going to take a look at the registration and normalization and motion correction.

Registration and Normalization check

Click the Registration tab on the webpage, and scroll down. You will see a lot of “brains”.

Each brain includes the red outlines on the top of a greyscale brain from background. The first column, **Summary Registration**, shows representative functional images in the fMRI time-series (volume) as the underlay, and the template brain as the red lines. If there were any problems in any of the previous registration or normalization steps, some obvious errors would be appear in here, such as the image being skewed or largely outside of the red outline. As the red outlines approximately trace the outline of the greyscale image. you also need to check the alignments for internal structures such as the ventricles. What's more, **Registration of example_func to highres**, stands for registering functional image to anatomical image, **Registration of highres to standard**, indicated that FSL normalize the anatomical image to the standard template. Make sure they are in a good shape as well.



Homework

Since we have done the preprocessing for one subject **sub-01**, please repeat all the procedures above for **sub-02** and **sub-03**

Statistics and modeling

After we finish all the preprocessing steps, we can go to the next - fit a model to the data. In order to understand model fitting, we need to know 3 fundamental components:

- 1 General Linear Model
- 2 The BOLD response
- 3 Time-series

General linear model

The General Linear Model (GLM) is a useful framework for comparing how several variables affect the continuous variables. Although it has many variants, the simplest form is described as: $Y = X + \epsilon$. In GLM, we can use one or more regressors(X) - independent variables, to fit a model for some outcome measure(Y) - or dependent variable. To do this we need to compute numbers called beta weights(), which are the relative weights assigned to each regressor in order to get the best fitting for the data. Any discrepancies between the model and the data are called residuals().

For example, imagine that we want to predict the chance of infecting Covid-19 based on social distance, daily interaction in person, and longitude. It is reasonable to find that social distance has a negative correlation with the infection of Covid-19, the more distance you keep with other people, the less chance you will get Covid-19. In terms of daily interaction in person, it has a positive association, the more you interact with people in person, you are more likely to be affected by the virus. lastly, longitude has no association in general.

Therefore, we need assign each of these regressors a beta weight so that the model can fit the real data best. To be more specific, maybe each additional meter for social distance is associated with an additional 0.5 decrease in infection of Covid-19, while each additional 0.5 times of personal interaction is associated with a 0.3 increase:

Chance of infecting Covid-19 (Y) = $-0.5X_1(\text{social distance}) + 0.3X_2(\text{personal interaction})$

This GLM can be expanded to include many regressors, but no matter how many regressors you have, the GLM assumes that the data can be modeled as a linear combination of each of the regressors. Keep GLM in your mind because we will apply it in our model soon.

BOLD response

BOLD signal

In 1990, a researcher at Bell Laboratories named Seiji Ogawa discovered that more deoxygenated blood leads to a decrease in the signal measured from a brain region. An increase in oxygenated blood, on the other hand, increases the signal - this increase in oxygenated blood was later shown to be correlated with increased neural firing. This change in signal is known as blood oxygen level dependent signal (BOLD signal). Shortly afterward, a researcher at Massachusetts General Hospital named Ken Kwong in 1992 demonstrated that the BOLD signal could be used as an indirect measure of neural activity. His experiment consisted of alternately showing a flashing checkerboard and a black screen to the subject for one minute each. The BOLD signal was recorded during each condition, as shown in the following video:

The General Linear Model (GLM)

**Uses one or more regressors (independent variables)
to predict an outcome measure (dependent variable)**

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

Y = Dependent variable

β = Beta Weights (parameter estimates)

X = Regressor

ε = Residual

Fig. 9: The symbols representing each of these terms are shown in the equation.

In the 1990's, empirical studies of the BOLD signal demonstrated that, after a stimulus was presented to the subject, any part of the brain responsive to that stimulus - say, the visual cortex in response to a visual stimulus - showed an increase in BOLD signal. The BOLD signal also appeared to follow a consistent shape, peaking around six seconds and then falling back to baseline over the next several seconds. This shape can be modeled with a mathematical function called a Gamma Distribution. As Gamma Distribution is created with parameters to best fit the BOLD response observed by these empirical studies, it is the canonical Hemodynamic Response Function(HRF).

Gamma Distribution

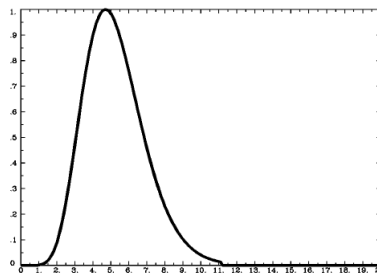
Gamma distribution is a distribution that arises naturally in processes for which the waiting times between events are relevant. It can be thought of as a waiting time between Poisson distributed events. For example, suppose you are fishing and you expect to get a fish once every 1/2 hour. Compute the probability that you will have to wait between 2 to 4 hours before you catch 4 fish. One fish every 1/2 hour means we would expect to get $= 1 / 0.5 = 2$ fish every hour on average. Using $\lambda = 2$ and $k = 4$, we can compute this as follows:

$$P(2 \leq X \leq 4) = \sum_{x=2}^4 \frac{x^{4-1} e^{-x/2}}{\Gamma(4) 2^4} = 0.12388$$

When applied to fMRI data, the Gamma Distribution is called a basis function because it is the fundamental element of the model we will create and fit to the time series of the data. Furthermore, if we know what the shape of the distribution looks like in response to a very brief stimulus, we can predict what it should look like in response to stimuli of varying durations, as well as any combination of stimuli presented over time.

The HRF for a Single impulse Stimulus

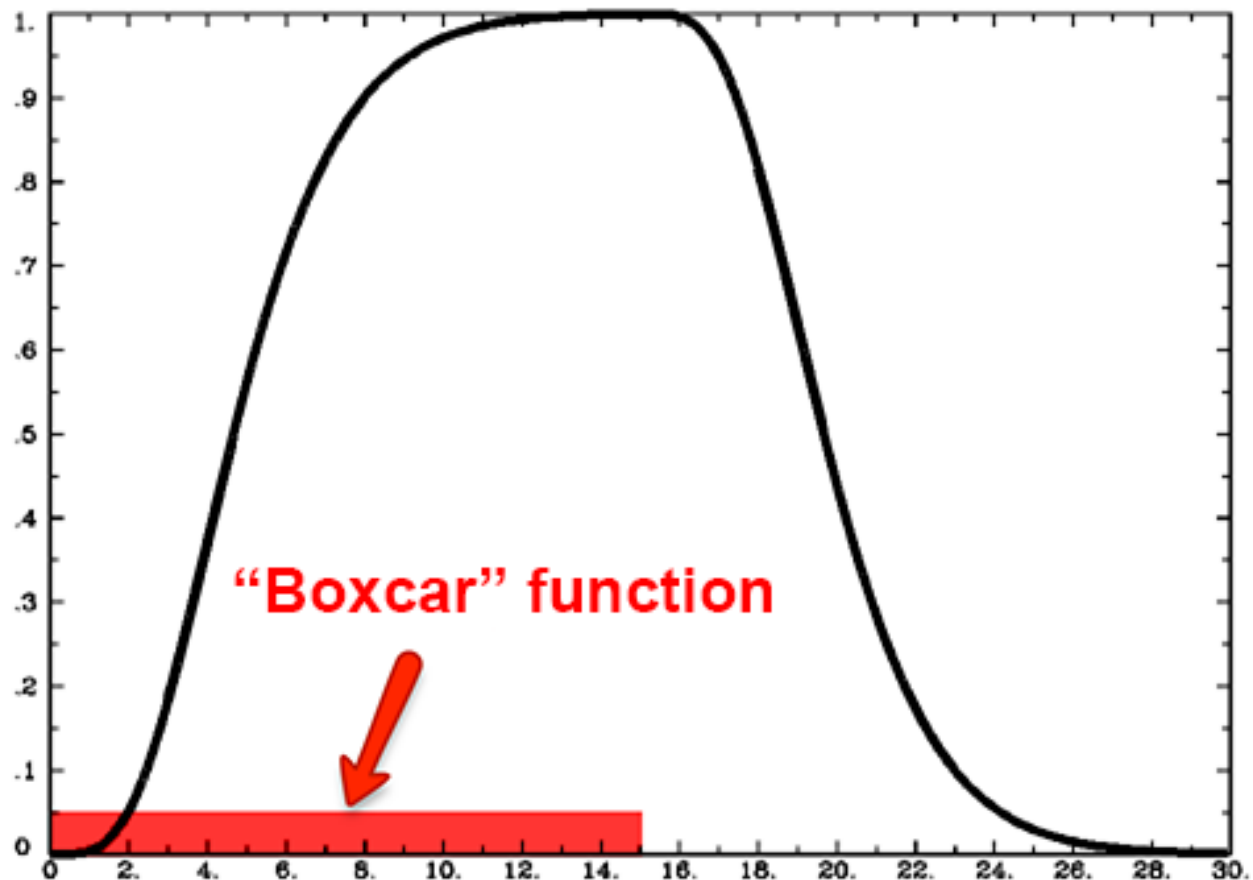
If the duration of a stimulus is very short, such as a snap of the fingers, we can say that it is an impulse stimulus - in other words, it has no duration. As you can see in the following figure, the shape of the BOLD signal looks like a typical Gamma Distribution, with a peak close to the beginning of the time axis (i.e., the x-axis) and a long tail to the right.



The HRF for a Single Boxcar Stimulus

Although many studies use stimuli lasting only a second or less, some studies present stimuli for longer periods of time. For example, imagine that the subject looks at a flashing checkerboard for fifteen seconds. In this case the shape of the HRF will be more spread out with a sustained peak proportional to the duration of the stimulus, falling back to baseline only after the stimulus has ended. This stimulus is called a boxcar stimulus, because it looks like a boxcar on a train.

In this case the Gamma Distribution is convolved with the boxcar stimulus. Convolution is the averaging of two functions over time; as a result, the Gamma Distribution broadens as it is averaged with the boxcar stimulus, and returns to baseline when the stimulus is removed.



Multiple HRFs overlapping

We have seen what the BOLD signal looks like after a stimulus is presented and how the HRF models the shape of that signal. But what happens if another stimulus is presented before the BOLD response for the previous stimulus has returned to baseline?

In that case, the individual HRFs are summed together. This creates a BOLD response that is a moving average of the individual HRFs, and the shape of the BOLD signal becomes more complex as more stimuli are presented close together.

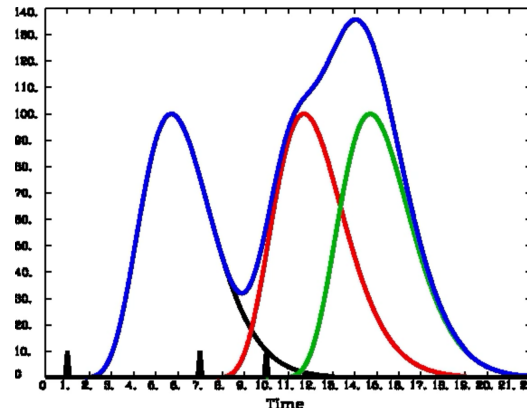


Fig. 10: Convolution of the HRFs for individual stimuli. The overall BOLD response (blue) is a moving average of the individual HRFs outlined in black, red, and green. The vertical black lines on the x-axis represent impulse stimuli. Figure created by Bob Cox of AFNI.

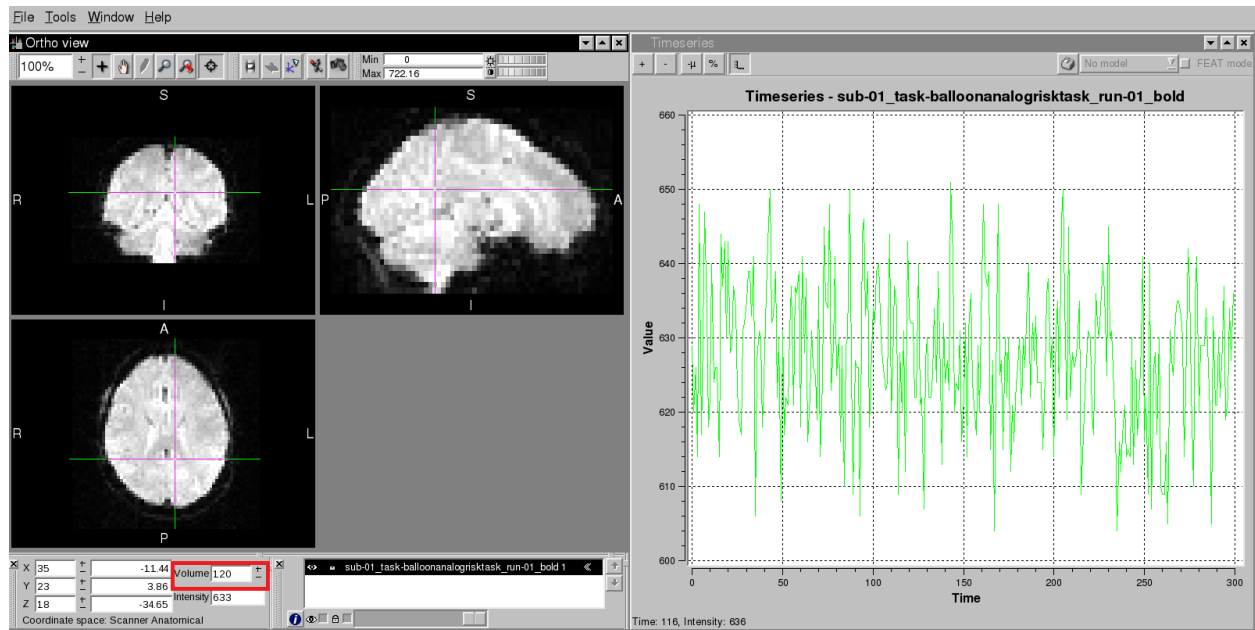
Fig. 11: Animations originally created by Bob Cox of AFNI

Time series

We have mentioned this concept several times before. As the basic composition of fMRI data. Remember that fMRI datasets contain several volumes strung together like beads on a string - we call this concatenated string of volumes a run of data. The signal that is measured at each voxel across the entire run is called a time-series. The time-series represents the signal that is measured at each voxel.

looking at the time series in FSL

In order to have a closer look at time series, you can open FSLview from the `sub-01` directory, and choose File to open 1 of 3 bold files from `func` directory. type `Ctrl+T` and take a look at the volume tab at the bottom. The right plot here indicates the voxel you point out at the different time series positions.



There is another material from FSL that can be very useful when it comes to the regression model and B design. Click [Here](#).

1st level analysis

I hope you had fun with FSL with the preprocessing learning. As you might remember, our goal is to gain the first-hand experience. With this in mind, we are ready go to the next, 1st level (individual) fMRI analysis. So, The first thing for 1st level analysis is to create the model with a ideal time-series so that we can apply the estimated beta weights into each regressors.

In the func directory of each subject from BART directory. You can find 3 files end with **events.tsv**. These files contain three important pieces of information that we need:

- 1 the experimental condition name
- 2 the onset time of trial for each condtion, relative to the onset of the scan
- 3 The duration of each trial

onset	duration	trial_type	cash_demean	control_pumps_demean	explode_demean	pumps_demean	response_time
0.061	0.772	pumps_demean	n/a	n/a	-2.000	2.420	
4.958	0.772	pumps_demean	n/a	n/a	-1.000	0.578	
7.179	0.772	pumps_demean	n/a	n/a	0.000	0.766	
10.416	0.772	pumps_demean	n/a	n/a	1.000	0.840	
13.419	0.772	pumps_demean	n/a	n/a	2.000	1.462	
16.754	0.772	explode_demean	n/a	n/a	1.700	n/a	
24.905	0.772	pumps_demean	n/a	n/a	-0.500	1.295	
27.454	0.772	pumps_demean	n/a	n/a	0.500	1.083	
30.111	0.772	cash_demean	-4.000	n/a	n/a	1.498	
38.449	0.772	pumps_demean	n/a	n/a	-1.500	0.656	
41.028	0.772	pumps_demean	n/a	n/a	-0.500	0.652	
44.529	0.772	pumps_demean	n/a	n/a	0.500	0.864	
47.692	0.772	pumps_demean	n/a	n/a	1.500	1.909	
51.102	0.772	cash_demean	-2.000	n/a	n/a	1.035	
59.031	0.772	pumps_demean	n/a	n/a	-2.000	0.833	
61.850	0.772	pumps_demean	n/a	n/a	-1.000	0.536	
63.516	0.772	pumps_demean	n/a	n/a	0.000	1.085	
67.007	0.772	pumps_demean	n/a	n/a	1.000	0.786	
69.693	0.772	pumps_demean	n/a	n/a	2.000	2.036	
74.286	0.772	cash_demean	-1.000	n/a	n/a	0.603	
82.940	0.772	pumps_demean	n/a	n/a	-2.500	1.109	
86.124	0.772	pumps_demean	n/a	n/a	-1.500	0.541	
88.648	0.772	pumps_demean	n/a	n/a	-0.500	1.001	
91.642	0.772	pumps_demean	n/a	n/a	0.500	1.375	
94.742	0.772	pumps_demean	n/a	n/a	1.500	1.331	
98.989	0.772	pumps_demean	n/a	n/a	2.500	1.549	
102.416	0.772	cash_demean	0.000	n/a	n/a	2.009	
108.445	0.772	control_pumps_demean	n/a	-5.000	n/a	n/a	1.053
111.308	0.772	control_pumps_demean	n/a	-4.000	n/a	n/a	0.333
114.044	0.772	control_pumps_demean	n/a	-3.000	n/a	n/a	0.270
117.302	0.772	control_pumps_demean	n/a	-2.000	n/a	n/a	0.404
120.638	0.772	control_pumps_demean	n/a	-1.000	n/a	n/a	0.332
123.096	0.772	control_pumps_demean	n/a	0.000	n/a	n/a	0.610
124.915	0.772	control_pumps_demean	n/a	1.000	n/a	n/a	1.087
128.429	0.772	control_pumps_demean	n/a	2.000	n/a	n/a	1.029
130.687	0.772	control_pumps_demean	n/a	3.000	n/a	n/a	1.699
133.919	0.772	control_pumps_demean	n/a	4.000	n/a	n/a	0.347
136.034	0.772	control_pumps_demean	n/a	5.000	n/a	n/a	1.655

All of these information can help us to build the time-series files for BART.

Create the time-series

We learned from the dataset previously, There are 4 different conditions for the BART experiment:

pump
control_pump
explode
cash out

Although it is really convenient see all the information from the **events.tsv** files, we still need to transfer the format and content of these files so that FSL can understand. we will create the timing file for each condition, and split them according to which run the condition was in. However, We will only analyze the **explode** and **cash out** condition in this documentation.

What we need from the time series files?

Timings files for the pump trials that occurred during the first run, second run and third run, which will be pump_run1.txt, pump_run2.txt and pump_run3.txt

Timings files for the control trials that occurred during the first run, second run and third run, which will be control_run1.txt, control_run2.txt and control_run3.txt

Each of these timing files will have same format consisting of three columns, in the following order:

- 1 Onset time, in seconds, relative to the start of the scan
- 2 Duration of the trial, in seconds
- 3 Parametric modulation(discuss later)

The script

It is not hard to imagine that if you manually extract the information and input the data into new files, it is time-consuming and very painful. Let alone the mistakes you might make during the process. Fortunately, there are some scripts that can help you with that.

Now, go to the BART directory and create a file with a text editor such as nano or vim, copy the script below and save it as **BART_timing.sh**:

```
#!/bin/bash

#Check whether the file subjList.txt exists; if not, create it
if [ ! -f subjList.txt ]; then
    ls -d sub-?? > subjList.txt
fi

#Loop over all subjects and format timing files into a format that FSL can understand
for subj in `cat subjList.txt` ; do
    cd $subj/func #Navigate to the subject's func directory, which contains the event_
    ↪files

    #Extract the onset and duration for the pump,control,explode, and cash out trials_
    ↪for each run.
    cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="pumps_
    ↪demean") {print $1, $2, "1"}}' > pump_run1.txt
    cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="pumps_
    ↪demean") {print $1, $2, "1"}}' > pump_run2.txt
    cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="pumps_
    ↪demean") {print $1, $2, "1"}}' > pump_run3.txt

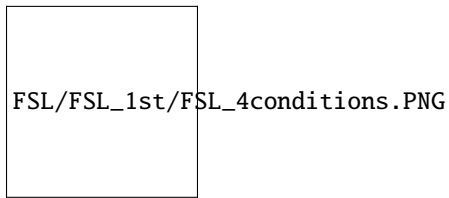
    cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="control_
    ↪pumps_demean") {print $1, $2, "1"}}' > control_run1.txt
    cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="control_
    ↪pumps_demean") {print $1, $2, "1"}}' > control_run2.txt
    cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="control_
    ↪pumps_demean") {print $1, $2, "1"}}' > control_run3.txt

    cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="explode_
    ↪demean") {print $1, $2, "1"}}' > explode_run1.txt
    cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="explode_
    ↪demean") {print $1, $2, "1"}}' > explode_run2.txt
    cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="explode_
    ↪demean") {print $1, $2, "1"}}' > explode_run3.txt

    cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="cash_
    ↪demean") {print $1, $2, "1"}}' > cash_run1.txt
    cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="cash_
    ↪demean") {print $1, $2, "1"}}' > cash_run2.txt
    cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="cash_
    ↪demean") {print $1, $2, "1"}}' > cash_run3.txt

    cd ../../
done
```

After you create the **BART_timing.sh**, run the script by the command `bash BART_timing.sh` and wait for a few seconds. you can check the results by `cd` to the func directory of each subject directory.



Eventually, you will see

```
16.754 0.772 1
157.899 0.772 1
269.218 0.772 1
309.930 0.772 1
320.442 0.772 1
364.626 0.772 1
552.418 0.772 1
566.909 0.772 1
578.603 0.772 1
600.409 0.772 1
```

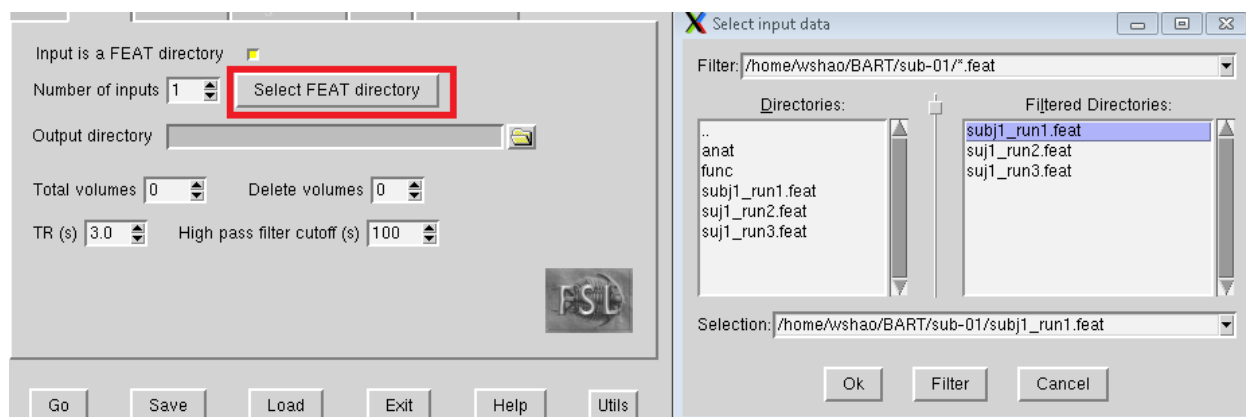
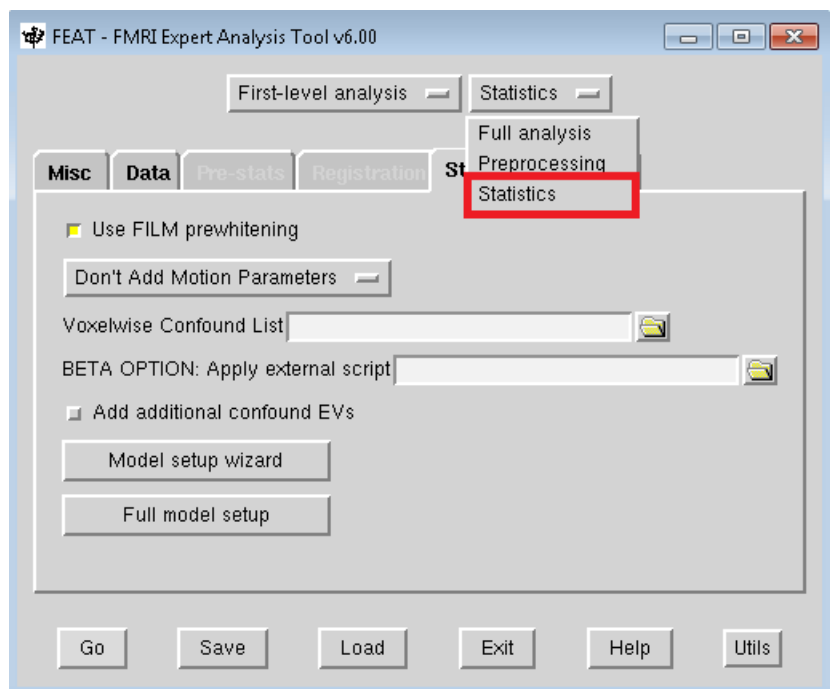
```
30.111 0.772 1
51.102 0.772 1
74.286 0.772 1
102.416 0.772 1
182.490 0.772 1
299.857 0.772 1
353.682 0.772 1
395.436 0.772 1
437.530 0.772 1
```

Run the first level analysis

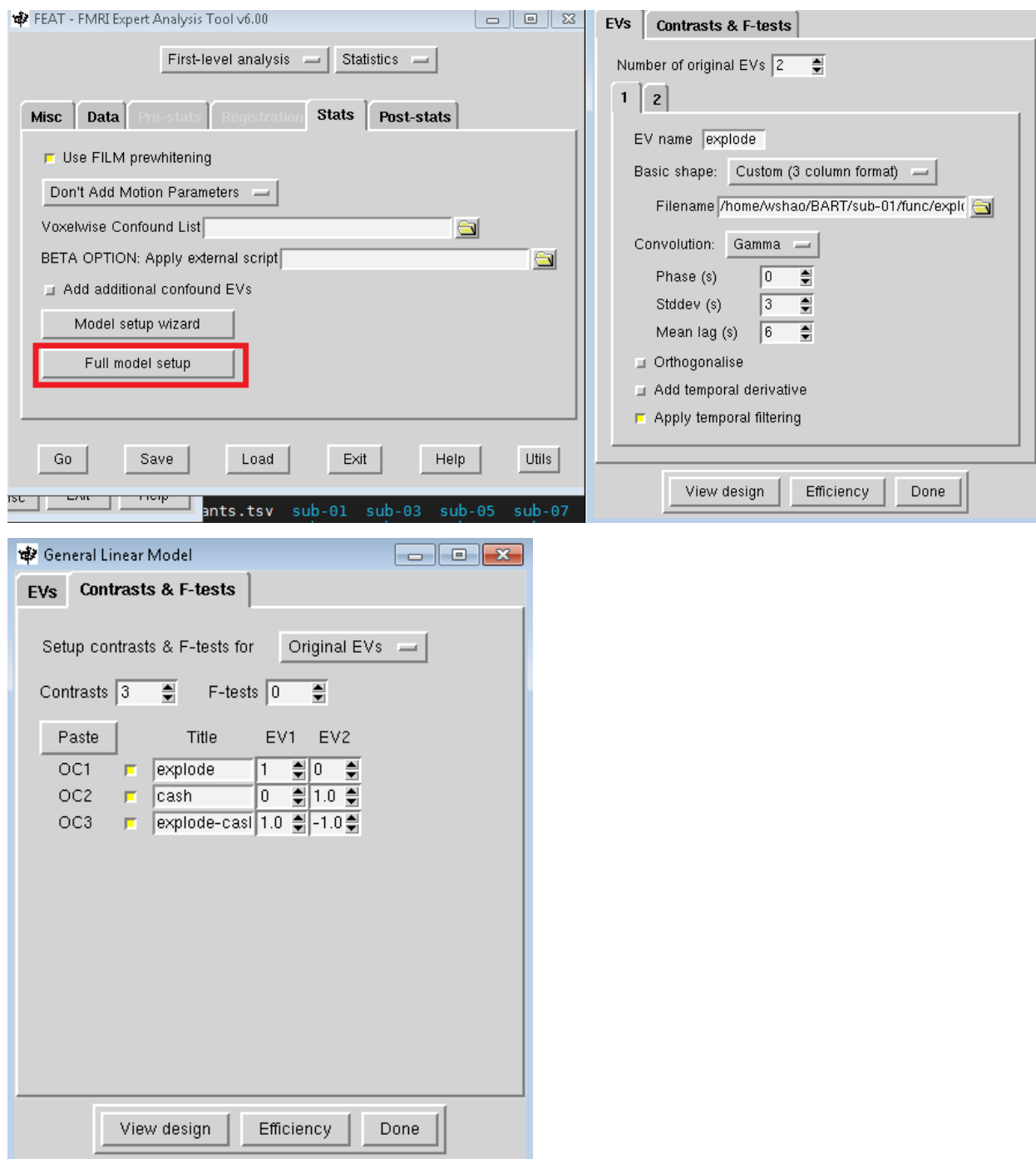
Let's have fun with individual analysis!

Data and Stats tab

Navigate to the sub-01 directory, and type `fs1` from the terminal, open the FEAT GUI, and from the dropdown menu in the upper right of the **Data** tab, change "Full Analysis" to "Statistics". This will grey out the Pre-stats and Registration tabs. You will also see a button called **Input is a FEAT director**. Click on the button, and select the FEAT directory `subj1_run1.feats` that you created before from the preprocessing. Click OK and ignore the warning message.



Next, go to the Stats tab. Click on “Full model setup”, and change the Number of original EVs (or Explanatory Variables, FSL’s term for regressors) to 2. This will create two tabs, one for each regressor. In the EV name field for regressor 1, type “explode”. Click on the dropdown menu next to Basic shape, and select “Custom (3 column format)”. This reveals a field called “Filename”; click on the folder icon to select the timing file explode_run1.txt. Uncheck the “Add temporal derivative” button and click on the “2” tab, repeat these steps, selecting the timing file “cash_run1.txt” at this time.



Design matrix

When you have done with the model set up, click on the **Contrasts & F-tests** tab. This is where you specify which contrast maps you would like to create after the beta weights for each condition have been estimated. In this experiment, we are interested in three contrasts:

- 1 The average beta weight for the explode condition compared to baseline
- 2 The average beta weight for the cash out condition compared to baseline
- 3 The difference of the average beta weights between the explode and cash conditions

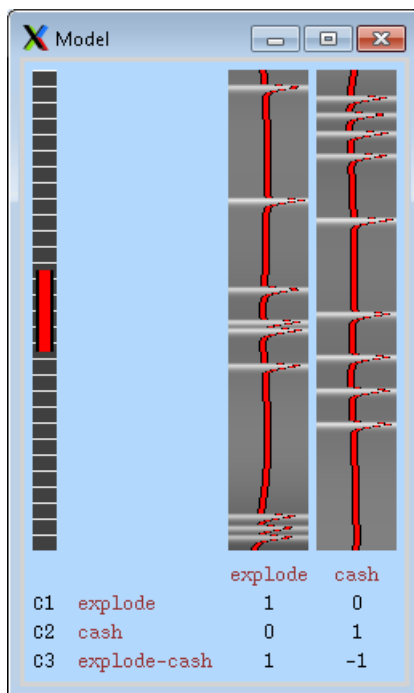
So, Set the number of contrasts to 3, and type the following contrast names in each row, along with the following contrast weights in the EV1 and EV2 columns:

1 explode [1 0]

2 cash out [0 1]

3 explode-cash [1 -1]

Click the Done button (or view design), which will open a **Design Matrix** window. The leftmost column represents the high-pass filter, which is the threshold that removes any frequencies that are longer than the length of the red bar. The two columns on the right represent the ideal time-series for both regressors(IV), and they correspond to the order that the timing files indicated; in other words, the first column is the ideal time-series for the explode condition, and the second column is the ideal time-series for the cash out condition.



The red line represents what we think the time-series of the voxel should look like if it is responsive to that regressor. The white bars represent the HRF that is convolved with the onset of each trial for that condition. The length of the white bar indicates the duration time. Take another look at the timing files for each condition and see if the correspondence between the onset times, duration and the design matrix makes sense.

Post-stats

The last tab in the FEAT GUI is called Post-stats. There are many options here, and the only two function you are likely to change are ones labeled “Z threshold” and “Cluster P threshold”, which are the thresholds that determine which voxels are statistically significant for each contrast. This makes more sense when we do the group-level analysis. For now, just leave it as it is.

After you filled the blanks, Click GO and go to make a nice coffee since it requires 7-10 minutes to process the data.

Automation

Since we have 16 subjects and each subject has 3 runs. In total, we need to repeat all the preprocessing and 1st level analysis 48 times! it is not hard to do but really tedious and you could make errors easily. As Joey from **Friends** said, there's gotta be a better way, and there is.

Here is the script that makes your life easier!

Create a design file

When you analyzed the run1 from sub-01 manually, a directory called run1.feats was created. There are many files and sub-directories within that directory. One of these files, **design.fsf** is the one that contains all of the information that transferred from the FEAT GUI created by you before. If you open up the design.fsf file in a text editor. You can see all the steps and data in which you made before.

Now, try to recall all the steps from your long-term memory system because we will create a design.fsf template for all the steps from preprocessing and 1st level analysis so that we can apply this design.fsf to all the subjects and runs with a few modifications.

First thing first, cd to sub-01 and rm all the **.feats** files we created. Reopen FSL GUI from the sub-01 directory. Click **FEAT FMRI analysis**:

1 Start from Data tab, select **Full Analysis**, use **Select 4D data** to feed the input bold.nii.gz file, name the output directory run1

2 select the brainmask stripping anat image we did at the first time (I hope you still keep this file) and to finish the **Registration** tab as we previous learn

3 go to the **Stats** tab and click the **Full model setup** to repeat the steps we did

After you finish all the steps, instead of clicking Go, click the Save button and name your file as **design_run1**

```
# FEAT version number
set fmri(version) 6.00

# Are we in MELODIC?
set fmri(inmelodic) 0

# Analysis level
# 1 : First-level analysis
# 2 : Higher-level analysis
set fmri(level) 1

# Which stages to run
# 0 : No first-level analysis (registration and/or group stats only)
# 7 : Full first-level analysis
# 1 : Pre-processing
# 2 : Statistics
set fmri(analysis) 7

# Use relative filenames
set fmri(relative_yn) 0

# Balloon help
set fmri(help_yn) 1

# Run Featwatcher
set fmri(featwatcher_yn) 1

# Cleanup first-level standard-space images
set fmri(sscleanup_yn) 0

# Output directory
set fmri(outputdir) "run1"

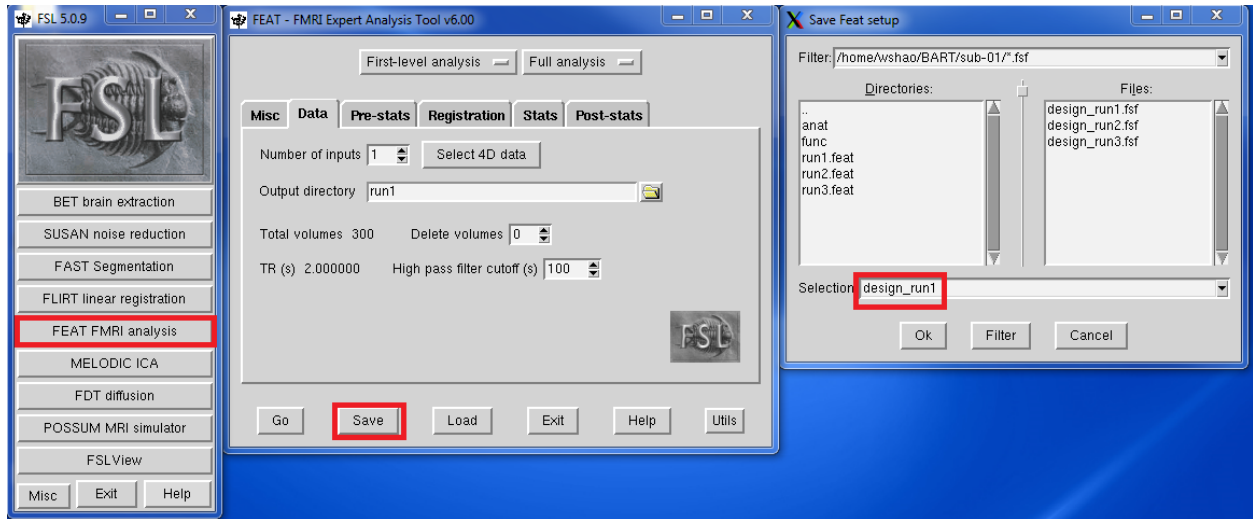
# TR(s)
set fmri(tr) 2.000000

# Total volumes
set fmri(npts) 300

# Delete volumes
set fmri(ndelete) 0

# Perfusion tag/control order
set fmri(tagfirst) 1
```

Fig. 12: for example, output directory is run1, TR is 2, and we have the total volumes 300



Adding the script

Once you create the template design file, what we need to do next is to apply this file into all 16 subjects directory with some changes. And execute the file with FSL.

Once again, here is the script you need to copy and save it as **prepro_model.sh** in your BART directory:

```
#!/bin/bash

# Generate the subject list to make modifying this script
for id in `seq -w 1 16` ; do
    subj="sub-$id"
    echo "====> Starting processing of $subj"
    echo
    cd $subj

    # If the brain mask doesn't exist, create it
    if [ ! -f anat/${subj}_T1w_brain.nii.gz ]; then
        echo "Skull-stripped brain not found, using bet with a fractional intensity_
↳ threshold of 0.35"
        bet2 anat/${subj}_T1w.nii.gz \
            anat/${subj}_T1w_brain_f02.nii.gz -f 0.35
    fi

    # Copy the design files into the all the subject directory, and change "sub-01"
↳ content of original design files to different subject accordingly
    cp ../design_run1.fsf .
    cp ../design_run2.fsf .
    cp ../design_run3.fsf .

    # Change "sub-01" content of original design files to match different subjects_
↳ accordingly
    sed -i "s|sub-01|${subj}|g" design_run1.fsf
    sed -i "s|sub-01|${subj}|g" design_run2.fsf
    sed -i "s|sub-01|${subj}|g" design_run3.fsf
```

(continues on next page)

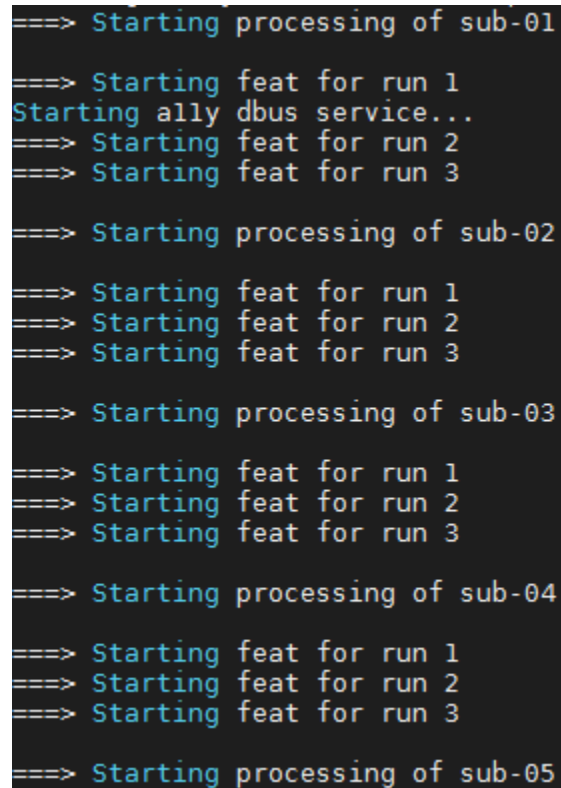
(continued from previous page)

```
# Now everything is set up to run feat
echo "====> Starting feat for run 1"
feat design_run1.fsf
echo "====> Starting feat for run 2"
feat design_run2.fsf
echo "====> Starting feat for run 2"
feat design_run3.fsf
echo

# Go back to the directory containing all of the subjects, and repeat the loop
cd ..
done

echo "job is done"
```

After everything is set, type `bash prepro_model.sh` to run the script and take a break, it is time for another episode from Friends.



```
====> Starting processing of sub-01
====> Starting feat for run 1
Starting ally dbus service...
====> Starting feat for run 2
====> Starting feat for run 3

====> Starting processing of sub-02
====> Starting feat for run 1
====> Starting feat for run 2
====> Starting feat for run 3

====> Starting processing of sub-03
====> Starting feat for run 1
====> Starting feat for run 2
====> Starting feat for run 3

====> Starting processing of sub-04
====> Starting feat for run 1
====> Starting feat for run 2
====> Starting feat for run 3

====> Starting processing of sub-05
```

The script will loop over all of the 16 subjects in the BART dataset and do the preprocessing, statistical model and 1st level analysis for each run. The time should take around 1-2 hours. Be sure to do quality checks for each subject after this has been done.

2nd level analysis

Once the script has been completed for all the runs of all subjects in the BART dataset. you can go further and do 2nd-level analysis. in FSL, 2nd-level is a group analysis in which we averaging together the parameter estimates within each subject and contrast estimates from the 1st-level analyses. Now, cd to the BART directory, open the FEAT GUI from the terminal. from the dropdown menu select “Higher-Level Analysis”.

Select the FEAT directories

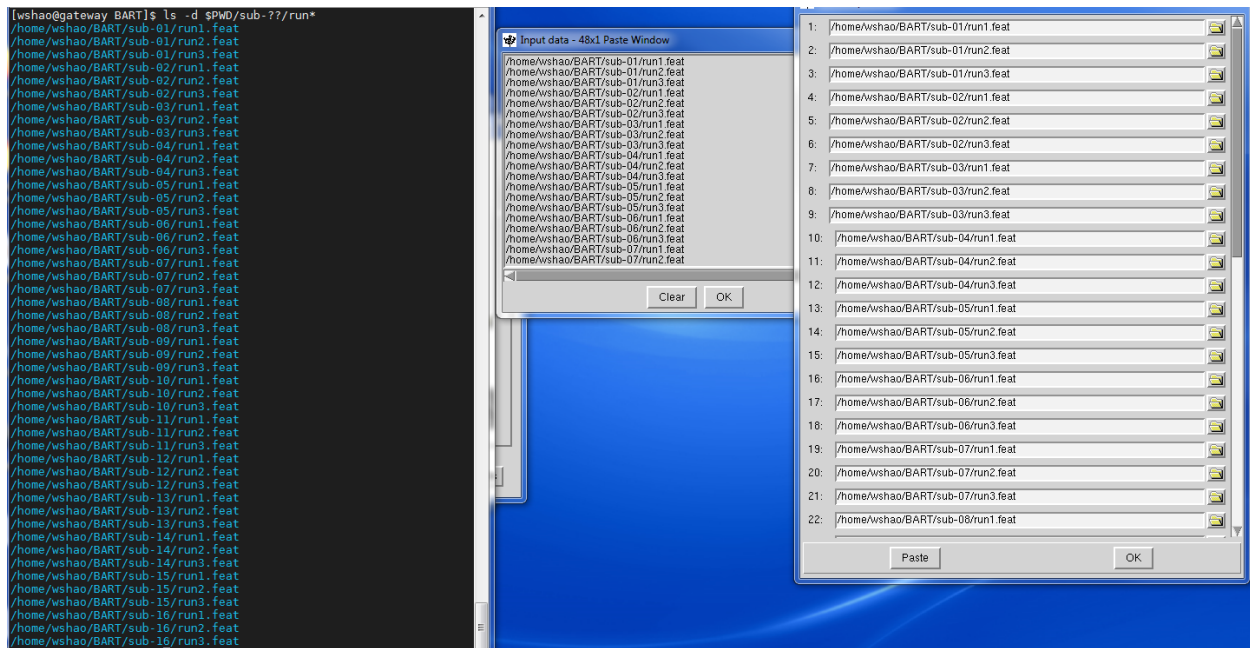
From Data tab, We have two options 1 **Inputs are lower-level FEAT directories** and 2 **Inputs are 3D cope images from FEAT directories**. Choosing option 1 since we have used the FSL to process all the data before. Option 1 is the default setting as well.

Since we have $16 \times 3 = 48$ FEAT directories in total. Change the Number of inputs to 48, and then click the Select FEAT directories, the Paste.

Now, Open a new terminal, cd to the BART directory, type:

```
ls -d $PWD/sub-??/run*
```

This will print an absolute path to each FEAT directory. The -d option means to only list directories, and \$PWD expands to an absolute path pointing to the current working directory. Within the current directory, any directory starting with sub- and ending with two digits (represented by the ??) is added to the path. Finally, within each subject directory, any directory beginning with the string run will be appended to the path name (e.g., run1.feats and run2.feats).



Use Ctrl+c and Ctrl+y to copy and paste the 48 runs.feats into the dashboard.

In the Data tab, you will also need ensure to select 3 lower-level copes boxes. This would tell FSL to run a 2nd-level analysis for 3 contrast, which correspond to:

- 1 The contrast estimate for the explode condition
- 2 The contrast estimate for the cash-out condition
- 3 The contrast estimate for explode vs cash-out

In the Output directory tab, type BART_2ndLevel1. This is where the results of the 2nd level analysis will be saved.

Creating the GLM

The Stats tab will look different from when you used it for 1st-level analysis - you can now choose different types of models:

1 Fixed Effects: Do not generalize from the sample - just take the average

2 Mixed Effects: Simple OLS (Ordinary Least Squares): This will perform a t-test on the average parameter estimates calculated for each subject, without taking into account the variability between the runs for each subject

3 Mixed Effects: FLAME 1: Weight each subject's parameter estimate by the variance of that contrast estimate. In other words, a subject with relatively low variance will be weighted more, and a subject with relatively high variance will be weighted less

4 Mixed Effects: FLAME 1+2: A more rigorous version of FLAME 1. It takes much longer, and is only helpful for analyzing small samples (e.g., 10 subjects or fewer)

Given we only want to know the average of the parameter estimates across the runs within each subject. So, we will use the Fixed Effects option. select it and go to ``Full Model Setup`` button.

This will display a window with the number of rows representing the number of individual parameter estimates - in our case, 48. For the Number of main EVs, change this to 16, which is the number of subjects in our dataset. Then change the numbers in each column to 1 where you want to take the average for the parameter estimates for that subject. In this case, the first three rows for column 1 would be changed to 1, and the next three rows for column 2 would be changed to 1, and so on.

	Group	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8	EV9	EV10	EV11	EV12	EV13	EV14	EV15	EV16
Input 1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 4	1	1	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 5	1	1	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 6	1	1	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 7	1	1	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 8	1	1	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 9	1	1	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0
Input 10	1	1	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0
Input 11	1	1	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0
Input 12	1	1	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0
Input 13	1	1	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0
Input 14	1	1	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0
Input 15	1	1	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0
Input 16	1	1	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0
Input 17	1	1	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0
Input 18	1	1	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0
Input 19	1	1	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0
Input 20	1	1	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0
Input 21	1	1	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0
Input 22	1	1	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0
Input 23	1	1	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0
Input 24	1	1	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0
Input 25	1	1	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0
Input 26	1	1	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0
Input 27	1	1	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0
Input 28	1	1	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0
Input 29	1	1	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0
Input 30	1	1	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0

When you have finished the EVs, click on the Contrasts & F-tests tab, and change the number of Contrasts to

16. Change all of the numbers on the diagonal to 1. This will create a single contrast estimate for each subject that is the average of that subject's parameter estimates.

General Linear Model

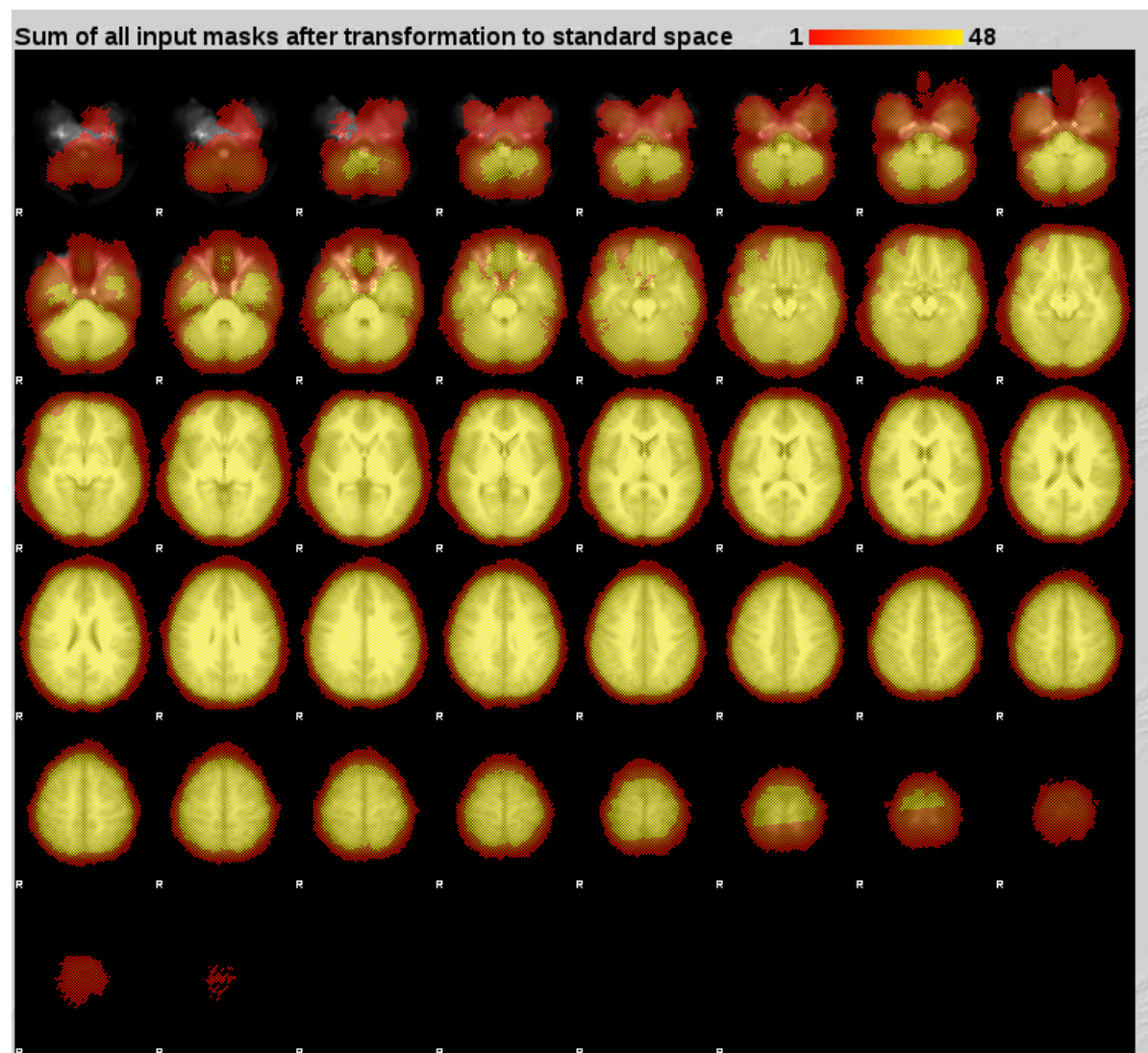
EVs Contrasts & F-tests

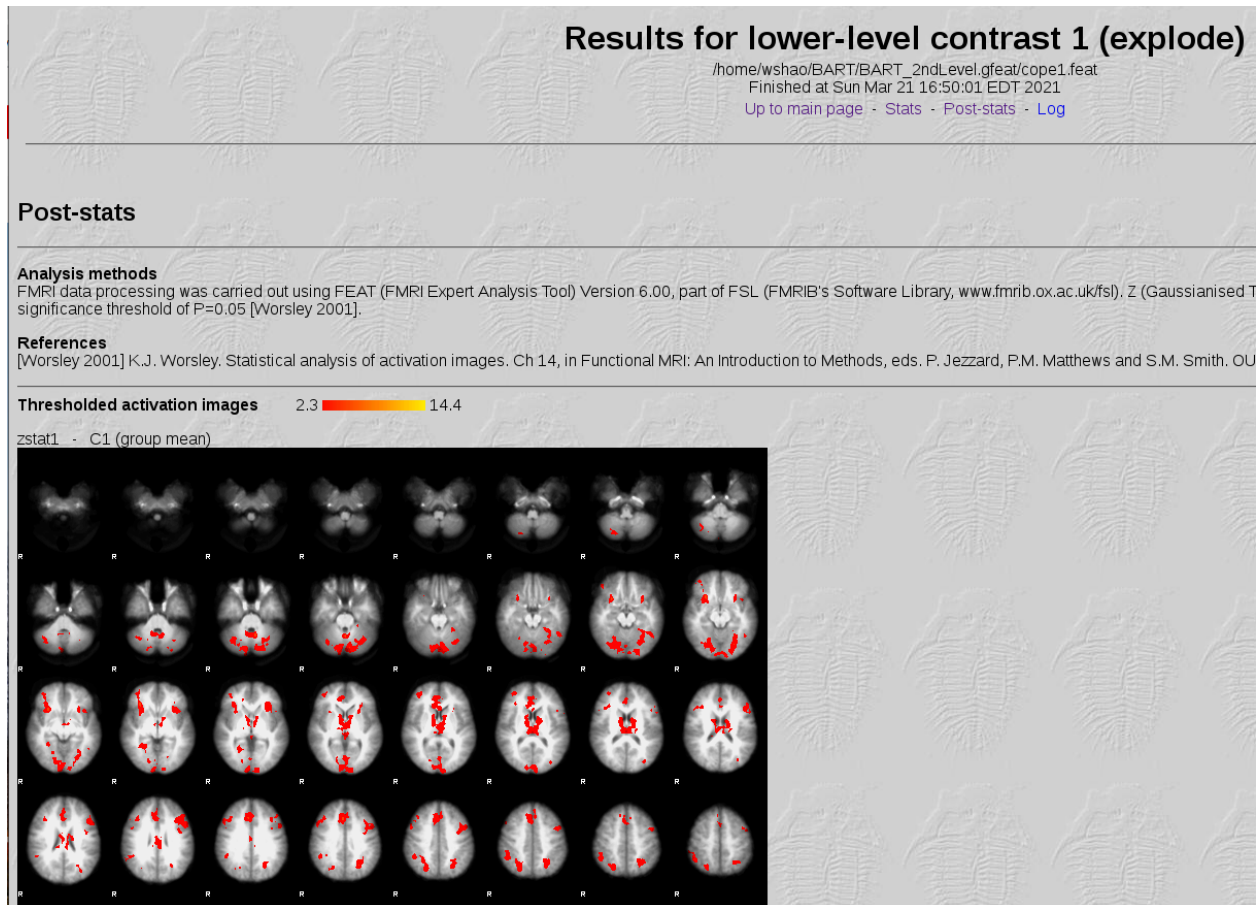
Contrasts 16 F-tests 0

Paste

	Title	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8	EV9	EV10	EV11	EV12	EV13	EV14	EV15	EV16
C1	group mean	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2		0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C3		0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0	0
C4		0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0	0
C5		0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0	0
C6		0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0	0
C7		0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0	0
C8		0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0	0
C9		0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0	0
C10		0	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0	0
C11		0	0	0	0	0	0	0	0	0	0	1.0	0	0	0	0	0
C12		0	0	0	0	0	0	0	0	0	0	0	1.0	0	0	0	0
C13		0	0	0	0	0	0	0	0	0	0	0	0	1.0	0	0	0
C14		0	0	0	0	0	0	0	0	0	0	0	0	0	1.0	0	0
C15		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0	0
C16		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.0

When you complete the GLM and contrasts setting, click Done, a design matrix will show up. make sure it is ok, and click Go.





Congratulations! You made it. Now, let's take a break and go to the 3rd level analysis.

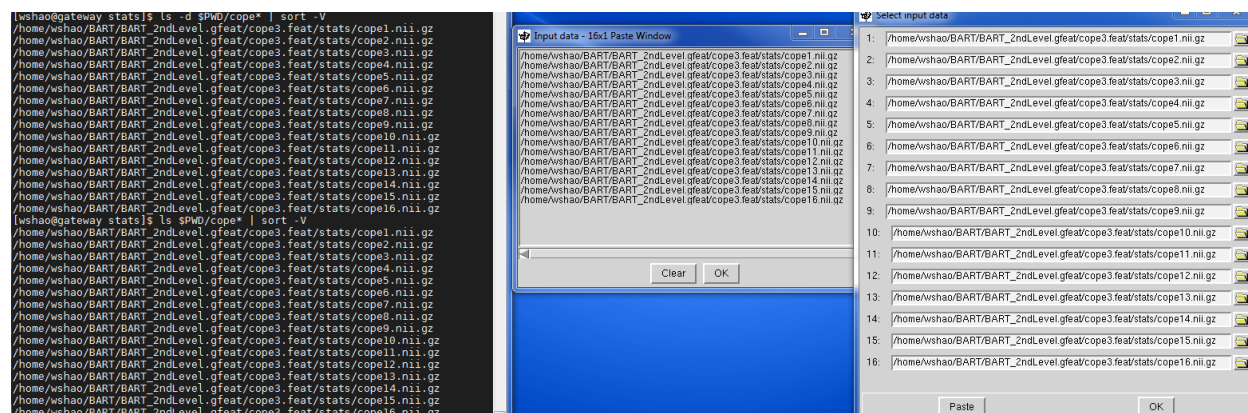
3rd level analysis

One of the goals in the fMRI analysis is to generalize the results of the sample to the population. If we see changes in brain activity in the sample, how can we ensure that these changes would likely be seen in the population? To answer that question, a 3rd-level analysis, group-level analysis, is needed. We need to calculate the standard error and the mean for a contrast estimate, and then test whether the average estimate is statistically significant.

FSL can only run one model at a time. In this example, we will run a 3rd-level analysis for the contrast of explode-cash (it is cope3 from the 2nd-level analysis because it was the third contrast that was specified).

Loading the Data

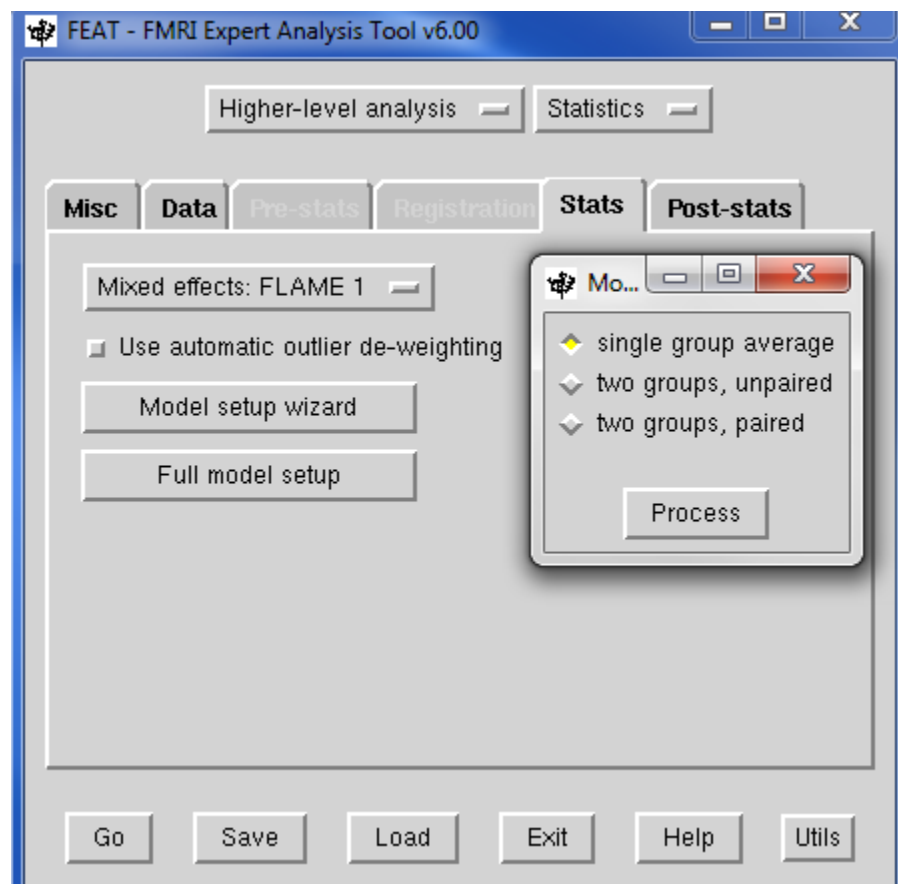
From the BART directory, open the FEAT GUI. As we did in the 2nd-level analysis, select Higher-level analysis, instead of input are lower-level FEAT directories, choose Inputs are 3D cope images from FEAT directories, and change the number of inputs to 16. Since the 2nd-level analysis generated an average contrast of parameter estimate (cope) for each subject for each contrast that was specified in our model. Regarding selecting the FEAT directories, we can copy and paste a list of the cope images: Click on Select cope images and then click the Paste button. In a new Terminal, navigate to the directory `BART_2ndLevel.gfeat/cope3.feats/stats`, and type `ls $PWD/cope* | sort -V`. This will list all of the cope images in numerical order, copy and paste the list into the Input data dashboard window by typing `Ctrl+c` and `ctrl+y`. After clicking OK, label the output directory `BART_3rdLevel_explode-cash`.



Creating the GLM

Move to the Stats tab. For a 3rd-level analysis, we will use the Mixed Effects FLAME 1. This model assigns weight based on the variance to ensure that our results of the sample are generalizable to the population our sample was drawn from. It provides accurate parameter estimates by using information about both within-subject and between-subject variability.

Due to the simple design, we can quickly create a GLM by using the Model setup wizard. as we have already taken the contrast for each subject, single group average would be a good choice. When you click Process, you should see a Model representation that looks like this:



The Post-Stats Tab

Now we finally could discuss the **Post-Stats** tab. we only care about in here the Thresholding options:

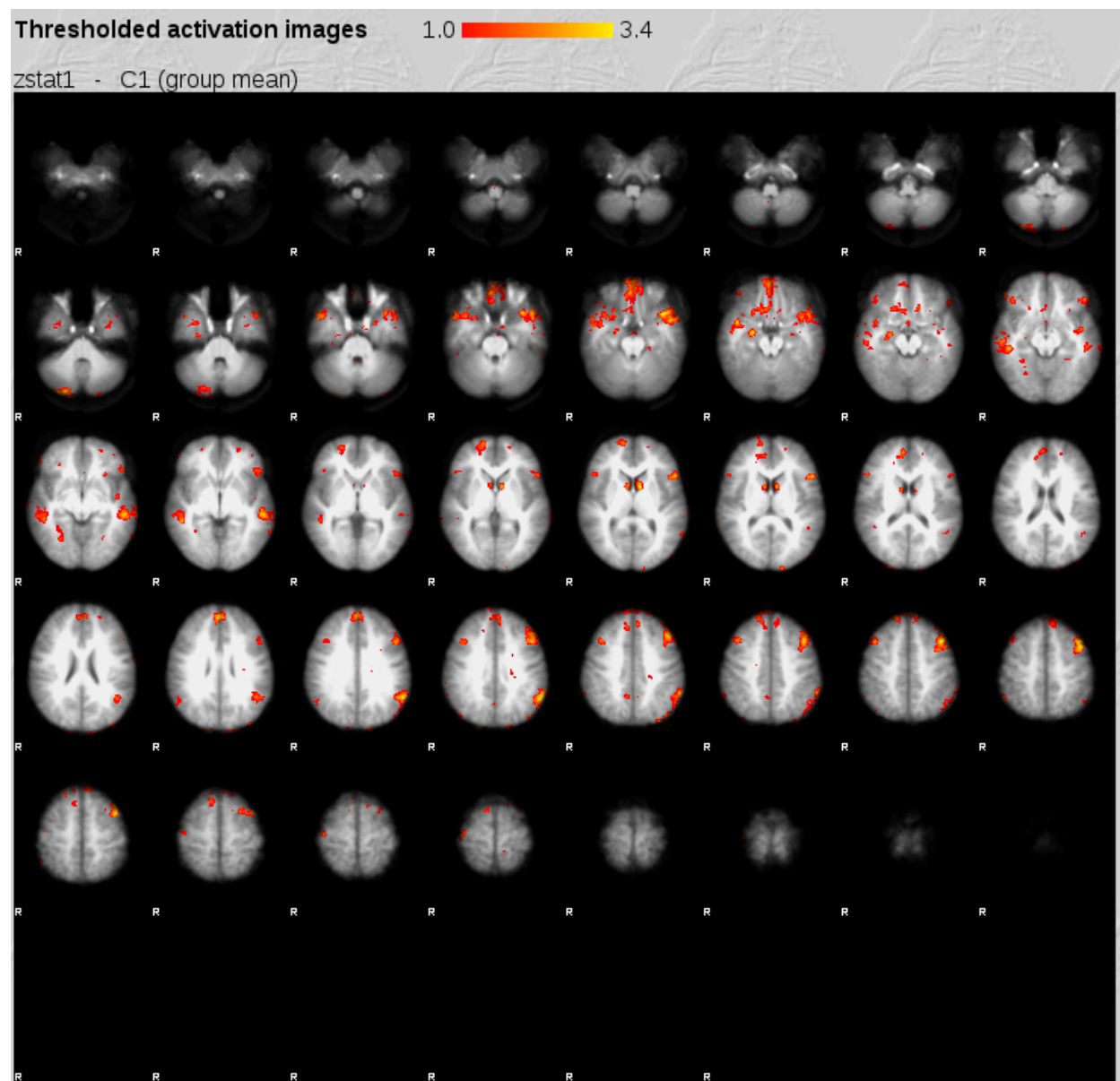
- 1 None won't do any thresholding (i.e., show the parameter estimate at every voxel, regardless of significance)
- 2 Uncorrected will allow any individual voxels to pass the threshold specified in Z-threshold (e.g., here we would only show voxels that have a value greater than 3.1);
- 3 Voxel will perform a type of maximum height thresholding based on Gaussian Random Field theory, which is less conservative than a Bonferroni test
- 4 Cluster, which uses a cluster-defining threshold (CDT) to determine whether a cluster of voxels is significant. The logic behind this approach is that neighboring voxels are not independent of one another, and this reduced degrees of freedom is taken into account when estimating significance.

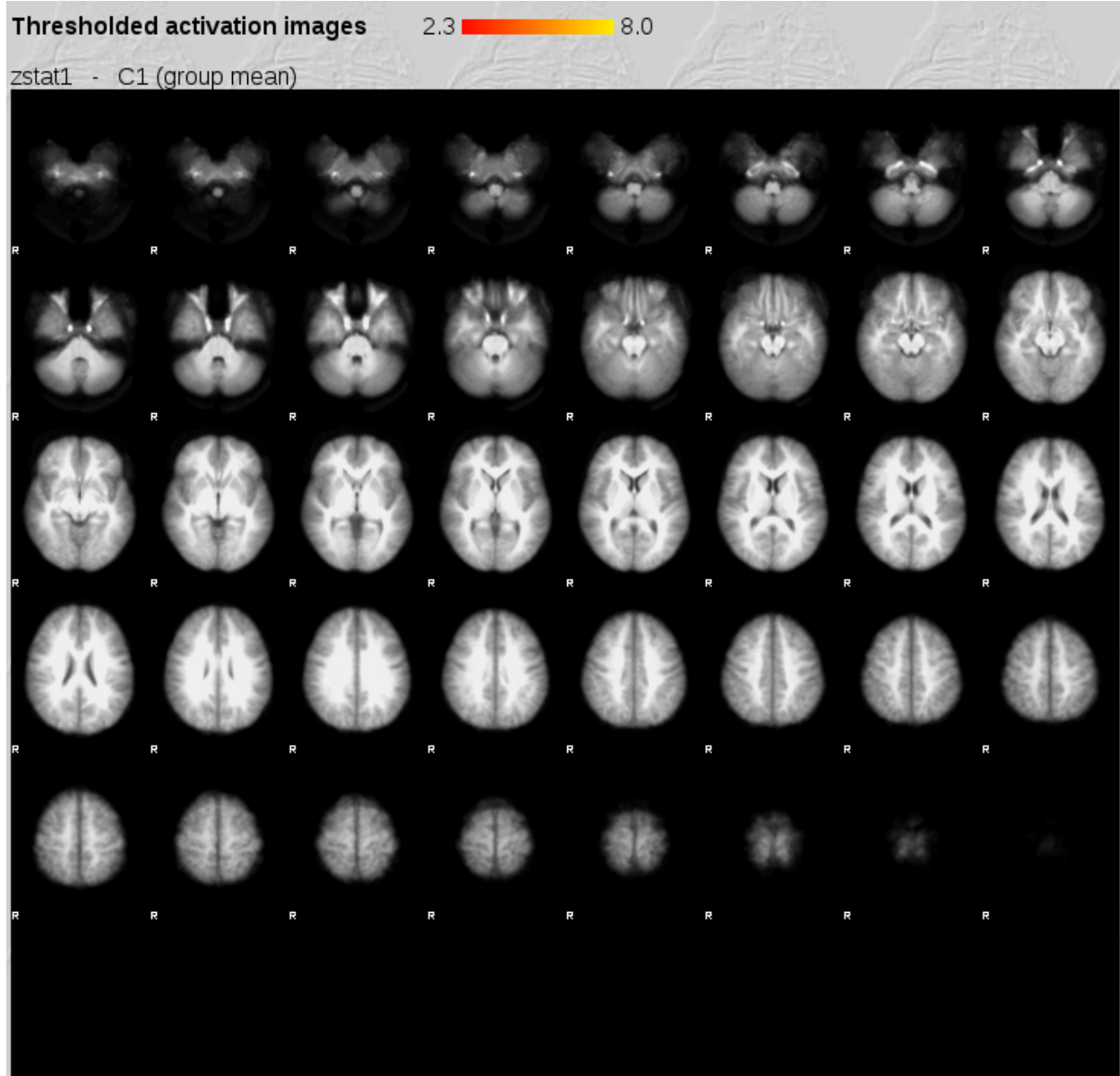
For example, if we leave our Z-threshold at 3.1 and our Cluster p-threshold at 0.05, we will look for clusters composed of voxels that each individually pass a z-threshold of 3.1. FSL runs simulations to see how often we would get clusters of certain sizes with each of their constituent voxels passing that z-threshold, and creates a distribution of cluster sizes for that CDT. Cluster sizes that occur less than 5% of the time in the simulations for that CDT are then determined to be significant.

The default of a Cluster correction setting for most analysis is that $z=3.1$ and a cluster threshold of $p=0.05$, which is reasonable unless you have a good reason to change it. Now click Go. This process will take about 5-10 minutes, depending on how powerful your computer is.

Reviewing the Output

In the FEAT HTML output, you will see the thresholded z-statistic image overlaid on a template MNI brain. These are axial slices, and they give you a quick overview of where the significant clusters are located.





As you can see, I have run two tests with $z = 1$ and 2.3. as you can see, the results will change according to the z score.

In order to take a closer look at the results, open `fslview` and load a standard template, such as `MNI152_T1_2mm_brain`. Then load the `thresh_zstat1.nii.gz` image, located in `BART_3rdLevel_explode-cash.gfeat/cope3.feats`. This image only shows those clusters that were determined to be significant based on the criteria you specified in the Post-stats tab. Change the color scheme to “Red-Yellow”, and change the “Min.” value to 3.1. You can also click on the Gear icon and change the interpolation to make the results look smoother. Lastly, click on a cluster in the dorsal medial prefrontal cortex area, and turn the crosshairs off by clicking on the crosshairs icon. You can then take a snapshot of this montage with the Camera icon, and include the image as a figure in your manuscript.

ROI analysis

As we just completed a group-level analysis, and identified some regions of the brain show a significant difference under the condition of the experiment. Now, we are going to continue our learning with **region of analysis(ROI)**. This is called a **whole-brain** or **exploratory analysis**. When we doesn't have a hypothesis to test, these types of studies are beneficial.

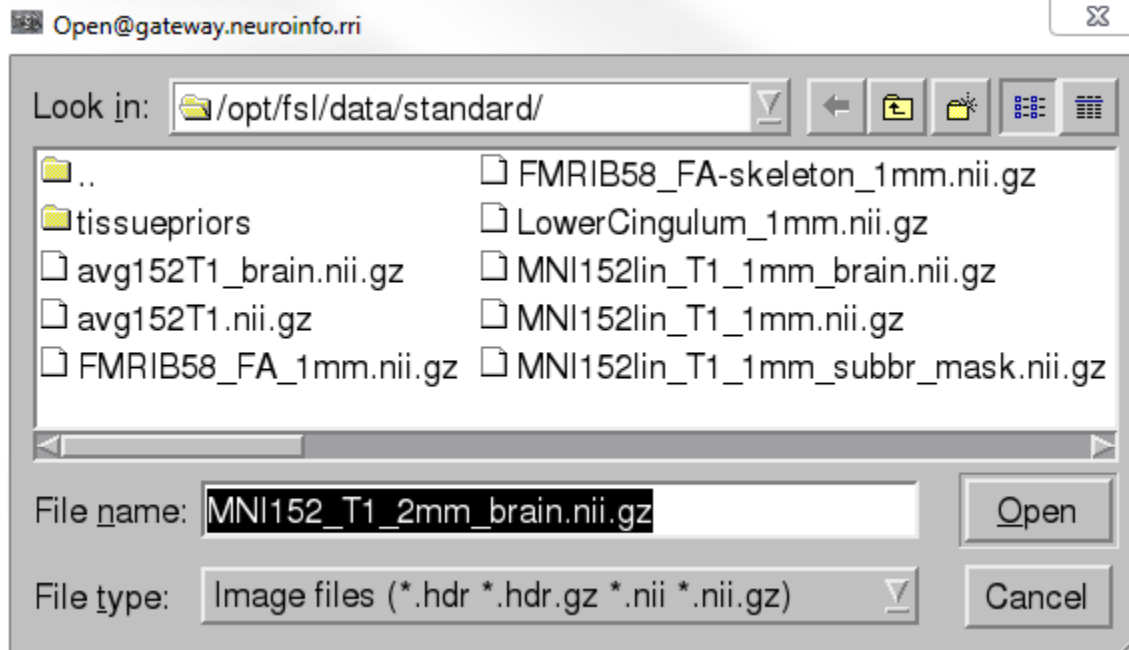
While a large number of studies have been run about a specific topic, we can begin to make more specific hypotheses about where we should find our results in the brain images. For instance, memory has been studied for many years, and many fMRI studies have been published about it using different paradigms that compare different memory tasks. In general, significant increases in the BOLD signal during various memories conditions are seen in a region of the brain known as the Hippocampus and medial temporal lobe. For this BART study, then, we could restrict our analysis to this region and only extract data from voxels within that region. This is known as a ROI analysis. In short, a general name for an analysis in which we choose to analyze a region selected before look at whole-brain results is called a confirmatory analysis.

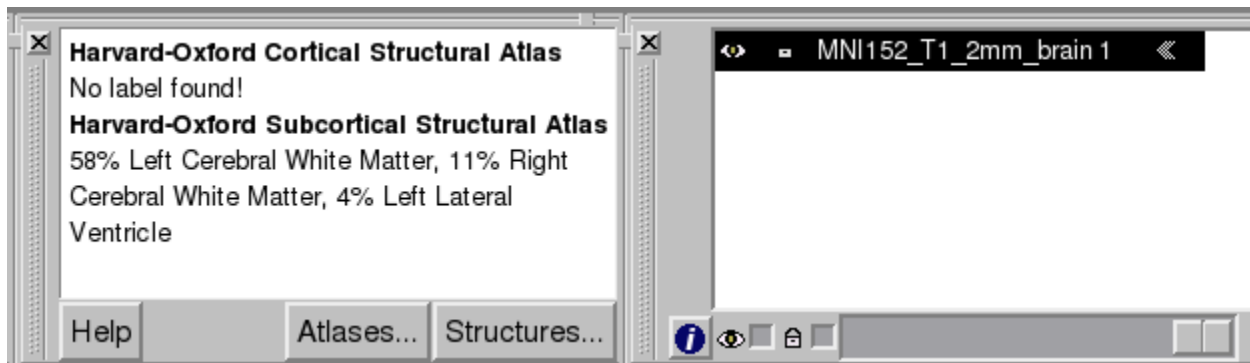
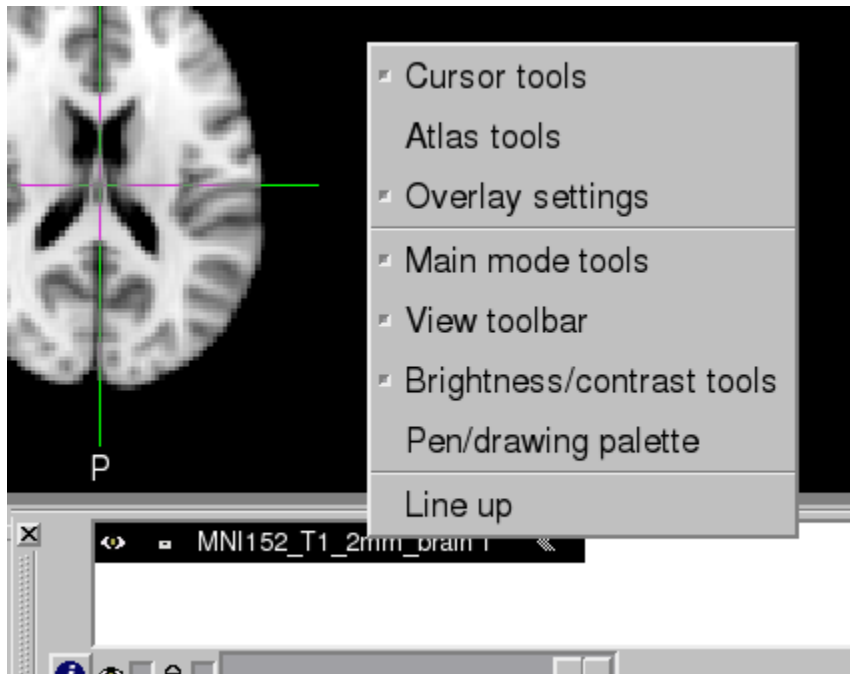
In terms of BART study, Whole-brain maps can hide important details about the effects that we're studying. As you may find a significant effect of BART conditions, but the reason the effect is significant could come from a greater change of cash than explode, or because explode is much more negative than cash, or some combination of the two. The only way to determine what is driving the effect is with ROI analysis, and this is especially important when dealing with interactions and more sophisticated designs.

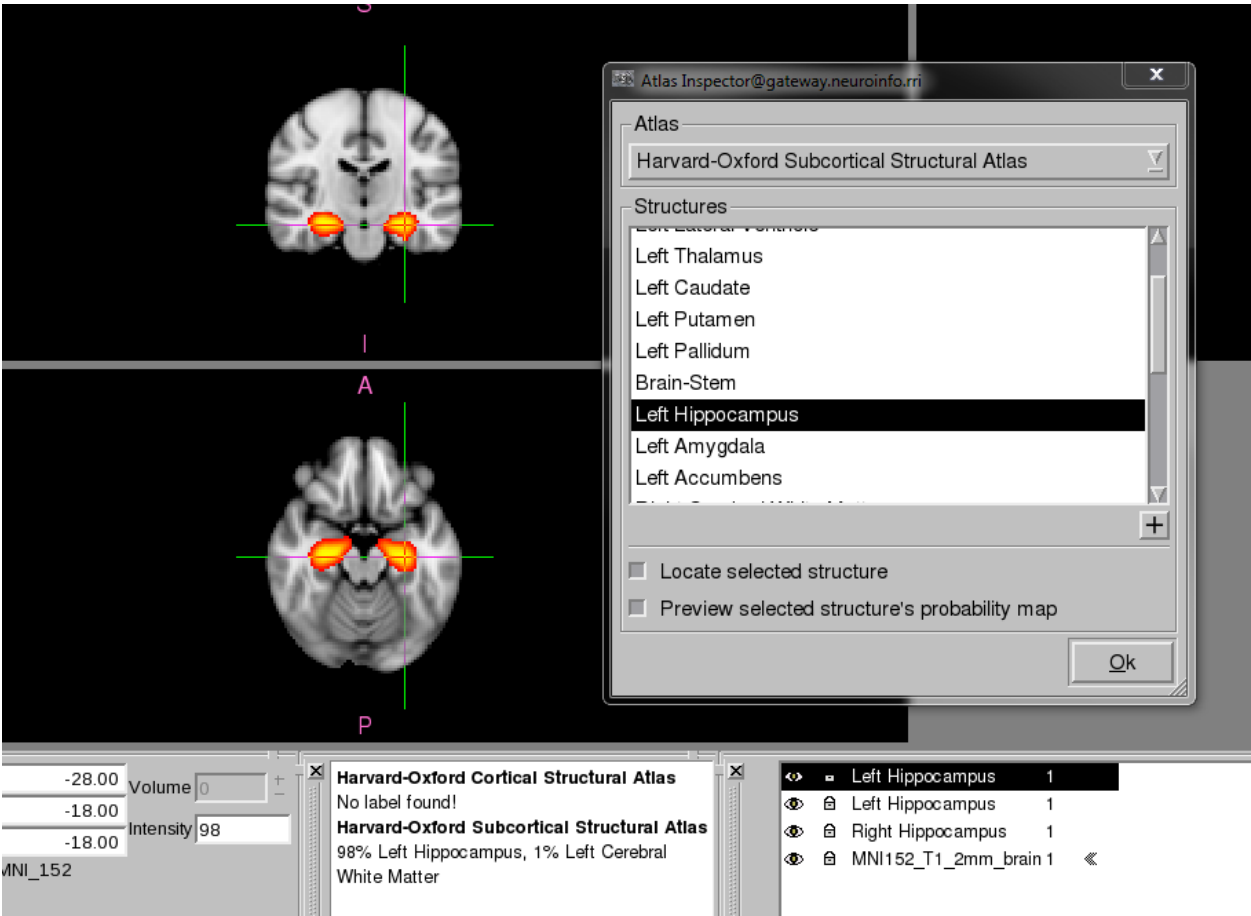
Atlases

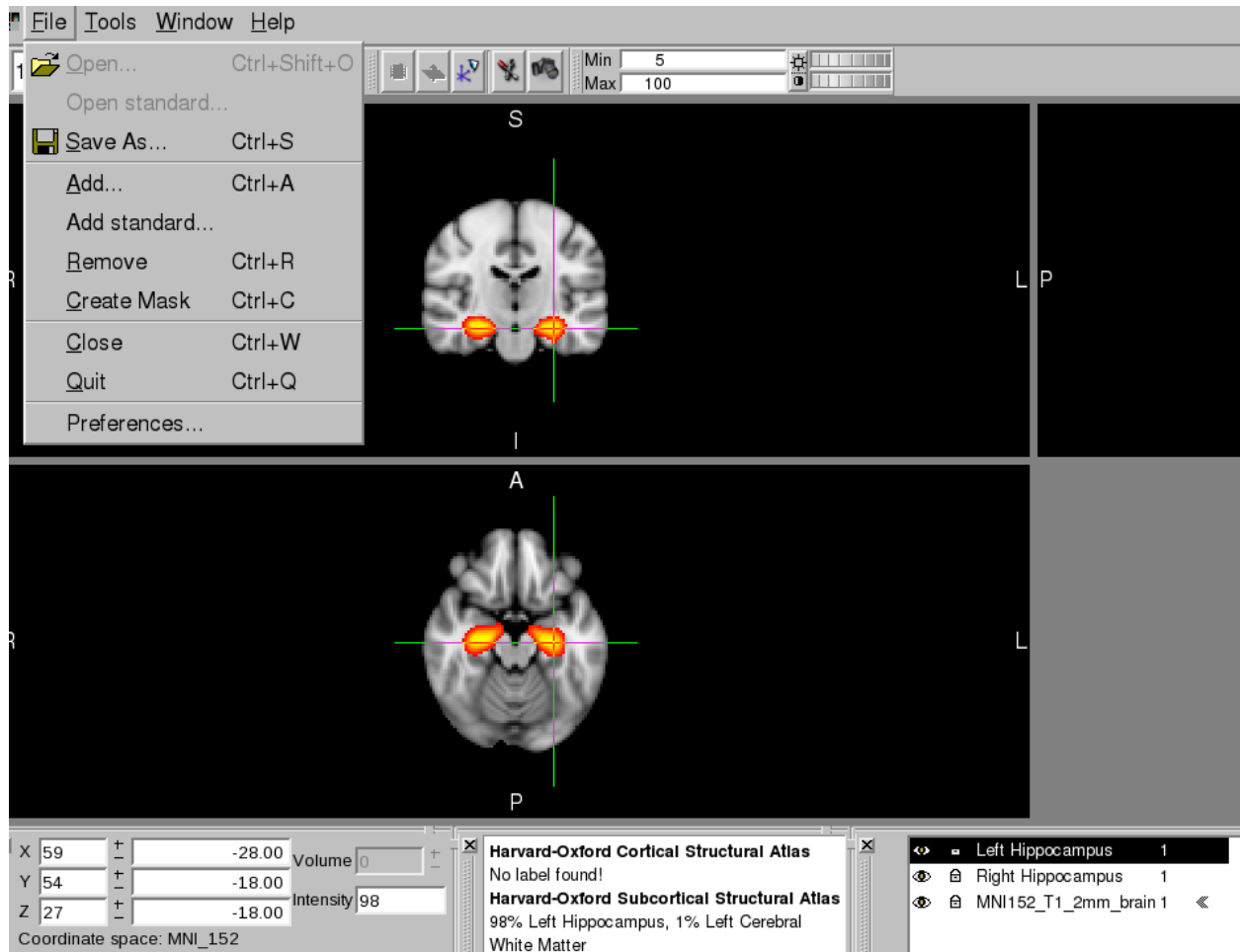
One way to do an ROI analysis is to use an atlas, a map that partitions the brain into anatomically distinct regions.

As you may know, Many atlases are already installed on FSL, and we can access them by using the FSL viewer. Open FSL view from FSL_gui and click file -> Open standard -> choose MNI standard space such as MNI152_T1_2mm_brain.nii.gz. you will see the Standard space, then, click the Atlas tools from the bottom box, a new window will appear on the left side, click Atlases to make sure the default Atlases Harvard-Oxford Cortical and Subcortical Atlases are loaded. Then, click the Structures and use Harvard-Oxford Subcortical Atlas, find the left hippocampus and click the + button to add the right hippocampus, click OK and you will see the two regions (left and right hippocampus) are highlighted, click file to save as lh_hippo.mask and rh_hippo.mask respectively.









Note: our results will have the same resolution as the template we used for normalization. The default in FSL is the MNI_152_T1_2mm_brain, which has a resolution

of 2x2x2mm. When you create a mask, it will have the same resolution as the template that it is overlaid on. When we extract data from the mask, the data and the mask need to have the same resolution. To avoid any errors due to different image resolutions, use the same template to create the mask that you used to normalize your data.

Extracting the data from mask

After generated the mask, we can use it to extract the contrast estimates for each participant. We want extract result from the 2nd-level analysis, not the 3rd-level analysis since the 3rd-level analysis is a image with a single number at each voxel. Our purpose in a ROI analysis is to extract the contrast estimate for each subject separately.

From the directory `BART_fsl/BART_2ndLevel.gfeat/cope3.feats/stats`, we can find tge subject's data maps for the cash-explode contrast condition such as T-statistic maps, cope pictures, and variance pictures, the data maps are calculated base from these methods. let's extract data from z-statistic maps since this data have been transformed into a normally distributed format and it is quit easier to plot and read.

We'll combine all of the z-statistic maps into a single dataset in order to make ROI analysis easier. It requires a combination of FSL and Unix commands to do this. cd to `BART_fsl/BART_2ndLevel.gfeat/cope3.feats/stats`, and type:

```
fslmerge -t allZstats.nii.gz `ls zstat*` | sort -V`
```

This command will combine all of the z-statistic images into a single dataset along the time series (flag -t option). The first argument specifies the name of the output dataset (allZstats.nii.gz), and the code in backticks use asterisk wildcard function to find all files begin with “zstat,” and then arranges them numerically from smallest to largest with the -V option.

Type `mv allZstats.nii.gz ../../..` to move the allZstats.nii.gz file up three levels to the main BART_fsl directory. Then, to extract the data from the lh_hippo.mask, use the `fslmeants` command:

```
fslmeants -i allZstats.nii.gz -m lh_hippo.mask.nii.gz
```

This will print 16 numbers, one per subject. Each number is the contrast estimate for that subject averaged across all of the voxels in the mask. To be more specific, the first number that corresponds to the average contrast estimate for cash-explode in sub-01. The second number, for sub-02, and so on. These numbers can be copied and pasted into your preferred statistical software tool (such as R), where you can then perform a t-test on them.

```
[wshao@gateway BART_fsl]$ fslmeants -i allZstats.nii.gz -m lh_hippo.mask.nii.gz
-0.151528
-0.744174
-1.168797
-1.025469
-0.374579
0.324745
-0.515714
-0.706206
0.605598
-0.289674
-0.655423
-1.146909
-0.257650
0.406815
-1.524155
0.360979
```

Now, you can use the right hippocampus mask and repeat the steps above to do an ROI analysis for right hippocampus on cash-explode condition. Have fun!

1.18 Change orientation in FSL

One of the useful function from FSL is `fslswapdim`, it can change the orientation of the image. Therefore, you can have the same orientation between T1 and T2 to run the further process.

let's say you have a T2 image looks like this:

```
Data Axes Approximate Orientation:
  first (x) = Right-to-Left
  second (y) = Inferior-to-Superior
  third (z) = Anterior-to-Posterior [-orient RIA]
R-to-L extent: -102.586 [R] -to- 116.984 [L] -step- 0.430 mm [512 voxels]
A-to-P extent: -3.715 [A] -to- 78.285 [P] -step- 2.000 mm [ 42 voxels]
I-to-S extent: -168.741 [I] -to- 50.829 [S] -step- 0.430 mm [512 voxels]
Number of values stored at each pixel = 1
-- At sub-brick #0 '?' datum type is short
```

but the T1 image like this

```

Data Axes Orientation:
  first (x) = Right-to-Left
  second (y) = Posterior-to-Anterior
  third (z) = Inferior-to-Superior [-orient RPI]
R-to-L extent:  -93.487 [R] -to-  97.513 [L] -step-  1.000 mm [192 voxels]
A-to-P extent: -157.601 [A] -to-  97.399 [P] -step-  1.000 mm [256 voxels]
I-to-S extent: -135.555 [I] -to-  23.445 [S] -step-  1.000 mm [160 voxels]
Number of values stored at each pixel = 1
-- At sub-brick #0 '?' datum type is short: 0 to 940
HISTORY

```

So, you can want to change the orientation so that the T1 and T2 can match, type:

```
fsflswapdim T2.nii.gz x -z y T2.reorientated.nii.gz
```

Now, you get the new orientated T2

```

Data Axes Approximate Orientation:
  first (x) = Right-to-Left
  second (y) = Posterior-to-Anterior
  third (z) = Inferior-to-Superior [-orient RPI]
R-to-L extent: -102.747 [R] -to-  116.824 [L] -step-  0.430 mm [512 voxels]
A-to-P extent: -16.256 [A] -to-   65.744 [P] -step-  2.000 mm [ 42 voxels]
I-to-S extent: -125.158 [I] -to-   94.412 [S] -step-  0.430 mm [512 voxels]
Number of values stored at each pixel = 1
-- At sub-brick #0 '?' datum type is short

```

1.19 Overview in AFNI



Let's enjoy the main course 3 AFNI!

So, What is AFNI?

AFNI (Analysis of Functional NeuroImages) is a software suite developed by C, Python, R programs and shell scripts, it is built for the analysis and display of multiple MRI modalities: anatomical, functional MRI (fMRI) and diffusion weighted (DW) data. It is freely available for research purposes. The software is made to run on virtually any Unix system with X11 and Motif displays. Binary packages are provided for MacOS and Linux systems such as Fedora, CentOS/Red Hat and Ubuntu (which includes the Windows Subsystem for Linux).

Please follow this [link](#) to find the suitable AFNI for your computer and install

The goal of this chapter is to show you how to run fMRI analysis with AFNI step by step. After successfully install AFNI and set up, we are going to analyze a dataset with AFNI so that we can gain some first-hand experiences. The data is Balloon Analogue Risk Task. I hope after this chapter, you will be able to apply the knowledge of AFNI in the future and practice with other datasets you have.

We will start with the dataset downloading, preprocessing, 1st level, and end with group analysis as follow:

1.19.1 BART and data downloading

One of the interesting topics from psychology is risky behavior. In order to gain a better understanding of how the brain reacts when people make a decision, there are many many interesting experiment paradigms, Balloon analog risk task is one of them.

In Balloon analog risk task(BART), Participants can manipulate the Balloon to control the risk and reward. Tom Schonberg and his colleagues scanned participants using fMRI while they completed the Balloon Analog Risk Task. As the picture shows, escalating risk-taking occurs under uncertainty and might be experienced either as the accumulation of greater potential rewards, or as exposure to increasing possible losses and decreasing expected value.

In the BART, subjects inflate simulated balloons, and accrue monetary rewards for each successive “pump” during a particular trial. A trial is defined as a balloon that can be pumped a certain number of times and the trial can conclude in two different ways. On the one hand, the participant can “cash-out” at any point during the trial and secure the cumulative winnings up to that point for that balloon in their cumulative total “bank.” On the other hand, a balloon may explode. In this case, participants would lose the money accumulated on that trial alone (but not the total accumulated during previous cash-out trials). Each trial began with a balloon displaying a value of \$0.25 and the value of the balloon increased by \$0.25 for each successive pump. During each trial, participants were presented with one of three types of “reward” balloons, each having a different explosion probability and signified by a different color: red, green, or blue. But participants was not informed the odds of explosion probability.

As a control task, participants intermittently inflated a gray “control” balloon (maximum 12 pumps) that did not explode and had no associated monetary value. The participants were instructed to inflate the control balloon until it disappeared from the screen, and the next trial began. Unlike with reward balloons, participants had no control over how many times they could inflate the control balloon before the trial ended.

for more information, please go [here](#)

Downloading the data

As an Open Source dataset, BART dataset has a standardized structure: Each subject folder contains an anatomical directory and a functional directory labeled anat and func, these contain the anatomical and functional images collected during the experiment. The func directory also contains onset times, or timestamps for when the subject underwent different trials. This format is known as Brain Imaging Data Structure, as we saw in previous chapter BIDS. As an example of the BIDS format. The func directory of BART contains functional data, three runs of timestamps of which condition happened at what time. You can open these as a text file or as a spreadsheet.

Now, go to [there](#) , download the data and save it as BART_afni in our home directory.

```
[wshao@gateway BART]$ ls
CHANGES      participants.tsv  sub-01  sub-03  sub-05  sub-07  sub-09  sub-11  sub-13  sub-15  task-balloonanalogrisktask_bold.json
dataset description.json  README      sub-02  sub-04  sub-06  sub-08  sub-10  sub-12  sub-14  sub-16  timing.sh
```

As you can see, we have 16 subjects. **participants.tsv** will tell you the demographic information of subjects, **task-balloonanalogrisktask_bold.json** contains TR information. You can preview all of these information from the Open-Neuro data web.

Let's take a close look

```
anat  func
```

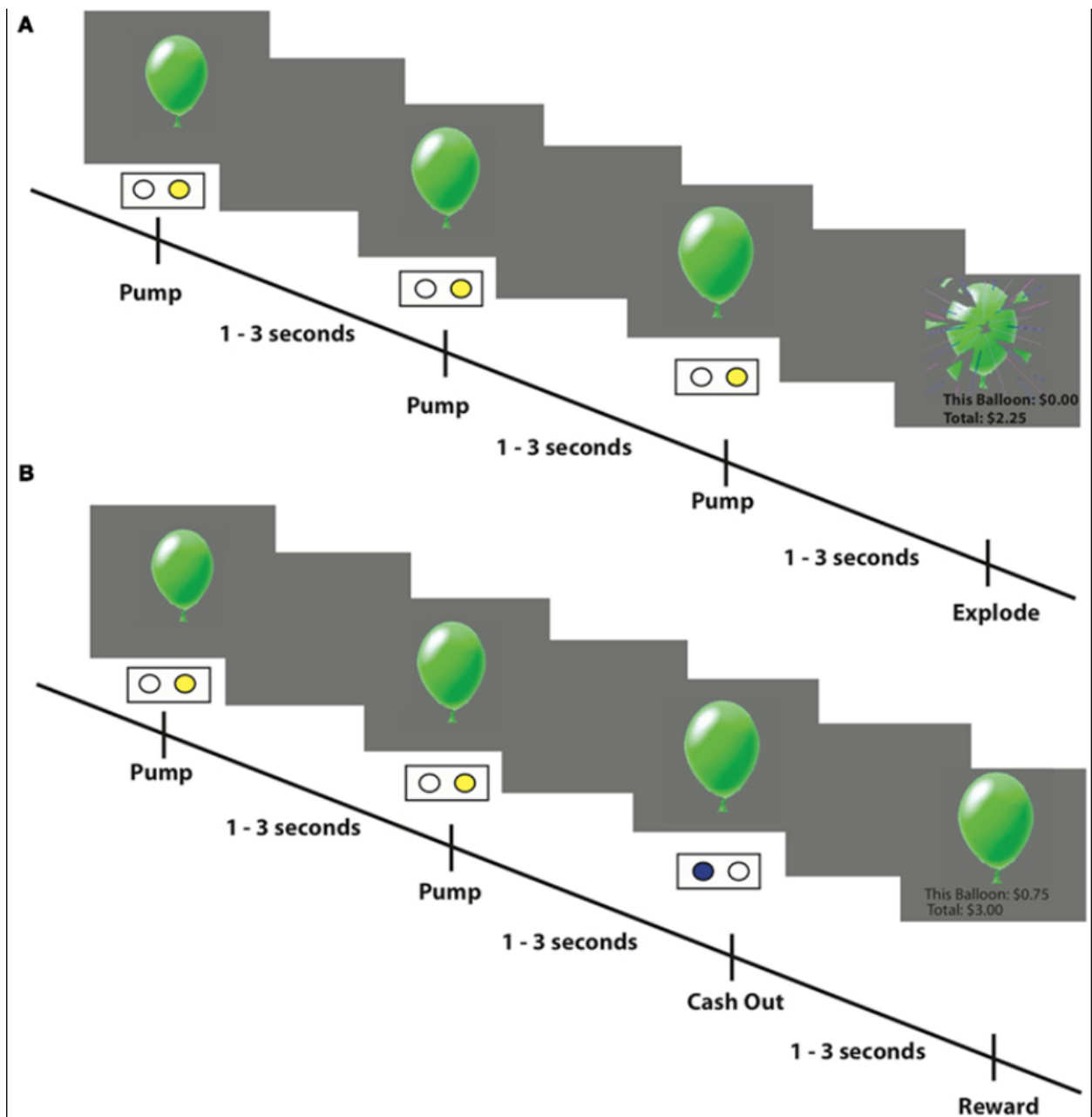


Fig. 13: (A) An example of an explosion trial: participants press one of two buttons to inflate puffs of air into a balloon presented on a computer screen. Every successful pump adds \$0.25 to their temporary bank for that trial. If the balloon explodes before the participant cashes out then nothing is won on that trial. However, an explosion does not affect the cumulative total winnings earned on prior trials. (B) An example of a cash-out trial where the participant decided to stop pumping the balloon and earn the amount accumulated up to that point.

cd to sub-01, you will find two directories, anat and func that correspond anatomical and functional. These are two important directories that we will visit most.

```
sub-01_inplaneT2.nii.gz  sub-01_T1w.nii.gz
```

anat includes all the anatomical images such as T1 and T2 (if possible).

```
sub-01_task-balloonanalogrisktask_run-01_bold.nii.gz  sub-01_task-balloonanalogrisktask_run-02_events.tsv
sub-01_task-balloonanalogrisktask_run-01_events.tsv  sub-01_task-balloonanalogrisktask_run-03_bold.nii.gz
sub-01_task-balloonanalogrisktask_run-02_bold.nii.gz  sub-01_task-balloonanalogrisktask_run-03_events.tsv
```

func has all the functional images, end with **bold.nii.gz** as well as trial-related files end with **events.tsv**.

Whenever you're ready, let's go to next

1.19.2 Preprocessing I

Remember our drink menu? It is a little different for what we should do in AFNI but overall there are very similar. Anyway, let's have a couple of drinks before the meal

Inspecting the image

Before we are actually running the analysis, it is beneficial for us to check the data for any problems such as scanner spikes, incorrect orientation, or poor contrast, and so on. Although it might be unnecessary for the open neuroimaging data, it is really important for you to check the image before when it comes to your own data.

Let's start from **sub-02**, cd to BART_afni/sub-02/anat directory and type afni to open the AFNI graphical user interface. you might see this

```
[wshao@gateway sub-02]$ afni
Precompiled binary linux_openmp_64: Feb 26 2018 (Version AFNI_18.0.22)

Thanks go to JA Bobholz for caloric input

Initializing: X11[Moba/X v 12004000].++ AFNI is detached from terminal.
[wshao@gateway sub-02]$ . Widgets..... Input files:
** Searching subdirectories of './' for data
  session # 1 = anat/ ==> 3 datasets
  session # 2 = func/ ==> 3 datasets
Fatal Signal 11 (SIGSEGV) received
  AFNI_read_inputs
  MAIN_workprocess
  AFNI:main
Bottom of Debug Stack
** AFNI version = AFNI_18.0.22  Compile date = Feb 26 2018
** [[Precompiled binary linux_openmp_64: Feb 26 2018]]
** Program Death **
** If you report this crash to the AFNI message board,
** please copy the error messages EXACTLY, and give
** the command line you used to run the program, and
** any other information needed to repeat the problem.
** You may later be asked to upload data to help debug.
** Memory usage: chunks=105361 bytes=23762370
** Crash log is appended to file /home/wshao/.afni.crashlog
```

That because AFNI will look for any images in the current directory by default - and load all of them into the program. If you want to load only the anatomical T1 image into the AFNI viewer, you would either go to sub-02/anat and type type: afni sub-02_T1w.nii.gz or type afni anat/sub-02_T1w.nii.gz from sub-02 directory

When you are done with the anatomical image, click on the Read button from the menu at the top of your screen. In the "Directories" sidebar, double-click on the filepath that ends in two dots such as .., which indicates one directory above

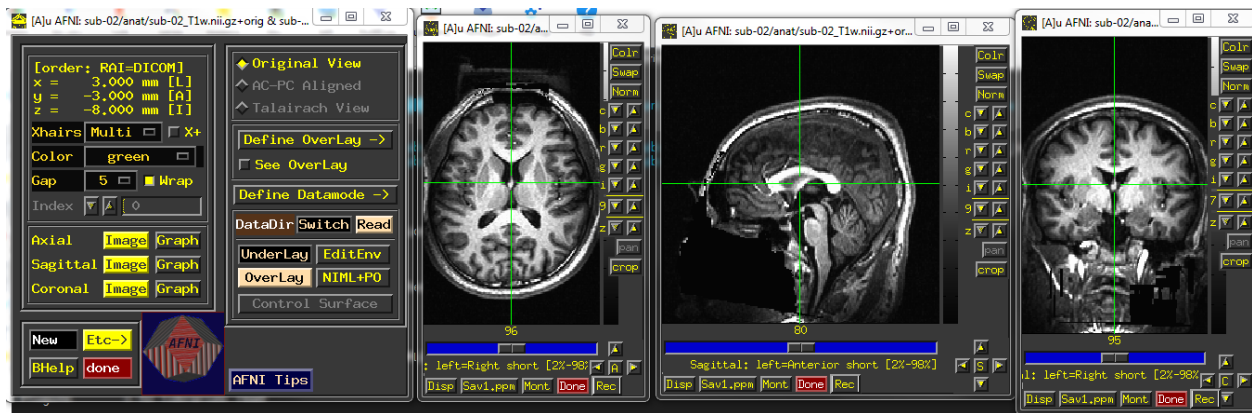


Fig. 14: T1 anat image of sub-02

the current directory. Then double-click on the func directory in the “Sessions” sidebar. This loads all of the images in the func directory, which you can look at the AFNI viewer.

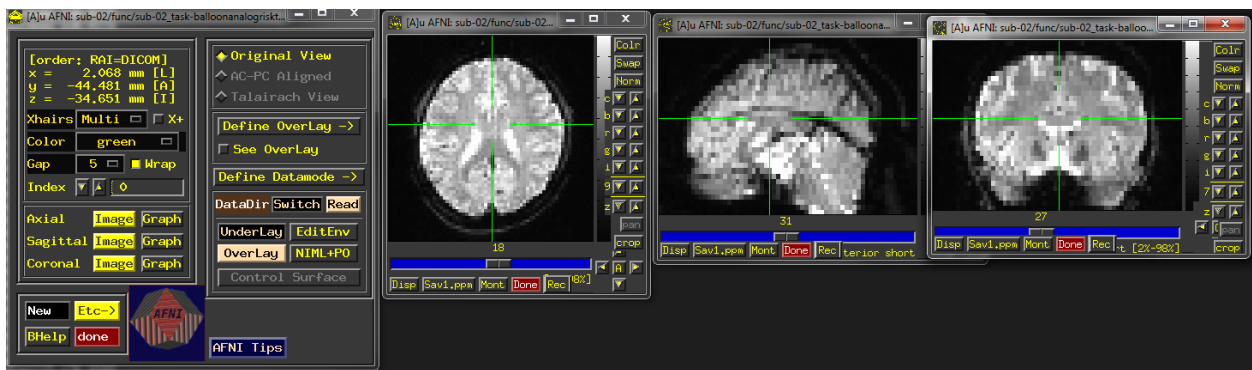


Fig. 15: functional image of sub-02

If you want to go back to look at the anat image again, click on **Switch** and select the another session. Inspect the image by clicking around in one of the viewing windows. Notice how the other viewing windows and crosshairs change as a result - this is because MRI data is collected as a three-dimensional image, and moving along one of the dimensions will change the other windows as well.

Be careful with two things:

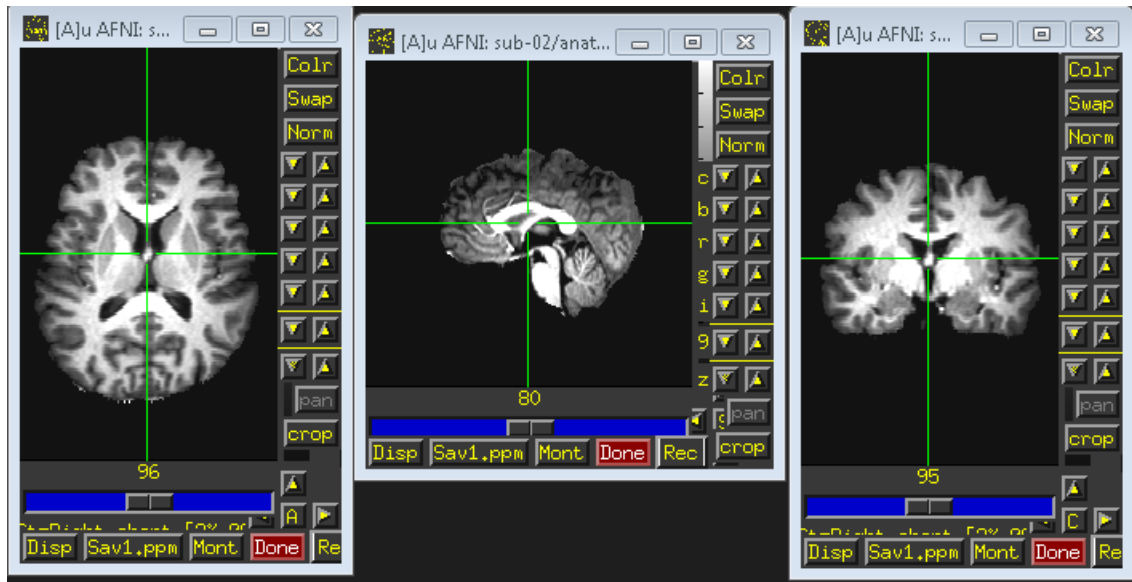
- 1 Lines that look like ripples in a pond. These are called **Gibbs Ringing Artifacts**, and they may indicate an error in the reconstruction of the MR signal from the scanner. These ripples may also be caused by the subject moving too much during the scan. In either case, if the ripples are large enough, they may cause preprocessing steps like brain extraction or normalization to fail.

- 2 Abnormal intensity differences within the grey or the white matter. These may indicate pathologies such as aneurysms or cavernomas, and they should be reported to your radiologist right away; make sure you are familiar with your laboratory's protocols for reporting artifacts. You can take a [gallery of pathologies](#) as a reference.

You might notice there is a black block/missing information on the face areas, it is because all the data from the open-source dataset need to be defaced for the purpose of privacy.

Skull stripping

When it comes to skull stripping, AFNI has 2 options for you. The first one would be 3dSkullStrip, which is the command for brain extraction. Now, cd to BART_afni/sub-02/anat and type 3dSkullStrip, you will see this doesn't work and the program give you error messages, it is because we need to specify more details such as the -input flag, which stands for the input command as well as the file sub-02_T1w.nii.gz. Therefore, the actual command would be 3dSkullStrip -input sub-02_T1w.nii.gz. type it and wait for a few minutes, you will see two new files **skull_strip_out+orig.BRIK** and **skull_strip_out+orig.HEAD**. use afni look at either one of the two files, 4 new windows will appear, and you can see the results.

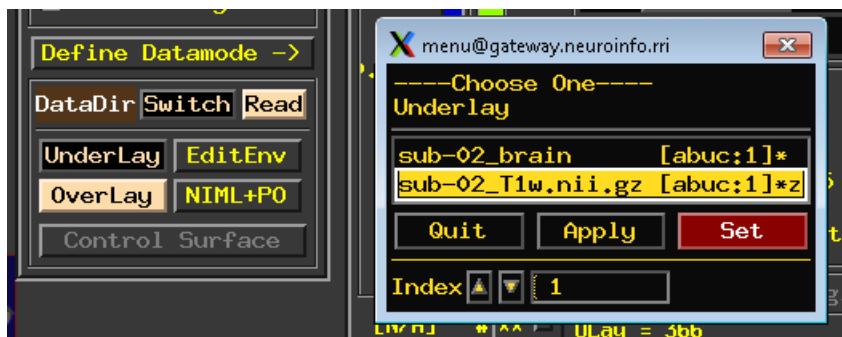


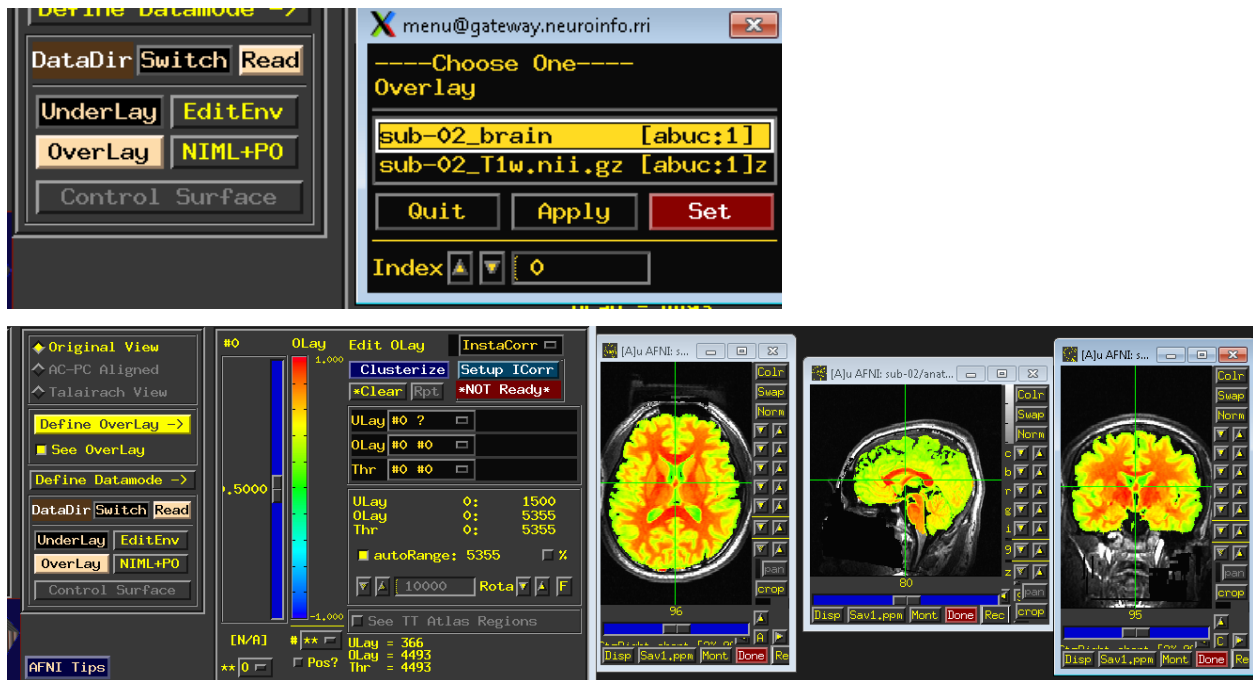
You might not be satisfied with the quality of skull stripping and the output file name also could be confusing. Don't worry, you also can add more, 3dSkullStrip -push_to_edge -input sub-02_T1w.nii.gz -prefix sub-02_brain would be a good choice. -push_to_edge can help you avoid removing any cortex parts since we rather have more dura than the lack of cortex. -prefix will label the output file name rather than the default setting. Therefore, the actual command for a good skull stripping would be:

```
3dSkullStrip -push_to_edge -input sub-02_T1w.nii.gz -prefix sub-02_brain
```

Check the quality

You can take the new skull stripping file as overlay with the original T1 anat file as underlay by type afni sub-02_brain+orig.BRIK sub-02_T1w.nii.gz, and go to the central area, apply the underlay and overlay.





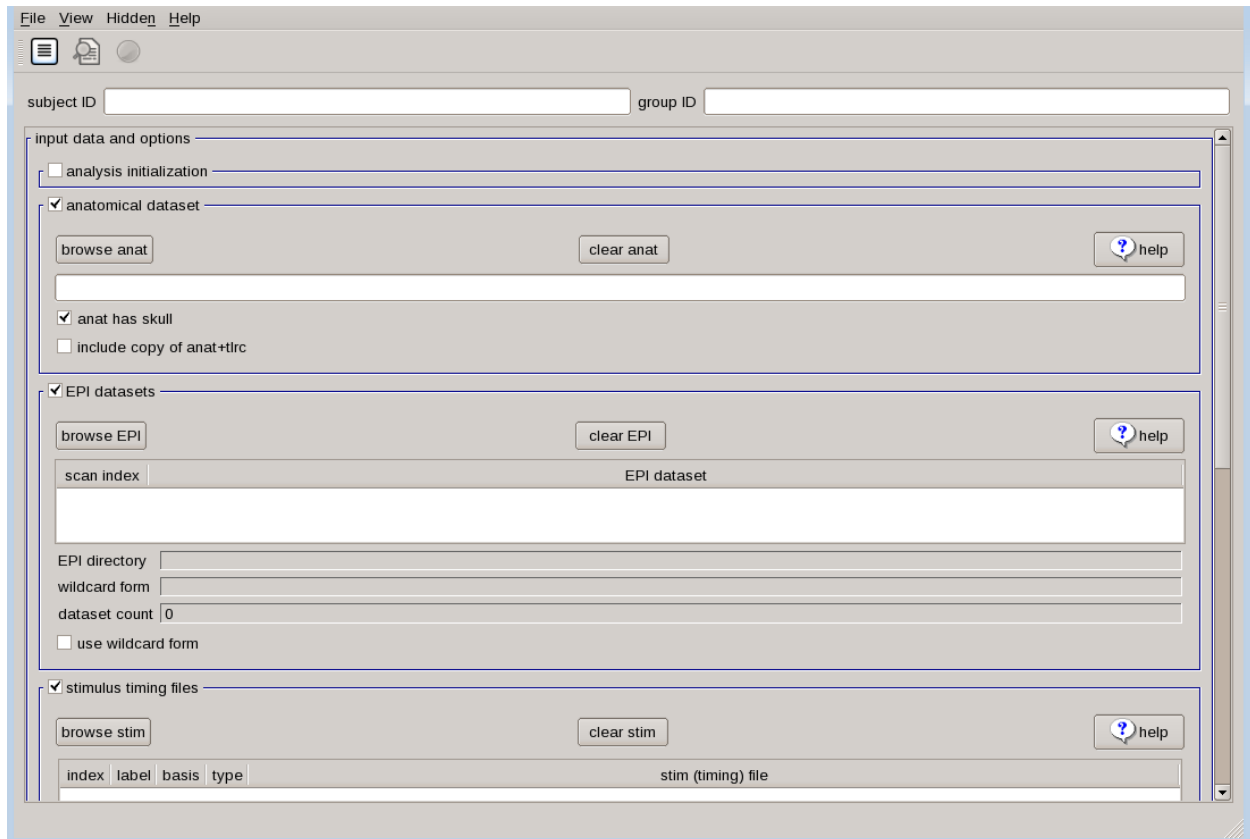
Type `3dSkullStrip -help` from your terminal to look for more information on this command.

The second option would be easier to use and more intuitive because it is associated with a new graphical interface developed by AFNI in recent years, `uber_subject.py`.

uber_subject

Go to the `sub-02` directory and type `uber_subject.py`, and you can see a window appear.

`uber_subject.py` is a graphical interface that can integrate all the input specified by the user and direct it into another program called `afni_proc.py`. The latter command will generate a large script that contains every AFNI command needed to run preprocessing.



Now, take a deep breath, let's go through this interface roughly. **subject ID** and **group ID** stand for the name you are going to label for the subject and group. "Analysis Initialization". By default, the data analysis type will be "task", which makes sense given most fMRI analyses have a task for the participants during the scanning, unless you are going to do Resting-state fMRI (we will talk about it in the future). The domain is "volume" because we are analyzing the volumetric data rather than surface analysis (a program of AFNI, SUMA, could do this kind of job). Regarding the words in the field after "processing blocks" are **tshift**, **align**, **tlrc**, **volreg**, **blur**, **mask**, **scale**, and **regress**, we will learn them in more details in the next, as well as what preprocessing step they correspond to, and why we do them.

Setting up the uber_subject

After the basic introduction of uber_subject, we will continue the setting. type sub_02 and BART in the sub ID and group ID accordingly. Go to processing tab from "Analysis Initialization" and remove the regress block. (it is a general linear model for each subject, we don't need it now). Then navigate to "anatomical dataset" and click on the browse anat button, find the anat directory, select the "sub-02_T1w.nii.gz", and check the "anat has skull" button. Next, select the functional images by clicking the "browse EPI" button from "EPI datasets" section, EPI = echo planar image. We will go to the func directory, and hold down shift to select the files "sub-02_task-balloonanalogrisktask_run-01_bold.nii.gz" "sub-02_task-balloonanalogrisktask_run-02_bold.nii.gz" "sub-02_task-balloonanalogrisktask_run-03_bold.nii.gz". The first half of the AFNI GUI should look as follows:

subject ID group ID

input data and options

☒ analysis initialization

type domain

processing blocks

☒ anatomical dataset

☒ anat has skull

☐ include copy of anat+tlrc

☒ EPI datasets

scan index	EPI dataset
1	sub-02_task-balloonanalogrisktask_run-01_bold.nii.gz
2	sub-02_task-balloonanalogrisktask_run-02_bold.nii.gz
3	sub-02_task-balloonanalogrisktask_run-03_bold.nii.gz

EPI directory

wildcard form

dataset count

☐ use wildcard form

As we are not yet doing regression, we will skip over the “stimulus timing files” and “symbolic GLTs” sections, and take a look at the default of “expected options” field. No TRs will be removed as 0 in “first TRS to remove”, the volume with the least amount of variability image will be used as the reference for alignment since the default is “MIN_OUTLIER” in volume registration base, a smoothing kernel of 4mm will be applied from “blur size”. Any volumes that contain movement of 0.3mm from TR to TR will be flagged.

Again, we are not doing regression model yet, just skip over the “extra regress options”, and go to “extra align options” and “extra tlrc options”. Choose “lpc+ZZ” in cost function and check the align box by using giant_move. This is prepare for functional and anatomical images are far away from each other by any case, it will be used with the align_epi_anat.py command in order to bring them into a closer initial alignment. Lastly, choose the MNI 152 T1 standard space (template) “MNI_avg152T1+tlrc” from the “extra tlrc options” sectio. the second half of the AFNI GUI looks like this:

subject ID group ID

☐ use wildcard form

☐ stimulus timing files

☐ symbolic GLTs

☒ expected options

first TRs to remove (per run)

volume registration base

blur size (FWHM in mm)

motion censor limit (mm, per TR)

☐ extra regress options

☒ extra align options

align: cost function

☒ align: use giant_move

☒ extra tlrc options

tlrc: template

☐ tlrc: OK maxite

Run the uber_subject

When you are done setting up, you can execute it by clicking the leftmost button of the GUI, ignore the warning message and click OK. There are three icons, from left to right, the first one generates the **afni_proc.py** command that includes everything that you just specified in the GUI. Click on this icon, and it will return two windows: One listing each of the options that were changed from the defaults and listing each of the inputs, and another showing the code of the “afni_proc.py” command. Take a look at it to see the commands and differences.

```

File
options changed from defaults (5):
    align_cost      : lpc+ZZ
    align_giant_move : yes
    blocks          : tshift align tlrc volreg blur mask scale
    tlrc_base       : MNI_avg152T1+tlrc
    volreg_base     : MIN_OUTLIER

applied subject variables (4):
    anat      : /home/wshao/BART_afni/sub-02/anat/sub-02_T1w.nii.gz
    epi       : [list of 3 elements]
    gid       : BART
    sid       : sub_02

File
# created by uber_subject.py: version 1.0 (December 29, 2017)
# creation date: Sat Mar 27 16:19:32 2021

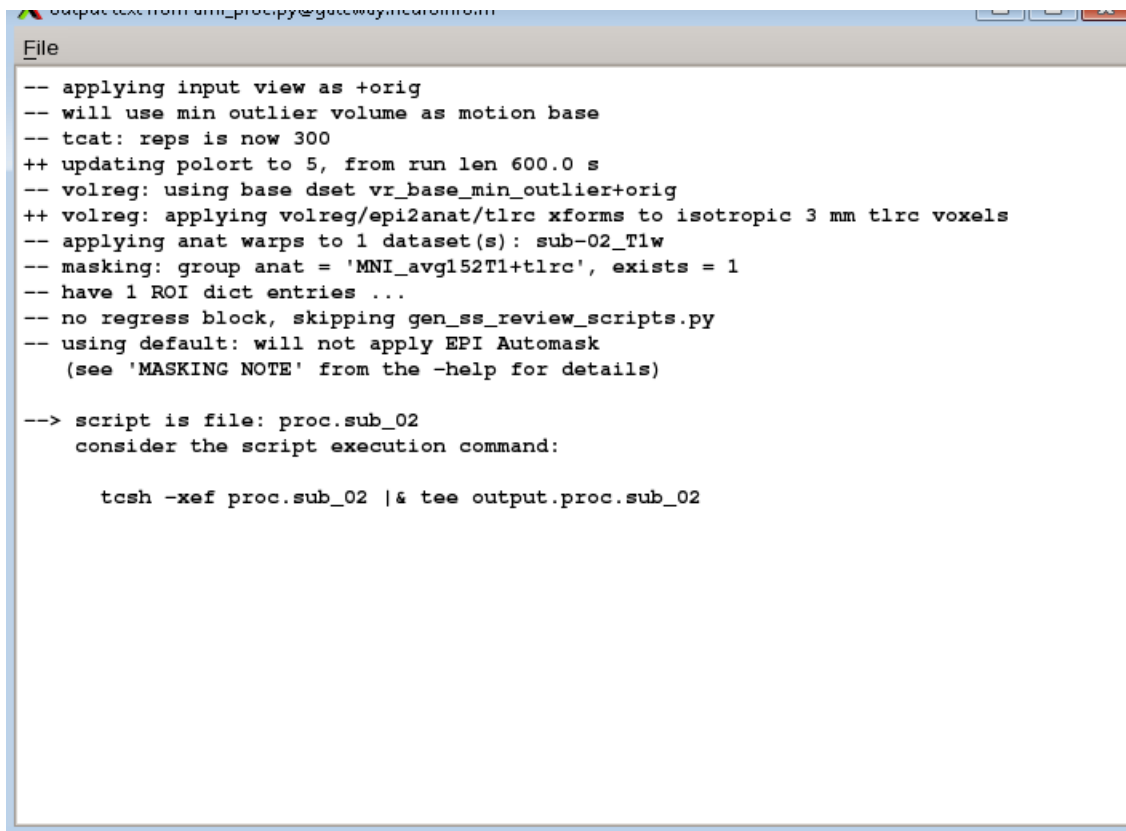
# set subject and group identifiers
set subj = sub_02
set gname = BART

# set data directories
set top_dir = /home/wshao/${gname}_afni/sub-02
set anat_dir = $top_dir/anat
set epi_dir = $top_dir/func

# run afni_proc.py to create a single subject processing script
afni_proc.py -subj_id $subj \
    -script proc.$subj -scr_overwrite \
    -blocks tshift align tlrc volreg blur mask scale \
    -copy_anat $anat_dir/sub-02_T1w.nii.gz \
    -dsets \
        $epi_dir/sub-02_task-balloonanalogriektask_run-01_bold.nii.gz \
        $epi_dir/sub-02_task-balloonanalogriektask_run-02_bold.nii.gz \
        $epi_dir/sub-02_task-balloonanalogriektask_run-03_bold.nii.gz \
    -tcats_remove_first_trs 0 \
    -align_opts_aea -cost lpc+ZZ -giant_move \
    -tlrc_base MNI_avg152T1+tlrc \
    -volreg_align_to MIN_OUTLIER \
    -volreg_align_e2a \
    -volreg_tlrc_warp \
    -blur_size 4.0

```

When you have looked over the paper, click on the next icon, a new sheet of paper will show up. This will execute the “afni_proc.py” code listed in the previous window, and return any warnings or errors that you should be aware of. You will also see a couple of lines of code specifying how to run the output of this command, which is a file called

proc.sub_02.


```

-- applying input view as +orig
-- will use min outlier volume as motion base
-- tcat: reps is now 300
++ updating polort to 5, from run len 600.0 s
-- volreg: using base dset vr_base_min_outlier+orig
++ volreg: applying volreg/epi2anat/tlrc xforms to isotropic 3 mm tlrc voxels
-- applying anat warps to 1 dataset(s): sub-02_T1w
-- masking: group anat = 'MNI_avg152T1+tlrc', exists = 1
-- have 1 ROI dict entries ...
-- no regress block, skipping gen_ss_review_scripts.py
-- using default: will not apply EPI Automask
  (see 'MASKING NOTE' from the -help for details)

--> script is file: proc.sub_02
    consider the script execution command:

    tcsh -xef proc.sub_02 |& tee output.proc.sub_02

```

Close the window, We are now ready to run this script. You can simply press the Green icon button at the top of the uber_subject.py GUI that next to the first two button. you will see another window open up that shows each command being run, and its corresponding output: and go back to your Terminal by Ctrl+Z and type **bg** to hide the executing task into background. From the sub-02 directory, you will notice that there is a new directory called **subject_results** in the current directory. **cd** to it and a directory named **group.BART** has been created, Navigate into this directory you will find **subj.sub_02**. There are 4 files within this directory:

1 cmd.ap.sub_02: This is the afni_proc.py file that was generated by uber_subject.py a.

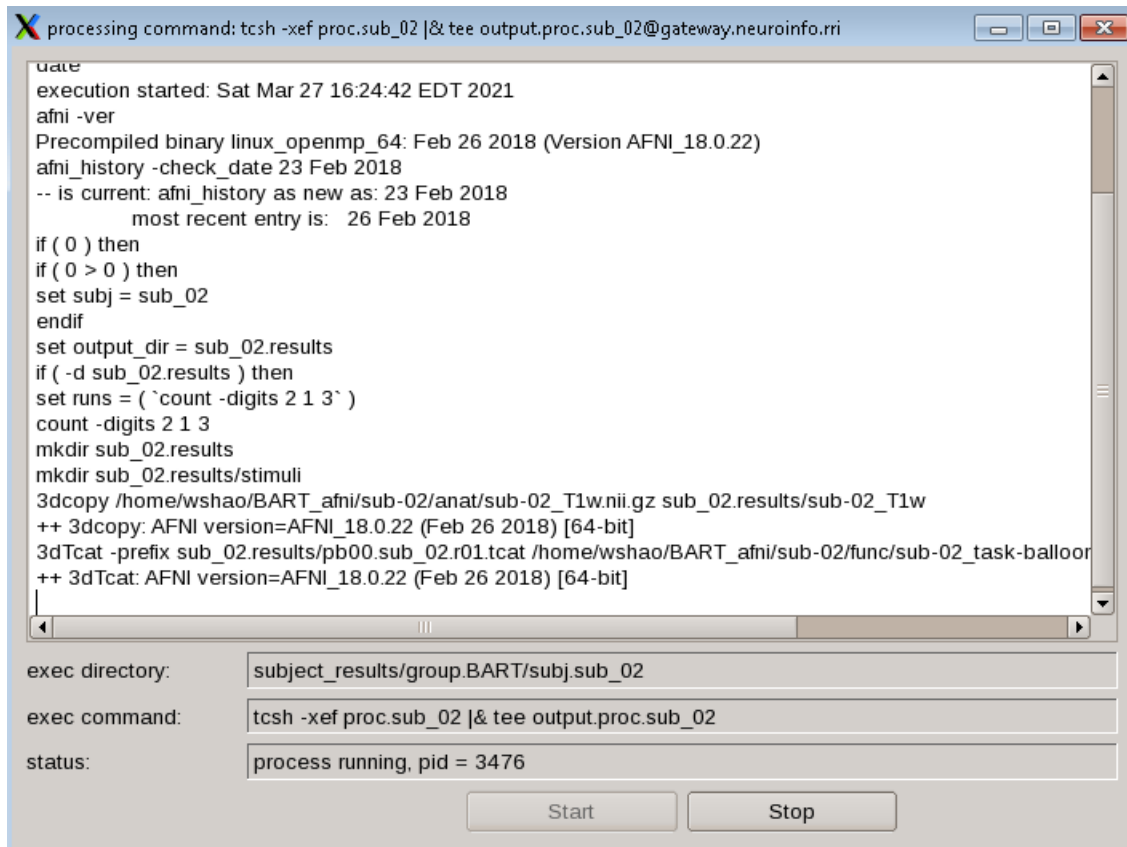
2 output.cmd.ap.sub_02: This is the output of afni_proc.py that generates any warnings or errors you should be aware of.

You have saw these two files when you click the first and second execute button

3 proc.sub_08: The AFNI script that perform the actual preprocessing.

4 output.proc.sub_02 the log files of afni_proc.py, which record the preprocessing information when it execute the commands

The last two are more important since they have all the information for you to debug when you do your own data with AFNI.



The preprocessing will take a few minutes, While it is running, let's review the preprocessing steps.

1.19.3 Preprocessing II

Slice-Timing Correction

There are no two identical wine in the world.

Most fMRI studies acquire one slice at a time, meaning that the signal recorded from one slice may be offset in time by up to several seconds when compared to another. Each of these slices takes tens to hundreds of milliseconds. The two most commonly used methods for creating volumes are sequential and interleaved slice acquisition. sequential acquisition slices have been acquired in sequential (1,2,3,4,5,6...), Interleaved slice acquisition (1,3,5,...2,4,6...) acquires every other slice and then fills in the gaps on the second pass o. Although slice timing differences may not be important for simple block design experiments, they can impact considerable errors in rapid, event-related fMRI studies if not accounted for. In addition, the statistics model requires the time-series for each slice needs to be shifted back in time by the duration it took to acquire that slice.

Timing slice correction in AFNI

In AFNI, you could find the block of code of slice-timing correction in the proc script, it take the first slice as the reference, the code for running slice-timing correction is specified by the -tzero 0 option of the 3dTshift command:

```
# ===== tshift =====
# time shift data so all slice timing is the same
foreach run ( $runs )
    3dTshift -tzero 0 -quintic -prefix pb01.$subj.r$run.tshift \
        pb00.$subj.r$run.tcat+orig
end
```

This will slice-time correct each run with the first slice as a reference. Everything in AFNI is starting from 0 in this case, the 0 in here represents the first slice of the volume. The command called -quintic, which resamples each slice using a 5th-degree polynomial. In other words, AFNI will replace the values of the voxels within a slice by using information from a larger number of other slices. the side effect of this method is some degree of correlation between the slices, which will be corrected later by **3dREMLfit** to de-correlate the data.

Motion correction

Another influential factor for the quality is the move, even a small motion would make a difference to the wine, just like they did to the scan.

Head motion is the largest source of error in fMRI studies, If the subject in the scanner is moving, the images will look blurry. If the subject moves a lot, we also risk measuring signal from a voxel that moves. We are then in danger of measuring signal from the voxel from a different region or tissue type. It is the false positive. In addition, motion can introduce confounds into the imaging data because motion generates signal. If the subject moves every time in response to a stimulus - for example, if he jerks his head every time he feels an electrical shock, then it can become impossible to determine whether the signal we are measuring is in response to the stimulus, or because of the movement. Therefore, a variety of strategies have been developed to cope with this problem such as immobilization of the head using padding and straps, bite bars, masks during the scanning. What's more, the standard motion correction method of neuroimage packages considers the head as a rigid body with three directions of translation (displacement) and three axes of rotation. A single functional volume of a run is chosen as the reference to which runs in all other volumes are aligned. Procedures are performed in which each volume is rotated and aligned with the reference, with the goal to minimize a cost function. This iterative adjustment terminates once no further improvement can be achieved..

motion correction in AFNI

Motion correction in AFNI is the command 3dvolreg. In a typical analysis script generated by uber_subject.py, there will be a block of code highlighted with the header =====volreg===== . There are several commands in this block, but the most important one for our present purposes is the line that begins with "3dvolreg":

```
# ===== volreg =====
# align each dset to base volume, align to anat, warp to tlrc space

# verify that we have a +tlrc warp dataset
if ( ! -f sub-02_T1w_ns+tlrc.HEAD ) then
    echo "*** missing +tlrc warp dataset: sub-02_T1w_ns+tlrc.HEAD"
    exit
endif

# register and warp
```

(continues on next page)

(continued from previous page)

```
foreach run ( $runs )
# register each volume to the base image
3dvolreg -verbose -zpad 1 -base vr_base_min_outlier+orig \
        -1Dfile dfile.r$run.1D -prefix rm.epi.volreg.r$run \
        -cubic \
        -1Dmatrix_save mat.r$run.vr.aff12.1D \
        pb01.$subj.r$run.tshift+orig
```

You will see several options used with this command. The `-base` option is the reference volume; in this case, the reference image is the volume that has the lowest amount of outliers in its voxels, as determined by an earlier command, `3dToutcount`. The `-1Dfile` command writes the motion parameters into a text file with a “1D” appended to it, and `-1Dmatrix_save` saves an affine matrix which indicates how much each TR would need to be “unwarped” along each of the affine dimensions in order to match the reference image:

```
# catenate volreg/epi2anat/tlrc xforms
cat_matvec -ONELINE \
        sub-02_T1w_ns+tlrc::WARP_DATA -I \
        sub-02_T1w_al_junk_mat.aff12.1D -I \
        mat.r$run.vr.aff12.1D > mat.r$run.warp.aff12.1D
```

This affine transformation matrix is then concatenated with the affine transformation matrices that were generated during the warping of the anatomical dataset to normalized space and during the alignment of the anatomical dataset to the fMRI data:

```
# apply catenated xform: volreg/epi2anat/tlrc 3dAllineate -base sub-02_T1w_ns+tlrc
        -input pb01.$subj.r$run.tshift+orig -1Dmatrix_apply mat.r$run.warp.aff12.1D -mast_dxyz 3 -
        prefix rm.epi.nomask.r$run
```

This concatenated affine matrix is used with the `3dAllineate` command to create fMRI datasets that are both motion corrected and normalized in one step (using the `1Dmatrix_apply` option):

These blocks of code may be difficult to understand at first, but always keep in mind the basic structure of the AFNI commands: The command name, followed by options, and usually including a “-prefix” option to label the output. The motion correction and normalization commands often include a “-base” and “-input” pair of options as well, to indicate which dataset is being aligned to which reference dataset. You will most likely not be editing these lines of code in the file generated by `uber_subject.py`, but it is still useful to know why they are written the way they are; and you may use these commands outside of this script to align other datasets if you wish.

```
# warp the all-1 dataset for extents masking 3dAllineate -base sub-02_T1w_ns+tlrc
        -input rm.epi.all1+orig -1Dmatrix_apply mat.r$run.warp.aff12.1D -mast_dxyz 3 -final NN -
        quiet -prefix rm.epi.1.r$run
```

There are some basic code structure of the AFNI commands: The command name, followed by options, and usually including a “-prefix” option to label the output. AFNI also use “-base” and “-input” pair of options to indicate which dataset is being aligned to which reference dataset. It is important for us to understand these coded since we might use these commands with different dataset individually.

Registration

The easiest and quickest way to know the quality of a wine is comparison, we need to take different wines to know which one is better and we also need a standard for a reference, we can compare each wine to each other and a standard wine for a quality assessment. So, here is the registration and normalization.

Although most people's brains are similar - everyone has 4 lobes and subcortical, there are also differences in brain size and shape. As a consequence, if we want to do a group analysis we need to ensure that each voxel for each subject corresponds to the same part of the brain. If we are measuring a voxel in the visual cortex, make sure that every subject's visual cortex is in alignment with each other. This alignment between the functional and anatomical images is called Registration. Most registration algorithms use the following steps:

- 1 Assume that the functional and anatomical images are in roughly the same location. If they are not, align the outlines of the images.
- 2 Take advantage of the fact that the anatomical and functional images have different contrast weightings - that is, areas where the image is dark on the anatomical image (such as cerebrospinal fluid) will appear bright on the functional image, and vice versa. This is called mutual information. The registration algorithm moves the images around to test different overlays of the anatomical and functional images, matching the bright voxels on one image with the dark voxels of another image, and the dark with the bright, until it finds a match that cannot be improved upon.
- 3 Once the best match has been found, then the same transformations that were used to warp the anatomical image to the template are applied to the functional images.

Registration with AFNI

The command `align_epi_anat.py` can do several preprocessing steps at once - registration, aligning the volumes of the functional images together, and slice-timing correction. In here, we will just use it for registration. The code for this step will be found in `proc.sub_02` script:

```
# ===== align =====
# for e2a: compute anat alignment transformation to EPI registration base
# (new anat will be intermediate, stripped, sub-02_T1w_ns+orig)
align_epi_anat.py -anat2epi -anat sub-02_T1w+orig \
    -save_skullstrip -suffix _al_junk \
    -epi vr_base_min_outlier+orig -epi_base 0 \
    -epi_strip 3dAutomask \
    -cost lpc+ZZ -giant_move \
    -volreg off -tshift off
```

We will introduce these different options in more details:

- 1 -anat2epi : will align anatomical to EPI dataset, which is the default setting in AFNI.
- 2 -anat: use anatomical dataset to align or to which to align.
- 3 save_skullstrip : save skull-stripped (not aligned).
- 4 -suffix with the string “_al_junk” to some of the intermediate stages of the registration, and for normalization later.
- 5 -epi_base : the epi base used in alignment, The “epi” options signalize that the functional volume with the least variability will be used as a reference image, and non-brain tissue is stripped by 3dAutomask , an alternative to 3dSkullStrip.
- 7 lpc (Localized Pearson Correlation) function. The ‘lpc’ cost function is computed by the 3dAllineate program.
- 8 -giant_move attempts to find a good initial starting alignment between the anatomical and functional images, giant_move option will usually work well too, but it adds significant time to the processing and allows for the possibility of a very bad alignment.

9 The last two options `-volreg off`, `-tshift off` indicate that we do not want to include alignment and slice-timing correction in the current command.

Normalization

Each brain needs to be transformed to have the same size, shape, and dimensions to be compared. We do this by normalizing (or warping) to a template. A template is a brain that has standard dimensions and coordinates. Each subjects' functional images will be transformed to match the general shape and large anatomical features of the template. Most researchers have agreed to use them when reporting their results. The dimensions and coordinates of the template brain are also referred to as standardized space.

Normalization with AFNI's `@auto_tlrc`

Once we have aligned the anatomical and functional images, the next step for us is to normalize the anatomical image to a template as well as functional image. We will use the `@auto_tlrc` command to do this. To normalize the anatomical image,; this and a following command, `cat_matvec`, are found in your **proc** script:

```
# ===== tlrc =====  
# warp anatomy to standard space  
@auto_tlrc -base MNI_avg152T1+tlrc -input sub-02_T1w_ns+orig -no_ss  
  
# store forward transformation matrix in a text file  
cat_matvec sub-02_T1w_ns+tlrc::WARP_DATA -I > warp.anat.Xat.1D
```

The first block of commands indicates that we take the image `MNI_avg152T1` as a template, and the skull-stripped anatomical image as a source image. The `-no_ss` option indicates that the anatomical image has already been skull-stripped. The anatomical image needs to be moved and transformed so that it can align the template and the anatomical image. The second command, `cat_matvec`, extracts this matrix and copies it into a file called `warp.anat.Xat.1D`.

Smoothing

If we have the best wine, why don't to mix it with other drinks in order to achieve the best taste? It is common for neuroimage software to smooth the functional data, or replace the signal at each voxel with a weighted average of that voxel's neighbors. This may seem strange at first - why would we want to make the images blurrier than they already are?

Spatial smoothing is the averaging of signals from adjacent voxels. This improves the signal-to-noise ratio (SNR) but decreases spatial resolution, blurs the image, and smears activated areas into adjacent voxels. The process can be justified because closely neighboring brain voxels are usually inherently correlated in their function and blood supply. The standard method is to convolve ("multiply") the fMRI data with a 3D Gaussian kernel ("filter") that averages signals from neighboring voxels with weights that decrease with increasing distance from the target voxel. In practice, the full width half maximum (FWHM) value of the Gaussian spatial filter is typically set to about 4-6 mm for single subject studies and to about 6-8 mm for multi-subject analyses. The benefits of smoothing can outweigh the drawbacks:

1 We know that fMRI data contain a lot of noise, and that the noise is frequently greater than the signal. By averaging over nearby voxels we can cancel out the noise and enhance the signal.

2 Smoothing can be good for group analyses in which all of the subjects' images have been normalized to a template. If the images are smoothed, there will be more overlap between clusters of signal, and therefore greater likelihood of detecting a significant effect. `3dmerge` in AFNI will do smoothing for us, you will find further details in **proc** script:

```
# ===== blur =====
# blur each volume of each run
foreach run ( $runs )
    3dmerge -1blur_fwhm 4.0 -doall -prefix pb03.$subj.r$run.blur \
        pb02.$subj.r$run.volreg+tlrc
end
```

The **-1blur_fwhm** specifies the amount to smooth the image in 4mm. **-doall** applies this smoothing kernel to each volume in the image, and the **-prefix** option, we you might know, specifies the name of the output dataset.

Masking

The last preprocessing steps will take these smoothed images and then scale them to have a mean signal intensity of 100 - so that deviations from the mean can be measured in percent signal change. Any non-brain voxels will then be removed by a mask, and these images will be ready for statistical analysis.

As you might see before, a volume of fMRI data includes both the areas that we are interested in and the irrelevant regions such as the brain and the surrounding skull, neck and ear. And we have large numbers of voxels that comprise the air outside the head. we can create a mask for our data to reduce the irrelevant regions of our datasets and speed up our analyses. A mask simply tells the programer which voxels are worth to be analyzed - any voxels within the mask have their original values, or can be assigned a value of 1, whereas any voxels outside mask are assigned a value of zero. Anything outside the mask will be assumed as noise.

Masks are created with 3dAutomask command, which requires arguments for input and output datasets:

```
# ===== mask =====
# create 'full_mask' dataset (union mask)
foreach run ( $runs )
    3dAutomask -dilate 1 -prefix rm.mask_r$run pb03.$subj.r$run.blur+tlrc
end
```

-dilate 1 means AFNI will dilate the mask outwards '1' times:

```
# create union of inputs, output type is byte
3dmask_tool -inputs rm.mask_r*+tlrc.HEAD -union -prefix full_mask.$subj

# ---- create subject anatomy mask, mask_anat.$subj+tlrc ----
# (resampled from tlrc anat)
3dresample -master full_mask.$subj+tlrc -input sub-02_T1w_ns+tlrc \
    -prefix rm.resam.anat

# convert to binary anat mask; fill gaps and holes
3dmask_tool -dilate_input 5 -5 -fill_holes -input rm.resam.anat+tlrc \
    -prefix mask_anat.$subj

# compute tighter EPI mask by intersecting with anat mask
3dmask_tool -input full_mask.$subj+tlrc mask_anat.$subj+tlrc \
    -inter -prefix mask_epi_anat.$subj

# compute overlaps between anat and EPI masks
3dABoverlap -no_automask full_mask.$subj+tlrc mask_anat.$subj+tlrc \
    |& tee out.mask_ae_overlap.txt
```

(continues on next page)

(continued from previous page)

```
# note Dice coefficient of masks, as well
3ddot -dodice full_mask.$subj+tlrc mask_anat.$subj+tlrc \
    |& tee out.mask_ae_dice.txt

# ---- create group anatomy mask, mask_group+tlrc ----
# (resampled from tlrc base anat, MNI_avg152T1+tlrc)
3dresample -master full_mask.$subj+tlrc -prefix ./rm.resam.group \
    -input /opt/afni/MNI_avg152T1+tlrc

# convert to binary group mask; fill gaps and holes
3dmask_tool -dilate_input 5 -5 -fill_holes -input rm.resam.group+tlrc \
    -prefix mask_group
```

The rest of the code within the “mask” block creates a union of masks that represents the extent of all of the individual fMRI datasets in the experiment. It then computes a mask for the anatomical dataset, and then takes the intersection of the fMRI and anatomical masks

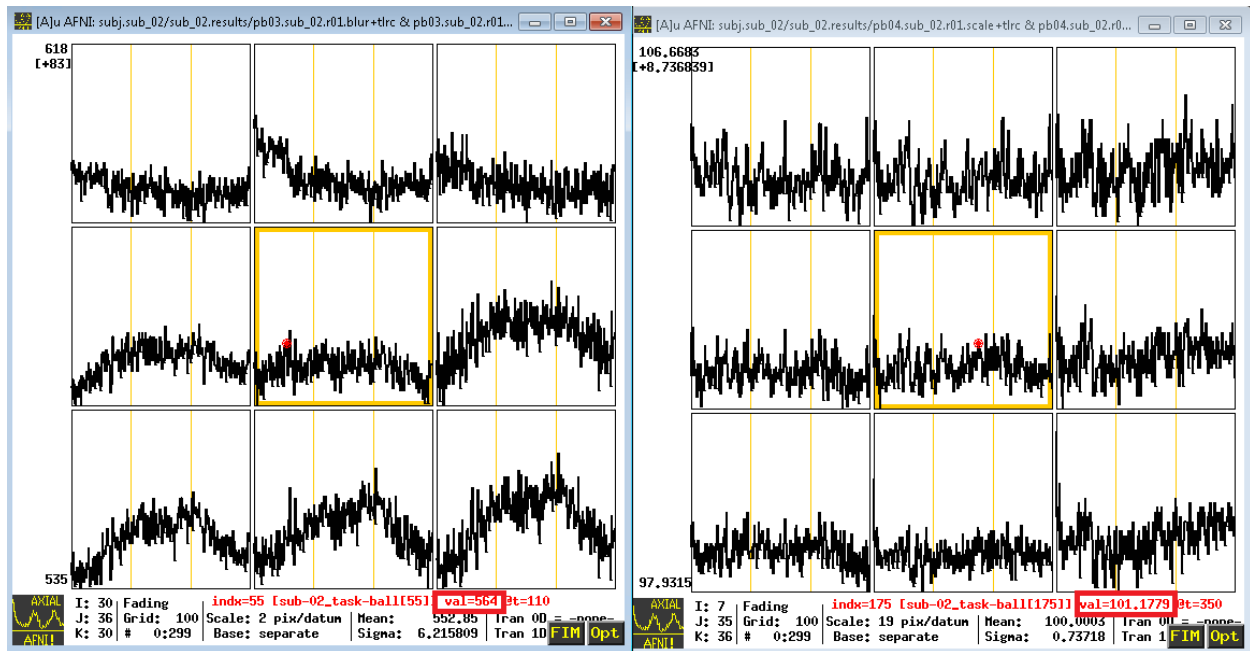
Scaling

One problem with fMRI data is that we collect data with units that are arbitrary, and of themselves meaningless. The intensity of the signal that we collect can vary from run to run, and from subject to subject. The only way to create a useful comparison within or between subjects is to take the contrast of the signal intensity between conditions, as represented by a beta weight (which will be discussed later in the chapter on statistics).

In order to make the comparison of signal intensity meaningful between studies as well, AFNI scales the timeseries in each voxel individually to a mean of 100:

```
# ===== scale =====
# scale each voxel time series to have a mean of 100
# (be sure no negatives creep in)
# (subject to a range of [0,200])
foreach run ( $runs )
    3dTstat -prefix rm.mean_r$run pb03.$subj.r$run.blur+tlrc
    3dcalc -a pb03.$subj.r$run.blur+tlrc -b rm.mean_r$run+tlrc \
        -c mask_epi_extents+tlrc \
        -expr 'c * min(200, a/b*100)*step(a)*step(b)' \
        -prefix pb04.$subj.r$run.scale
end
```

You can see these changes in the time-series graph below. Note that the values in the first image are relatively high - in the 600s - and that they are arbitrary; they could just as easily be around 500 or 900 in another subject. By scaling each subject's data to the same mean, as in the right image, we can place each run of each subject's data on the same scale to compare.



1.19.4 Check the preprocessed data

After the script generated by `uber_subject.py` has completed, navigate to the directory containing the preprocessed data. By default, AFNI will create a new directory tree in the following format:

```
cd sub-02/subject_results/group.BART/subj.sub_02/sub_02.results
```

In which subjects name and group name are specified in the subject ID and group ID tab from the `uber_subject.py` GUI we did.

Look at the Preprocessed files

You can see all the different images and files from each step of preprocessing. Especially, these files starting from “pb” (Processing Block) are preprocessed functional images and the files with “T1w” are the preprocessed anatomical images.

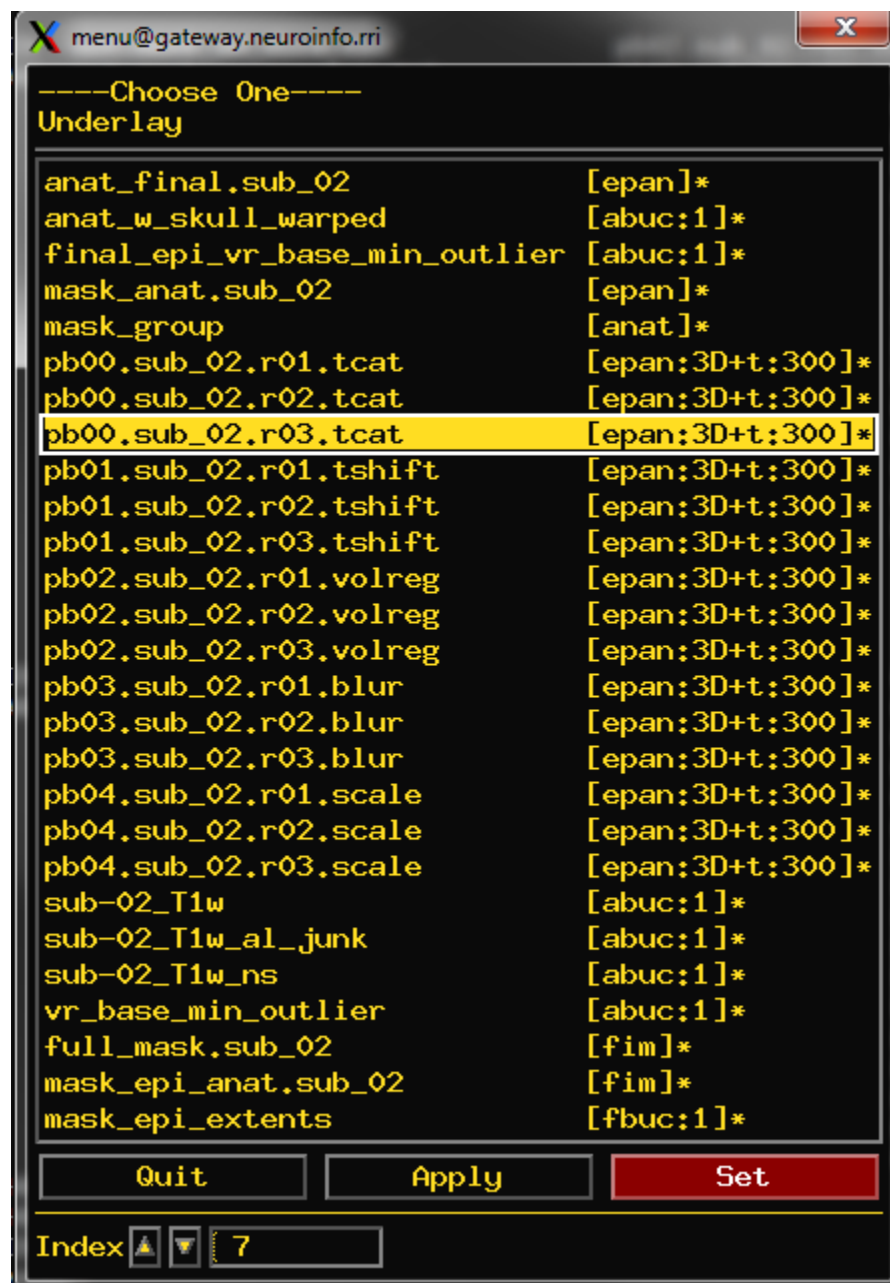
```
anat_final.sub_02+tlrc.BRIK
anat_final.sub_02+tlrc.HEAD
anat_w_skull_warped+tlrc.BRIK
anat_w_skull_warped+tlrc.HEAD
dfile_r01.1D
dfile_r02.1D
dfile_r03.1D
dfile_rall.1D
@epi_review.sub_02
final_epi_vr_base_min_outlier+tlrc.BRIK
final_epi_vr_base_min_outlier+tlrc.HEAD
full_mask.sub_02+tlrc.BRIK
full_mask.sub_02+tlrc.HEAD
task_anat.sub_02+tlrc.BRIK
task_anat.sub_02+tlrc.HEAD
task_epi_anat.sub_02+tlrc.BRIK
task_epi_anat.sub_02+tlrc.HEAD
mask_epi_extents+tlrc.BRIK
mask_epi_extents+tlrc.HEAD
mask_group+tlrc.BRIK
mask_group+tlrc.HEAD
mat_basewarp.aff12.1D
mat_r01.vr.aff12.1D
mat_r01.warp.aff12.1D
mat_r02.vr.aff12.1D
mat_r02.warp.aff12.1D
mat_r03.vr.aff12.1D
mat_r03.warp.aff12.1D
motion.sub_02_enorm.1D
out_attcostx.txt
outcount_r01.1D
outcount_r02.1D
outcount_r03.1D
outcount_rall.1D
out_mask_ae_dice.txt
out_mask_ae_overlap.txt
out_min_outlier.txt
out_pre_ss_warn.txt
out_ss_review.sub_02.txt
pb00.sub_02_r01.tcatt+orig.BRIK
pb00.sub_02_r01.tcatt+orig.HEAD
pb00.sub_02_r02.tcatt+orig.BRIK
pb00.sub_02_r02.tcatt+orig.HEAD
pb00.sub_02_r03.tcatt+orig.BRIK
pb00.sub_02_r03.tcatt+orig.HEAD
pb01.sub_02_r01.tshift+orig.BRIK
pb01.sub_02_r01.tshift+orig.HEAD
pb01.sub_02_r02.tshift+orig.BRIK
pb01.sub_02_r02.tshift+orig.HEAD
pb01.sub_02_r03.tshift+orig.BRIK
pb01.sub_02_r03.tshift+orig.HEAD
pb02.sub_02_r01.volreg+tlrc.BRIK
pb02.sub_02_r01.volreg+tlrc.HEAD
pb02.sub_02_r02.volreg+tlrc.BRIK
pb02.sub_02_r02.volreg+tlrc.HEAD
pb02.sub_02_r03.volreg+tlrc.BRIK
pb02.sub_02_r03.volreg+tlrc.HEAD
pb03.sub_02_r01.blur+tlrc.BRIK
pb03.sub_02_r01.blur+tlrc.HEAD
pb03.sub_02_r02.blur+tlrc.BRIK
pb03.sub_02_r02.blur+tlrc.HEAD
pb03.sub_02_r03.blur+tlrc.BRIK
pb03.sub_02_r03.blur+tlrc.HEAD
pb04.sub_02_r01.scale+tlrc.BRIK
pb04.sub_02_r01.scale+tlrc.HEAD
pb04.sub_02_r02.scale+tlrc.BRIK
pb04.sub_02_r02.scale+tlrc.HEAD
pb04.sub_02_r03.scale+tlrc.BRIK
pb04.sub_02_r03.scale+tlrc.HEAD
pb04.sub_02_r03.scale+tlrc.HEAD
stimuli
sub_02_T1w_at_junk_mat.aff12.1D
sub_02_T1w_at_junk+orig.BRIK
sub_02_T1w_at_junk+orig.HEAD
sub_02_T1w_ns+orig.BRIK
sub_02_T1w_ns+orig.HEAD
sub_02_T1w_ns+tlrc.BRIK
sub_02_T1w_ns+tlrc.HEAD
sub_02_T1w_ns_warped+orig.BRIK
sub_02_T1w_ns_warped+orig.HEAD
sub_02_T1w_ns_xaff12.1D
sub_02_T1w_ns_xat.1D
sub_02_T1w+orig.BRIK
sub_02_T1w+orig.HEAD
vr_base_min_outlier+orig.BRIK
vr_base_min_outlier+orig.HEAD
warp_anat.Xat.1D
```

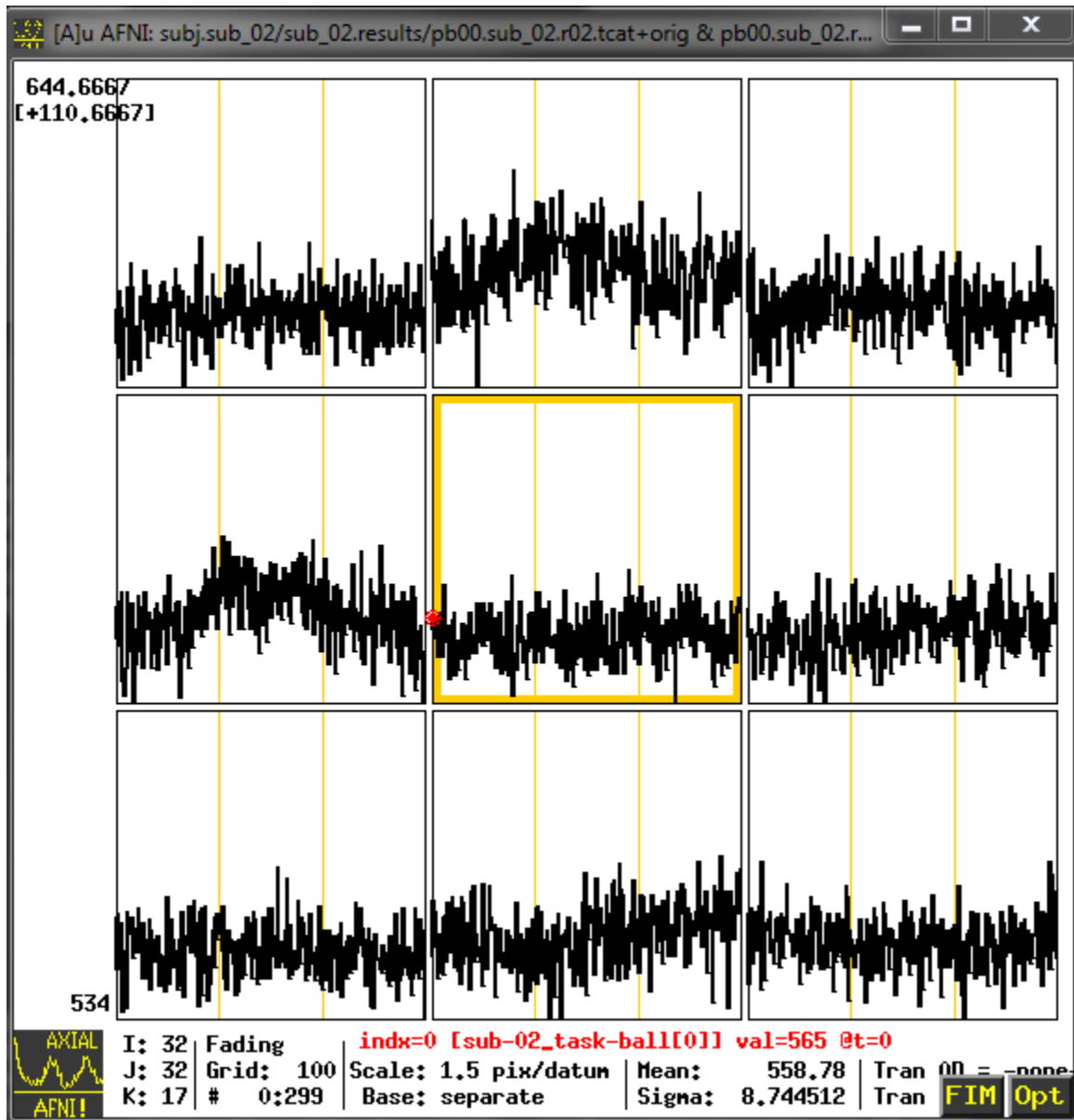
Fig. 16: The files containing the “pb” are the preprocessed functional images, the files with the “T1w” string are the preprocessed anatomical images. the name “3dTshift” means that these images have been slice-time corrected by the “3dTshift” command.

Processed Functional Image

After check with different files in the preprocessed data directory, let's take a close look, type `afni` to open the AFNI GUI. Click the "Underlay", and choose "pb00.sub_02.r01.tcat", click on the "Graph" next to any of the Axial, Sagittal, or Coronal views to view the time-series. You also can see the same image when you open "pb00.sub_02.r02.tcat" and "pb00.sub_02.r03.tcat"; it is because the initial volumes of dataset had been in OpenNeuro.

The **Underlay** menu has two columns: The left column is the file name, and the right contains header information about the file. "epan" indicates that it is an echo-planar image (functional image as we introduced from last chapter), whereas "abuc" stands for an anatomical image. "3D+t:300" indicates that it is a 3-dimensional with 300 volumes (time points) image



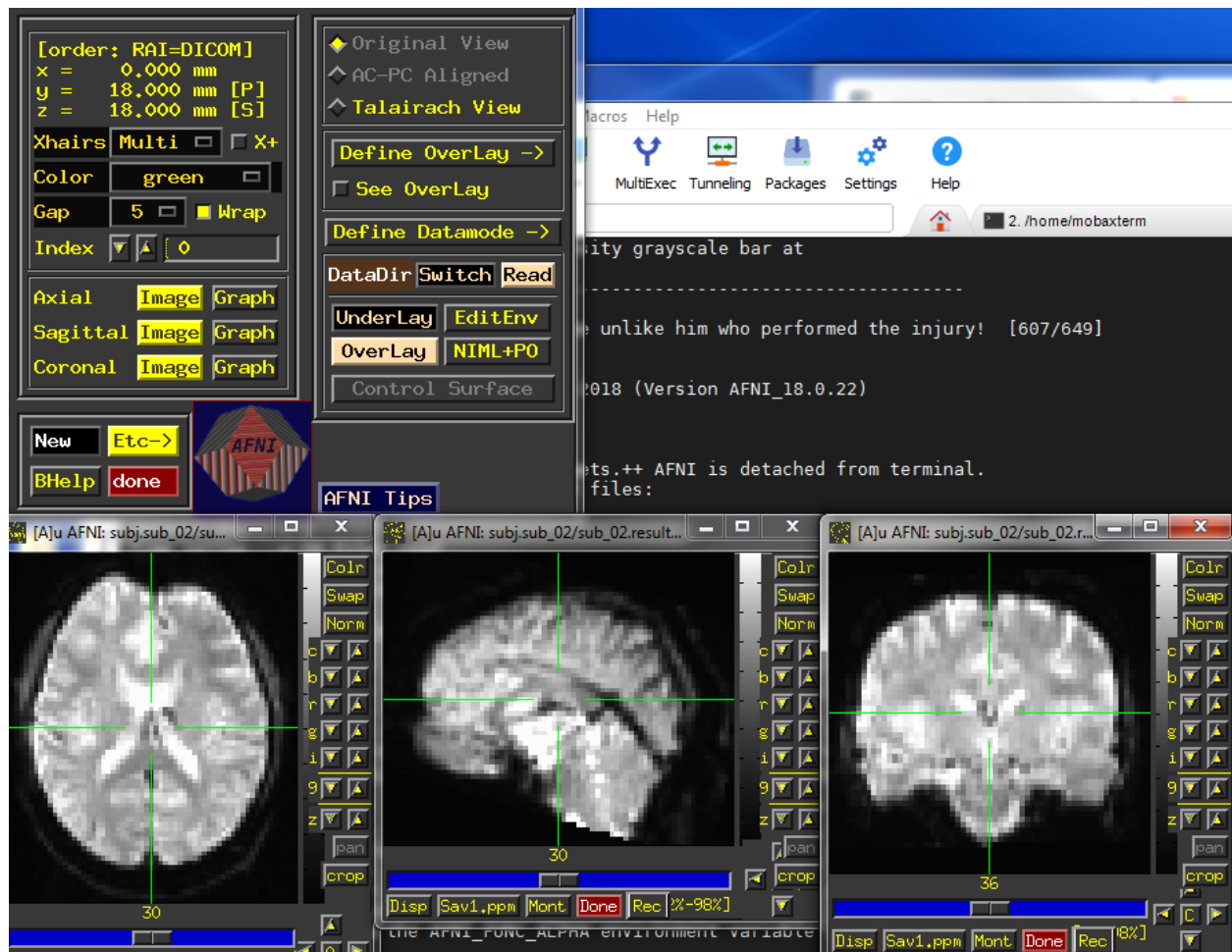


Aligned and Co-Registered image

The next file to look at is the “pb02.sub_02.r01.volreg+tlrc.BRIK”, which has 3 meanings:

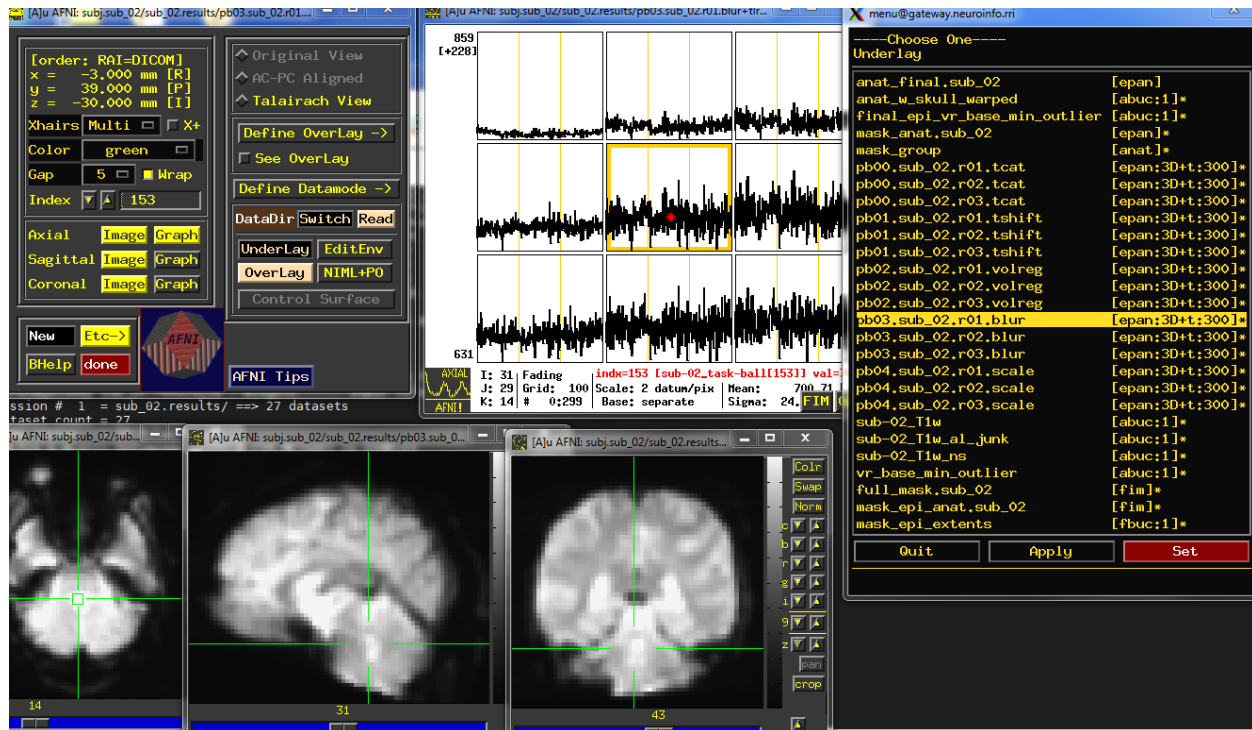
- 1 Motion-corrected, each volume in the time-series for this run has been aligned to a reference volume.
- 2 Co-registered to the anatomical image, the functional image has registered into anatomical image.
- 3 the images also have been normalized to a standardized space, which is MNI152 template.

If you click on the pb02... images, you will notice that there is a section of the AFNI GUI that has “Original View”, “AC-PC Aligned”, and “Talairach View”. In this images, the “Talairach View” is highlighted, which indicated that these images have been normalized. you can click the different locations of the images to see the differences.



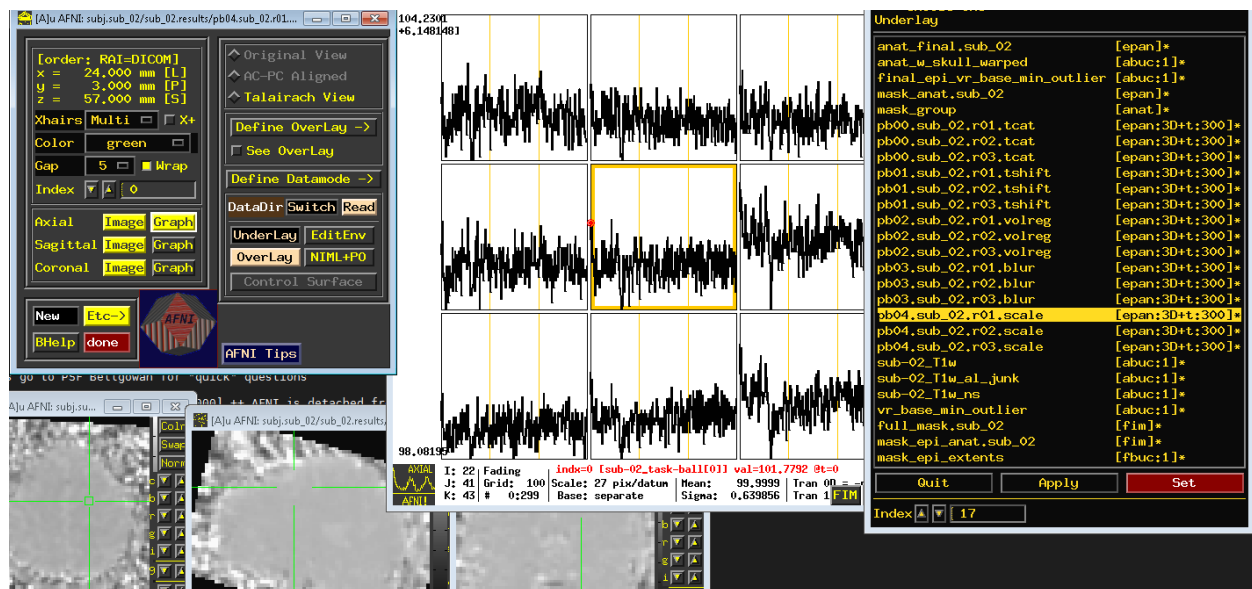
Smoothed image

The following preprocessing step is smoothing, which averages the signal of nearby voxels together in order to boost any signal that is there, and to cancel out noise. These images will look more blurry as a function of the size of the smoothing kernel that you apply to the data; in this case, a smoothing kernel of 4mm will blur the data slightly, but not by much. Look at the images to make sure that the blurring looks reasonable, as in the figure below.



Scaled image

The next step for preprocessing is scaled images, each voxel has a mean signal intensity of 100. This allows us to notice any relative changes to the mean as percent signal change such as a value of 101 could be interpreted as a signal change of 1%. these scaled images will be very blurred because the signal voxels within the brain are uniform compared to the variability of voxel outside of the brain. However it is still visible to see the outline of the brain, and the time-series values within the brain should be close to 100 as we saw previously.

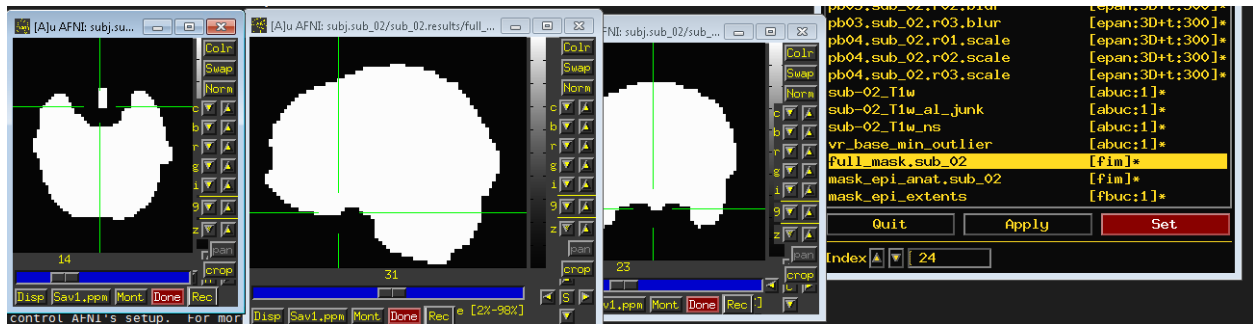


the Masks

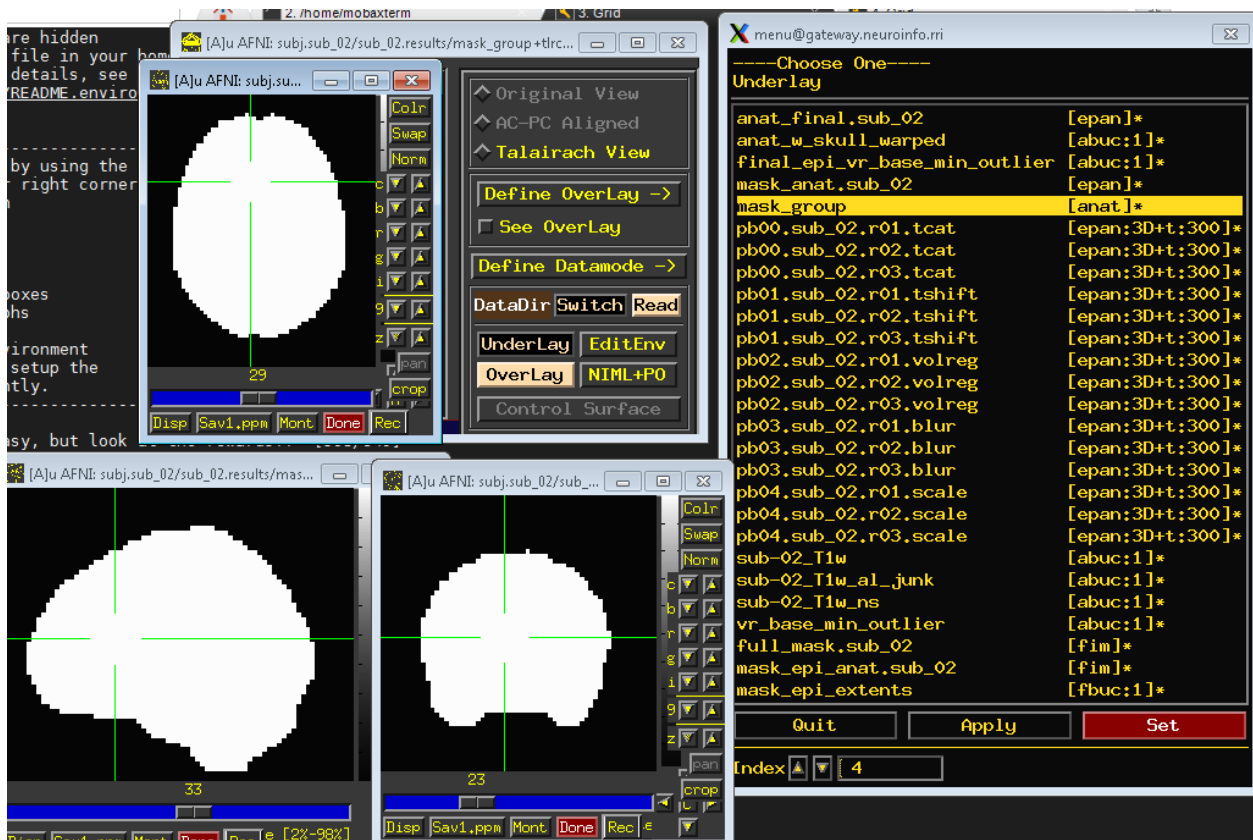
As it looks like, the mask is for the purpose in which we need to exclude all the signals outside of our interested areas. The mask is binary: 1 represents the voxels in the brain, and 0 indicates the outside of the brain (you can have more liberal or conservative masks to cover whatever you interested in)

There are two masks that you can choose between

1 full_mask is a union of all signal intensity of the individual functional image, Voxels with very low signal intensity are not considered brain voxels.

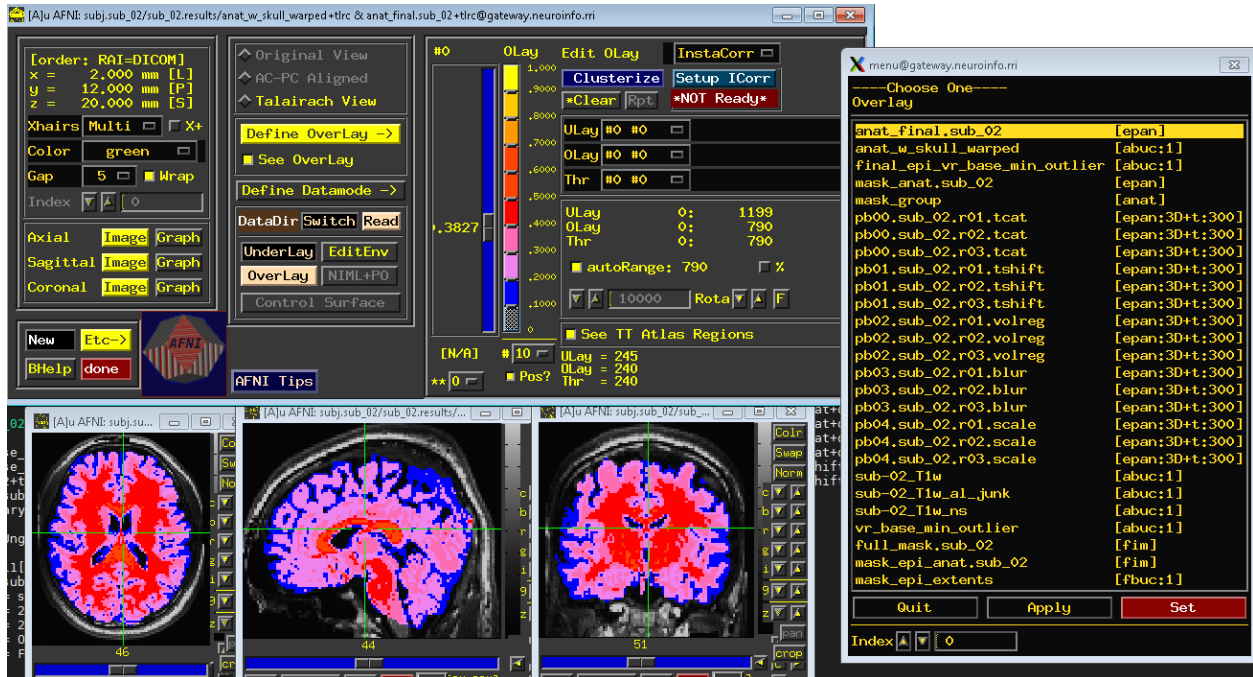


2 mask_group is a more liberal mask that match the template that you have chosen



Anatomical Images

The next look would be the anatomical images.



We can use `anat_w_skull_warped` as the underlay and `anat_final.sub_02` as the overlay.

All the output log from preprocessing steps would be store in the `output.proc.sub_02`, which is above the current directory. AFNI will keep move on even if it encouters errors but you can ckeck it with a text editor.

```

if ( `ldeval -a outcount.r$run.1D"{0}" -expr "step(a-0.4)"` ) then
ldeval -a outcount.r03.1D{0} -expr step(a-0.4)
end
cat outcount.r01.1D outcount.r02.1D outcount.r03.1D
set minindex = `3dTstat -argmin -prefix - outcount_rall.1D\`
3dTstat -argmin -prefix - outcount_rall.1D'
++ 3dTstat: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ Authored by: KR Hammett & RW Cox
^[[7m*+ WARNING:^[[0m Input dataset is not 3D+time; assuming TR=1.0
set ovals = ( `ld_tool.py -set_run_lengths $tr_counts
               -index_to_run_tr $minindex` )
ld_tool.py -set_run_lengths 300 300 300 -index_to_run_tr 332
set minouttr = 02
set minouttr = 32
echo min outlier: run 02, TR 32
tee out_min_outlier.txt
min outlier: run 02, TR 32
foreach run ( 01 02 03 )
3dTshift -tzero 0 -quintic -prefix pb01.sub_02.r01.tshift pb00.sub_02.r01.tcat+orig
++ 3dTshift: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
^[[7m*+ WARNING:^[[0m dataset is already aligned in time!
^[[7m*+ WARNING:^[[0m ==> output dataset is just a copy of input dataset
end
3dTshift -tzero 0 -quintic -prefix pb01.sub_02.r02.tshift pb00.sub_02.r02.tcat+orig
++ 3dTshift: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
^[[7m*+ WARNING:^[[0m dataset is already aligned in time!
^[[7m*+ WARNING:^[[0m ==> output dataset is just a copy of input dataset
end
3dTshift -tzero 0 -quintic -prefix pb01.sub_02.r03.tshift pb00.sub_02.r03.tcat+orig
++ 3dTshift: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
^[[7m*+ WARNING:^[[0m dataset is already aligned in time!
^[[7m*+ WARNING:^[[0m ==> output dataset is just a copy of input dataset
end
3dbucket -prefix vr_base_min_outlier pb01.sub_02.r02.tshift+orig[32]
++ 3dbucket: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
align_epi_anat.py -anat2epi -anat sub-02_T1w+orig -save_skullstrip -suffix _al_junk -epi vr_base_min_outlier+orig
#++ align_epi_anat version: 1.57
#++ turning off volume registration
#Script is running (command trimmed):
3dAttribute DELTA ./vr_base_min_outlier+orig
#Script is running (command trimmed):
3dAttribute DELTA ./vr_base_min_outlier+orig
#Script is running (command trimmed):
3dAttribute DELTA ./sub-02_T1w+orig

```

1.19.5 Statistics and modeling

After we finish all the preprocessing steps, we can go to the next - fit a model to the data. In order to understand model fitting, we need to know 3 fundamental components:

- 1 General Linear Model
- 2 The BOLD response
- 3 Time-series

General linear model

The General Linear Model (GLM) is a useful framework for comparing how several variables affect the continuous variables. Although it has many variants, the simplest form is described as: $Y = X + \epsilon$. In GLM, we can use one or more regressors(X) - independent variables, to fit a model for some outcome measure(Y) - or dependent variable. To do this we need to compute numbers called beta weights(), which are the relative weights assigned to each regressor in order to get the best fitting for the data. Any discrepancies between the model and the data are called residuals().

For example, imagine that we want to predict the chance of infecting Covid-19 based on social distance, daily interaction in person, and longitude. It is reasonable to find that social distance has a negative correlation with the infection of Covid-19, the more distance you keep with other people, the less chance you will get Covid-19. In terms of daily interaction in person, it has a positive association, the more you interact with people in person, you are more likely to be affected by the virus. lastly, longitude has no associati in general.

The General Linear Model (GLM)

**Uses one or more regressors (independent variables)
to predict an outcome measure (dependent variable)**

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

Y = Dependent variable

β = Beta Weights (parameter estimates)

X = Regressor

ε = Residual

Fig. 17: The symbols representing each of these terms are shown in the equation.

Therefore, we need assign each of these regressors a beta weight so that the model can fit the real data best. To be more specific, maybe each additional meter for social distance is associated with an additional 0.5 decrease in infection of Covid-19, while each additional 0.5 times of personal interaction is associated with a 0.3 increase:

Chance of infecting Covid-19 (Y) = $-0.5X_1(\text{social distance}) + 0.3X_2(\text{personal inteaction})$

This GLM can be expanded to include many regressors, but no matter how many regressors you have, the GLM assumes that the data can be modeled as a linear combination of each of the regressors. Keep GLM in your mind because we will apply it in our model soon.

BOLD response

BOLD signal

In 1990, a researcher at Bell Laboratories named Seiji Ogawa discovered that more deoxygenated blood leads to a decrease in the signal measured from a brain region. An increase in oxygenated blood, on the other hand, increases the signal - this increase in oxygenated blood was later shown to be correlated with increased neural firing. This change in signal is known as blood oxygen level dependent signal (BOLD signal). Shortly afterward, a researcher at Massachusetts General Hospital named Ken Kwong in 1992 demonstrated that the BOLD signal could be used as an indirect measure of neural activity. His experiment consisted of alternately showing a flashing checkerboard and a black screen to the subject for one minute each. The BOLD signal was recorded during each condition, as shown in the following video:

In the 1990's, empirical studies of the BOLD signal demonstrated that, after a stimulus was presented to the subject, any part of the brain responsive to that stimulus - say, the visual cortex in response to a visual stimulus - showed an increase in BOLD signal. The BOLD signal also appeared to follow a consistent shape, peaking around six seconds and then falling back to baseline over the next several seconds. This shape can be modeled with a mathematical function called a Gamma Distribution. As Gamma Distribution is created with parameters to best fit the BOLD response observed by the these empirical studies, it is the canonical Hemodynamic Response Function(HRF).

Gamma Distribution

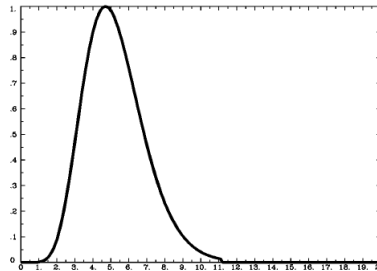
Gamma distribution is a distribution that arises naturally in processes for which the waiting times between events are relevant. It can be thought of as a waiting time between Poisson distributed events. For example, suppose you are fishing and you expect to get a fish once every 1/2 hour. Compute the probability that you will have to wait between 2 to 4 hours before you catch 4 fish. One fish every 1/2 hour means we would expect to get $= 1 / 0.5 = 2$ fish every hour on average. Using $\lambda = 2$ and $k = 4$, we can compute this as follows:

$$P(2 \leq X \leq 4) = \sum_{x=2}^4 \frac{x^{4-1} e^{-x/2}}{\Gamma(4)2^4} = 0.12388$$

When applied to fMRI data, the Gamma Distribution is called a basis function because it is the fundamental element of the model we will create and fit to the time series of the data. Furthermore, if we know what the shape of the distribution looks like in response to a very brief stimulus, we can predict what it should look like in response to stimuli of varying durations, as well as any combination of stimuli presented over time.

The HRF for a Single impulse Stimulus

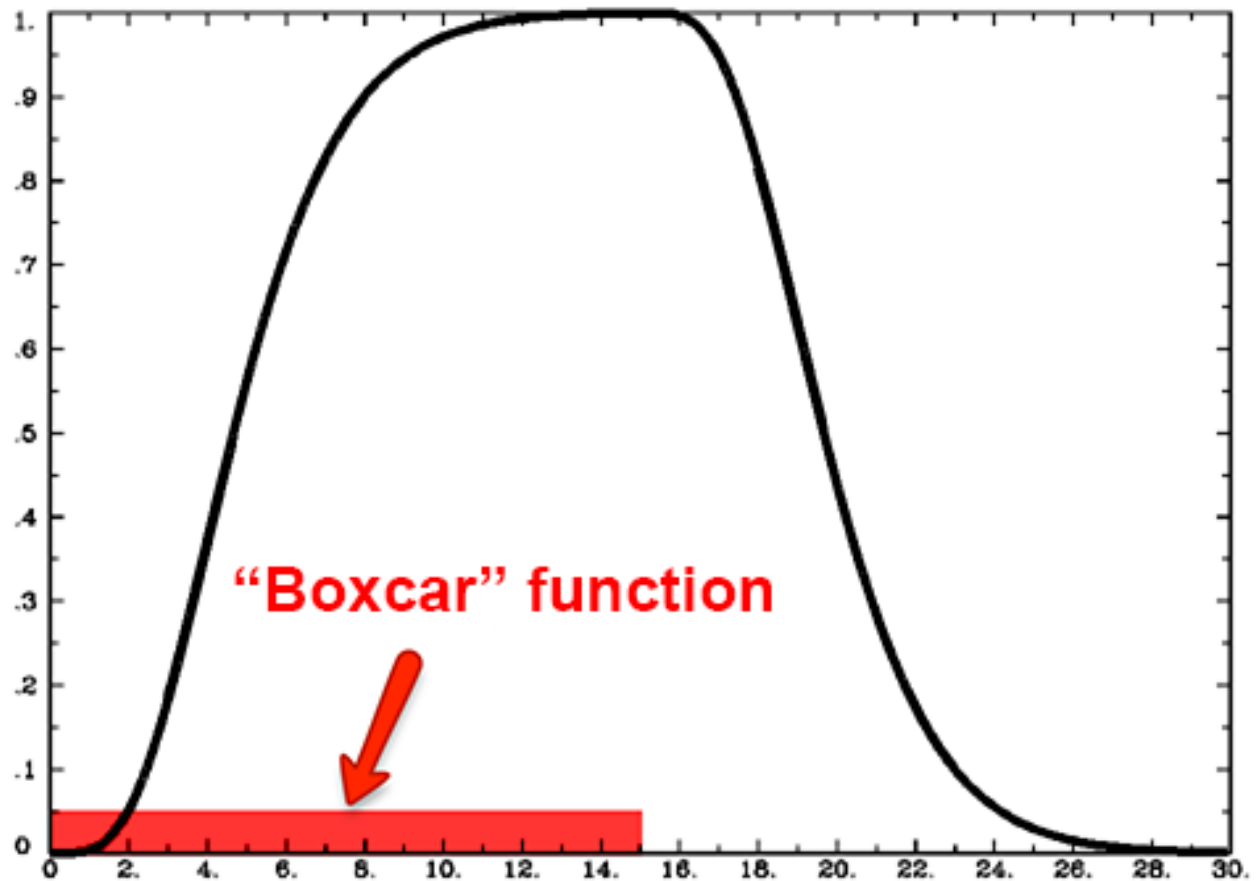
If the duration of a stimulus is very short, such as a snap of the fingers, we can say that it is an impulse stimulus - in other words, it has no duration. As you can see in the following figure, the shape of the BOLD signal looks like a typical Gamma Distribution, with a peak close to the beginning of the time axis (i.e., the x-axis) and a long tail to the right.



The HRF for a Single Boxcar Stimulus

Although many studies use stimuli lasting only a second or less, some studies present stimuli for longer periods of time. For example, imagine that the subject looks at a flashing checkerboard for fifteen seconds. In this case the shape of the HRF will be more spread out with a sustained peak proportional to the duration of the stimulus, falling back to baseline only after the stimulus has ended. This stimulus is called a boxcar stimulus, because it looks like a boxcar on a train.

In this case the Gamma Distribution is convolved with the boxcar stimulus. Convolution is the averaging of two functions over time; as a result, the Gamma Distribution broadens as it is averaged with the boxcar stimulus, and returns to baseline when the stimulus is removed.



Multiple HRFs overlapping

We have seen what the BOLD signal looks like after a stimulus is presented and how the HRF models the shape of that signal. But what happens if another stimulus is presented before the BOLD response for the previous stimulus has returned to baseline?

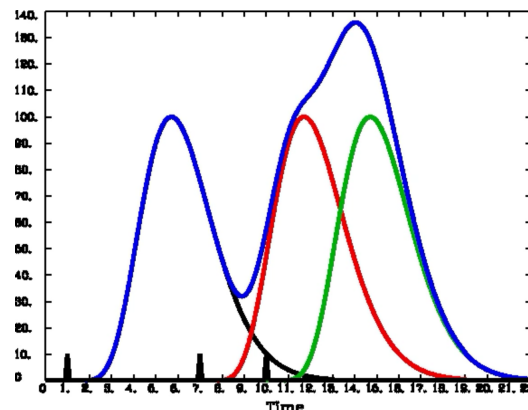


Fig. 18: Convolution of the HRFs for individual stimuli. The overall BOLD response (blue) is a moving average of the individual HRFs outlined in black, red, and green. The vertical black lines on the x-axis represent impulse stimuli. Figure created by Bob Cox of AFNI.

In that case, the individual HRFs are summed together. This creates a BOLD response that is a moving average of

the individual HRFs, and the shape of the BOLD signal becomes more complex as more stimuli are presented close together.

Fig. 19: Animations originally created by Bob Cox of AFNI

Time series

We have mentioned this concept several times before. As the basic composition of fMRI data. Remember that fMRI datasets contain several volumes strung together like beads on a string - we call this concatenated string of volumes a run of data. The signal that is measured at each voxel across the entire run is called a time-series. and this time-series represents the signal that is measured at each voxel during the whole scan..

Creating the time series

Since one of our goals is to create the ideal time-series so that we can use it to estimate the beta weights for our GLM, we need to create the ideal time-series first.

What do we need? Let's take a look at the BART dataset. you could find some "event.tsv" files in the subject's func directory. These files contain three pieces of information for the timing files. There are:

- 1 the experimental condition name
- 2 the onset time of trial for each conditon, relative to the onset of the scan
- 3 The duration of each trial

This information needs to be extracted from the events.tsv files and be transformed into a format that the AFNI can read. our job is to create a timing file for explode and cash experimental condition, and then split the file based on which run the condition was in. We will have 6 timing files:

- 1 Timings for the explode trials that occurred during the first run (explode_run1.txt)
- 1.2 Timings for the explode trials that occurred during the first run (explode_run2.txt)
- 1.3 Timings for the explode trials that occurred during the first run (explode_run3.txt)
- 2 Timings for the cash trials that occurred during the first run (cash_run1.txt)
- 2.2 Timings for the cash trials that occurred during the second run (cash_run2.txt)
- 2.3 Timings for the cash trials that occurred during the third run (cash_run3.txt)

Each of these timing files will have three columns:

- 1 Onset time, in seconds, relative to the start of the scan
- 2 Duration of the trial, in seconds
- 3 Parametric modulation(discuss later)

Next, we will condense the timing files for each run into a single file for each condition, which will be called explode.1D and cash.1D. The "1D" is a specification to AFNI; it indicates that the file contains the text information arranged in rows and columns.

Admittedly, we will do this by our hands but not every details. There is a script that can help us to create these timing files, you can copy this can create a script in the BART directory by a text editor and name it as `timing.sh`, and type `bash timing.sh`. If everthong goes smoothly, This will create timing files for each run for each subject on the two conditions, and save them in each subject's func directory. You will see "cash.1D", "control.1D" as well as "cash_run1.txt" "control_run1.txt" from "sub-02/func/:

```
#!/bin/bash

#Check whether the file subjList.txt exists; if not, create it
if [ ! -f subjList.txt ]; then
    ls -d sub-?? > subjList.txt
fi

#Loop over all subjects and format timing files into FSL format
for subj in `cat subjList.txt` ; do
    cd $subj/func #Navigate to the subject's func directory, which contains the
    ↪ timing files

    #Extract the onset times for the explode and cash trials for each run.

    cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3==
    ↪ "explode_demean") {print $1, $2, "1"}}' > explode_run1.txt
    cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3==
    ↪ "explode_demean") {print $1, $2, "1"}}' > explode_run2.txt
    cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3==
    ↪ "explode_demean") {print $1, $2, "1"}}' > explode_run3.txt

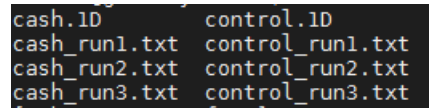
    cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="cash_
    ↪ demean") {print $1, $2, "1"}}' > cash_run1.txt
    cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="cash_
    ↪ demean") {print $1, $2, "1"}}' > cash_run2.txt
    cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="cash_
    ↪ demean") {print $1, $2, "1"}}' > cash_run3.txt

#Now convert to AFNI format

    timing_tool.py -fsl_timing_files explode*.txt -write_timing explode.1D

    timing_tool.py -fsl_timing_files cash*.txt -write_timing cash.1D

    cd ../../
done
```



```
cash.1D      control.1D
cash_run1.txt control_run1.txt
cash_run2.txt control_run2.txt
cash_run3.txt control_run3.txt
```

Once the timing files has been created, you are now ready to use them to fit a model to the fMRI data.

Homework

In order to understand the script bettwem, you can makde some modification to extract the information for pump, control trial and make the timing files.

1.19.6 Running the 1st level analysis

Previously, we have used **uber_subject.py** to set up the preprocessing for a single subject. You may remember that we removed one of the blocks called “regress”. Now, we will add the regress back and combine both preprocessing and 1st-level analysis. Let's start a new script for the same subject, sub-02. First of all, we need to remove the preprocessed result by typing `rm -r subject_results` from sub-02 directory. And type `uber_subject.py` from the terminal. We will leave “analysis initialization” as the default setting like all the blocks keep as they are.

Let's copy what we did in the preprocessing such as `fill the name tab`, `anatomical dataset`, `EPI dataset`, `expect options`, `extra align options`, `extra tlrc options``. But this time, we need to do more with 3 more sections:

1 stimulus timing files 2 symbolic GLTs 3 extra regress options

Stimulus Timing Files

As we have created the timing files in the last chapter, click the **browse stim** to select the “explode.1D” and “cash.1D” files from sub-02/func. After you click the OK, there are 3 more columns jump up; “label”, “basis”, and “type”. “Label” is how the timing reference files in the command that does the model fitting (i.e., 3dDeconvolve), “basis” and “type” indicated that whaty kind of function we will apply to the timing files.

The default basis function is GAM, which specifies that the onset times should be convolved with the canonical HRF as we discussed before from the statistics and modeling chapter. This basis function requires the height of the HRF as the parameter for estimation, which roughly corresponds to the amount of neural activity in response to that condition. The tab below, “init file types” specify the type of convolution we are going to use. The default setting is times, which means to convolve all of the time-points specified in the timing file and generate a parameter estimate for the HRF that on average fits all of the occurrences.

☒ stimulus timing files

browse stim clear stim ? help

index	label	basis	type	stim (timing) file
1	cash	GAM	times	cash.1D
2	explode	GAM	times	explode.1D

stim directory: /home/wshao/BART_afni/sub-02/func

wildcard form: *1D

stim file count: 2

init basis funcs: choose GAM

init file types: choose times

☐ use wildcard form

Symbolic GLTs

The following section, Symbolic GLTs allows you to specify general linear tests after the AFNI estimated the beta weights from the different conditions from the above section. let's take the "init with examples" as an example. After you click it, there are two contrasts: C-E, and mean. CE. The first contrast uses contrast weights to specify how the contrast between different conditions will be calculated, and it prepares the contrast weights to the labels we have specified from "stimulus timing files" section. the default of no sign in a contrasting weight is +1, while a negative sign will assign a contrasting weight of -1. Therefore, the line here shows that we assign weight to the parameter estimate for the Cash condition with +1, and assign weight to the parameter estimate for the explode condition with -1. The second contrast, use different computation, takes the average of the two conditions by multiplying 0.5 for both.

In general, the sum of contrast weights of different conditions should be 0, and the sum of the average of contrast weights across conditions should be 1.

Let's change this GLT syntax table and to set up the contrasts. In the first row, change the "label" column to "cash_explode", and in the later "symbolic GLT" column, do **cash -explode**. similarly, the second row we need to specify a contrast of explode_cash as well as explode -cash. Then, click **insert glt row** and put the **0.5*cash +0.5*explode** the section will look like this:

	label	symbolic GLT
1	cash_explode	cash -explode
2	explode_cash	explode -cash
3	mean.CE	0.5*cash +0.5*explode

GLT count | 3

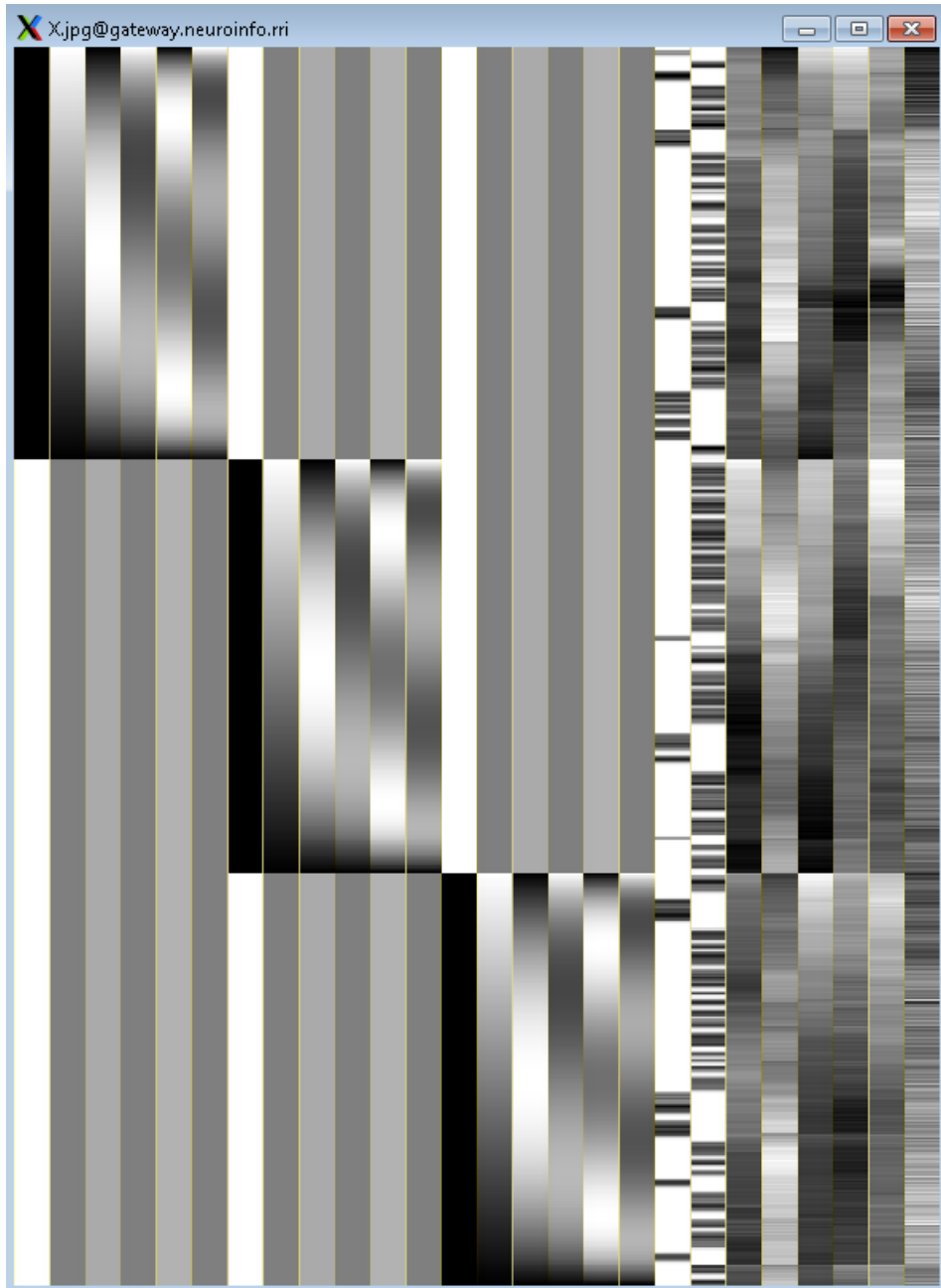
Extra Regress Options

Lastly, let's check out the other options. The first one, "outlier censor limit", will remove any TRs from this analysis that has been censored outlier fraction greater than the number we put on the right side. (For example, by 3dToutcount, if we type 3 in there, which will signify any voxels in a TR that have signal intensity greater than 3 standard deviations from the other voxels in brain mask.) If this is left at 0.0, then will be no TRs will be censored based on the output from 3dToutcount. We will use the default setting now.

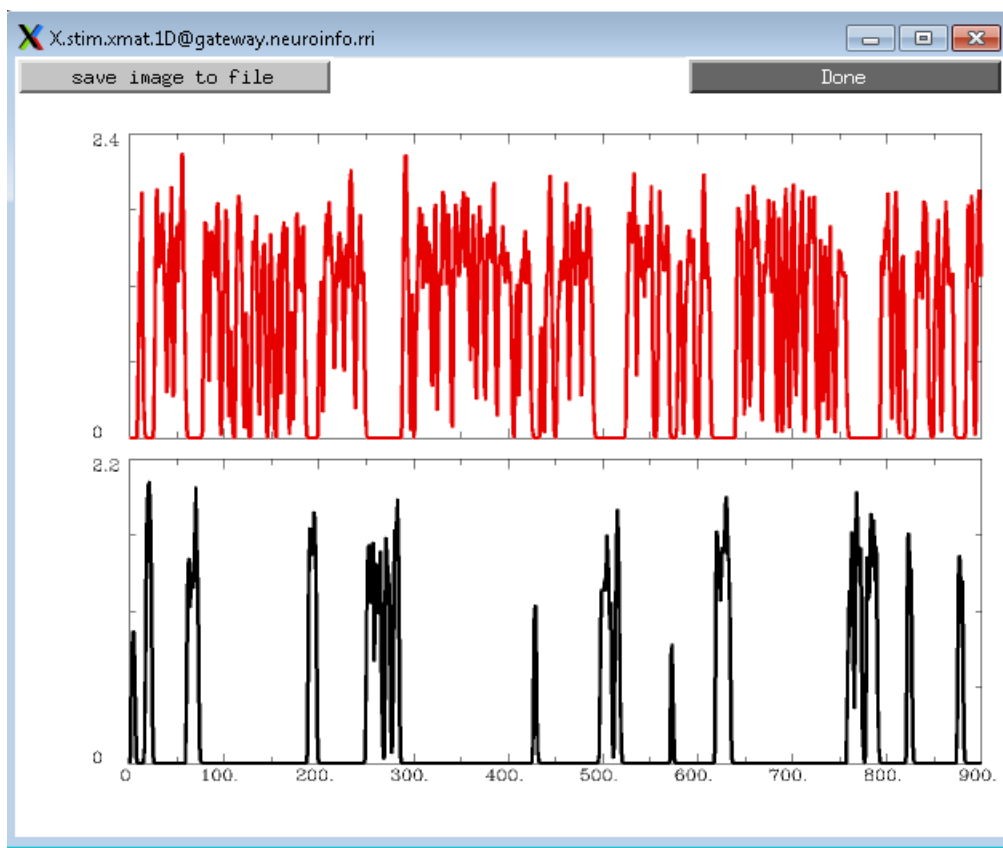
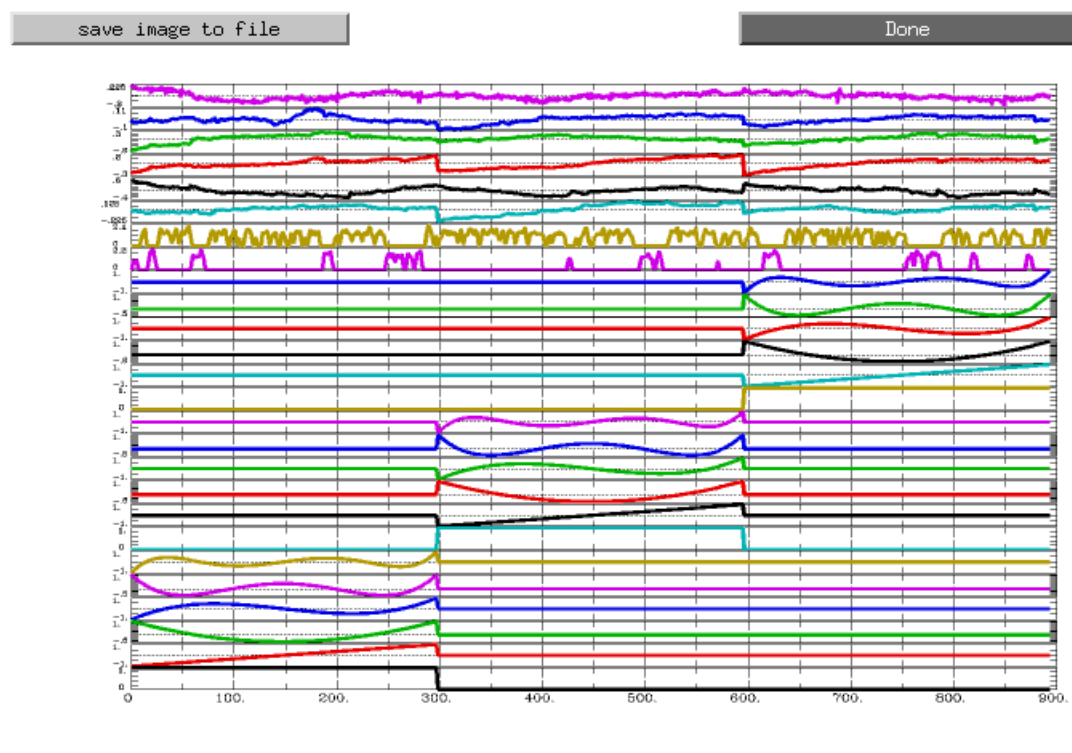
The next section, "jobs for regression (num CPUs)", indicate how many processors we can use for the regression analysis. You can use the maximum number of CPUs that you can spare. We will set it to 8. "GOFORIT level (override 3dD warnings)" will ignore any warnings about the design matrix detected by 3dDeconvolve. In general, you will want 3dDeconvolve to throw a warning and stop running when it finds a high level of collinearity among two or more regressors.

The "bandpass in regression" tab will be used for resting-state analyses to remove both low and high frequency fluctuations. And low-pass filtering (i.e., removing high frequency signal) risks will be applied to removing actual signal related to the task.

There are four more options in addition. 1 "Regress motion derivatives" will model higher-order derivatives of the motion regressors, which can capture more complex head movements. This is useful for populations that tend to move a lot, such as children or clinical subjects; and as long as you have a long time-series of data (more than 200 TRs in a run). 2 The "run cluster simulation" box unchecked, as this computes whether a cluster is statistically significant in real time as you change the thresholding slider. We will do inference later at a group level later. 3 I do, however,

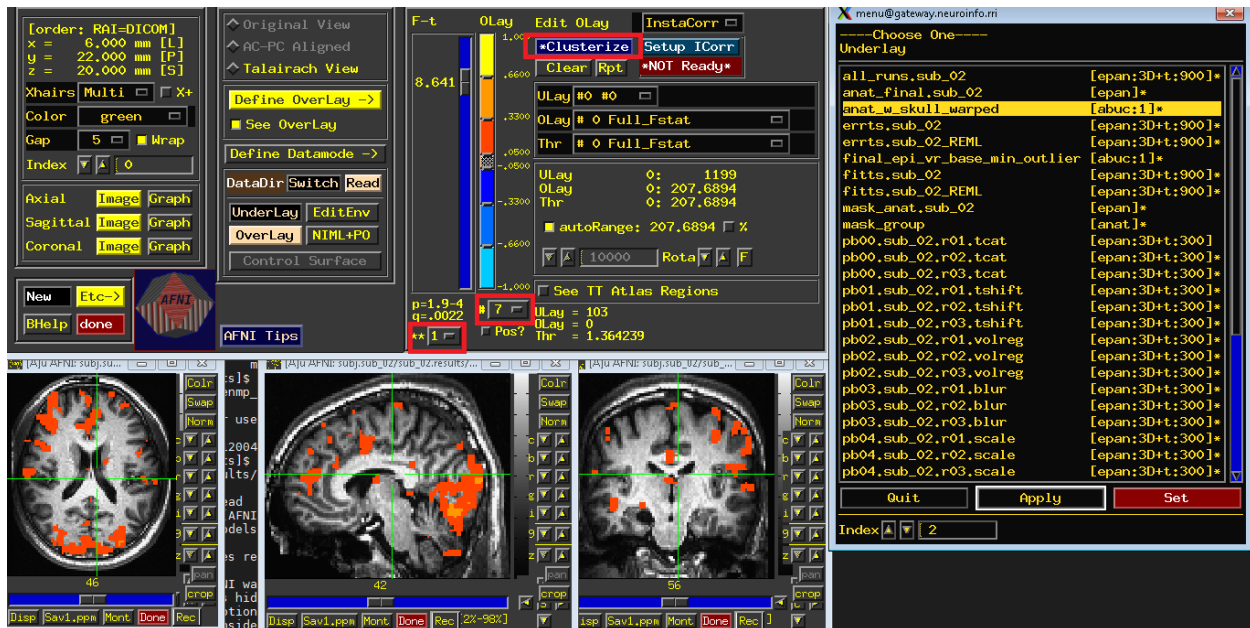


For a different view, looking at all of the regressors and stimulus in separate rows, type `1dplot -sepscl X.xmat.1D` and `X.stim.xmat.1D`.



The Statistics Files

The final view is to look at the contrast maps of our data. Open up the GUI, you can select “anat_w_skull_warped” or “anat_final.sub_02” as the Underlay and you can choose select stats.sub_02 as Overlay. Then, Go to the button of F-t column and click 1 and select the 7 for the box on the right side. move on, click Clusterize tab and select the 30 for the voxel tab. You will see something like this:



1.19.7 Automation

Since we have 16 subjects and each subject has 3 runs. In total, we need to repeat all the preprocessing and 1st level analysis in AFNI_GUI 48 times! it is not hard but it is a really tedious job and you could make errors easily. As Joey from Friends said, there's gotta be a better way, and there is.

Here is the script that makes your life easier!

Create a design file

We have analyzed the first subject, sub-02, and created a file called **proc.sub_02**. This file contained a list of AFNI commands, composed in a manner determined by `uber_subject.py` GUI which we played before. Now it is time to `cp` this file to the BART directory that has all of your subjects, and rename it as **proc_BART.sh**. Let's go to the directory which has the "pro.sub_02" file and type this:

```
cp proc.sub_02 ../../../../proc.BART.sh
```

After that, you are able to find the `proc.BART.sh` in your BART directory, We are going to 2 things:

1 Remove every reference to sub-02, and turn those strings into a new variable. For example, we will change the script so that it will replace the variable in our script with the string whatever we want, and analyze the corresponding subject's data.

2 We will replace the paths to be more generalizable so that AFNI can find the corresponding input data.

To begin with, open the `proc_BART.sh` with a text editor like nano or other editors. Scroll down a little bit, which contains the following code:

```
# the user may specify a single subject to run with
if ( $#argv > 0 ) then
    set subj = $argv[1]
else
    set subj = sub_02
endif
```

This is a conditional statement. The first few lines state that if the user provides an argument as input, then set the variable "subj" to whatever the argument is. If you look through the rest of the script **proc_BART.sh**, you will see numerous lines that contain the variable "\$subj". which is our argument. Another part we need to change is the paths which involves the sub-02 such as we will need to replace these with the "\$subj" variable. typy "Ctrl+W" and "Ctrl+R" to replace it as `${subj}`.

Next, we will need to replace the relative path. As you can see in the script, there are several lines of code that contain paths starting with `/home/wshao/BART_afni/`. We will replace this with the `$PWD` or the whatever current working directory path you have, This will ensure that the script will be adapted to the your work environment. For example, you can change the `/home/wshao/BART_afni/` to `$PWD`

You can download the script from [here](#) can make the some changes for you dataset if necessary.

Automating the Analysis

When you done the set up for the **proc_BART.sh**, use this updated preprocessing script in a for-loop to loop over all of the subjects in our dataset. with this code:

```
for subj in `cat subjList.txt`; do
    tcsh proc.BART.sh $subj;
    mv ${subj}.results $subj;
done
```

you can write this code in a new script and bash it or run this code in your terminal directly.

```
[wshao@gateway BART_afni]$ for subj in `cat subjList.txt`; do
>   tcsh proc.BART.sh $subj;
>   mv ${subj}.results $subj;
> done
auto-generated by afni_proc.py, Thu Apr  8 19:08:24 2021
(version 6.06, February 26, 2018)
execution started: Fri Apr  9 19:21:19 EDT 2021
Precompiled binary linux openmp 64: Feb 26 2018 (Version AFNI 18.0.22)
```

```
[wshao@gateway BART_afni]$ for subj in `cat subjList.txt`; do
>   tcsh proc.BART.sh $subj;
>   mv ${subj}.results $subj;
> done
auto-generated by afni_proc.py, Thu Apr  8 19:08:24 2021
(version 6.06, February 26, 2018)
execution started: Thu Apr  8 19:56:13 EDT 2021
Precompiled binary linux openmp 64: Feb 26 2018 (Version AFNI 18.0.22)
++ 3dcopy: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ 3dTcat: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ elapsed time = 2.6 s
++ 3dTcat: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ elapsed time = 3.0 s
++ 3dTcat: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ elapsed time = 2.6 s
++ 3dToutcount: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ 32809 voxels passed mask/clip
++ 3dToutcount: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ 32845 voxels passed mask/clip
++ 3dToutcount: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ 32854 voxels passed mask/clip
++ 3dTstat: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
++ Authored by: KR Hammett & RW Cox
*+ WARNING: Input dataset is not 3D+time; assuming TR=1.0
min outlier: run 01, TR 134
++ 3dTshift: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
*+ WARNING: dataset is already aligned in time!
*+ WARNING: ==>> output dataset is just a copy of input dataset
++ 3dTshift: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
*+ WARNING: dataset is already aligned in time!
*+ WARNING: ==>> output dataset is just a copy of input dataset
```

1.19.8 Group_analysis

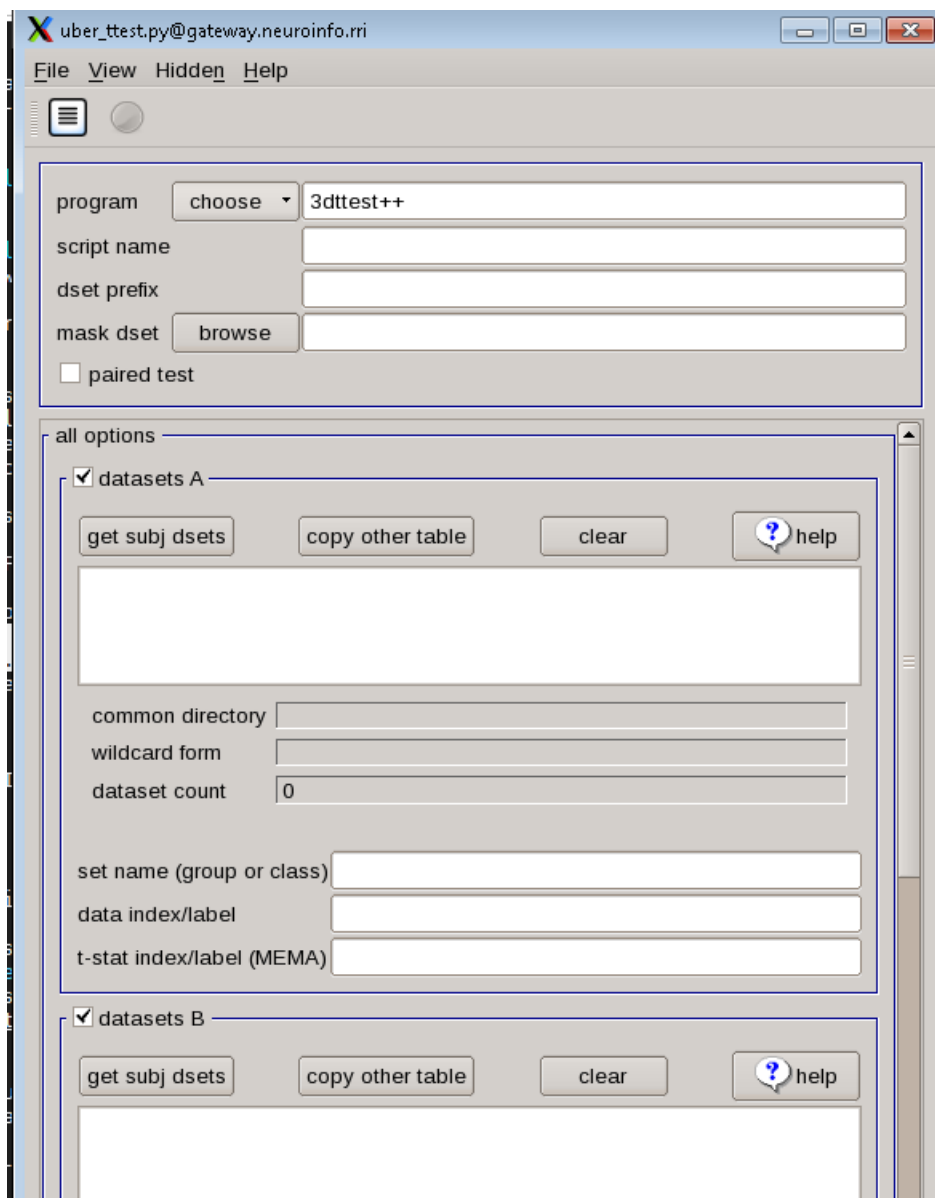
one of the goals in fMRI analysis is to generalize the results from the sample to the population. In other words, if we see the pattern of brain activity in our sample, it is reasonable to infer that these changes would most likely happen in the population as well?

In order to test this, a group-level analysis (second-level analysis) is needed. we need to calculate the standard error and the mean for a contrast estimate, and then test whether the average estimate is statistically significant. There are two ways to do this group-level analysis in AFNI:

1 3dttest, which uses only the contrast estimates in testing for statistical significance 2 3dMEMA, which accounts for both the difference between the parameter estimates, and the variability of that contrast.

uber_ttest.py

We can use an AFNI GUI to set up our group-level analyses like we did with `uber_subjects.py`. type the command `uber_ttest.py` from the terminal and press return, you will see:



The first tab, “program”, allows 3dtttest and 3dMEMA. The difference between these two is that 3dMEMA will count the variability of the estimate as well, and give more weight to those subjects who have lower variability in their estimates. For now, we use “3dtttest”. In the next part, the “script name” and “dset prefix” tab, type “cash-explode_ttest”. For the “mask dset”, choose the **mask_group+tlrc.HEAD** located in the results directories.

Using wildcards, we can select each of the subject statistical datasets. Click “get subj dsets” from “datasets A”, select a representative file. Select a subjects’ statistical datasets such as “stats.sub-02+tlrc.HEAD”, and replace the last two numbers of sub-“02” with two question marks “sub-??”. Then click on “apply pattern”. If all of the subjects we analyzed are the same way and have the same directory structure, there will be 16 entries in the field below.

At the bottom of the “datasets A” section, you will see a few additional fields. In “set name (group of class)”, write cash-explode, and in “data index/label” type 7.

Why 7?, type 3dinfo -verb stats.sub-02+tlrc from any results directories such as **sub-02.results**. you will see this:

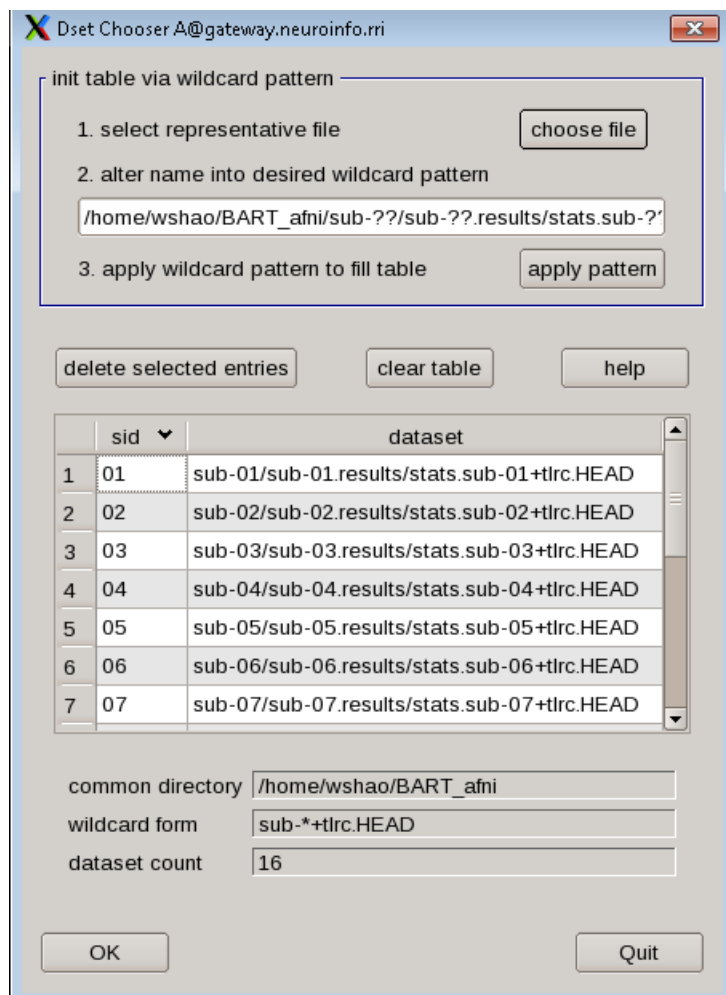
```

Dataset File: stats.sub-02+tlrc
Identifier Code: XYZ_dZZ2zWUKzdAdlx7cixeg3A Creation Date: Thu Apr 8 20:53:57 2021
Template Space: TT_N27
Dataset Type: Func-Bucket (-fbuc)
Byte Order: LSB_FIRST [this CPU native = LSB_FIRST]
Storage Mode: BRIK
Storage Space: 11,059,200 (11 million [mega]) bytes
Geometry String: "MATRIX(3,0,0,-79.5,0,3,0,-79.5,0,0,3,-63.5):54,64,50"
Data Axes Tilt: Plumb
Data Axes Orientation:
  first (x) = Right-to-Left
  second (y) = Anterior-to-Posterior
  third (z) = Inferior-to-Superior [-orient RAI]
R-to-L extent: -79.500 [R] -to- 79.500 [L] -step- 3.000 mm [ 54 voxels]
A-to-P extent: -79.500 [A] -to- 109.500 [P] -step- 3.000 mm [ 64 voxels]
I-to-S extent: -63.500 [I] -to- 83.500 [S] -step- 3.000 mm [ 50 voxels]
Number of values stored at each pixel = 16
-- At sub-brick #0 'Full_Fstat' datum type is float: 0 to 219.011
  statcode = fitt; statpar = 2 868
-- At sub-brick #1 'cash#0_Coef' datum type is float: -11.9985 to 18.9309
-- At sub-brick #2 'cash#0_Tstat' datum type is float: -6.13623 to 14.4952
  statcode = fitt; statpar = 868
-- At sub-brick #3 'cash_Fstat' datum type is float: 0 to 210.112
  statcode = fitt; statpar = 1 868
-- At sub-brick #4 'explode#0_Coef' datum type is float: -10.287 to 9.94113
-- At sub-brick #5 'explode#0_Tstat' datum type is float: -7.78975 to 17.3863
  statcode = fitt; statpar = 868
-- At sub-brick #6 'explode_Fstat' datum type is float: 0 to 302.284
  statcode = fitt; statpar = 1 868
-- At sub-brick #7 'C-E_GLT#0_Coef' datum type is float: -15.4523 to 15.9057
-- At sub-brick #8 'C-E_GLT#0_Tstat' datum type is float: -5.59533 to 10.2448
  statcode = fitt; statpar = 868
-- At sub-brick #9 'C-E_GLT_Fstat' datum type is float: 0 to 104.955
  statcode = fitt; statpar = 1 868
-- At sub-brick #10 'E-C_GLT#0_Coef' datum type is float: -15.9057 to 15.4523
-- At sub-brick #11 'E-C_GLT#0_Tstat' datum type is float: -10.2448 to 5.59533
  statcode = fitt; statpar = 868
-- At sub-brick #12 'E-C_GLT_Fstat' datum type is float: 0 to 104.955
  statcode = fitt; statpar = 1 868
-- At sub-brick #13 'mean.CE_GLT#0_Coef' datum type is float: -8.47223 to 14.1633
-- At sub-brick #14 'mean.CE_GLT#0_Tstat' datum type is float: -6.92524 to 20.3839
  statcode = fitt; statpar = 868
-- At sub-brick #15 'mean.CE_GLT_Fstat' datum type is float: 0 to 415.503
  statcode = fitt; statpar = 1 868

```

This will show all of the sub-briks and the labels accordingly. As we are looking for the contrast estimates for cash-explode. The output of 3dinfo shows that this is located in sub-brik 7(C-E). The sub-brik that contains the label “Coef” means that it is a parameter (or contrast) estimate. “Tstat” indicates a t-statistic. “Fstat” means an F-statistic.

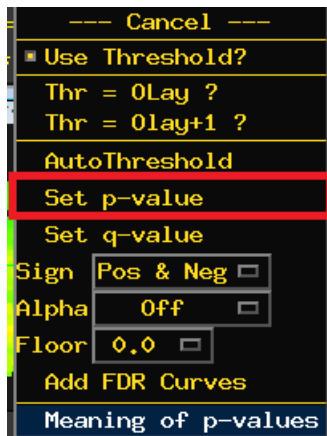
After you have done all the steps, we should see a form looks like this:

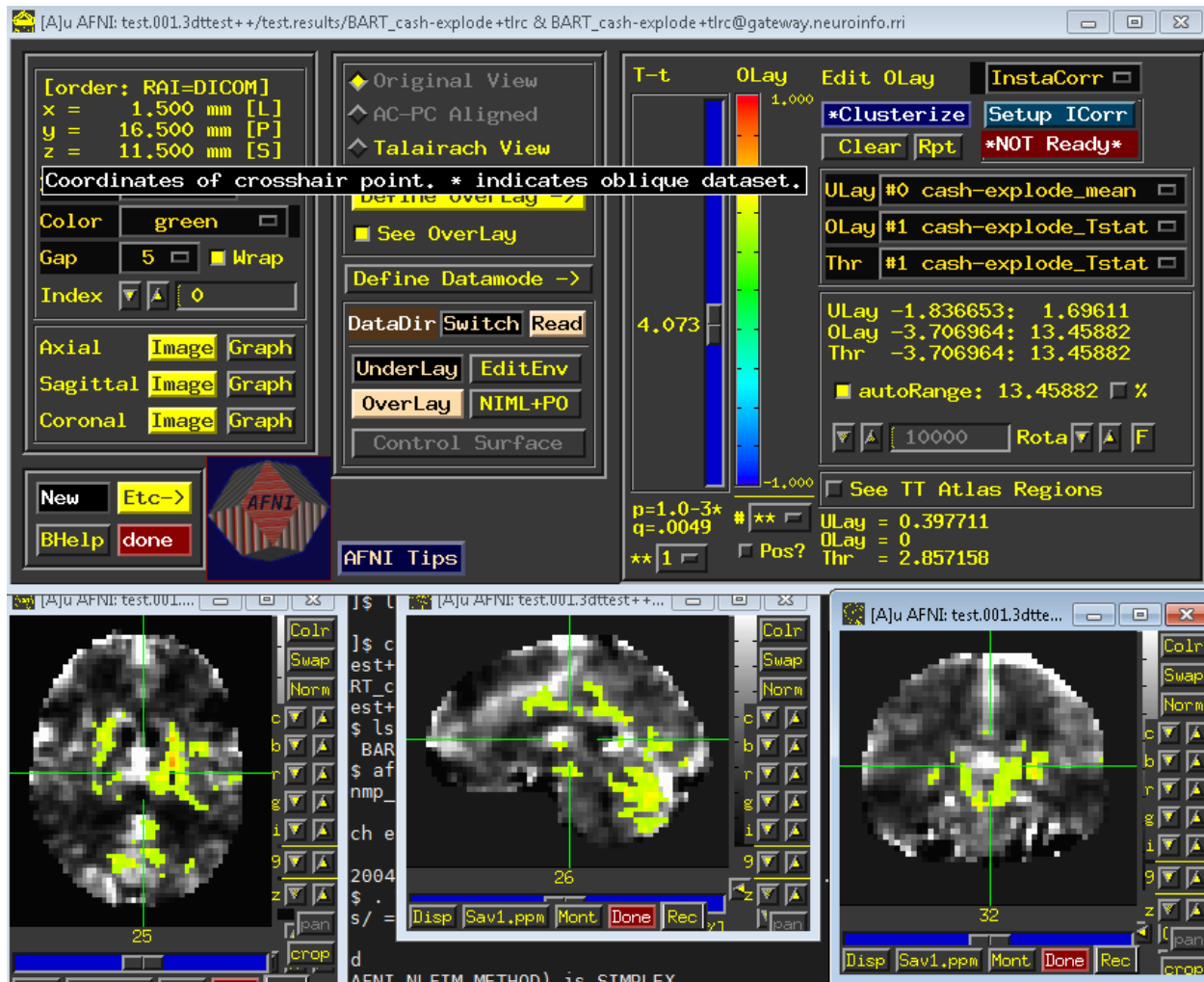


As with the `uber_subject.py` script, there are buttons at the top of the GUI for both generating the script and then running the script. First click on the icon that looks like a sheet of paper with lines on it, which will show you the command that has been generated. Review it to see how it has inserted all of your inputs into a command called `3dttest++`, which will run the actual group-level analysis. Then click on the green “Go” icon to run the test.

Viewing the Results

When Go has been finished, You will see a new directory called “group_results” has been created if you go back to Terminal and type `ls, cd to test.001.3dttest++`. Which contains the script we just produced (“BART_cash-explode_ttest”), another folder called `test.results`, which contains the group-level output. Load this in the afni viewer, set an uncorrected p-value of 0.001 (by right-clicking on the “p=”) and clusterize the data to 40, which show clusters with an extent of 30 voxels or more.



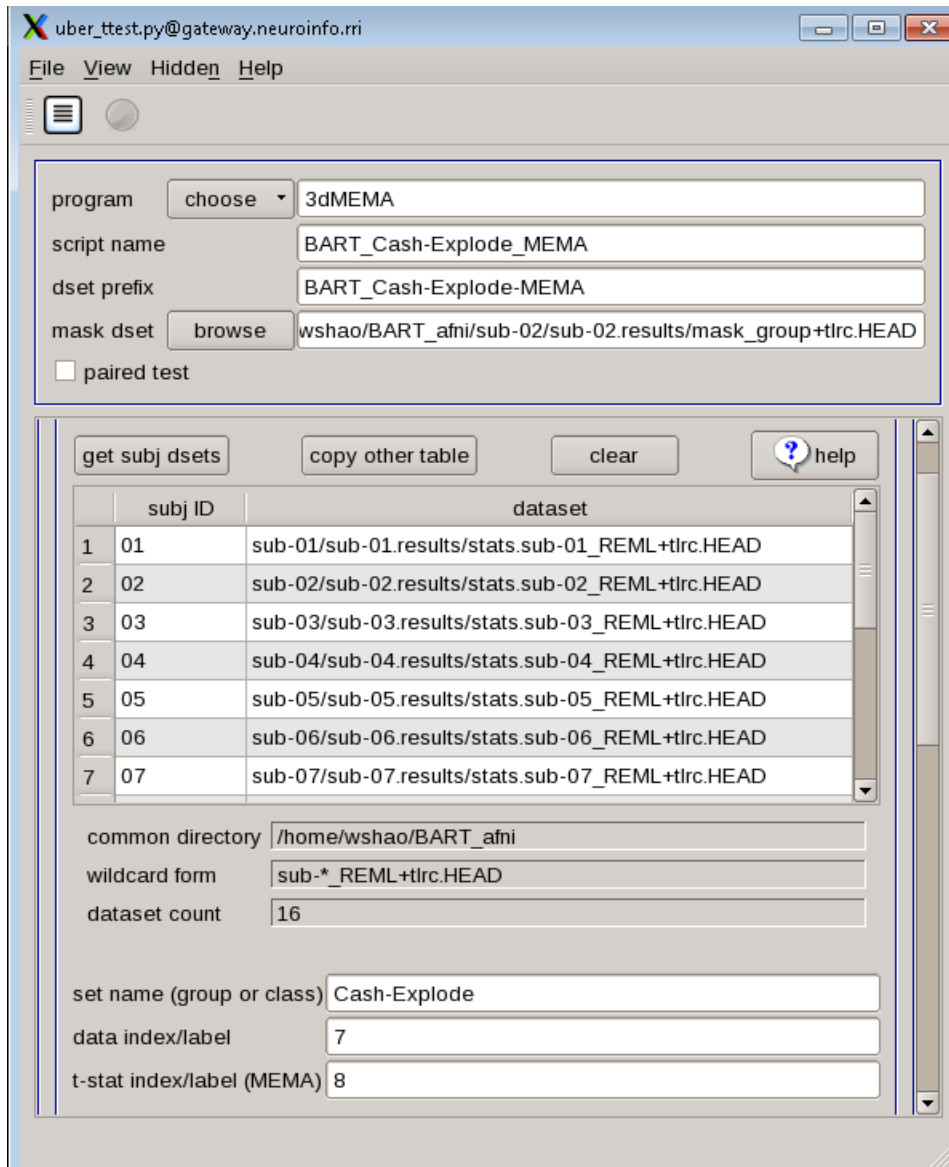


3dMEMA

Close the AFNI viewer, and go back to the uber_ttest.py GUI, and make the following changes:

- 1 Change the “program” from 3dttest++ to 3dMEMA.
- 2 Change the “script name” and “dset prefix” to BART_Cash-ExplodeCash-Explode .
- 3 Click on “get subj dsets”, and select a subject’s statistical dataset such as stats.sub-02_REML+tlrc. Use the wildcards as above to select all of the subjects’ REML datasets.
- 4 In the field “t-stat index/label (MEMA)”, type 8. The sub-briks of the REML dataset, which should be in an order identical to the non-REML statistical dataset, indicate that sub-brik #8 is the t-statistic associated with the contrast estimate of “incongruent-congruent.”

As before, click on the script generator icon, and then click on the green “Go” button. This model estimation will take longer, and you will see a progress report for each slice that has been analyzed; in total, it should take only a couple of minutes.



program 3dMEMA

script name BART_Cash-Explode_MEMA

dset prefix BART_Cash-Explode-MEMA

mask dset wshao/BART_afni/sub-02/sub-02.results/mask_group+tlrc.HEAD

☐ paired test

	subj ID	dataset
1	01	sub-01/sub-01.results/stats.sub-01_REML+tlrc.HEAD
2	02	sub-02/sub-02.results/stats.sub-02_REML+tlrc.HEAD
3	03	sub-03/sub-03.results/stats.sub-03_REML+tlrc.HEAD
4	04	sub-04/sub-04.results/stats.sub-04_REML+tlrc.HEAD
5	05	sub-05/sub-05.results/stats.sub-05_REML+tlrc.HEAD
6	06	sub-06/sub-06.results/stats.sub-06_REML+tlrc.HEAD
7	07	sub-07/sub-07.results/stats.sub-07_REML+tlrc.HEAD

common directory /home/wshao/BART_afni

wildcard form sub-*_REML+tlrc.HEAD

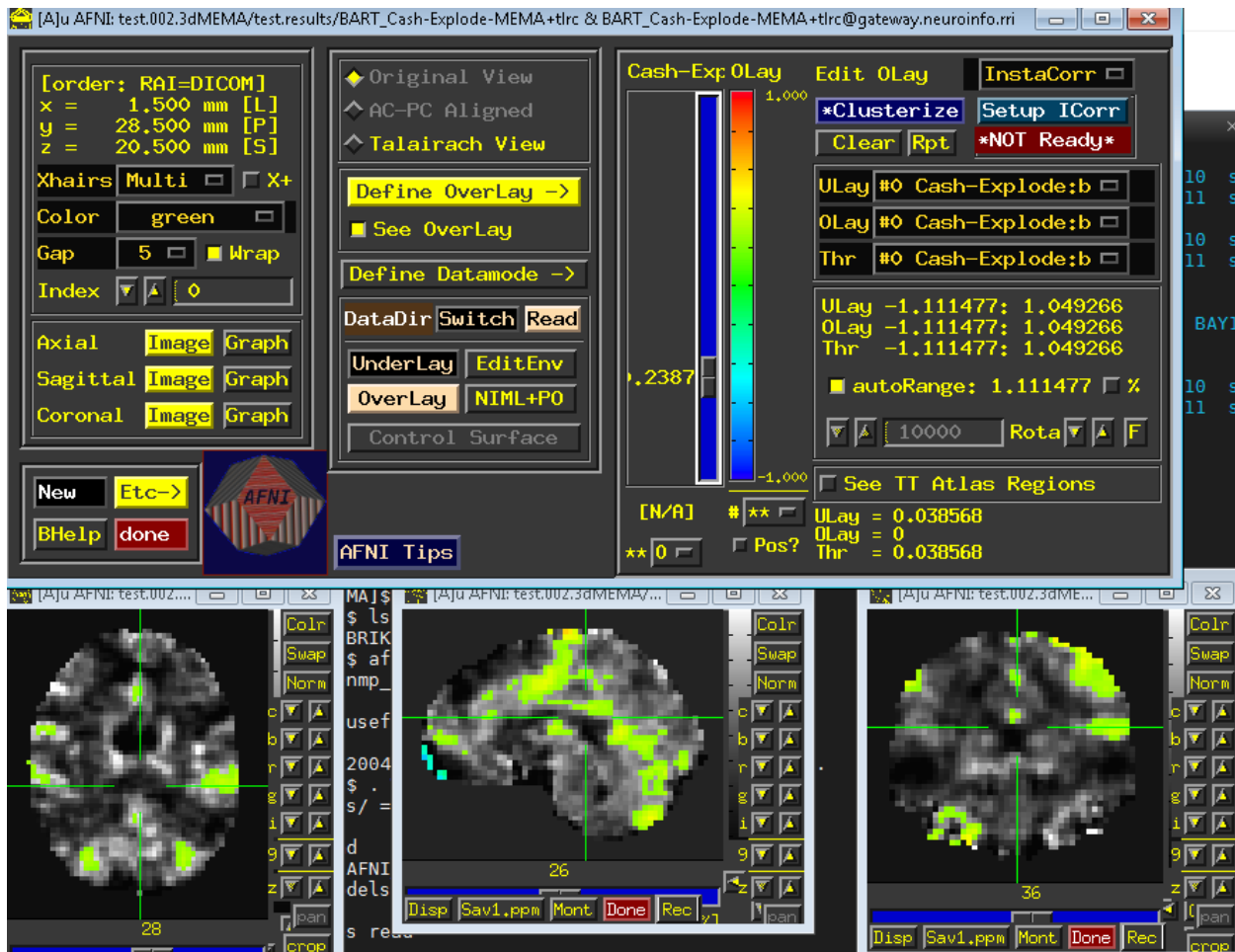
dataset count 16

set name (group or class) Cash-Explode

data index/label 7

t-stat index/label (MEMA) 8

When it has finished, you will find a new directory in the group_results folder called test.002.3dMEMA,. “cd” to the sub-directory, use AFNI viewer and overlay the results as before. you will see a similar image like this:



1.19.9 ROI analysis

As we just completed a group-level analysis, and identified some regions of the brain show a significant difference under the condition of the experiment. Now, we are going to continue our learning with **region of analysis(ROI)**. This is called a **whole-brain** or **exploratory analysis**. When we doesn't have a hypothesis to test, these types of studies are beneficial.

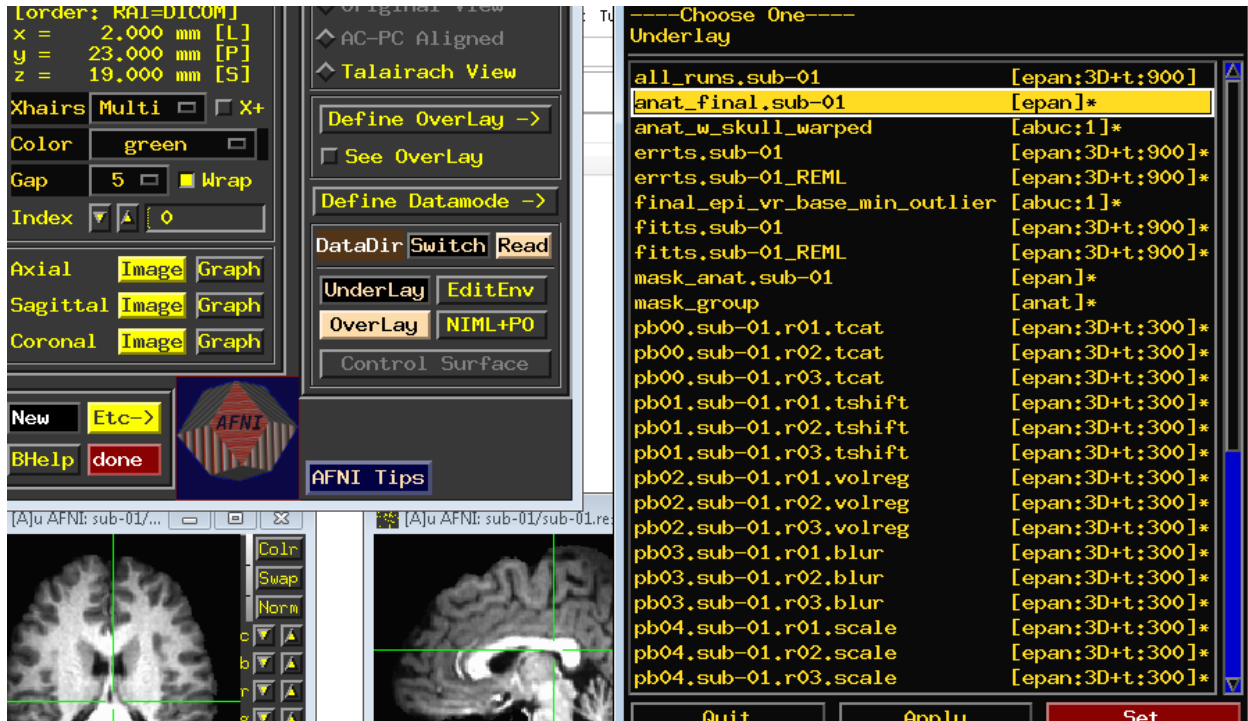
While a large number of studies have been run about a specific topic, we can begin to make more specific hypotheses about where we should find our results in the brain images. For instance, memory has been studied for many years, and many fMRI studies have been published about it using different paradigms that compare different memory tasks. In general, significant increases in the BOLD signal during various memories conditions are seen in a region of the brain known as the Hippocampus and medial temporal lobe. For this BART study, then, we could restrict our analysis to this region and only extract data from voxels within that region. This is known as a ROI analysis. In short, a general name for an analysis in which we choose to analyze a region selected before look at whole-brain results is called a confirmatory analysis.

In terms of BART study, Whole-brain maps can hide important details about the effects that we're studying. As you may find a significant effect of BART conditions, but the reason the effect is significant could come from a greater change of cash than explode, or because explode is much more negative than cash, or some combination of the two. The only way to determine what is driving the effect is with ROI analysis, and this is especially important when dealing with interactions and more sophisticated designs.

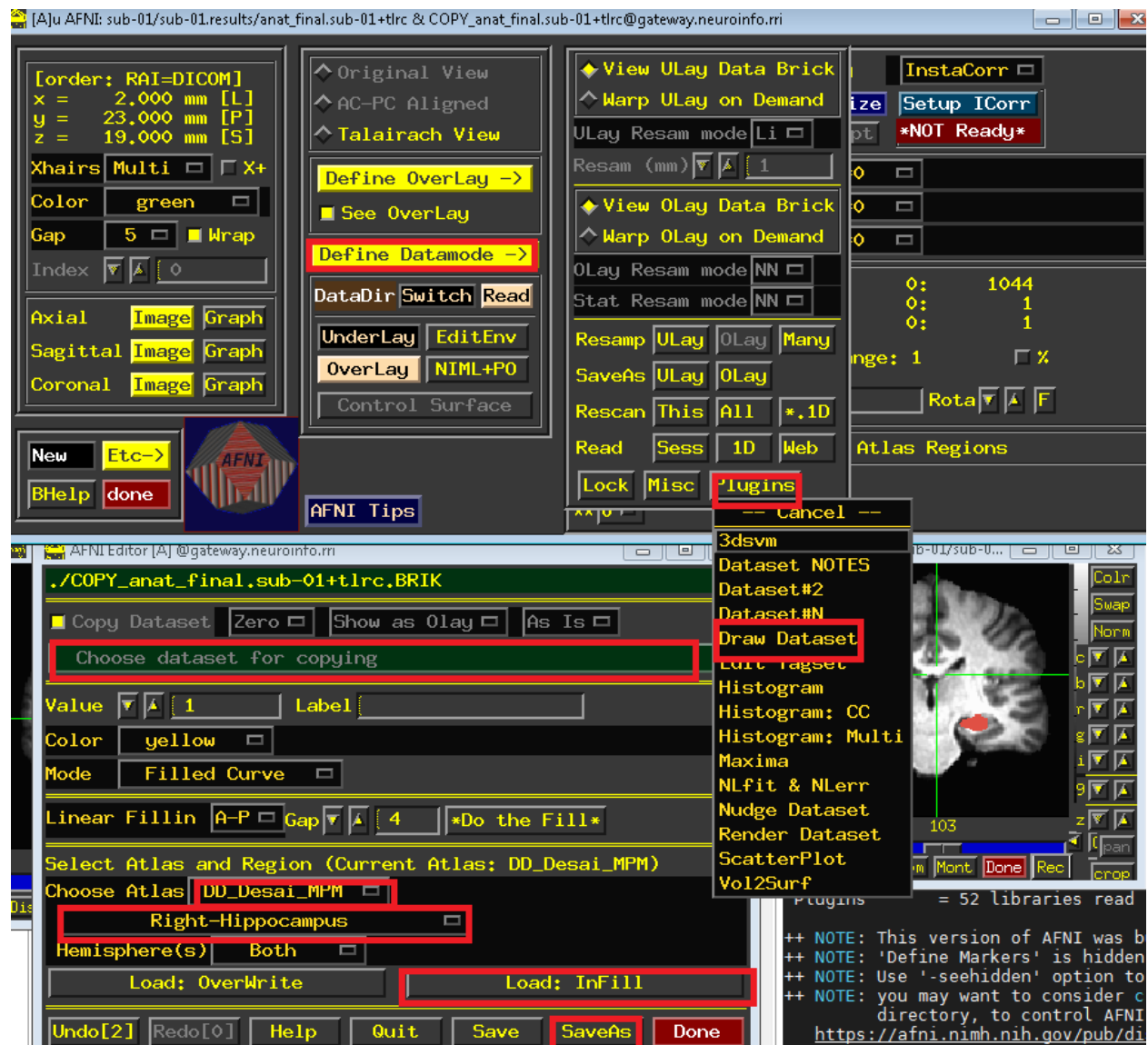
Atlas

One way to do an ROI analysis is to use an atlas, a map that partitions the brain into anatomically distinct regions.

AFNI has a number of atlases in both Talairach and MNI space that may be accessed via the AFNI GUI. It may be difficult for the beginner to locate the atlases; first, Open the AFNI_GUI from the sub-01/sub-01.results directory and choose the anat_final.sub-01 as underlay.



Then, you must first click on Define Datamode, then Plugins, and then Draw Dataset from the dropdown menu. the procedure is depicted in the picture below.



Like painter need to drawing on a whiteboard. The The goal of copying that dataset is to generate a “clean” dataset with the same dimensions as the other images, but one that we can write on by designating which voxels correspond to our ROI. In this case, continue the Draw Dataset from the directory sub-01/sub-01.results, click the button Choose dataset for copying, select the file sub-01/sub-01.results/anat_final.sub-01+tlrc.

Next, select the atlas DD Desai MPM from the AFNI_GUI, then click the dropdown option underneath it. You can choose from a variety of regions, and the voxel region indicated by each label can usually be deduced from the name.

After selecting left hippocampus and right hippocampus, click the Load: InFill button. All of the voxels in that atlas region will be highlighted in red, and then SaveAs. The hippo_mask output is what you should call it. This will generate a new file with values of “1” in the region’s voxels and zeros everywhere else; this is referred to as a mask. Click Done once you’ve completed.

All the step:

Define Datamode Plugins Draw Dataset Choose dataset for copying Choose Atlas SaveAs

The mask dataset in AFNI has a different resolution than the normalised anatomical image and the template used for normalisation by default. AFNI template was the MNI avg152T1+tlrc file, which has a resolution of 2x2x2mm, whereas our statistics dataset has a resolution of 3x3x3mm. To use a mask for a ROI analysis, it must have the same

resolution as the dataset from which it is being extracted.

We can match the resolutions of our mask dataset and our statistics dataset by using AFNI's `3dresample` command. This command requires both a “master” dataset, which we will be resampling to, and an “input” dataset, which will have its dimensions and resolution changed to match the master dataset:

```
3dresample -master stats.sub-01+tlrc -input hippo_mask+tlrc -prefix hippo_mask_rs+tlrc
```

This will create a new file, `hippo_mask_rs` (rs stands for re-sampled). Move this mask to our data subject directory by typing `mv hippo_mask_rs+tlrc* .././`. We can then use it to extract data for our ROI analysis.

Extracting the data from mask

After we generated the mask, we can use it to extract the contrast estimates for each subject. We can extract our contrast of interest of cash-explode in one ways: Extract the individual beta weights for cash and explode separately, and then take the difference between the two.

This method allows us to figure out what is causing the effect; for example, a significant effect might be caused by both beta weights being positive but the cash betas being more positive, or both weights being negative but the explode betas being more negative, or a mix of the two. This can only be determined by extracting both sets of beta weights.

From the subjects directory type:

```
3dinfo -verb sub-01/sub-01.results/stats.sub-01+tlrc.
```

```
[wshao@gateway BART_afni]$ 3dinfo -verb sub-01/sub-01.results/stats.sub-01+tlrc.
++ 3dinfo: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]

Dataset File:      stats.sub-01+tlrc
Identifier Code:   XYZ_GE8S07THhdub3sumLvXxrA  Creation Date: Thu Apr  8 20:15:59 2021
Template Space:    TT_N27
Dataset Type:      Func-Bucket (-fbuc)
Byte Order:        LSB_FIRST [this CPU native = LSB_FIRST]
Storage Mode:      BRIK
Storage Space:     11,059,200 (11 million [mega]) bytes
Geometry String:   "MATRIX(3,0,0,-79.5,0,3,0,-79.5,0,0,3,-63.5):54,64,50"
Data Axes Tilt:    Plumb
Data Axes Orientation:
  first (x) = Right-to-Left
  second (y) = Anterior-to-Posterior
  third (z) = Inferior-to-Superior [-orient RAI]
R-to-L extent:    -79.500 [R] -to- 79.500 [L] -step- 3.000 mm [ 54 voxels]
A-to-P extent:    -79.500 [A] -to- 109.500 [P] -step- 3.000 mm [ 64 voxels]
I-to-S extent:    -63.500 [I] -to- 83.500 [S] -step- 3.000 mm [ 50 voxels]
Number of values stored at each pixel = 16
-- At sub-brick #0 'Full_Fstat' datum type is float:      0 to      83.5981
  statcode = fitt; statpar = 2 874
-- At sub-brick #1 'cash#0_Coef' datum type is float:    -16.957 to    16.4743
  statcode = fitt; statpar = 874
-- At sub-brick #2 'cash#0_Tstat' datum type is float:   -9.60273 to    11.1894
  statcode = fitt; statpar = 874
-- At sub-brick #3 'cash_Fstat' datum type is float:      0 to    125.201
  statcode = fitt; statpar = 1 874
-- At sub-brick #4 'explode#0_Coef' datum type is float:  -17.29 to    16.3795
  statcode = fitt; statpar = 874
-- At sub-brick #5 'explode#0_Tstat' datum type is float: -8.57224 to    9.66493
  statcode = fitt; statpar = 874
-- At sub-brick #6 'explode_Fstat' datum type is float:      0 to    82.173
  statcode = fitt; statpar = 1 874
-- At sub-brick #7 'C-E_GLT#0_Coef' datum type is float: -18.1939 to    17.2486
  statcode = fitt; statpar = 874
-- At sub-brick #8 'C-E_GLT#0_Tstat' datum type is float: -5.81146 to    7.79921
  statcode = fitt; statpar = 874
-- At sub-brick #9 'C-E_GLT_Fstat' datum type is float:      0 to    60.8276
  statcode = fitt; statpar = 1 874
-- At sub-brick #10 'E-C_GLT#0_Coef' datum type is float: -17.2486 to    18.1939
  statcode = fitt; statpar = 874
-- At sub-brick #11 'E-C_GLT#0_Tstat' datum type is float: -7.79921 to    5.81146
  statcode = fitt; statpar = 874
-- At sub-brick #12 'E-C_GLT_Fstat' datum type is float:      0 to    60.8276
  statcode = fitt; statpar = 1 874
-- At sub-brick #13 'mean.CE_GLT#0_Coef' datum type is float: -16.6019 to    9.70739
  statcode = fitt; statpar = 874
-- At sub-brick #14 'mean.CE_GLT#0_Tstat' datum type is float: -9.50902 to   12.8593
  statcode = fitt; statpar = 874
-- At sub-brick #15 'mean.CE_GLT_Fstat' datum type is float:      0 to   165.361
  statcode = fitt; statpar = 1 874
```

The sub-briks indicate which volume in the dataset has which beta weight. In this example, the cash condition's beta weight is sub-brik 1, the explode condition's beta weight is sub-brik 4, and the contrast weight for cash-explode is sub-brik 7. Sub-briks 1 and 4 will be extracted and saved in separate files, and the values for each subject will be extracted from a ROI.

To extract the individual sub-briks:

```
#!/bin/bash

for subj in `cat subjList.txt`; do

    3dbucket -aglueto Cash_betas+tlrc.HEAD ${subj}/${subj}.results/stats.${subj}+tlrc
    ↪ '[1]'
    3dbucket -aglueto Explode_betas+tlrc.HEAD ${subj}/${subj}.results/stats.${subj}
    ↪ +tlrc'[4]'

done
```

When it finishes, you will have generated two new datasets: Cash_betas and Explode_betas. You can now extract data from the anatomical mask by using the 3dmaskave command:

```
3dmaskave -quiet -mask hippo_mask_rs+tlrc Cash_betas+tlrc
```

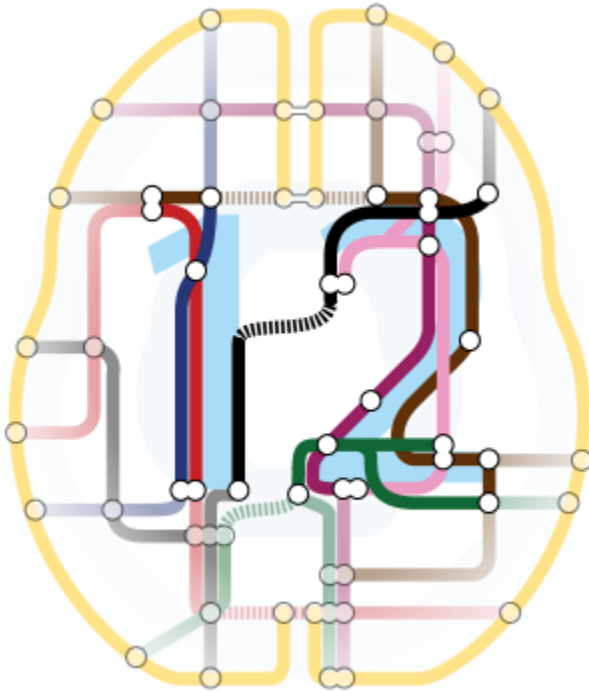
And the same command for the explode betas as well:

```
3dmaskave -quiet -mask hippo_mask_rs+tlrc Explode_betas+tlrc
```

```
[wshao@gateway BART_afni]$ 3dmaskave -quiet -mask hippo_mask_rs+tlrc. Cash_betas+tlrc.
++ 3dmaskave: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
+++ 350 voxels survive the mask
0.00828946
0.0156234
0.22995
0.14858
0.0963133
-0.0114821
-0.00169905
-0.0601025
0.26196
0.0663501
-0.0111806
0.211177
0.000166241
-0.0519291
0.0272612
0.112369
[wshao@gateway BART_afni]$ 3dmaskave -quiet -mask hippo_mask_rs+tlrc. Explode_betas+tlrc.
++ 3dmaskave: AFNI version=AFNI_18.0.22 (Feb 26 2018) [64-bit]
+++ 350 voxels survive the mask
-0.179978
0.0673552
-0.0426225
-0.0872705
-0.000499946
0.0651911
-0.159568
-0.159074
0.0153901
0.0235492
-0.160995
-0.0338155
-0.100837
-0.0198984
-0.719971
0.236533
```

This command returns a number that corresponds to the contrast estimate used in the analysis. The first value, for example, corresponds to the average contrast estimate for cash-explode for sub-01, the second number, for sub-02, and so on. After that, you may use statistics software like R to do a t-test on them.

1.20 SPM

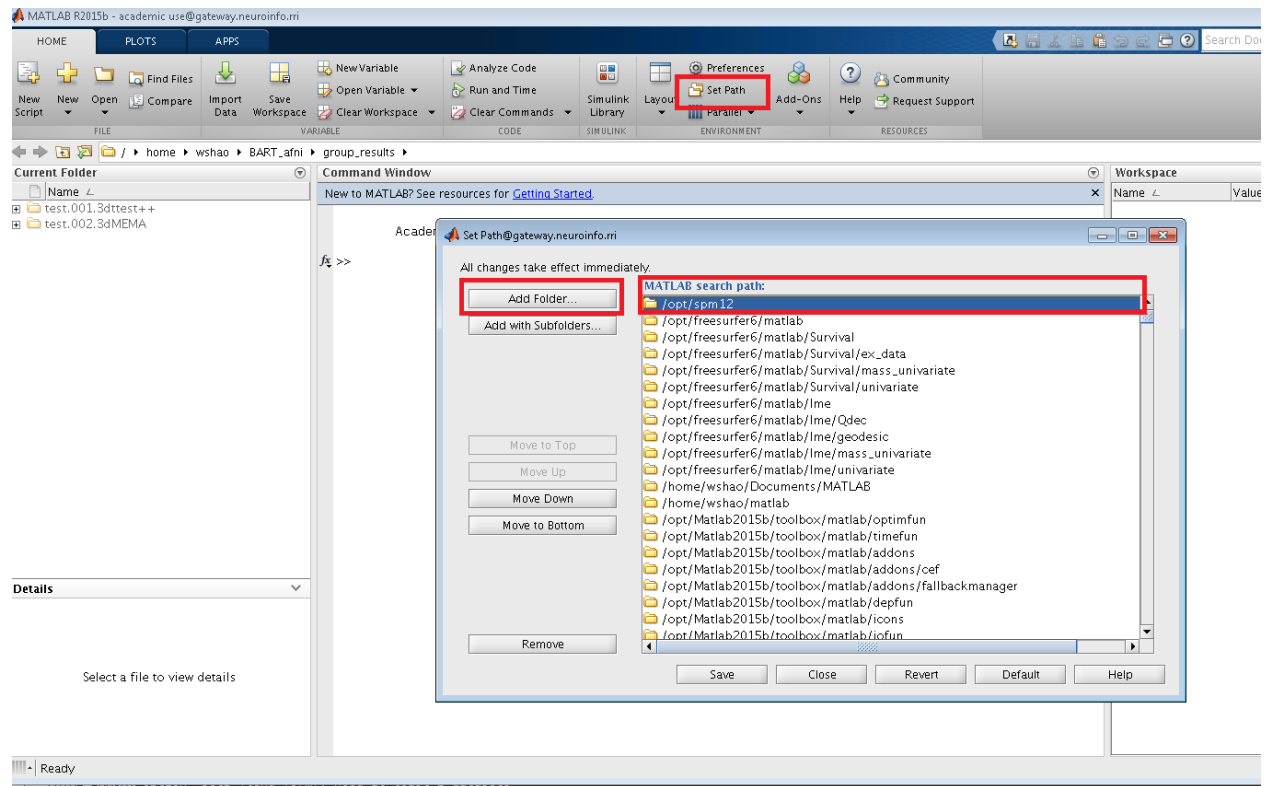


What is SPM?

SPM (Statistical Parametric Mapping) is an interactive application that performs statistical analyses of brain imaging data. SPM refers to the construction and assessment of spatially extended statistical processes to test functional imaging data. The data could be a series of images from different cohorts or time-series from the same subject. The current release version of SPM, SPM12, is designed for the analysis of fMRI, PET, SPECT, EEG and MEG. SPM12 has had several updates since 2014. First released 1st October 2014 and last updated 13th January 2020

1.20.1 Downloading and Installing SPM

Unlike AFNI or FSL, SPM is able to run on any operating system as long as it has Matlab installed. Matlab is proprietary software that is quite expensive, but if you are a student or university employee you may be able to obtain a copy for free. Once you have installed Matlab, the [SPM website](#) has instructions on how to install the software package. Click on the “download form” button to fill out some personal details such as your position and what you will be using the software for, which will enable you to download the software.

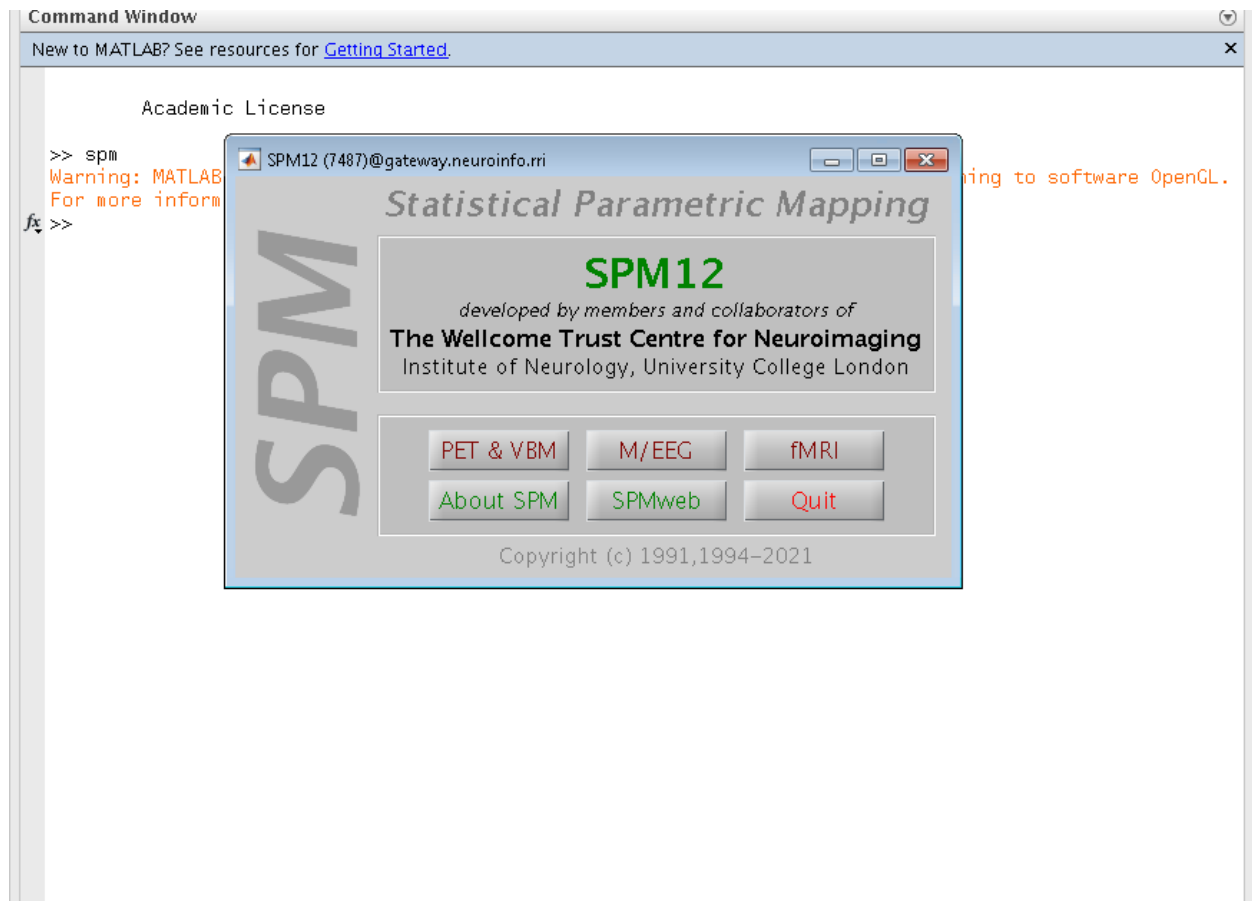


Once you have downloaded the SPM package, place it your home directory. Open up Matlab, click on the “Home” tab, and then click the “Set Path” button. Select the spm12 directory, and then click “Add with Subfolders”. Click the “Save” button to ensure that the path is set every time Matlab is opened, and then close the window.

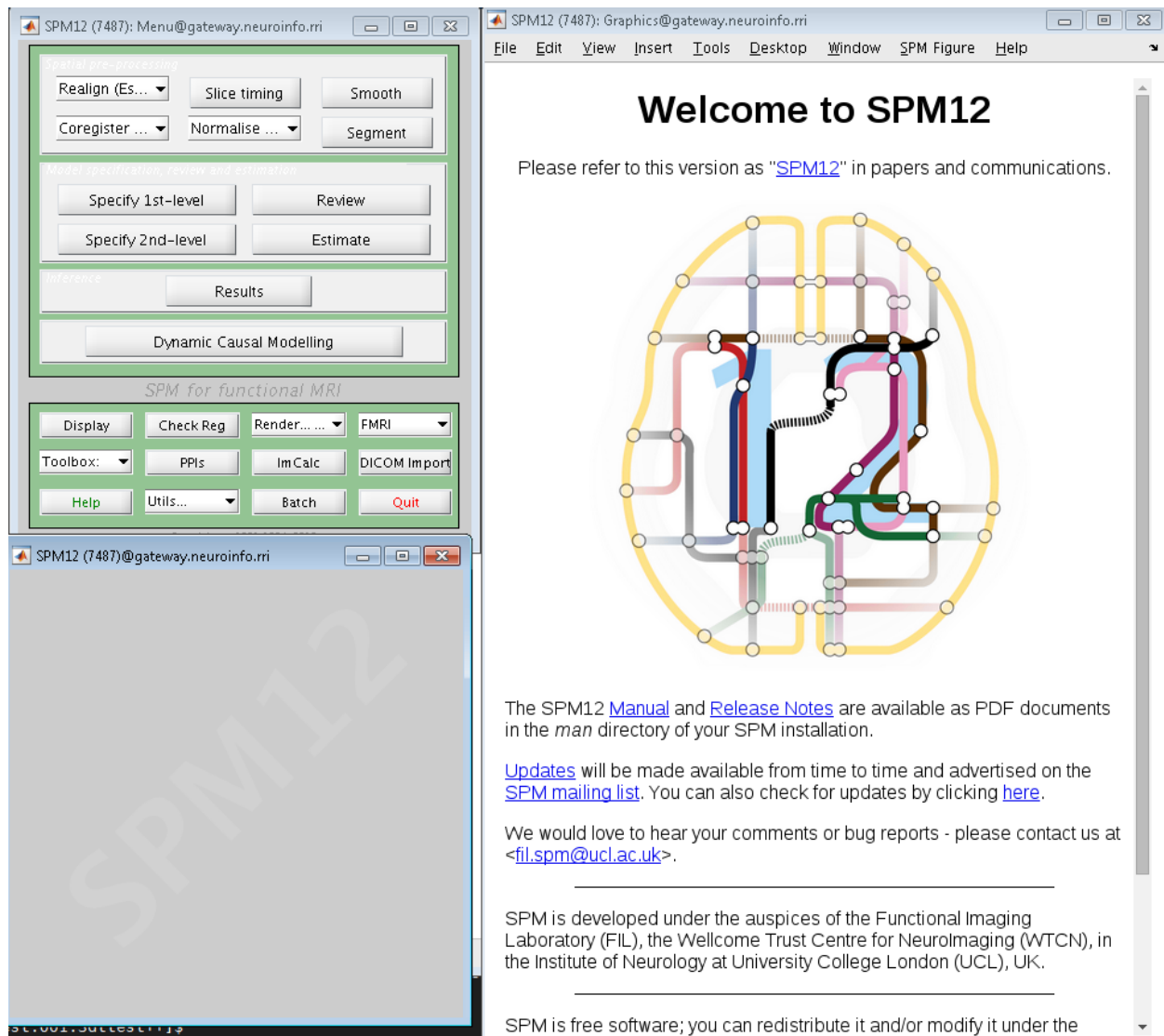
After you have set the path, type the following from the Matlab terminal:

```
spm
```

Wait for a few seconds, you will see a SPM GUI



Click the fMRI, navigate to the fMRI_GUI



The goal of this chapter is to show you how to run fMRI analysis with SPM step by step. After successfully install SPM and set up, we are going to analyze a dataset with SPM so that we can gain some first-hand experiences. The data is Balloon Analogue Risk Task. I hope after this chapter, you will be able to apply the knowledge of AFNI in the future and practice with other datasets you have.

We will start with the dataset downloading, preprocessing, 1st level, and end with group analysis as follow:

BART and data downloading

One of the interesting topics from psychology is risky behavior. In order to gain a better understanding of how the brain reacts when people make a decision, there are many many interesting experiment paradigms, Balloon analog risk task is one of them.

In Balloon analog risk task(BART), Participants can manipulate the Balloon to control the risk and reward. Tom Schonberg and his colleagues scanned participants using fMRI while they completed the Balloon Analog Risk Task. As the picture shows, escalating risk-taking occurs under uncertainty and might be experienced either as the accumulation of greater potential rewards, or as exposure to increasing possible losses and decreasing expected value.

In the BART, subjects inflate simulated balloons, and accrue monetary rewards for each successive “pump” during a

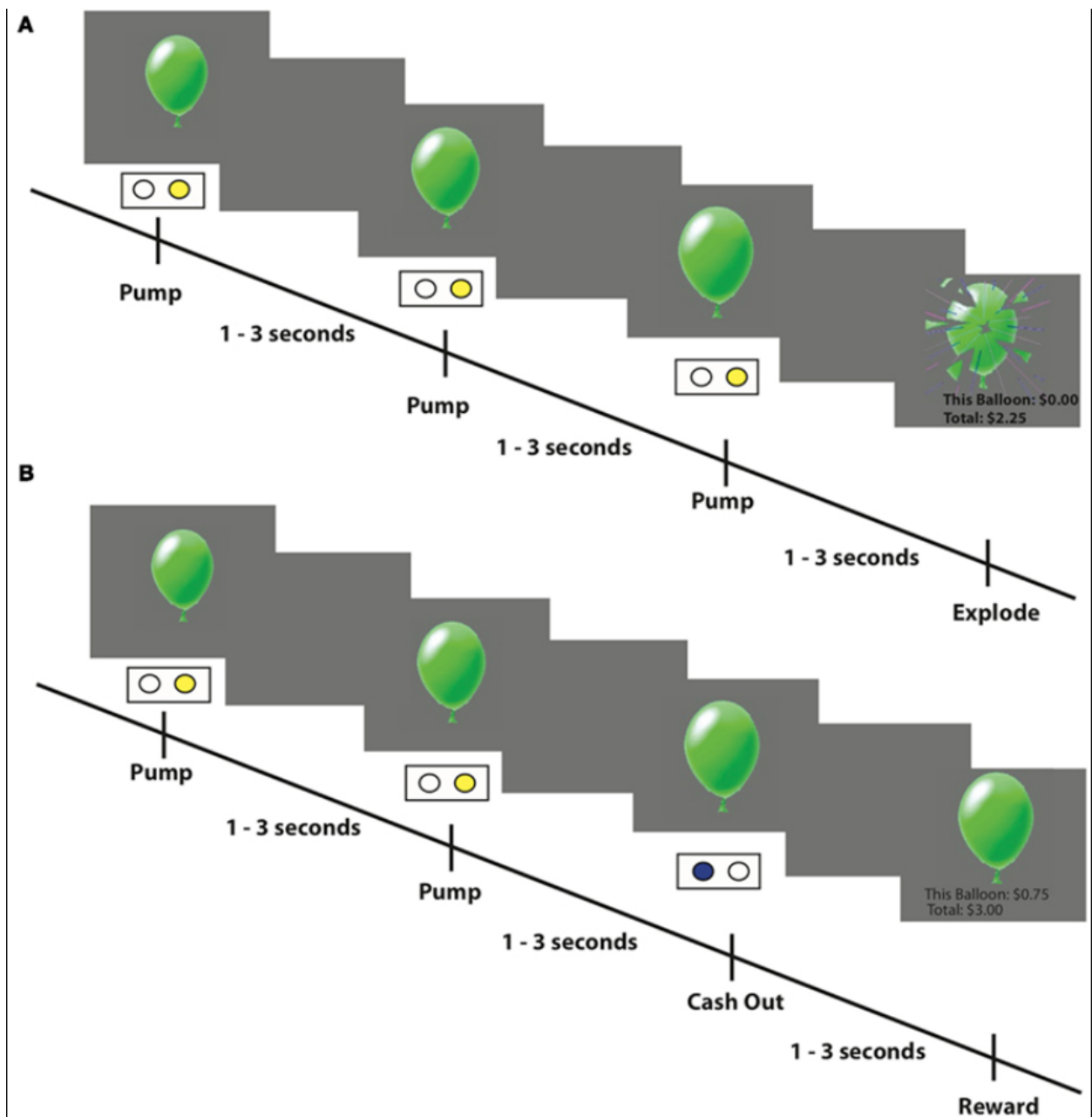


Fig. 20: (A) An example of an explosion trial: participants press one of two buttons to inflate puffs of air into a balloon presented on a computer screen. Every successful pump adds \$0.25 to their temporary bank for that trial. If the balloon explodes before the participant cashes out then nothing is won on that trial. However, an explosion does not affect the cumulative total winnings earned on prior trials. (B) An example of a cash-out trial where the participant decided to stop pumping the balloon and earn the amount accumulated up to that point.

particular trial. A trial is defined as a balloon that can be pumped a certain number of times and the trial can conclude in two different ways. On the one hand, the participant can “cash-out” at any point during the trial and secure the cumulative winnings up to that point for that balloon in their cumulative total “bank.” On the other hand, a balloon may explode. In this case, participants would lose the money accumulated on that trial alone (but not the total accumulated during previous cash-out trials). Each trial began with a balloon displaying a value of \$0.25 and the value of the balloon increased by \$0.25 for each successive pump. During each trial, participants were presented with one of three types of “reward” balloons, each having a different explosion probability and signified by a different color: red, green, or blue. But participants was not informed the odds of explosion probability.

As a control task, participants intermittently inflated a gray “control” balloon (maximum 12 pumps) that did not explode and had no associated monetary value. The participants were instructed to inflate the control balloon until it disappeared from the screen, and the next trial began. Unlike with reward balloons, participants had no control over how many times they could inflate the control balloon before the trial ended.

For more information, please go [here](#)

Downloading the data

As an Open Source dataset, BART dataset has a standardized structure: Each subject folder contains an anatomical directory and a functional directory labeled anat and func, these contain the anatomical and functional images collected during the experiment. The func directory also contains onset times, or timestamps for when the subject underwent different trials. This format is known as Brain Imaging Data Structure, as we saw in previous chapter [BIDS](#). As an example of the BIDS format. The func directory of BART contains functional data, three runs of functional data - and corresponding “events.tsv” files, which contain onsets, or timestamps of which condition happened at what time. You can open these as a text file or as a spreadsheet.

Now, go to [there](#) , download the data and save it as BART_spm in our home directory.

```
[wshao@gateway BART_spm]$ ls
CHANGES      participants.tsv  spm_2021Apr08.ps  sub-02  sub-04  sub-06  sub-08  sub-10  sub-12  sub-14  sub-16
dataset_description.json  README          sub-01          sub-03  sub-05  sub-07  sub-09  sub-11  sub-13  sub-15  task-balloonanalogrisktask_bold.json
```

As you can see, we have 16 subjects. **participants.tsv** will tell you the demographic information of subjects, **task-balloonanalogrisktask_bold.json** contains TR information. You can preview all of these information from the Open-Neuro data web.

Let's take a close look

```
anat func
```

cd to sub-01, you will find two directories, anat and func that correspond anatomical and functional. These are two important directories that we will visit most.

```
sub-01_inplaneT2.nii.gz  sub-01_T1w.nii.gz
```

anat includes all the anatomical images such as T1 and T2 (if possible).

```
sub-01_task-balloonanalogrisktask_run-01_bold.nii.gz  sub-01_task-balloonanalogrisktask_run-02_events.tsv
sub-01_task-balloonanalogrisktask_run-01_events.tsv  sub-01_task-balloonanalogrisktask_run-03_bold.nii.gz
sub-01_task-balloonanalogrisktask_run-02_bold.nii.gz  sub-01_task-balloonanalogrisktask_run-03_events.tsv
```

func has all the functional images, end with **bold.nii.gz** as well as trial-related files end with **events.tsv**.

Whenever you're are ready, let's go!

Preprocessing

Remember our drink menu? It is a little different for what we should do in SPM but overall there are very similar, Anyway, let's have a couple of drinks before the meal

Since we have download and rename the dataset, let's look at our data first to check if there are any artifacts or problems with the image data. call the `Matlab` and navigate to **BART_spm** directory.

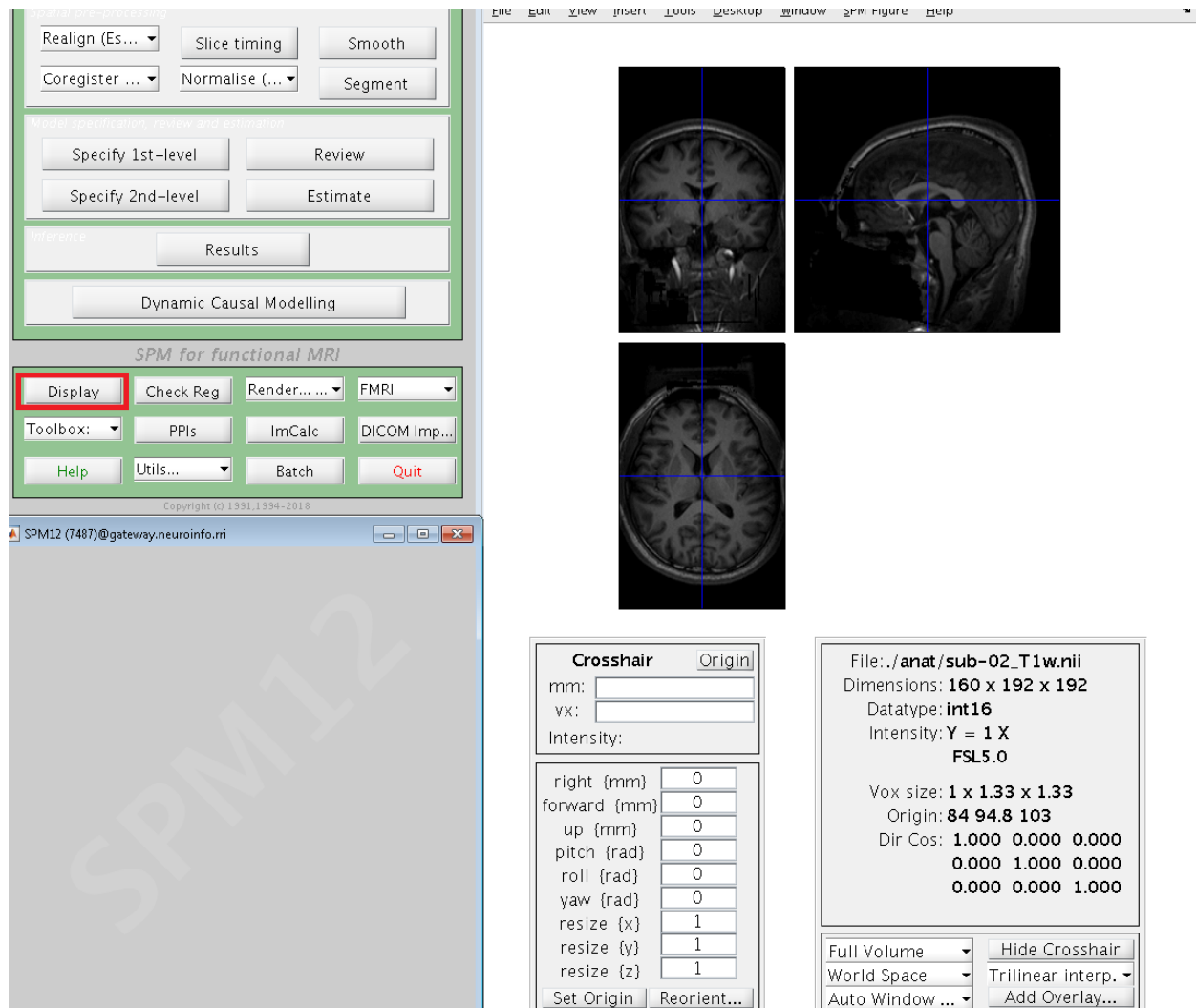
Viewing the Anatomical images

Whenever you download imaging data, check the anatomical and functional images for any artifacts - scanner spikes, incorrect orientation, poor contrast, and so on. It will take some time to develop an eye for what these problems look like, but with practice it will become quicker and easier to do.

To begin, let's take a look at the anatomical image in the anat folder for sub-02. If you haven't already opened SPM, navigate to the sub-02 folder and then type:

```
spm_fmri
```

and press return, which will open the SPM graphical user interface. If you click on the `Display` button, you will be prompted to select an image.



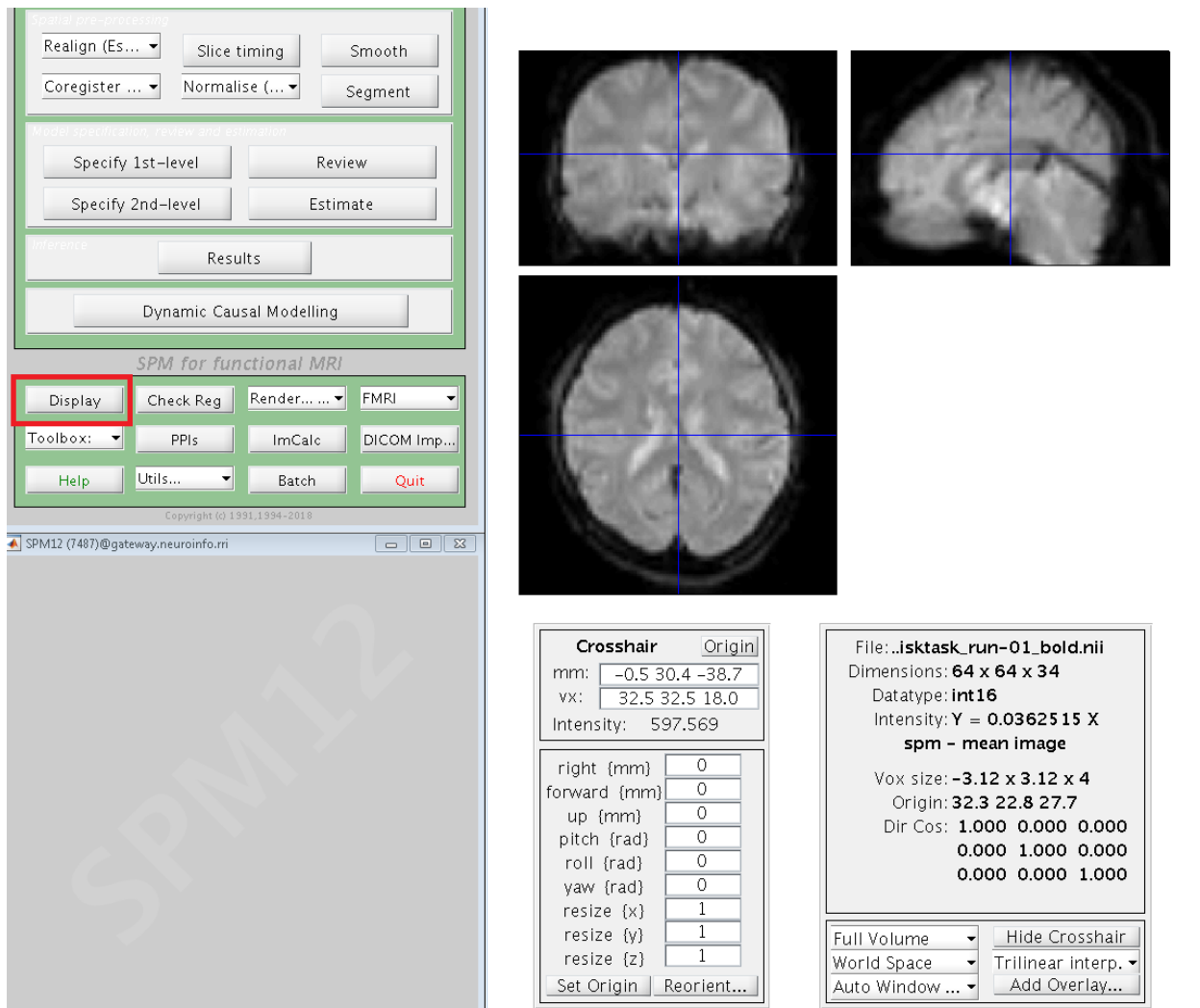
Note: SPM can read any image that are in NIFTI format, but they cannot be compressed - that is, if the datasets end with a .gz extension, you will first need to unzip them by navigating to the directory

containing the images and then type `gunzip *.gz`

Viewing the Functional images

When you are done looking at the anatomical image, click on the Display button again, navigate to the func directory, and select the **run-01** functional image.

A new image will be displayed in the orthogonal viewing windows. This image also looks like a brain, but it is not as clearly defined as the anatomical image. This is because the resolution is lower. It is typical for a study to collect a high-resolution T1-weighted (i.e., anatomical) image and lower-resolution functional images, which are lower resolution in part because they are collected at a very fast rate. One of the trade-offs in imaging research is between spatial resolution and temporal resolution: Images collected at higher temporal resolution will have lower spatial resolution, and vice versa.

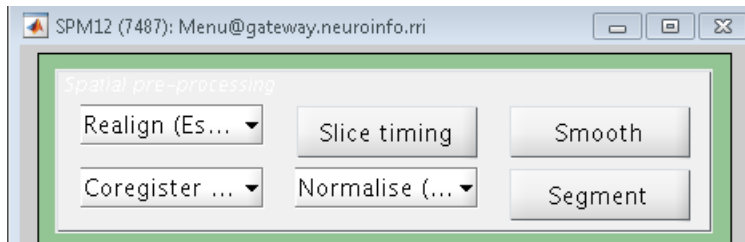


During the Realignment preprocessing step you will generate a movement parameter file showing how much motion there was between each volume.

Realignment and Slice-Timing Correction

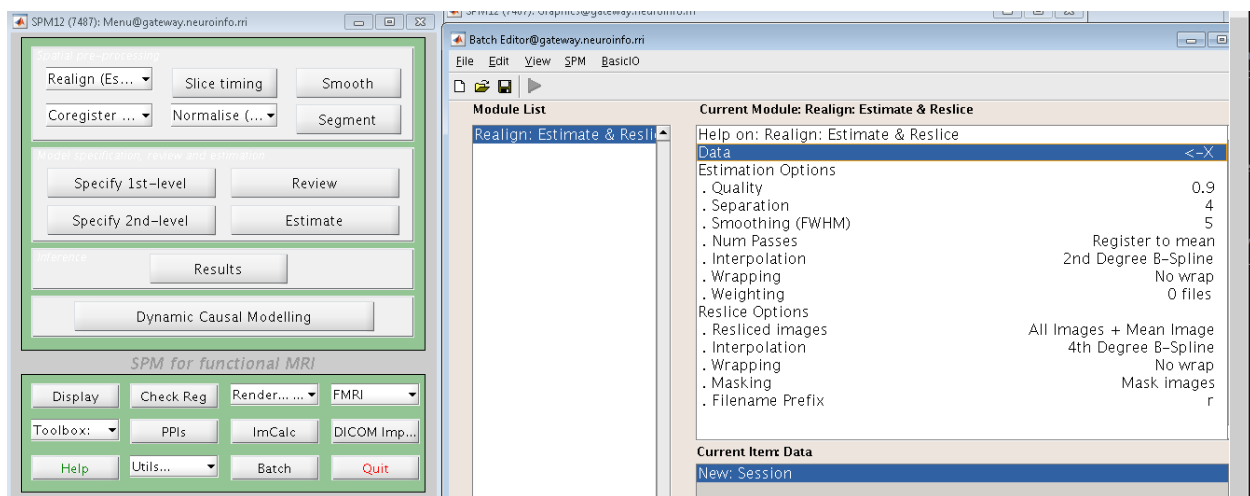
Similarly, when we preprocess fMRI data we are cleaning up the three-dimensional images that we acquire every TR. An fMRI volume contains not only the signal that we are interested in - changes in oxygenated blood - but also fluctuations that we are not interested in, such as head motion, random drifts, breathing, and heartbeats. We call these other fluctuations noise, since we want to separate them from the signal that we are interested in. Some of these can be regressed out of the data by modeling them (which is discussed in the chapter on modeling fitting), and others can be reduced or removed by preprocessing.

To begin preprocessing sub-02 data, read through the following chapters. We will begin with Realignment and Slice-Timing Correction, which correct misalignments and timing errors in the functional images, before moving on to Coregistration and Normalization, which align the functional and structural images and move them both to a standardized space. Finally, the images are Smoothed in order to increase signal and cancel out noise. The typical sequence of preprocessing steps is numbered in the image below:



The first step of preprocessing is to realign the functional images. If you think of a time-series as a deck of cards, with each volume as a separate card, realignment will put all the cards in the same orientation and make the sides line up - similar to what you do after you shuffle a deck of cards.

If you click on the button Realign (Estimate & Reslice), a window opens up showing the options for realigning and reslicing the data. The Estimate part refers to estimating the amount that each volume is out of alignment with a reference volume, and Reslice indicates that these estimates will be used to nudge each of the volumes into alignment with the reference volume. The reference volume is set in the field “Num Passes”, which allows you to specify whether the volumes will be aligned to the mean of all of the volumes, or to the first volume. For this tutorial, leave it as the default, and leave the rest of the defaults alone, as well.



Loading the Images

In this experiment, there were three runs of data in per subject (each run of SPM refers as a session). Look at the Data tab, click on **New: Session** to add three sessions. An <-X appeared to the right of each Session field. Click Specify at the button to open up the Image Selection window. Navigate to the func directory of sub-02 and select the file “sub-02_task-balloonanalogrisktask_run-01_bold.nii,1”, 1 in here indicates that only the first frame, or volume, is available. However, In order to select all of the volumes for this run, type 1:300 and press enter in the “Frames field” (underneath the Filter field) to expand the 300 number of frames available for selection.

Note: If you don't know how many frames are in the current dataset, you can use `3dinfo` and `mri_info` command to check the frrams if you have installed AFNI or freesurfer. or you can set the upper

bound to an high number like 1:10000. The list of files will max out at the number of available frames, and so will ensure that you do not miss any.

```

Dataset File:      sub-02_task-balloonanalogrisktask_run-01_bold.nii
Identifier Code:   NII_zgZHvhcw_SEUOU-K6Y0TRg  Creation Date: Wed Apr 14 17:15:39 2021
Template Space:    TLRC
Dataset Type:      Echo Planar (-epan)
Byte Order:        LSB_FIRST {assumed} [this CPU native = LSB_FIRST]
Storage Mode:      NIFTI
Storage Space:     83,558,400 (84 million [mega]) bytes
Geometry String:   "MATRIX(3.125,0,0,-97.93152,0,-3.125,0,68.01921,0,0,4,-106.6506):64,64,34"
Data Axes Tilt:    Plumb
Data Axes Orientation:
  first (x) = Right-to-Left
  second (y) = Posterior-to-Anterior
  third (z) = Inferior-to-Superior  [-orient RPI]
R-to-L extent:    -97.932 [R] -to- 98.943 [L] -step- 3.125 mm [ 64 voxels]
A-to-P extent:    -128.856 [A] -to- 68.019 [P] -step- 3.125 mm [ 64 voxels]
I-to-S extent:    -106.651 [I] -to- 25.349 [S] -step- 4.000 mm [ 34 voxels]
Number of time steps = 300  Time step = 2.00000s  Origin = 0.00000s
-- At sub-brick #0 '?' datum type is short
-- At sub-brick #1 '?' datum type is short
-- At sub-brick #2 '?' datum type is short
** For info on all 300 sub-bricks, use '3dinfo -verb' **

```

```

[wshao@gateway func]$ mri_info sub-02_task-balloonanalogrisktask_run-01_bold.nii
Volume information for sub-02_task-balloonanalogrisktask_run-01_bold.nii
  type: nii
  dimensions: 64 x 64 x 34 x 300
  voxel sizes: 3.125000, 3.125000, 4.000000
  type: FLOAT (3)
  fov: 200.000
  dof: 0
  xstart: -100.0, xend: 100.0
  ystart: -100.0, yend: 100.0
  zstart: -68.0, zend: 68.0
  TR: 2000.00 msec, TE: 0.00 msec, TI: 0.00 msec, flip angle: 0.00 degrees
  nframes: 300
  PhEncDir: UNKNOWN
  FieldStrength: 0.000000
ras xform present
  xform info: x_r = -1.0000, y_r = 0.0000, z_r = 0.0000, c_r = -2.0685
               : x_a = 0.0000, y_a = 1.0000, z_a = 0.0000, c_a = 31.9808
               : x_s = 0.0000, y_s = 0.0000, z_s = 1.0000, c_s = -38.6506
Orientation : LAS
Primary Slice Direction: axial

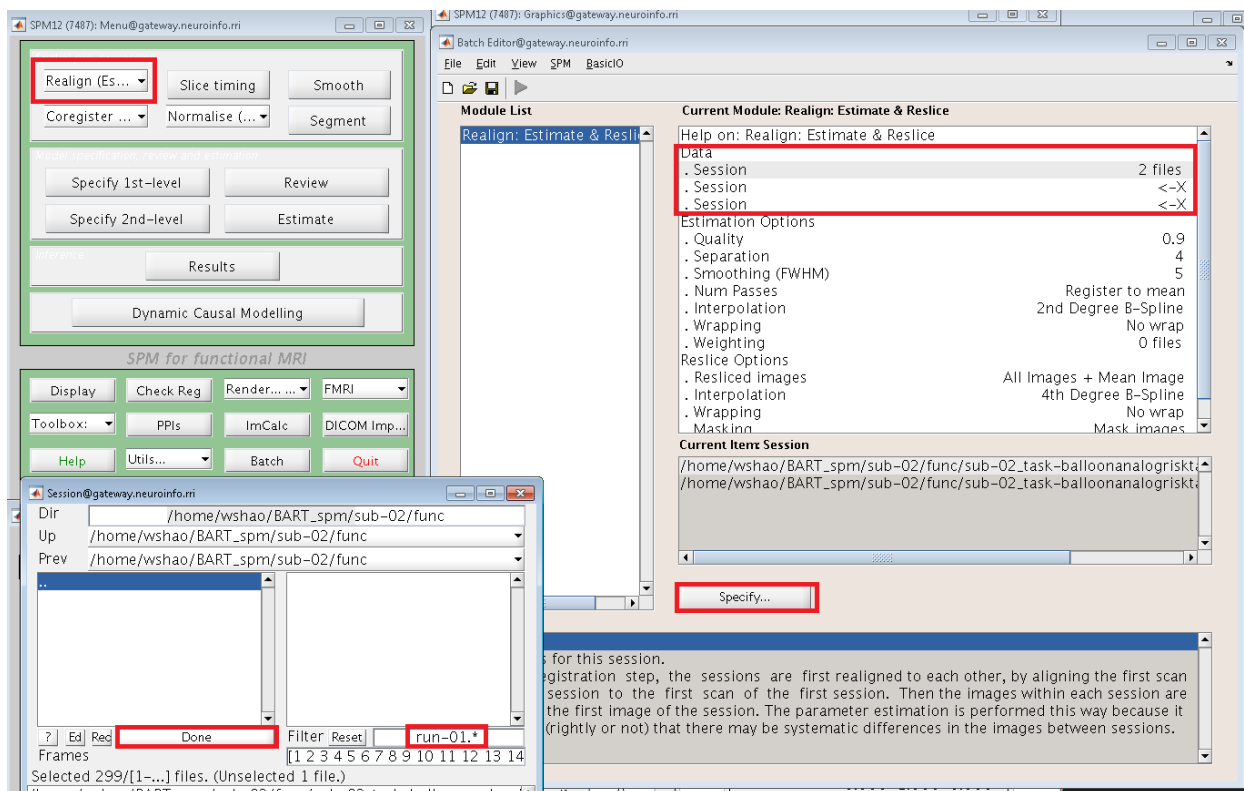
voxel to ras transform:
  -3.1250  0.0000  0.0000  97.9315
  0.0000  3.1250  0.0000 -68.0192
  0.0000  0.0000  4.0000 -106.6506
  0.0000  0.0000  0.0000  1.0000

voxel-to-ras determinant -39.0625

ras to voxel transform:
  -0.3200 -0.0000 -0.0000  31.3381
  -0.0000  0.3200 -0.0000  21.7661
  -0.0000 -0.0000  0.2500  26.6626
  -0.0000 -0.0000 -0.0000  1.0000

```

You might notice that all of the **frames** for run-01, run-02 and run-03 have been selected, even though we only want run-01 frames in this session. To limit our file selection to only the frames we are interested in, we can use the “Filter field”. In this case, to the left of the .* characters that are already in the field, type run-01 and press return. This will refresh the screen to display only those frames which include the string run-01. Either click “rec” or right click select all of the images.



You need to repeat all the steps above and choose 300 frames from the choose **run-02** and **run-03**

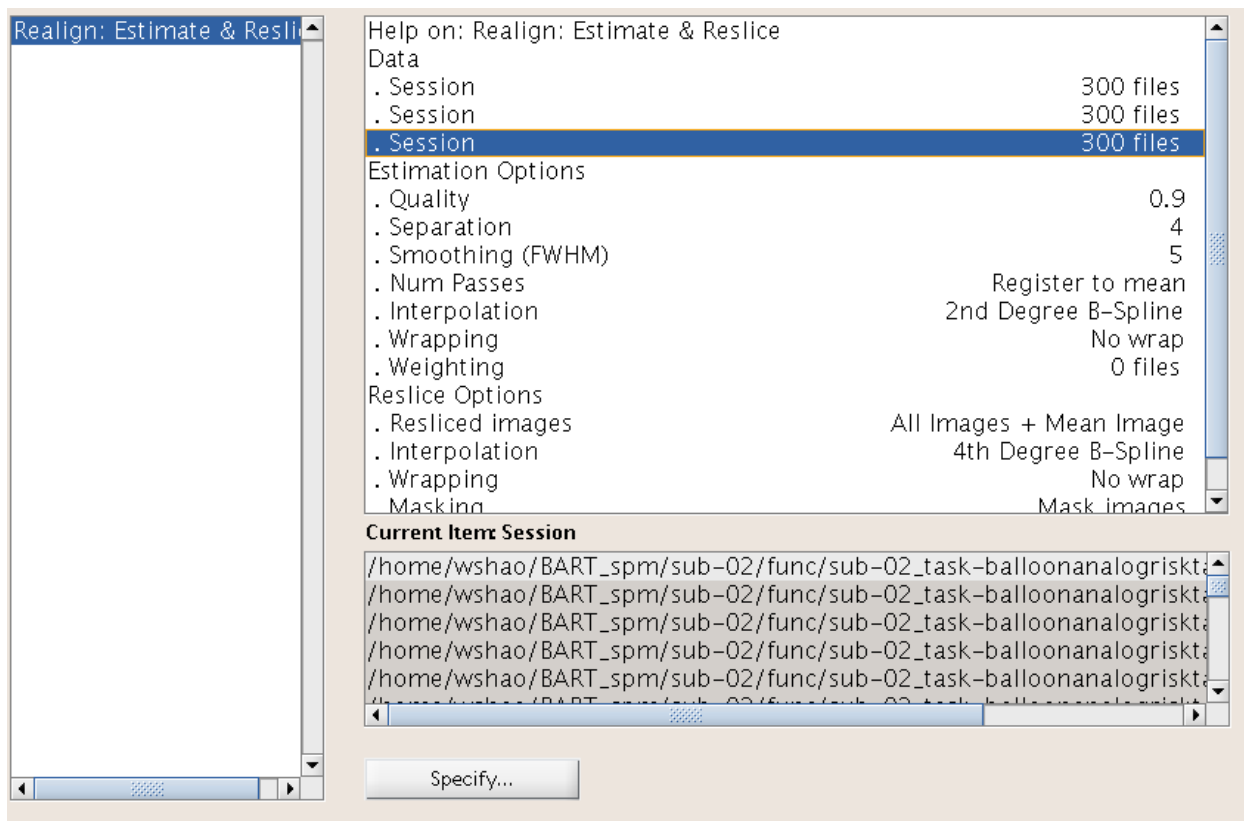
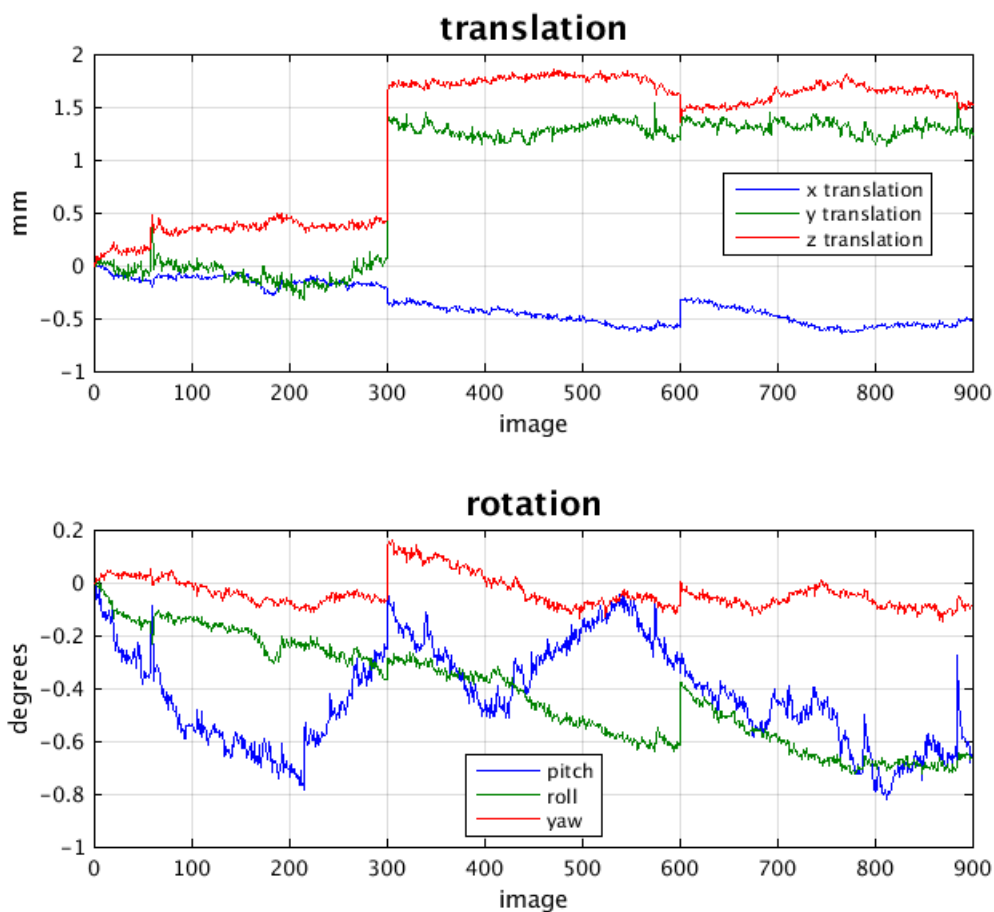


Image realignment

```

1 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
2 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
3 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
4 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
5 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
6 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
7 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
8 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
9 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
10 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
11 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
12 /home/wshao/BART_spm/sub-02/func/sub-02_task-balloonanalogrisktask_run-01_
..... etc

```



Slice-Timing Correction in SPM

Similar to what we did with Realignment, we will first click on the Slice Timing button in the SPM GUI. Click on the Data field and create two new Sessions. Double-click on the first Session, and in the Filter column type `^rsub-02_task-flanker_run-1.*`. In the Frames field, enter 1:300 and press enter; select all of the frames that are displayed, and click Done. Do the same procedure for the run-02 and run-03 files for the second session.

For the Number of Slices field, we will need to find out how many slices there are in each of the volumes in our dataset. From the Matlab terminal navigate to the directory sub-02/func and type:

```
V = spm_vol('sub-02_task-balloonanalogrisktask_run-01_bold.nii')
```

This will load the header of the image into a variable called V. If you now type V and press return, you will see that it contains the following fields:

```
300x1 struct array with fields:
```

```
fname
dim
dt
pinfo
mat
n
descrip
private
```

fname is the name of the file, and dim contains the dimensions for each volume in the file. (all you need to know is that they contain other header information that SPM needs to read the file.)

It will return the dimensions of the first volume in the time-series in the x-, y-, and z-directions. You should see something like this:

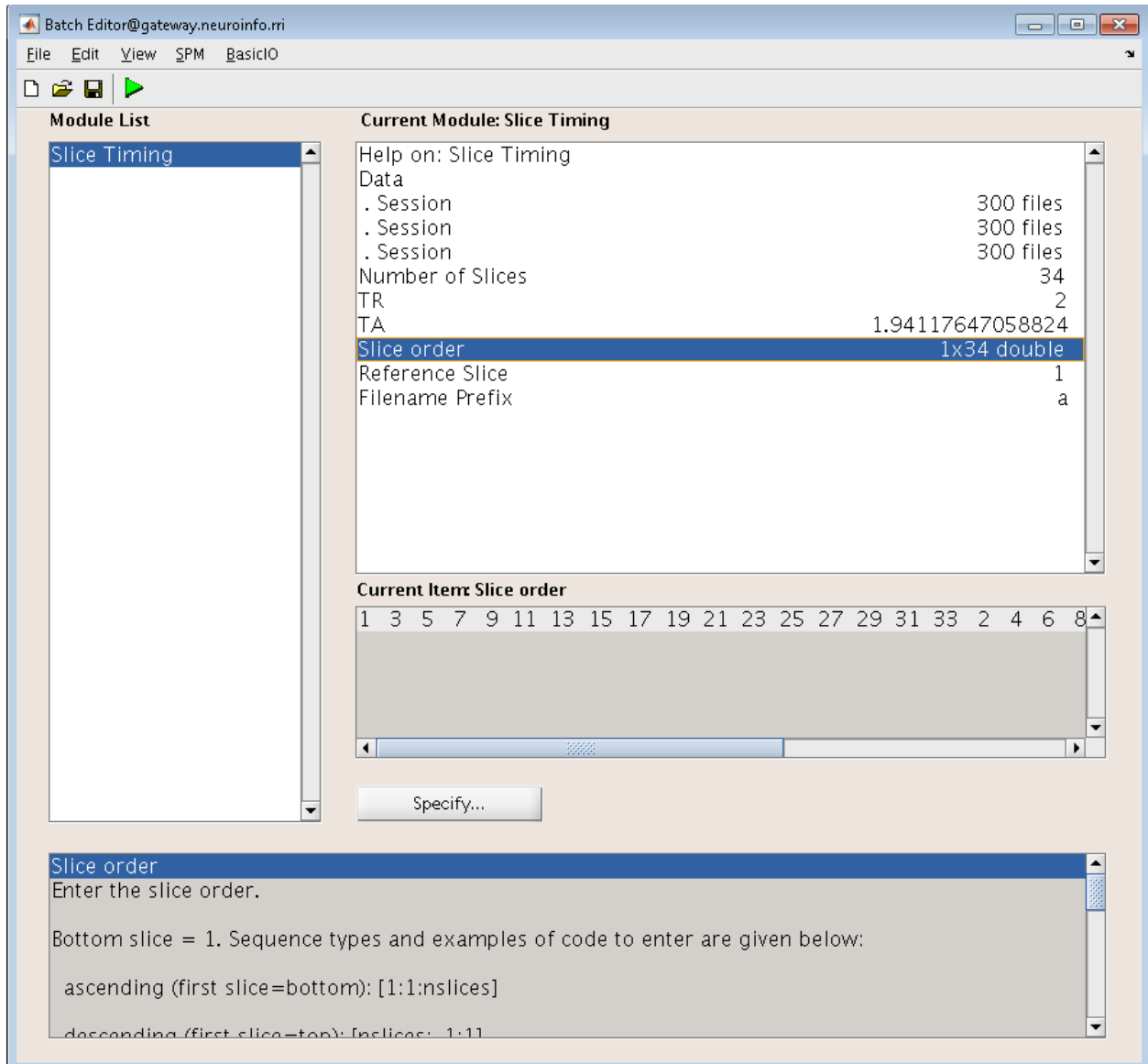
```
>> V = spm_vol('sub-02_task-balloonanalogrisktask_run-01_bold.nii')
V =
300x1 struct array with fields:
    fname
    dim
    dt
    pinfo
    mat
    n
    descrip
    private

Did you mean:
>> V(1).dim
ans =
    64    64    34
```

This means that the first volume of the time-series has the dimensions of 64x64x34 voxels, with 34 being the number of slices in the z-dimensions. We will assume that the dimensions of each image and the number of slices will be the same for every volume in the subject's functional data.

Go back to the SPM_GUI window, double-click on Number of Slices, enter a value of 34, and click OK.

Enter 2 for TR, use the formula enter 2-(2/34) for the TA. Enter [1:2:34 2:2:34] for Slice order, and for Reference Slice enter a value of 1. Leave the filename prefix as a. Do this same procedure for run-02 and run-03 as well. When you are finished, the preprocessing window should look like this:



When the images have been slice-time corrected, you are ready to coregister the functional data to the anatomical data; in other words, we will align the two sets of images as best we can.

Co-registration with SPM

To co-register the functional and anatomical images, go back to the SPM GUI and click on Coregister (Estimate & Reslice). This will open up a batch editor window with only two fields that need to be filled in - a Reference Image and a Source Image.

The Reference Image is the image that will remain stationary; the Source Image, on the other hand, is moved around until a best fit is found between the Reference and the Source image, using the cost functions described above. For most experiments, you will want to use a representative of the functional data as the Reference Image, and the anatomical data as the Source Image, since we generally want to introduce as few edits as possible to the functional data.

Double-click on the Reference Image, and select the `meansub-08_task-flanker_run-1_bold.nii`. For the source image, navigate to the `anat` directory and select the file `sub-08_T1w.nii`. Then click the green Go button. This step should only take a few moments.

When it finishes, another window will be generated showing the coregistration results with the mean functional image on the left and the anatomical image on the right. Click and drag the crosshairs in either image to see how well the images are aligned - in addition to the outlines of the brains being matched, you should also check to make sure that internal structures such as the ventricles are aligned as well. Remember that the intensities will be flipped: Darker areas in the anatomical image (such as the ventricles) will appear brighter in the functional image.

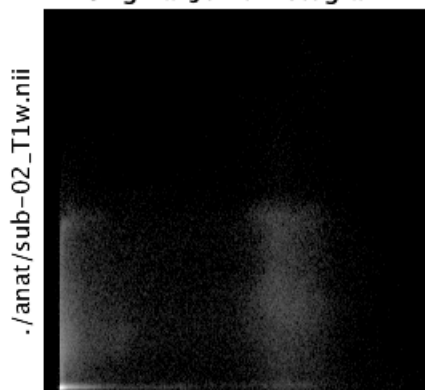
Normalised Mutual Information Coregistration

$$X1 = -0.316*X - 0.069*Y + 0.004*Z + 63.531$$

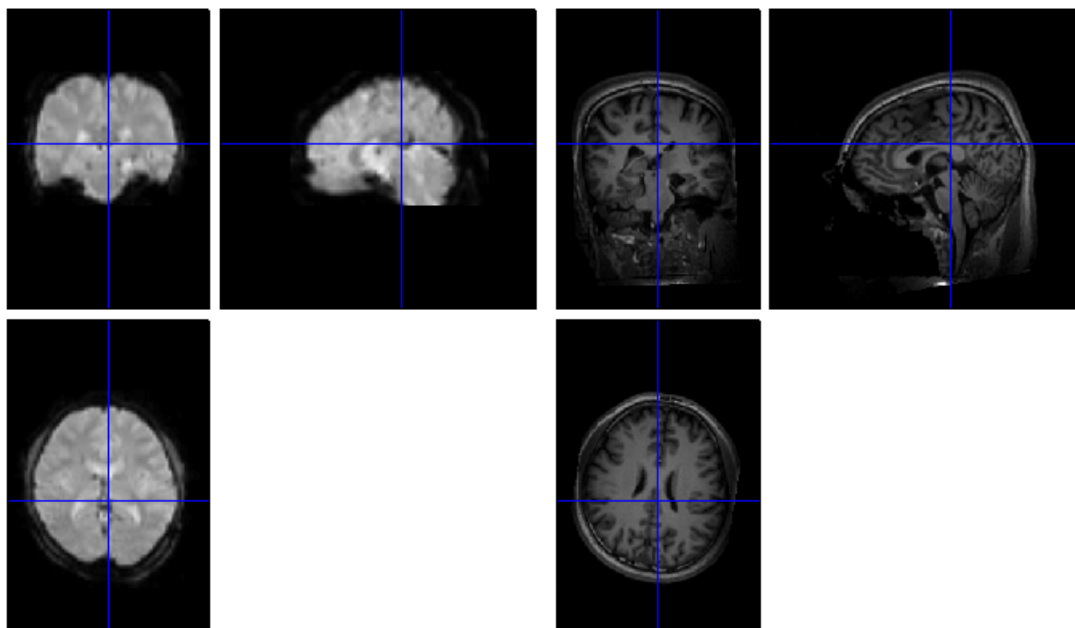
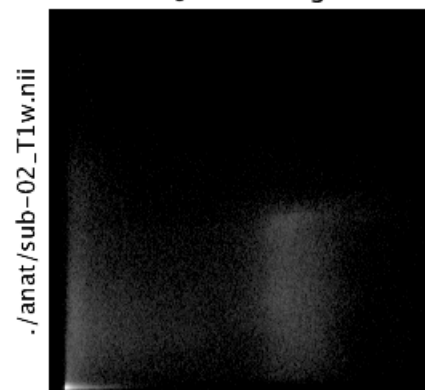
$$Y1 = -0.050*X + 0.416*Y + 0.067*Z - 6.138$$

$$Z1 = 0.009*X - 0.051*Y + 0.329*Z - 14.997$$

Original Joint Histogram



Final Joint Histogram

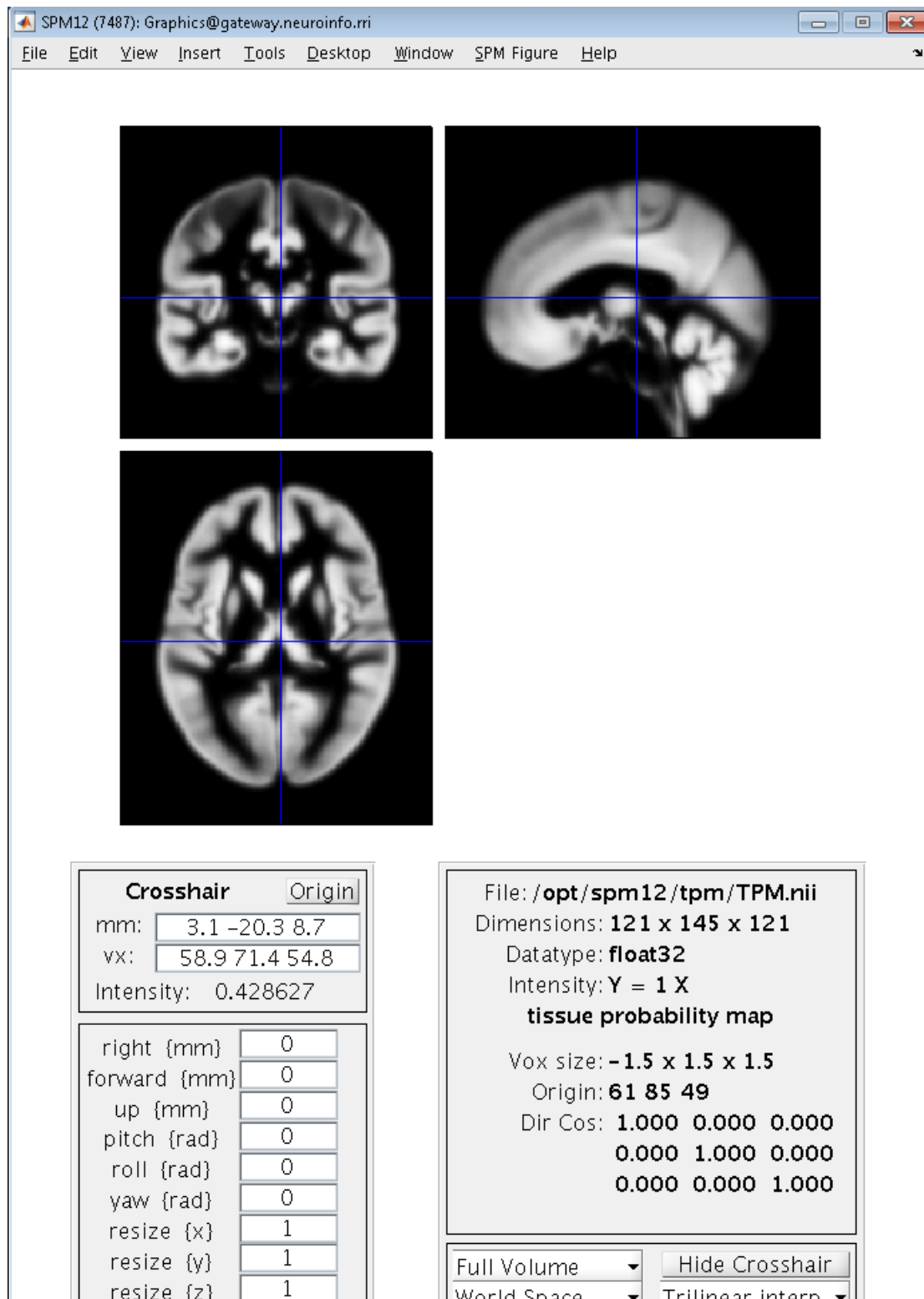


Tissue Probability Maps

The brain is composed of two major tissue types: 1 Grey matter (containing high densities of unmyelinated neurons) and 2 white matter (containing high densities of myelinated neurons). The brain is also surrounded by cerebrospinal fluid (CSF), and large amounts of CSF are contained in internal spaces within the brain called ventricles.

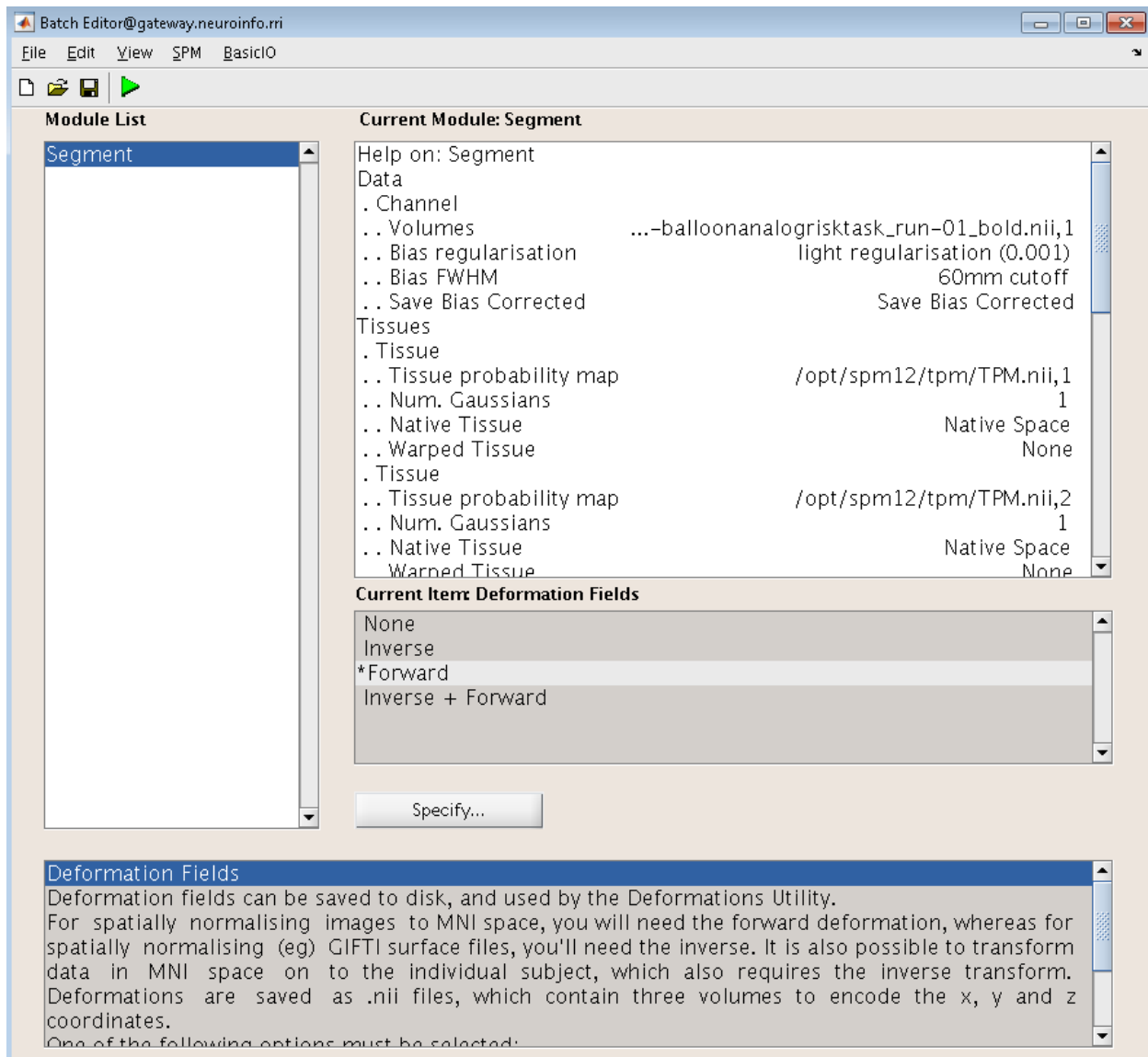
Knowing which voxels belong to which tissue type can assist in Normalizing the anatomical image, warping it to match a template in standardized space. SPM has images of six tissue priors representing their best guess as to which voxel in standardized space belongs to which tissue type. Accurately mapping the tissues of our anatomical image to the tissues of the template will increase the accuracy of our registration.

Why six priors, instead of the three we just listed? The anatomical image also contains non-brain tissues, such as soft tissue (e.g., dura mater) and skull. The last tissue type is reserved for all other tissues not captured by any of the above; usually this just means air inside the sinuses and outside the brain, but it can also detect abnormal tissue, such as tumors.



Segmentation

let's setting up the Segmentation, which requires the realigned anatomical file as input. Open Segmentation button on the SPM GUI, and click the "volumes field". Select the file rsub-02_T1w.nii, and then set the Save Bias Corrected field from Save Nothing to Save Bias Corrected. Lastly, at the very bottom of the menu, change Deformation Fields to Forward. Then click the green Go button.



Once the segmentation has finished, we are now ready to use the information generated by this step for Normalization.

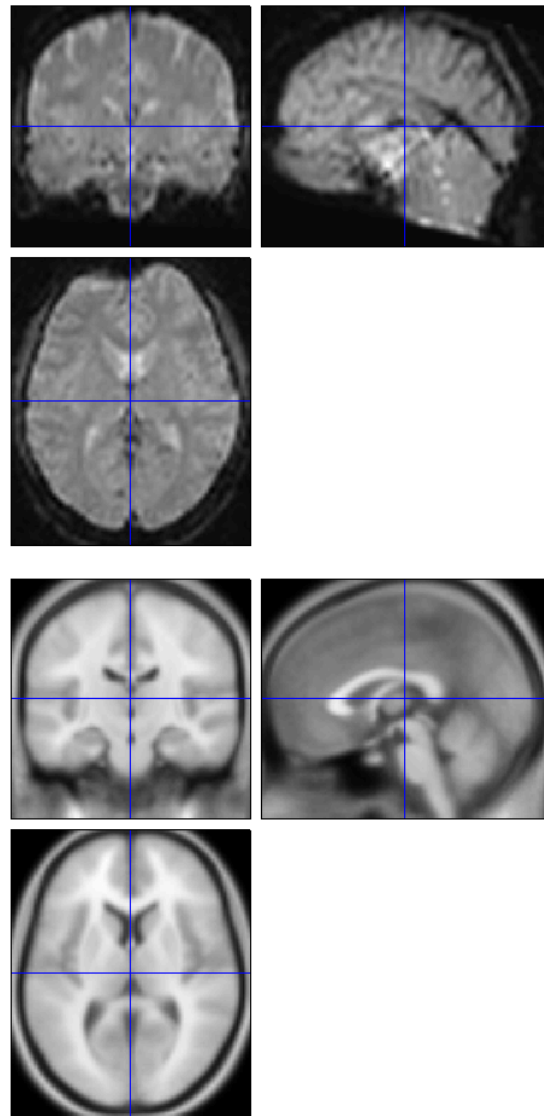
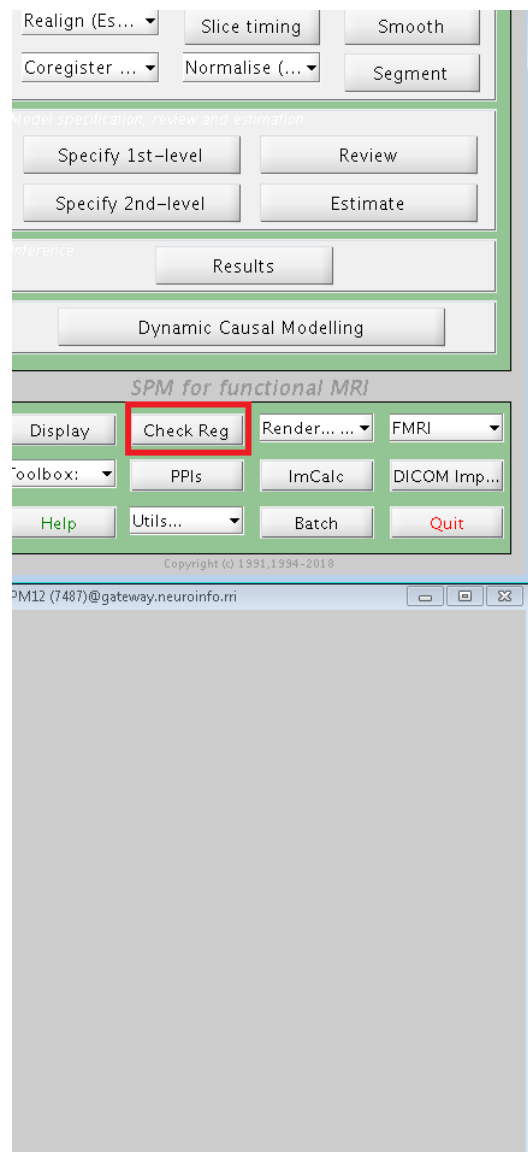
Normalization

After the anatomical image has been segmented, we can use those segmentations to normalize the image. From the SPM GUI, click on **Normalize (Write)**, from **Data** field and create a new Subject. Select the **Deformation Field** that you created in the **func** directory during **Segmentation** (the files should be called “y_rsub-02_T1w.nii”), next, **Images to Write**, select all of the realigned and slice-time corrected images for the run-1, which means you need select all 900 files from the 3 runs. Typing `^ar.*` in the “Filter field”, and entering 1:300 in the “Frames field”.

In the **Writing Options** section, we can change the voxel resolution of the images that are warped. The default of 2x2x2 will create higher-resolution images, but the files will take up more space on your computer. But you also can choose smaller files with lower resolution by changing this to 3 3 3.

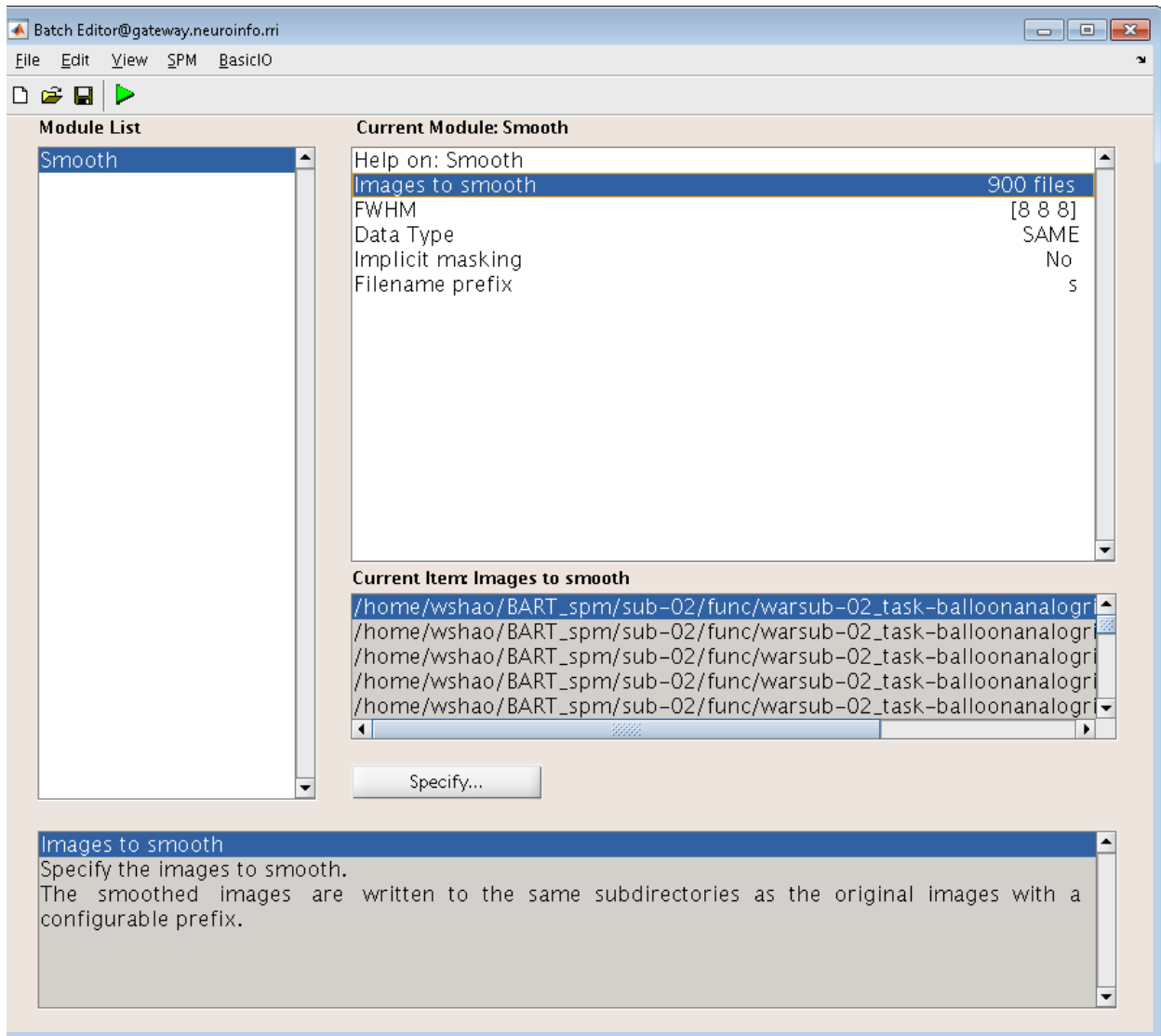
Checking the Output

Once the functional images have been normalized, let's check out the output. From SPM GUI, click on **Check Reg**, and select one of our functional volumes that has a **w** prepended (**w** = warped, that is, normalized). We also need choose the second image, which is a template image. Go to the **spm** canonical directory such as `...spm12/canonical` and select a averaged template like `avg152T1.nii`, please carefully make sure that both the outlines of the brains and the internal structures are well-aligned between these two.



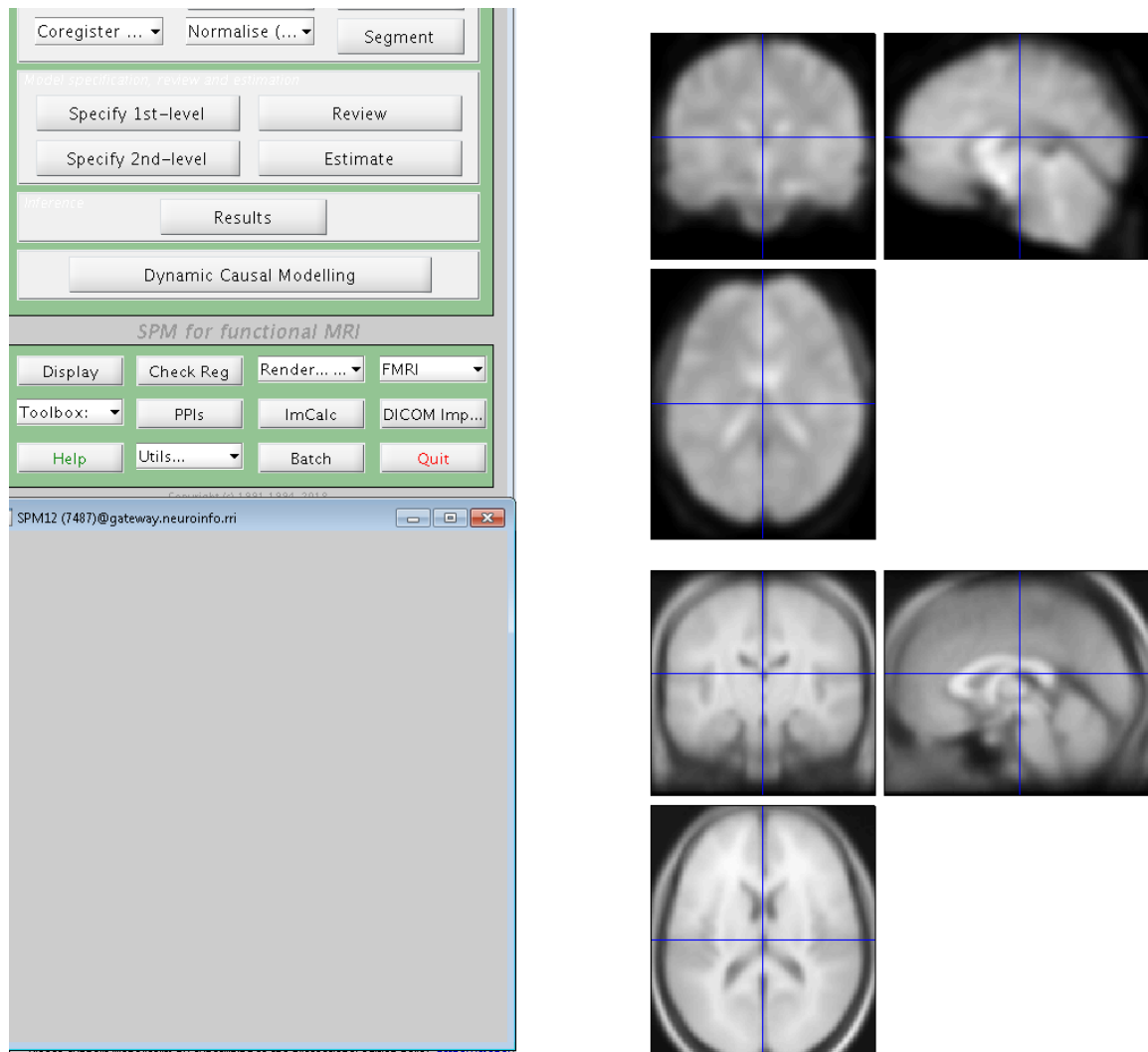
Smooth

Click the **Smooth** from SPM GUI and select **Images** to **Smooth**. choose all the 300 warped frame functional images of each run. (you can use ^warsub in “Filter fields” and 1:900 in “Frames fields” to select the images), then click **Go** button.



Smoothed Images

As before, use **Check Reg** to load a representative volume from the output file, and view it with a warped functional image that hasn't been smoothed side by side.



Statistics and modeling

After we finish all the preprocessing steps, we can go to the next - fit a model to the data. In order to understand model fitting, we need to know 3 fundamental components:

- 1 General Linear Model
- 2 The BOLD response
- 3 Time-series

General linear model

The General Linear Model (GLM) is a useful framework for comparing how several variables affect the continuous variables. Although it has many variants, the simplest form is described as: $Y = X + \cdot$. In GLM, we can use one or more regressors(X) - independent variables, to fit a model for some outcome measure(Y) - or dependent variable. To do this we need to compute numbers called beta weights(), which are the relative weights assigned to each regressor in order to get the best fitting for the data. Any discrepancies between the model and the data are called residuals().

The General Linear Model (GLM)

**Uses one or more regressors (independent variables)
to predict an outcome measure (dependent variable)**

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \varepsilon$$

Y = Dependent variable

β = Beta Weights (parameter estimates)

X = Regressor

ε = Residual

Fig. 21: The symbols representing each of these terms are shown in the equation.

For example, imagine that we want to predict the chance of infecting Covid-19 based on social distance, daily interaction in person, and longitude. It is reasonable to find that social distance has a negative correlation with the infection of Covid-19, the more distance you keep with other people, the less chance you will get Covid-19. In terms of daily interaction in person, it has a positive association, the more you interact with people in person, you are more likely to be affected by the virus. lastly, longitude has no association in general.

Therefore, we need assign each of these regressors a beta weight so that the model can fit the real data best. To be more specific, maybe each additional meter for social distance is associated with an additional 0.5 decrease in infection of Covid-19, while each additional 0.5 times of personal interaction is associated with a 0.3 increase:

Chance of infecting Covid-19 (Y) = $-0.5X_1$ (social distance) + $0.3X_2$ (personal interaction) $\rightarrow +$

This GLM can be expanded to include many regressors, but no matter how many regressors you have, the GLM assumes that the data can be modeled as a linear combination of each of the regressors. Keep GLM in your mind because we

will apply it in our model soon.

BOLD response

BOLD signal

In 1990, a researcher at Bell Laboratories named Seiji Ogawa discovered that more deoxygenated blood leads to a decrease in the signal measured from a brain region. An increase in oxygenated blood, on the other hand, increases the signal - this increase in oxygenated blood was later shown to be correlated with increased neural firing. This change in signal is known as blood oxygen level dependent signal (BOLD signal). Shortly afterward, a researcher at Massachusetts General Hospital named Ken Kwong in 1992 demonstrated that the BOLD signal could be used as an indirect measure of neural activity. His experiment consisted of alternately showing a flashing checkerboard and a black screen to the subject for one minute each. The BOLD signal was recorded during each condition, as shown in the following video:

In the 1990's, empirical studies of the BOLD signal demonstrated that, after a stimulus was presented to the subject, any part of the brain responsive to that stimulus - say, the visual cortex in response to a visual stimulus - showed an increase in BOLD signal. The BOLD signal also appeared to follow a consistent shape, peaking around six seconds and then falling back to baseline over the next several seconds. This shape can be modeled with a mathematical function called a Gamma Distribution. As Gamma Distribution is created with parameters to best fit the BOLD response observed by the these empirical studies, it is the canonical Hemodynamic Response Function(HRF).

Gamma Distribution

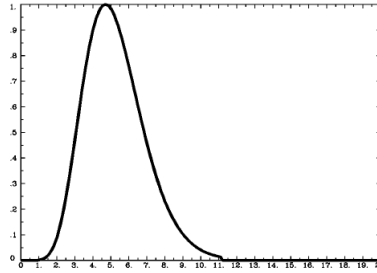
Gamma distribution is a distribution that arises naturally in processes for which the waiting times between events are relevant. It can be thought of as a waiting time between Poisson distributed events. For example, suppose you are fishing and you expect to get a fish once every 1/2 hour. Compute the probability that you will have to wait between 2 to 4 hours before you catch 4 fish. One fish every 1/2 hour means we would expect to get $\lambda = 1 / 0.5 = 2$ fish every hour on average. Using $\lambda = 2$ and $k = 4$, we can compute this as follows:

$$P(2 \leq X \leq 4) = \sum_{x=2}^4 \frac{x^{4-1} e^{-x/2}}{\Gamma(4) 2^4} = 0.12388$$

When applied to fMRI data, the Gamma Distribution is called a basis function because it is the fundamental element of the model we will create and fit to the time series of the data. Furthermore, if we know what the shape of the distribution looks like in response to a very brief stimulus, we can predict what it should look like in response to stimuli of varying durations, as well as any combination of stimuli presented over time.

The HRF for a Single impulse Stimulus

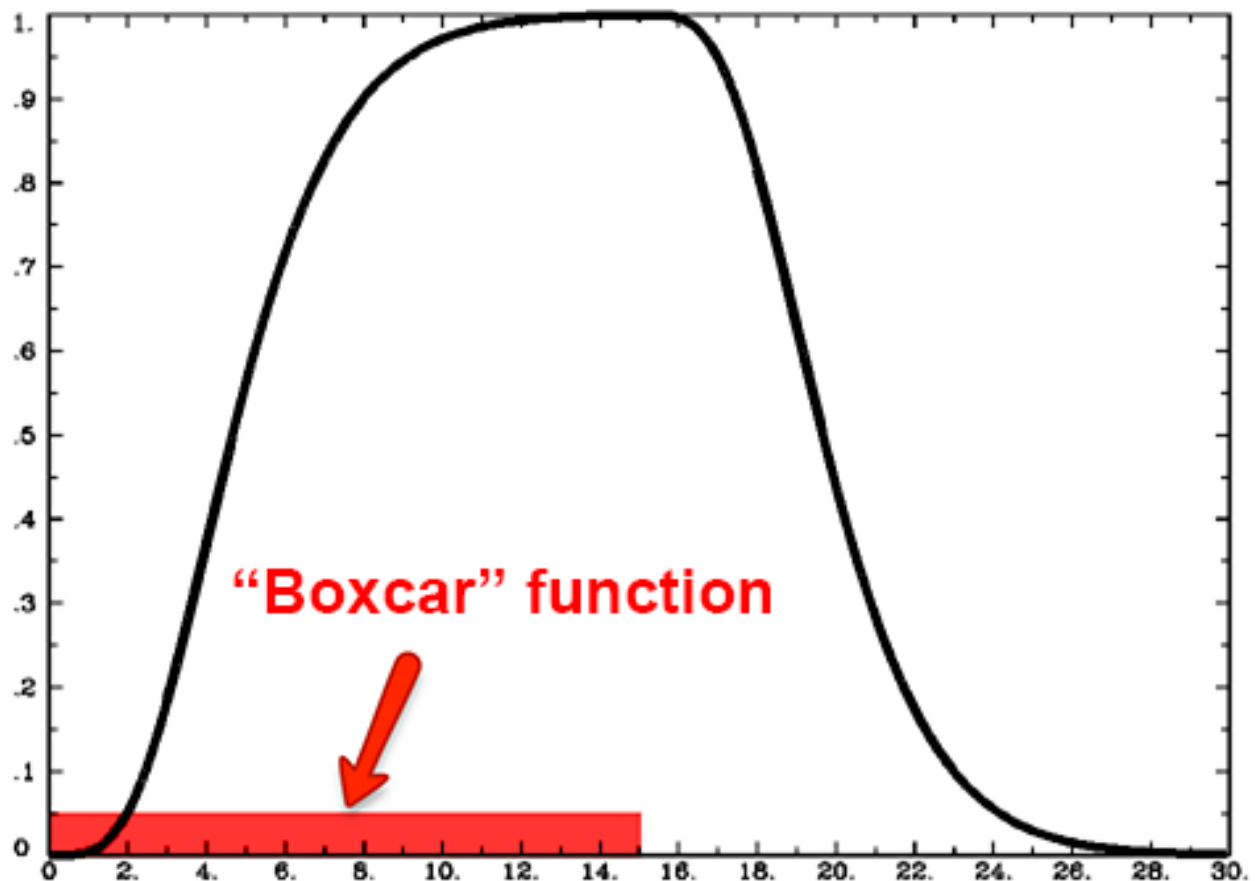
If the duration of a stimulus is very short, such as a snap of the fingers, we can say that it is an impulse stimulus - in other words, it has no duration. As you can see in the following figure, the shape of the BOLD signal looks like a typical Gamma Distribution, with a peak close to the beginning of the time axis (i.e., the x-axis) and a long tail to the right.



The HRF for a Single Boxcar Stimulus

Although many studies use stimuli lasting only a second or less, some studies present stimuli for longer periods of time. For example, imagine that the subject looks at a flashing checkerboard for fifteen seconds. In this case the shape of the HRF will be more spread out with a sustained peak proportional to the duration of the stimulus, falling back to baseline only after the stimulus has ended. This stimulus is called a boxcar stimulus, because it looks like a boxcar on a train.

In this case the Gamma Distribution is convolved with the boxcar stimulus. Convolution is the averaging of two functions over time; as a result, the Gamma Distribution broadens as it is averaged with the boxcar stimulus, and returns to baseline when the stimulus is removed.



Multiple HRFs overlapping

We have seen what the BOLD signal looks like after a stimulus is presented and how the HRF models the shape of that signal. But what happens if another stimulus is presented before the BOLD response for the previous stimulus has returned to baseline?

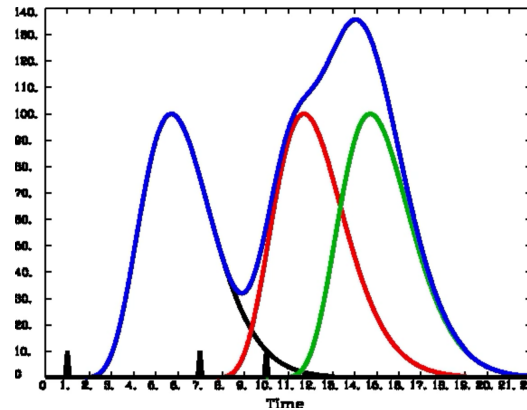


Fig. 22: Convolution of the HRFs for individual stimuli. The overall BOLD response (blue) is a moving average of the individual HRFs outlined in black, red, and green. The vertical black lines on the x-axis represent impulse stimuli. Figure created by Bob Cox of AFNI.

In that case, the individual HRFs are summed together. This creates a BOLD response that is a moving average of the individual HRFs, and the shape of the BOLD signal becomes more complex as more stimuli are presented close together.

Fig. 23: Animations originally created by Bob Cox of AFNI

Time series

We have mentioned this concept several times before. As the basic composition of fMRI data. Remember that fMRI datasets contain several volumes strung together like beads on a string - we call this concatenated string of volumes a run of data. The signal that is measured at each voxel across the entire run is called a time-series. The time-series represents the signal that is measured at each voxel. We just saw how we can use several regressors, or independent variables, to estimate an outcome such as chance of infecting Covid-19. Conceptually, we're doing the same thing when we use several regressors to estimate brain activity, which is our outcome measure with fMRI data: We estimate the average amplitude of the BOLD signal in response to each condition in our model.

Creating the time series

Since one of our goals is to create the ideal time-series so that we can use it to estimate the beta weights for our GLM, we need to create the ideal time-series first.

What do we need? Let's take a look at the BART dataset. you could find some "event.tsv" files in the subject's func directory. These files contain three pieces of information for the timing files. There are:

1 the experimental condition name 2 the onset time of trial for each condition, relative to the onset of the scan 3 The duration of each trial

onset	duration	trial_type	cash_demean	control_pumps_demean	explode_demean	pumps_demean	response_time
0.061	0.772	pumps_demean	n/a	n/a	-2.000	2.420	
4.958	0.772	pumps_demean	n/a	n/a	-1.000	0.578	
7.179	0.772	pumps_demean	n/a	n/a	0.000	0.766	
10.416	0.772	pumps_demean	n/a	n/a	1.000	0.840	
13.419	0.772	pumps_demean	n/a	n/a	2.000	1.462	
16.754	0.772	explode_demean	n/a	n/a	1.700	n/a	
24.905	0.772	pumps_demean	n/a	n/a	-0.500	1.295	
27.454	0.772	pumps_demean	n/a	n/a	0.500	1.083	
30.111	0.772	cash_demean	-4.000	n/a	n/a	1.498	
38.449	0.772	pumps_demean	n/a	n/a	-1.500	0.656	
41.028	0.772	pumps_demean	n/a	n/a	-0.500	0.652	
44.529	0.772	pumps_demean	n/a	n/a	0.500	0.864	
47.692	0.772	pumps_demean	n/a	n/a	1.500	1.909	
51.102	0.772	cash_demean	-2.000	n/a	n/a	1.035	
59.031	0.772	pumps_demean	n/a	n/a	-2.000	0.833	
61.850	0.772	pumps_demean	n/a	n/a	-1.000	0.536	
63.516	0.772	pumps_demean	n/a	n/a	0.000	1.085	
67.007	0.772	pumps_demean	n/a	n/a	1.000	0.786	
69.693	0.772	pumps_demean	n/a	n/a	2.000	2.036	
74.286	0.772	cash_demean	-1.000	n/a	n/a	0.603	
82.940	0.772	pumps_demean	n/a	n/a	-2.500	1.109	
86.124	0.772	pumps_demean	n/a	n/a	-1.500	0.541	
88.648	0.772	pumps_demean	n/a	n/a	-0.500	1.001	
91.642	0.772	pumps_demean	n/a	n/a	0.500	1.375	
94.742	0.772	pumps_demean	n/a	n/a	1.500	1.331	
98.989	0.772	pumps_demean	n/a	n/a	2.500	1.549	
102.416	0.772	cash_demean	0.000	n/a	n/a	2.009	
108.445	0.772	control_pumps_demean	n/a	-5.000	n/a	n/a	1.053
111.308	0.772	control_pumps_demean	n/a	-4.000	n/a	n/a	0.333
114.044	0.772	control_pumps_demean	n/a	-3.000	n/a	n/a	0.270
117.302	0.772	control_pumps_demean	n/a	-2.000	n/a	n/a	0.404
120.638	0.772	control_pumps_demean	n/a	-1.000	n/a	n/a	0.332
123.096	0.772	control_pumps_demean	n/a	0.000	n/a	n/a	0.610
124.915	0.772	control_pumps_demean	n/a	1.000	n/a	n/a	1.087
128.429	0.772	control_pumps_demean	n/a	2.000	n/a	n/a	1.029
130.687	0.772	control_pumps_demean	n/a	3.000	n/a	n/a	1.699
133.919	0.772	control_pumps_demean	n/a	4.000	n/a	n/a	0.347
136.034	0.772	control_pumps_demean	n/a	5.000	n/a	n/a	1.655

This information needs to be extracted from the events.tsv files and be transformed into a format that the AFNI can read. our job is to create a timing file for explode and cash experimental condition, and then split the file based on which run the condition was in. We will have 6 timing files:

1 Timings for the explode trials that occurred during the first run (explode_run1.txt) 1.2 Timings for the explode trials that occurred during the first run (explode_run2.txt) 1.3 Timings for the explode trials that occurred during the first run (explode_run3.txt) 2 Timings for the cash trials that occurred during the first run (cash_run1.txt) 2.2 Timings for the cash trials that occurred during the second run (cash_run2.txt) 2.3 Timings for the cash trials that occurred during the third run (cash_run3.txt)

These need to be extracted from the “events.tsv” files and formatted in a way that the SPM software can read(txt file). In this case, we will create a timing file for two conditions, and then split that file according to which run the condition was in. In total, then, we will create four timing files:

Each of these timing files will have the same format consisting of two columns, in the following order:

1 Onset time, in seconds, relative to the start of the scan; and 2 Duration of the trial, in seconds.

The code:

```
#!/bin/bash

#Check whether the file subjList.txt exists; if not, create it
if [ ! -f subjList.txt ]; then
    ls -d sub-?? > subjList.txt
fi

#Loop over all subjects and format timing files into a format that FSL can understand
for subj in `cat subjList.txt` ; do
    cd $subj/func #Navigate to the subject's func directory, which contains the event_
    ↪files

    #Extract the onset and duration for the pump,control,explode, and cash out trials_
    ↪for each run.
```

(continues on next page)

(continued from previous page)

```

cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="pumps_
↪demean") {print $1, $2, "1"}}' > pump_run1.txt
cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="pumps_
↪demean") {print $1, $2, "1"}}' > pump_run2.txt
cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="pumps_
↪demean") {print $1, $2, "1"}}' > pump_run3.txt

cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="control_
↪pumps_demean") {print $1, $2, "1"}}' > control_run1.txt
cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="control_
↪pumps_demean") {print $1, $2, "1"}}' > control_run2.txt
cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="control_
↪pumps_demean") {print $1, $2, "1"}}' > control_run3.txt

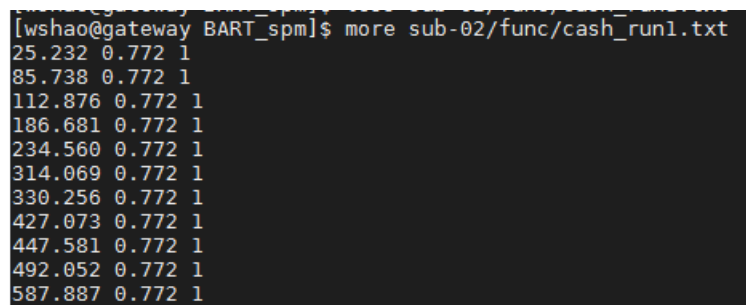
cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="explode_
↪demean") {print $1, $2, "1"}}' > explode_run1.txt
cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="explode_
↪demean") {print $1, $2, "1"}}' > explode_run2.txt
cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="explode_
↪demean") {print $1, $2, "1"}}' > explode_run3.txt

cat ${subj}_task-balloonanalogrisktask_run-01_events.tsv | awk '{if ($3=="cash_demean
↪") {print $1, $2, "1"}}' > cash_run1.txt
cat ${subj}_task-balloonanalogrisktask_run-02_events.tsv | awk '{if ($3=="cash_demean
↪") {print $1, $2, "1"}}' > cash_run2.txt
cat ${subj}_task-balloonanalogrisktask_run-03_events.tsv | awk '{if ($3=="cash_demean
↪") {print $1, $2, "1"}}' > cash_run3.txt

cd ../../
done

```

Copy this block of code, write them into a script file save as timing.sh. Then, place it in the directory that containing the all subjects, type `bash timing.sh`. This will create all the timing files for each run for each subject and store them in func directory accordingly. we can type `more sub-02/func/cash_run1.txt` to check the data. You are supposed to see the figure below:



```

[wshao@gateway BART_spm]$ more sub-02/func/cash_run1.txt
25.232 0.772 1
85.738 0.772 1
112.876 0.772 1
186.681 0.772 1
234.560 0.772 1
314.069 0.772 1
330.256 0.772 1
427.073 0.772 1
447.581 0.772 1
492.052 0.772 1
587.887 0.772 1

```

Once we have created the timing files, we are now ready to use them to fit a model to the fMRI data.

First level analysis in SPM

Specifying the Model

Since we have created the timing files previously, it is time for us to use them in conjunction with our imaging data to create statistical parametric maps. These maps could indicate that the correlation between the ideal time-series (the onset times convolved with the HRF in our model) and the time-series collected in this experiment. When we use the SPM to construct contrasts, beta weight represents the amount of modulation of the HRF, which is then transformed into a t-statistic.

To get started, create a sub-directory in sub-02 called 1stlevel so we can organize the data. Then, Open SPM GUI from Matlab terminal and select 1st-Level, select the 1stLevel directory we just created. All of the output of the 1st-level analysis will be kept in this folder. After that, we'll fill out the section on Timing parameters. Select Seconds for the design unit, and enter a value of 2 for Interscan Interval. Then go to Data & Design. and build three new sessions by clicking three times on New: Subject/Session. Go to the func directory and use the "Filter and Frames fields" we used before to pick all 300 volumes of the warped usable data(file start with swar) for the first session's Scans. And do the same for the other two session.

Return to the field for the first time. In the experiment, there are two conditions, and both conditions occur in each run. To construct two new condition fields, go to conditions and then New: Condition twice. Double-click on Name and type cash and explode. for the first condition.

In order to find out the onset times for each occurrence of the cash condition. From the Matlab terminal, navigate to the func directory and type:

```
cashRun1 = importdata('cash_run1.txt'); IncRun1(:,1)
```

Which will give you the onset times for cash condition in run_1, copy and paste the onset times onto the Onsets sector. Then we can therefore enter the number 0.772 in the Durations field, (you can check the duration time by less sub-02_task-balloonanalogrisktask_run-01_events.tsv) and SPM will assume that it is the same duration for every trial.

Repeat the process for the explode condition in run 01, as well as the cash and explode conditions in run 02 and run 03, remembering to set the duration time to 2. Here's the code for displaying the onset times for the remaining onset times:

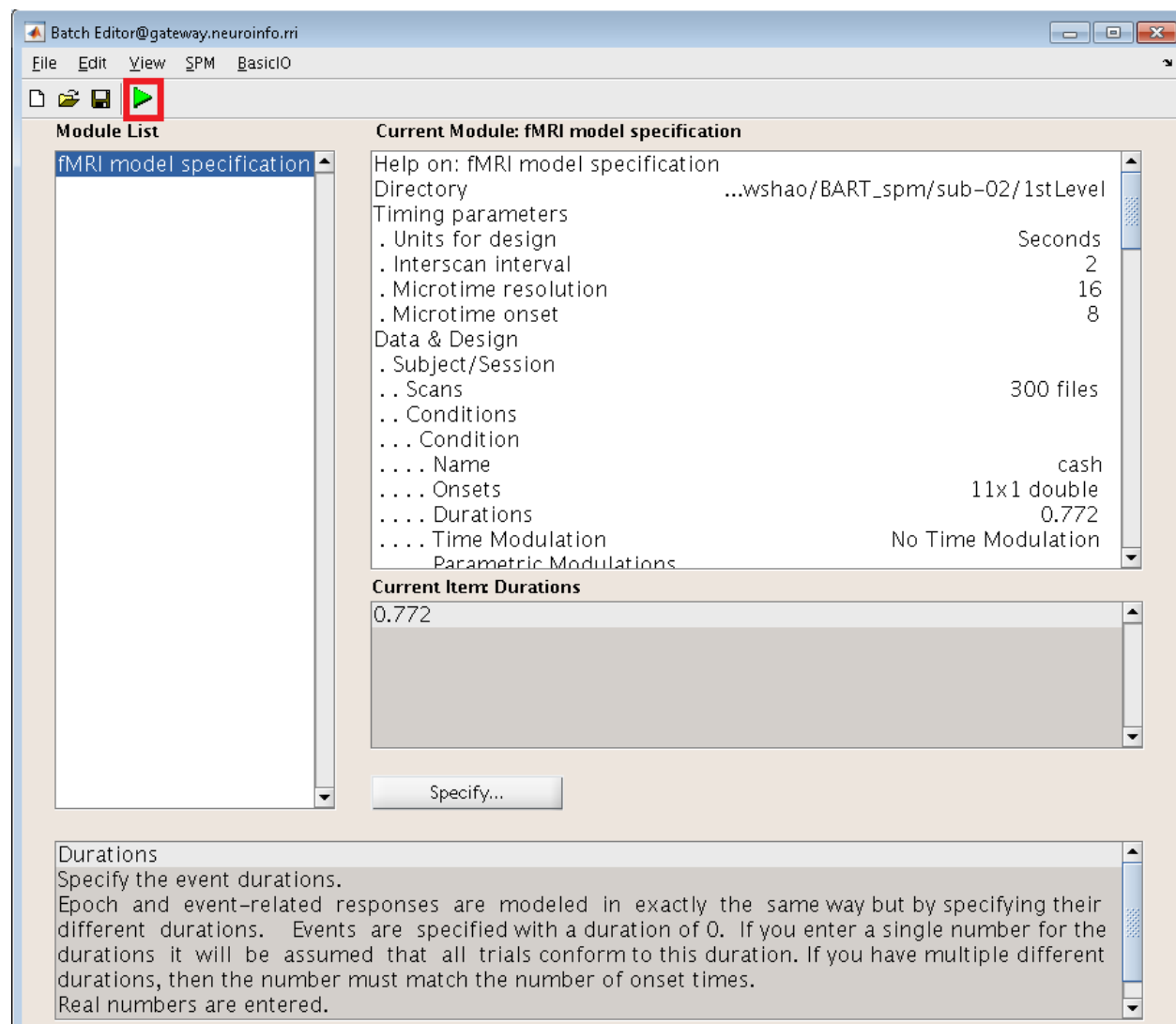
```
explodeRun1 = importdata('explode_run1.txt');
explodeRun1(:,1)

cashRun2 = importdata('cash_run2.txt');
explodeRun2(:,1)

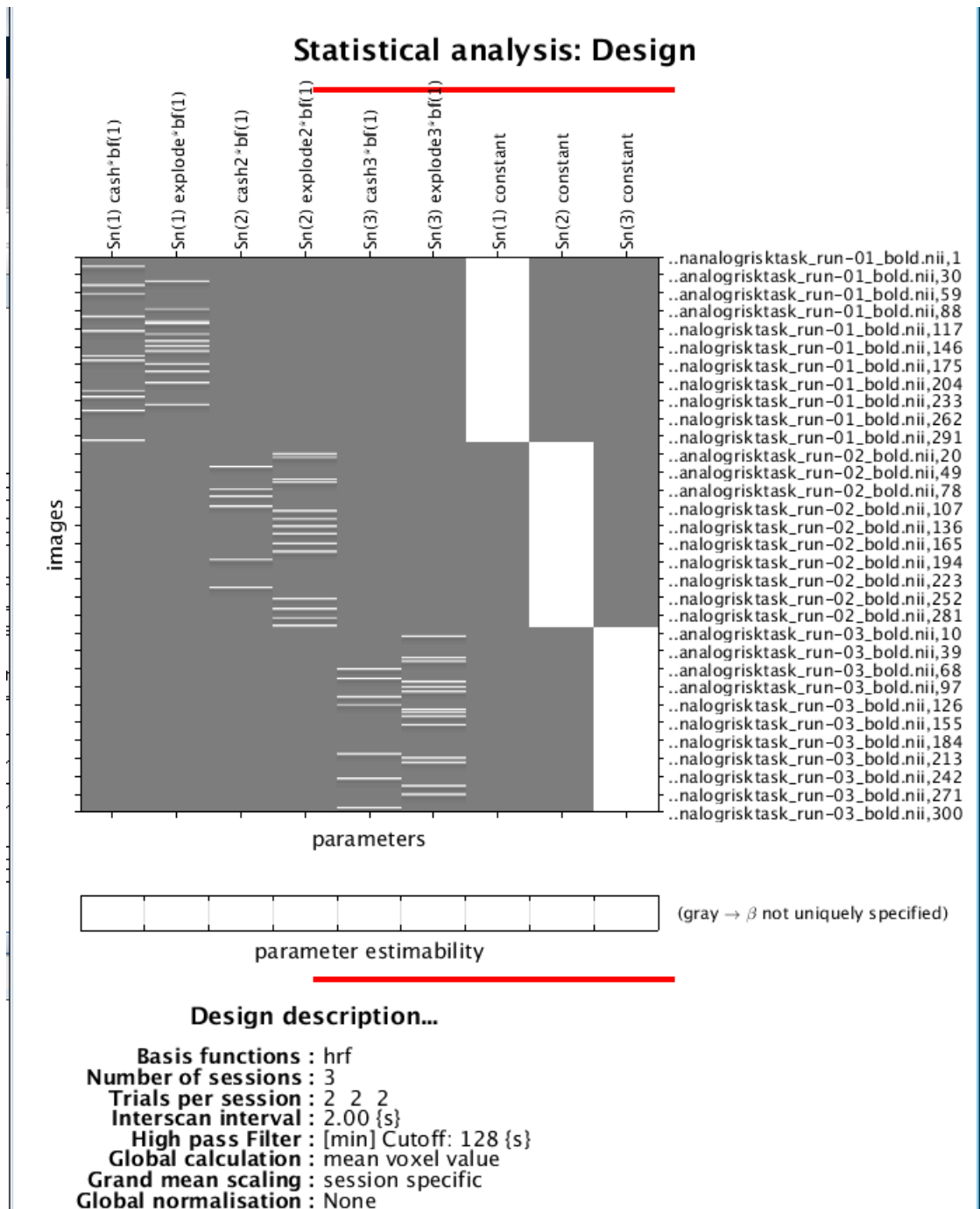
explodeRun2 = importdata('explode_run2.txt');
explodeRun2(:,1)

cashRun3 = importdata('cash_run3.txt');
explodeRun3(:,1)

explodeRun3 = importdata('explode_run3.txt');
explodeRun3(:,1)
```



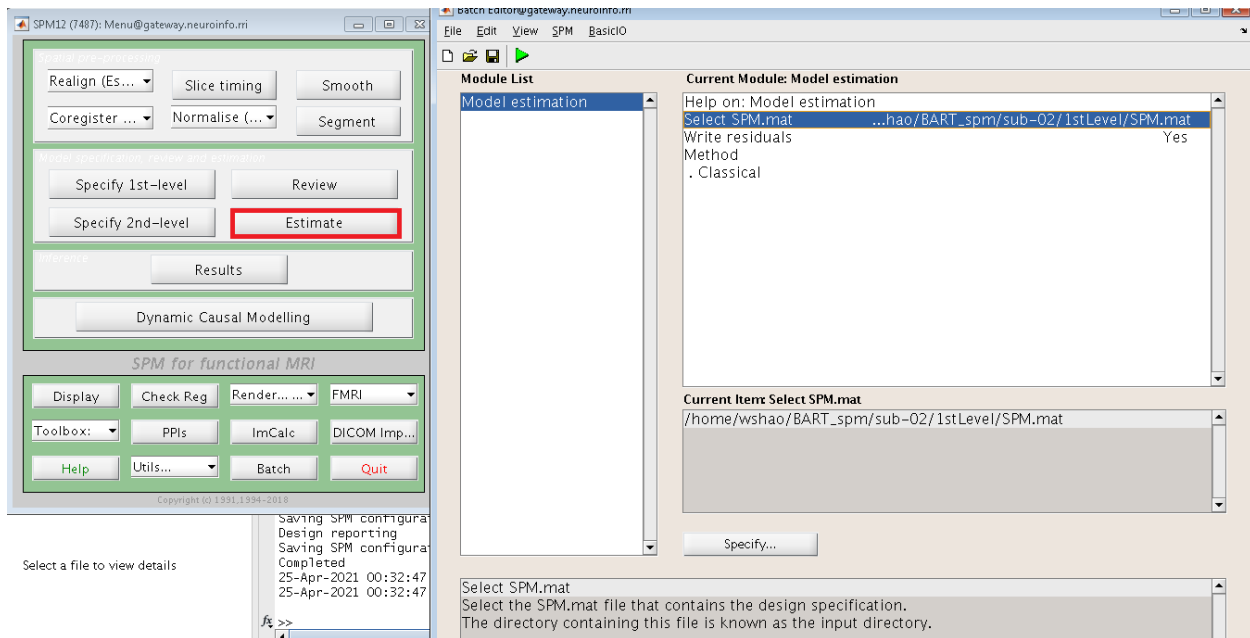
The names will be stored in a file called SPM.mat in the 1stLevel directory which we will look at later in more detail. Now, click the green Go button when you're done. It should only take a few moments to estimate the model. When it's all said and done, it should look like this:



The General Linear Model for For a single subject. The ideal time-series for the cash and explode conditions for the first session are shown in the first two columns, while the ideal time-series for the conditions of run 02 are shown in the next two columns, the next two columns indicate the ideal time-series for the conditions of run 03. The last three columns are baseline regressors that capture the mean signal of each run. In this figure, time runs from top to bottom, and lighter colors represent more activity.

Estimating the Model

We have created our GLM, the next step is to estimate the beta weights for each condition. Click **Estimate** from the SPM GUI, and select the SPM.mat from **select SPM.mat** tab from sub-02/1stLevel. Change the “Write residuals option” to Yes. and then click the green **Go** button. This will take a few minutes to run.

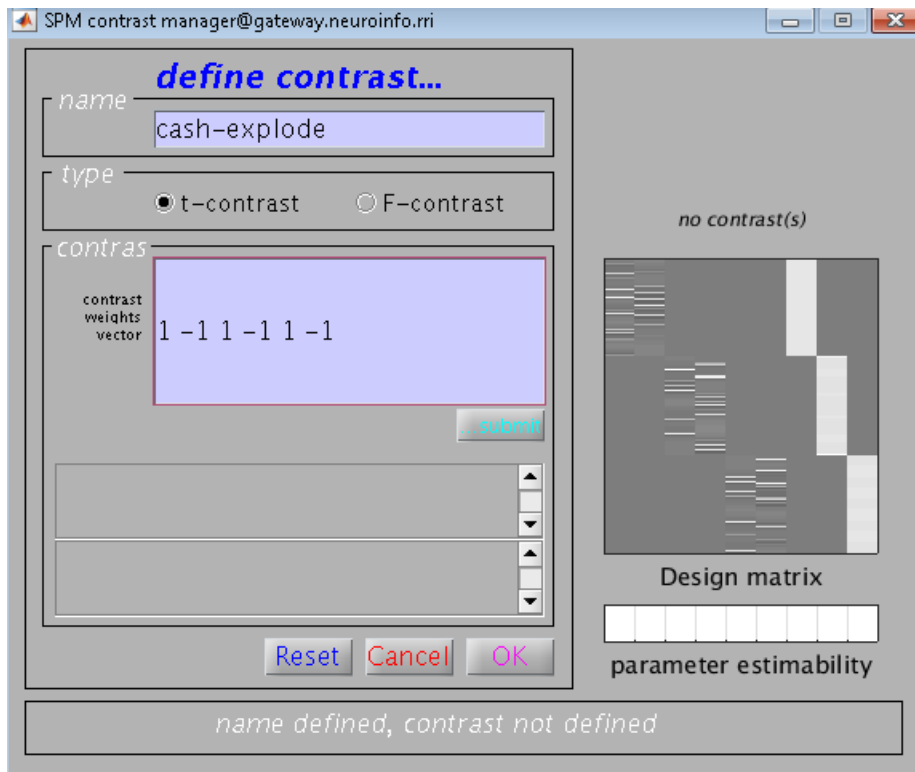


The Contrast Manager

When you've done estimating the model, it's time to start making contrasts. what we need to do is estimate a beta weight for the cash condition and a beta weight for the explode condition. To be more specific, we can determine a contrast estimate at each voxel in the brain by take the difference between these two conditions. A contrast map will be generated if by doing this way.

To make these contrasts, go to the **Result** button from SPM GUI and select the SPM.mat file that was created when the model was estimated. The design matrix is located on the right side of the screen. Select **Define New Contrast** and put cash-explode in the Name field.

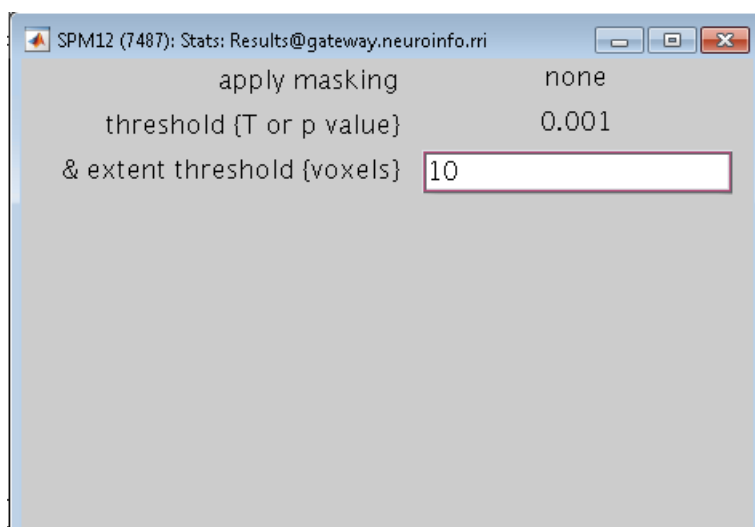
type 1 -1 1 -1 1 -1 in the contrast vector window, then press submit. If the contrast is right, you will see “name identified, contrast defined” at the bottom of the window. Make sure your contrast manager looks like the image below, and then click OK



Examining the Output

Now, since we have set the contrast manager, click done, you will see a few options:

1 apply masking: set this to “none” because we want to examine all of the voxels and don’t want to restrict our analysis. 2 p value adjustment to control: Click on “none”, and set the uncorrected p-value to 0.001. This will test each voxel individually at a p-threshold of 0.01. 3 extent threshold {voxels}: Set this to 10 for now, which will only show clusters of 10 or more contiguous voxels.

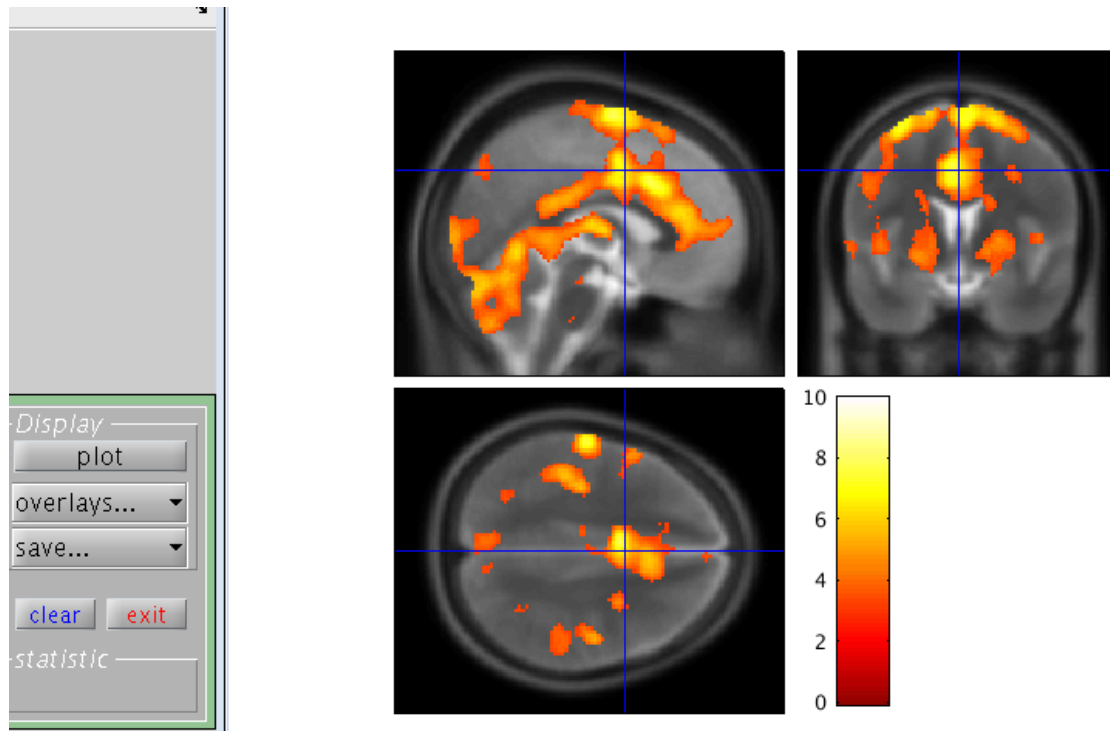


We will see our results in a standardized space in three orthogonal planes, which reflected on a glass brain once you’ve finished defining the options. The dark spots reflect clusters of voxels that passed our statistical threshold. The top-right

side is a copy of our design matrix and the contrast that we are currently looking at.

The coordinates and statistical significance of each cluster are mentioned in a table at the bottom. Set-level is the first column, and it shows the likelihood of seeing the current number of clusters, *c*. Next, the cluster-level column shows the significance for each cluster (measured in number of voxels, or *kE*) using different correction methods. The “t- and z-statistics” of the peak voxel within each cluster are shown in the peak-level column, with the main clusters in bold and any sub-clusters marked in lighter font below the main cluster. Finally, in the rightmost column, the MNI coordinates of the peak for each cluster and sub-cluster are shown. The coordinates for a cluster will be highlighted in red and the cursor in the glass brain view will switch to those coordinates if you left-click on them. You can click and drag the red arrow header in the glass brain, then right-click on it and choose one of the options for jumping to the nearest suprathreshold voxel or the nearest local maximum.

From the left-button window, click on overlays/sections and then pick a standard space other than the glass brain. Go to the `spm12/canonical` directory and choose a brain scan such as “avg152T2”.



The results will now be shown on the template as a heatmap. The statistical table will reappear after you position the crosshairs over a specific cluster and press the “current cluster” button in the Results window, highlighting the coordinates of the cluster you picked.

Scripting

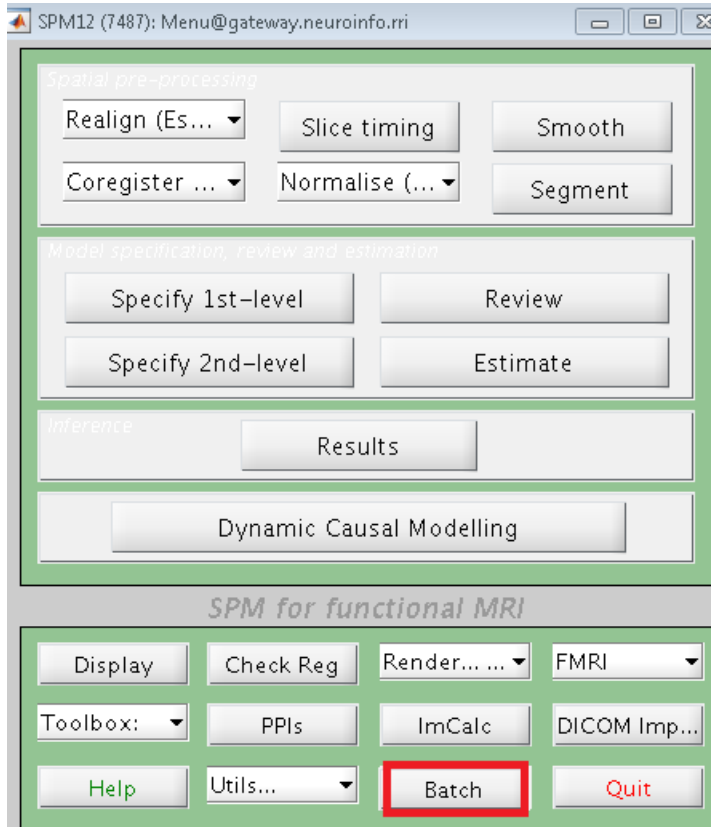
Since we have 16 subjects and each subject has 3 runs. In total, we need to repeat all the preprocessing and 1st level analysis in AFNI_GUI 48 times! it is not hard but it is a really tedious job and you could make errors easily. As Joey from **Friends** said, there’s gotta be a better way, and there is.

Here is the script that makes your life easier!

Creating the Template

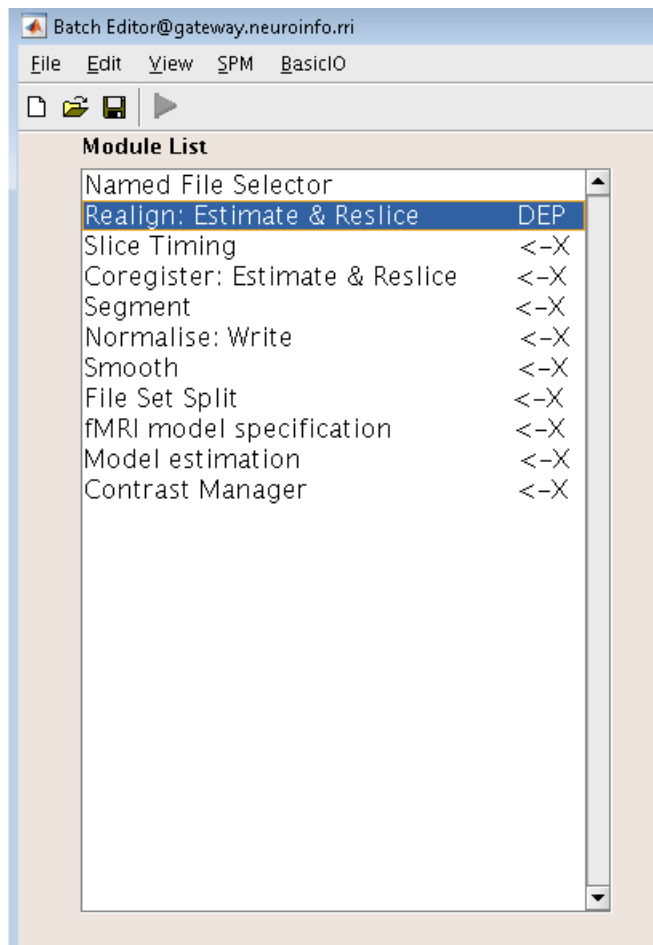
To begin with, let's create a script template for our data. As you have done all the preprocessing of sub-02 from SPM GUI, what we need to do in here is clicking on each preprocessing button of the SPM GUI and add each of the preprocessing modules to our template. Then, just as in the previous tutorials, we'll fill in the inputs for each preprocessing and statistical modelling portion, and translate what we see from the SPM GUI into Matlab code.

Now, Open the SPM GUI and select Batch from the menu.



Select the following modules in the order from the SPM at the top menu of the Batch Editor window:

```
BasicIO -> File/Dir Operations -> File Operations -> Named File Selector
SPM -> Spatial -> Realign -> Estimate & Reslice
SPM -> Temporal -> Slice Timing
SPM -> Spatial -> Coregister: Estimate & Reslice
SPM -> Spatial -> Segment
SPM -> Spatial -> Normalise -> Write
SPM -> Spatial -> Smooth
BasicIO -> File/Dir Operations -> File Operations -> File Set Split
SPM -> Stats -> fMRI Model Specification
SPM -> Stats -> Model Estimation
SPM -> Stats -> Contrast Manager
```

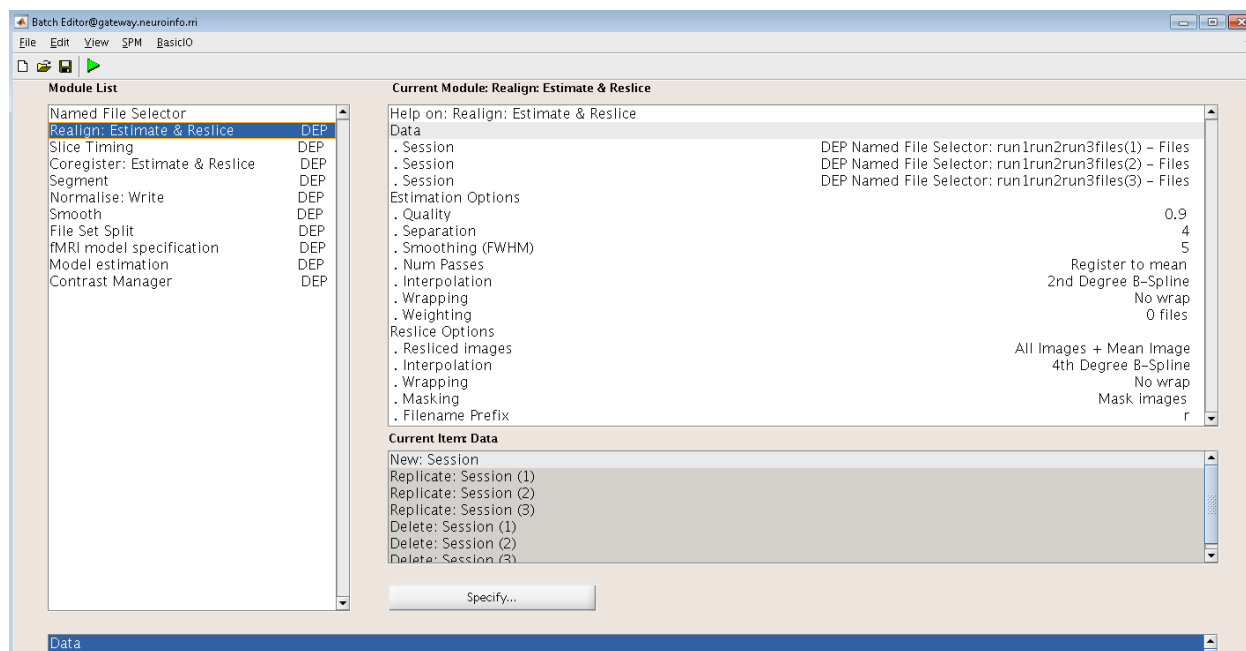
File Selection and File Splitting

First and foremost, we'll address the first additional modules that aren't listed above. `Named File Selector` module.

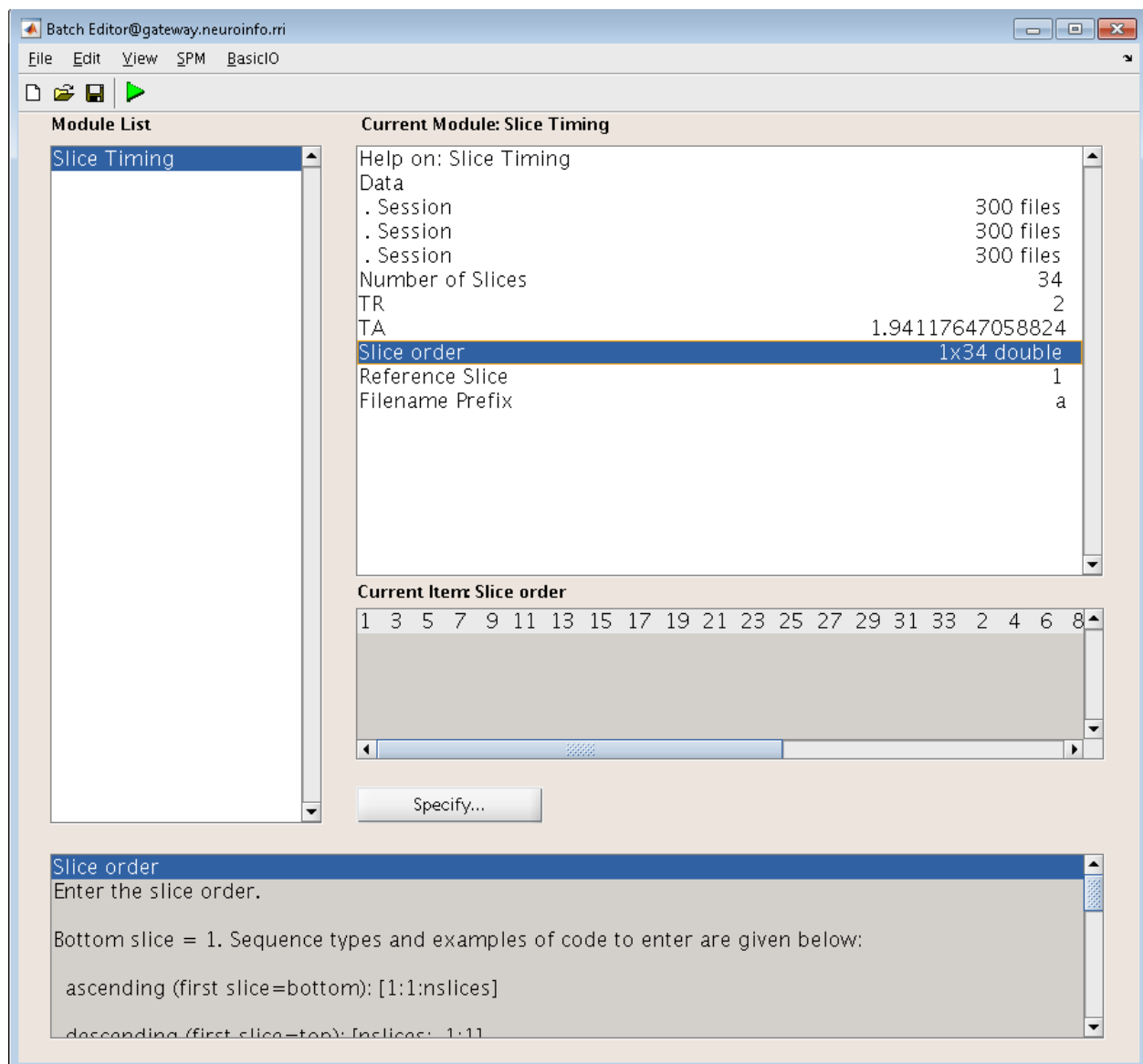
The first module requires an input name and sets of files. We will create three file sets, and enter the **run-01_bold.nii**, **run-02_bold.nii** and **run-03_bold.nii** files for each set after you use `gunzip` to unzip the file. After that, we will deal with the preprocessing modules, Realignment, create three sessions and enter the files for each session respectively.

Preprocessing Modules

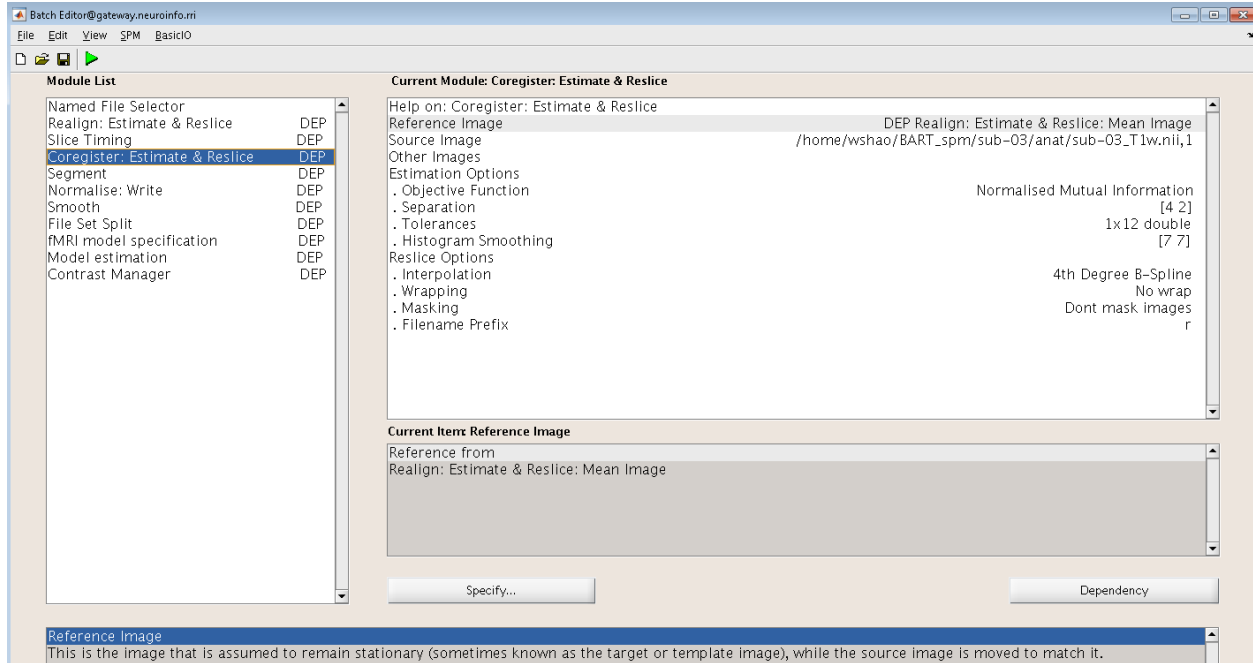
Just as we did in the previous chapters, we must now fill in each of the appropriate fields. This is the most time-consuming aspect of the tutorial, but keep in mind that if you don't script your analysis, you'll have to do it manually for each session. In case you may wonder what you are supposed to input for a later preprocessing step if the required data hasn't been created yet. The `Batch Editor` can allow you to use "Dependencies" from earlier steps, indicating that the input should come from the previous step's output step. If you click the `Dependency` button in the `Realign` module for the first Session, for example, you can pick the option "`Named File Selector: run1run2run3files(1)`," Once you've filled it in, it should look like this: `Realign` module



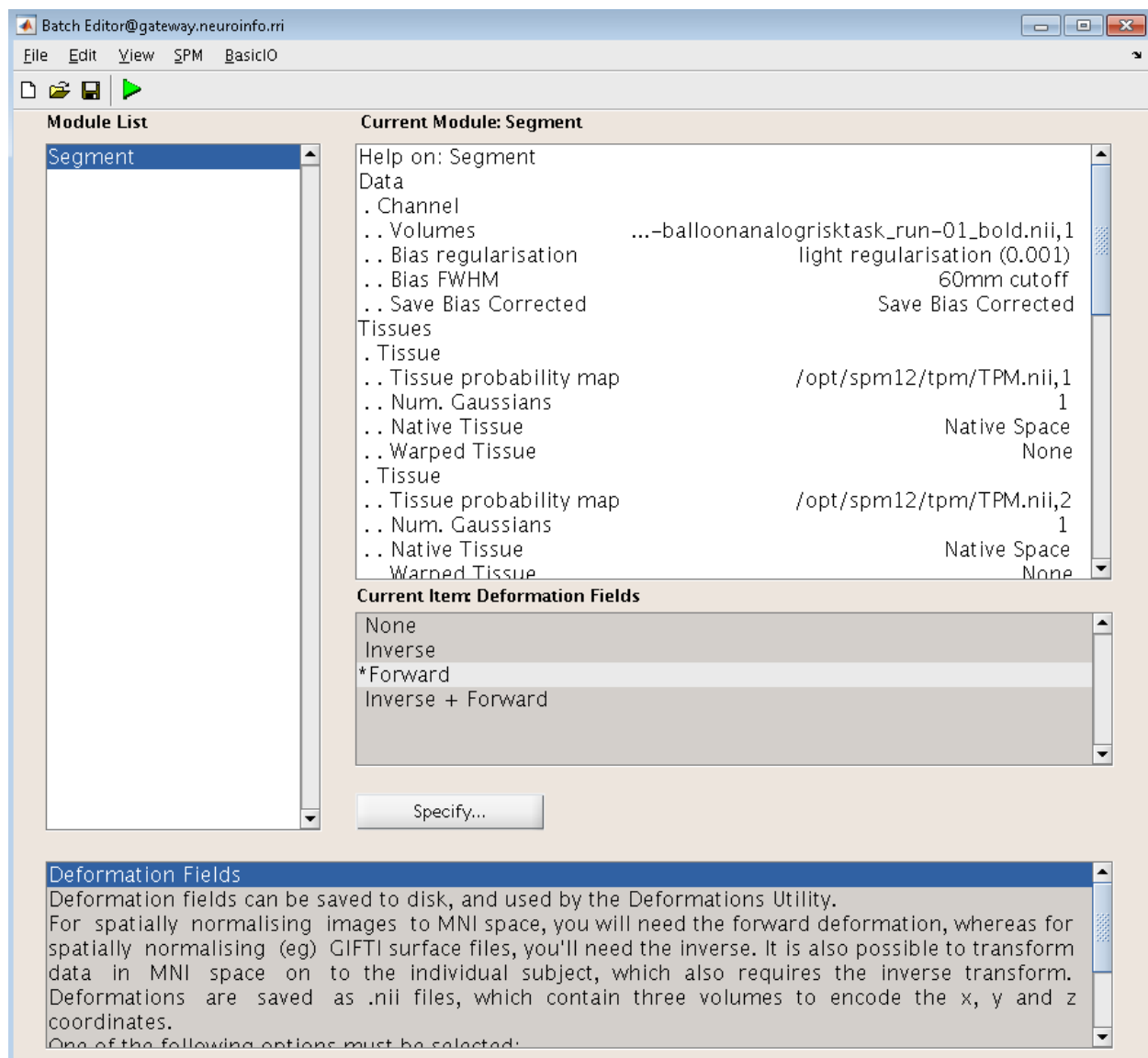
And the same with Slice Timing module,



keep continue with the coregister module, we can use Reference Image with the mean functional image generated during Realignment:

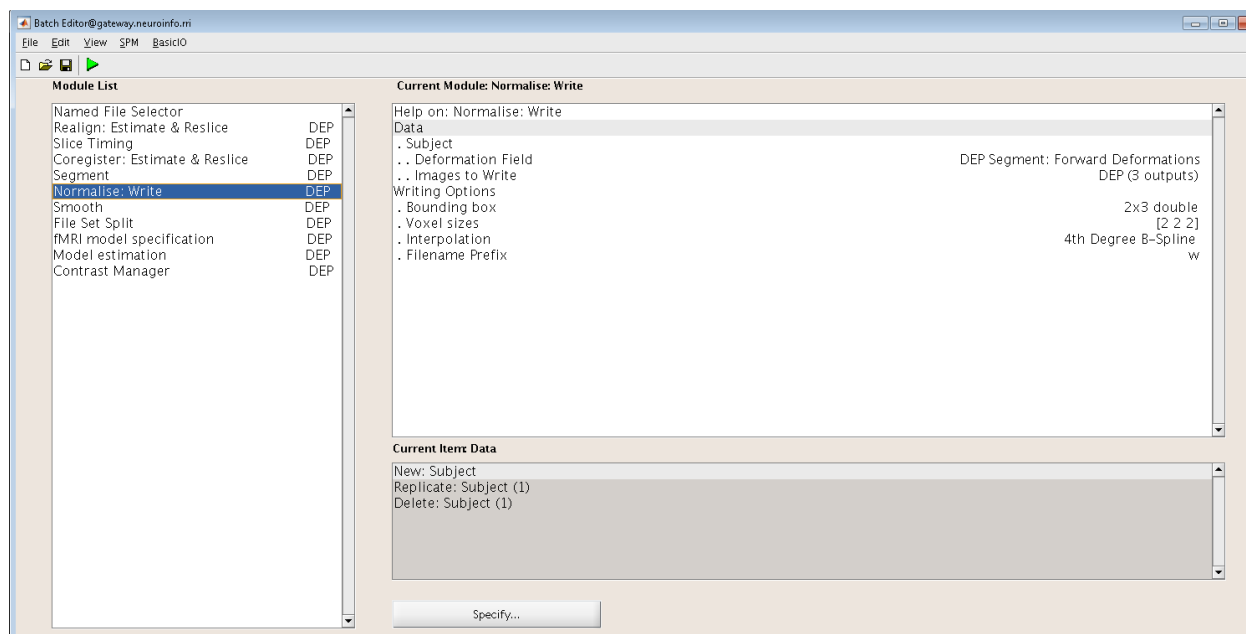


Followed by “Segmentation module”, which we need to change a few parameters such as `volumes``save Bias Corrected``` and `Deformation Fields`

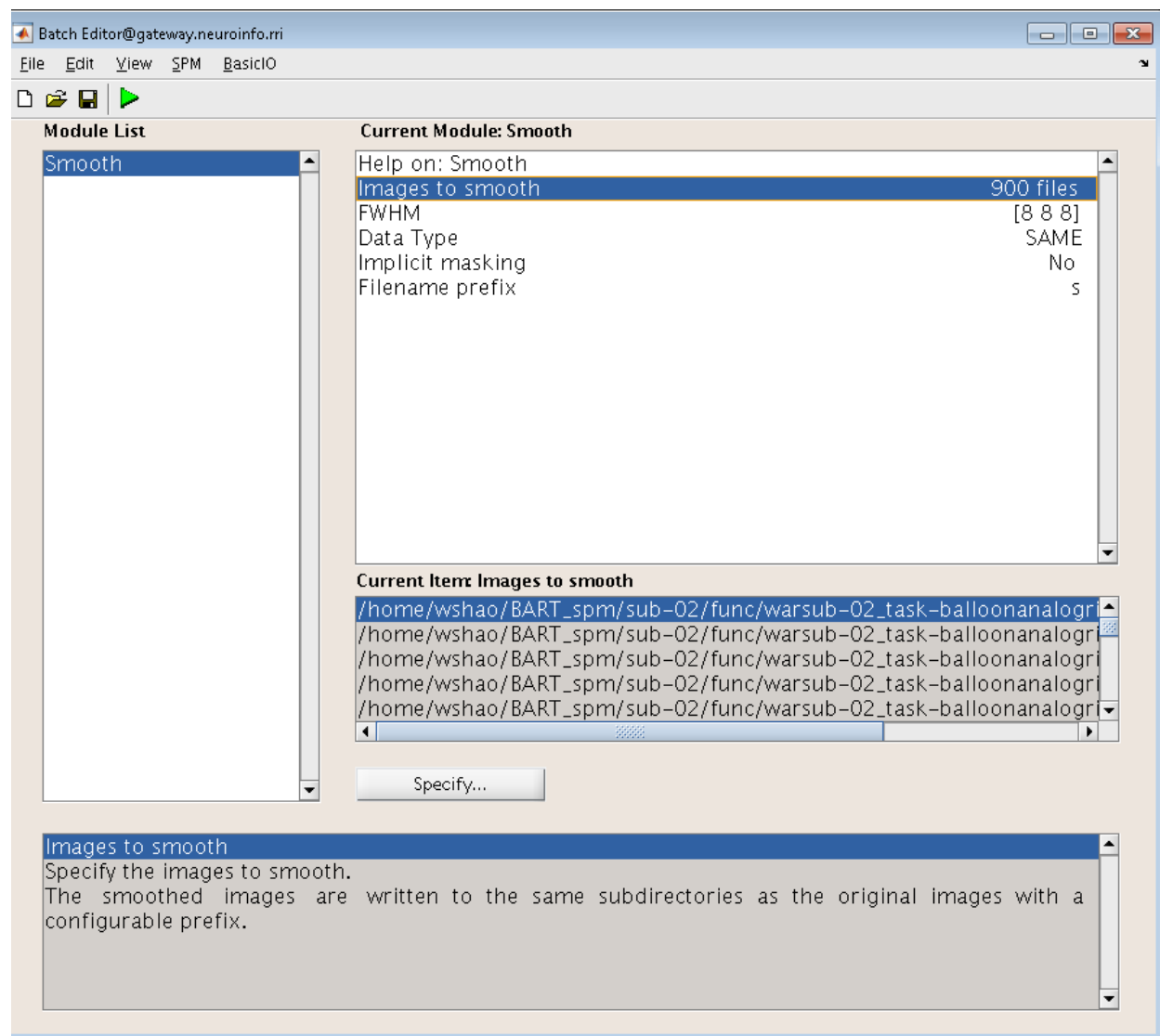


The Normalise preprocessing step requires both the Forward Deformation fields from Segmentation, as well as both the Slice Timing outputs from Sessions 1, 2 and 3

Normalise module

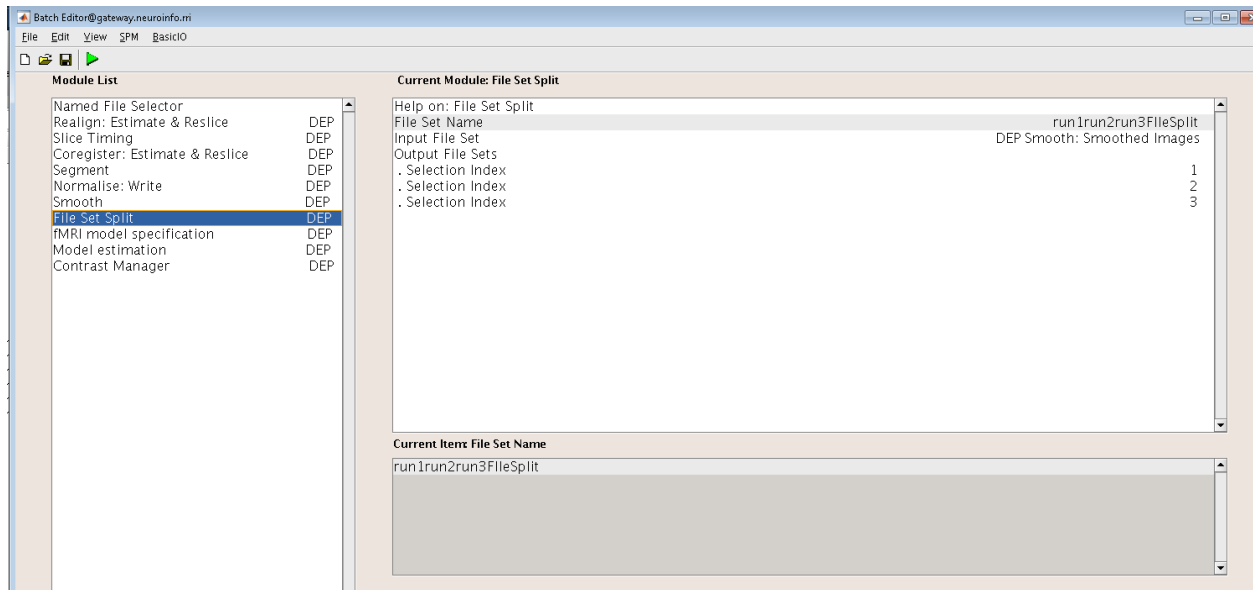


The Smooth module will use the images generated by Normalization



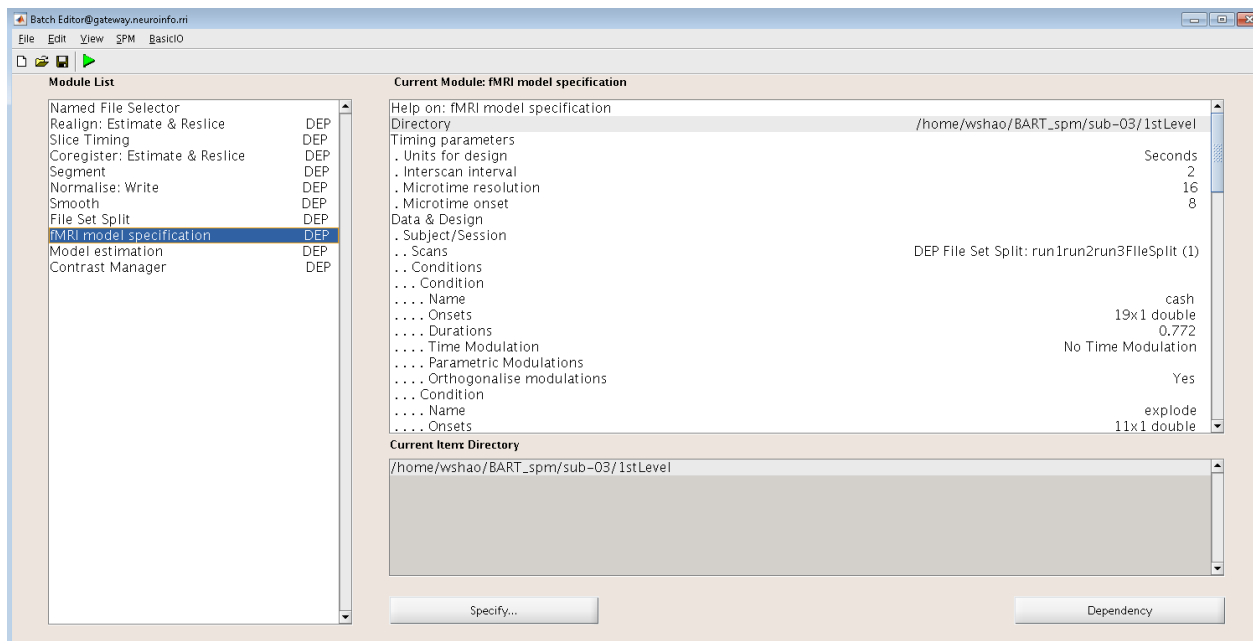
label the **File Set Name** as `run1run2run3FileSplit` (this name is simply a label for reference). The **Input File Set** is the smoothed images from the Smoothing module, and as with the **Named File Selector** module, we create two **Output File Sets**. The **Selection Index** for the first one is 1, and the **Selection Index** for the second one is 2. and The **Selection Index** for the third one is 3, This directs the module to split the smoothed images into three separate sets, based on how they were labeled by the previous **Named File Selector** module.

File Set Split module



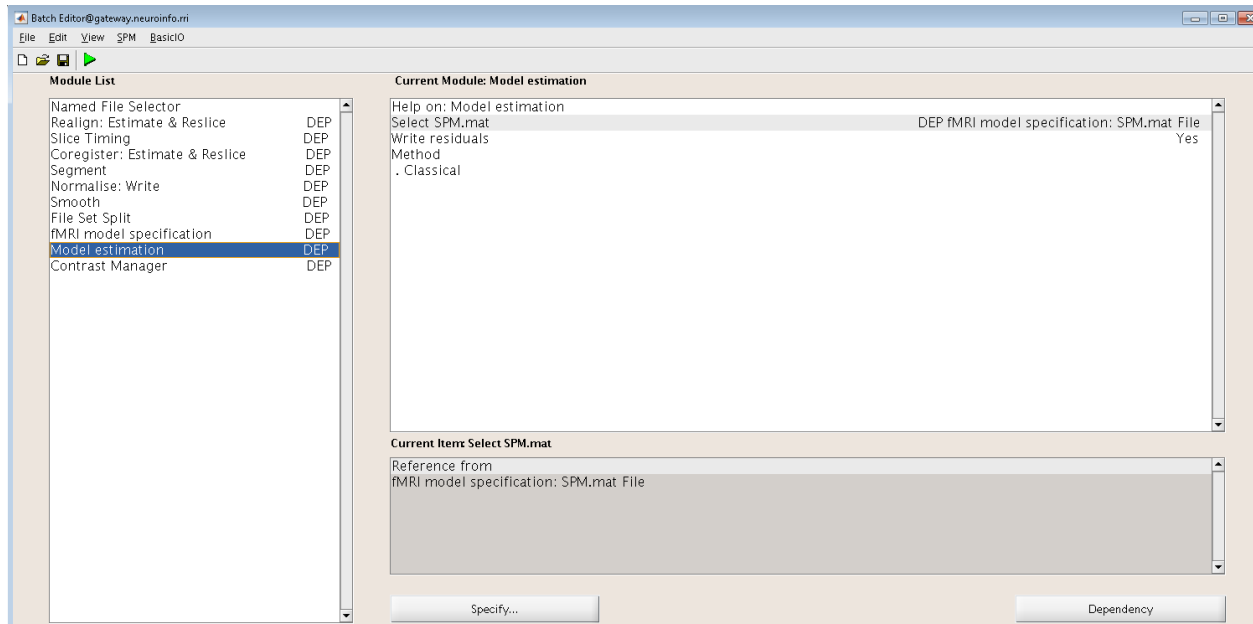
Model Specification module probably is the most tedious module, it will use the images created during Smoothing, and you also need fill the name, onset time, and Durations as well.

fMRI model specification



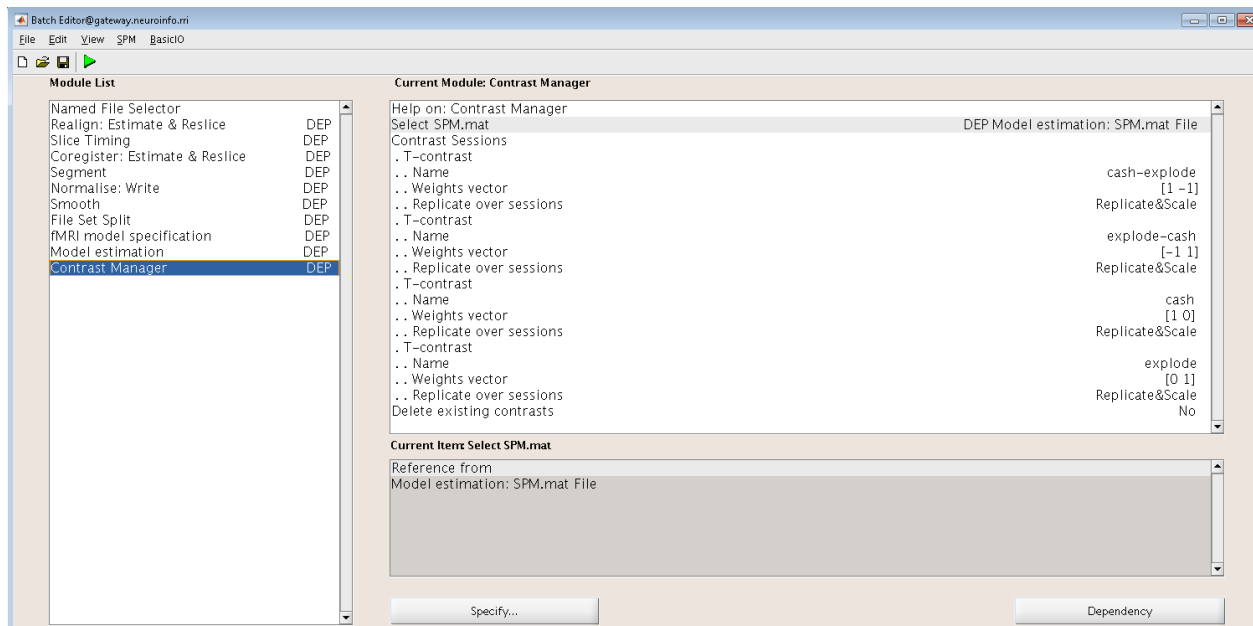
The Model Estimation module analyzes the data output from Model Specification:

Model estimation



Lastly, the contrast manager will load the SPM.mat file created by the Model Estimation module, For the contrast module, we select the “Replicate&Scale” option. This will replicate the contrast weights across all of the sessions for that subject, and scale them in inverse proportion to the number of sessions.

Contrast Manager



Editing the Matlab file

The Batch module we have just created is specific to sub-03 so that we have used sub-03's images and timing files, and the results will only apply to sub-03. We can click the green Go button to run and test the script. More importantly, after a few adjustments, we can adapt this script to all of the rest subjects in this study.

First of all, we need to save the modules into a Matlab script. Click on File -> Save Batch and Script, and label the file BART_runprproc. Save it to the BART directory that contains all of your subjects. This will create a Matlab script file that you can open in the Matlab window.

Open the Matlab terminal, navigate to the BART directory which contains the BART_runprproc.m script, Open it by type open BART_runprproc.m from the matlab terminal. We will need to make the following edits:

- 1 Replace the subject number "03" with a variable containing a different subject number on each instance of a for-loop;
- 2 Replace the username ("wshao") with a variable pointing to the username of whichever machine name is currently being used by you.
- 3 change the input directory accordingly to fit in your machine

These three changes will allow us to place the existing code in a for-loop which will run over a set of numbers indicating each subject in the study.

At the beginning of the script, type the following code:

```
subjects = [01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16]; % Replace with a list of
↳all of the subjects you wish to analyze

user = getenv('USER'); % Will return the username for OSX operating systems

for subject=subjects

subject = num2str(subject, '%02d');

if exist(['/home/wshao/BART_spm/sub-' subject '/func/sub-' subject '_task-
↳balloonanalogrisktask_run-01_bold.nii']) == 0
    display('Run 1 has not been unzipped; unzipping now')
    gunzip(['/home/wshao/BART_spm/sub-' subject '/func/sub-' subject '_task-
↳balloonanalogrisktask_run-01_bold.nii.gz'])
else
    display('Run 1 is already unzipped; doing nothing')
end

if exist(['/home/wshao/BART_spm/sub-' subject '/func/sub-' subject '_task-
↳balloonanalogrisktask_run-02_bold.nii']) == 0
    display('Run 2 has not been unzipped; unzipping now')
    gunzip(['/home/wshao/BART_spm/sub-' subject '/func/sub-' subject '_task-
↳balloonanalogrisktask_run-02_bold.nii.gz'])
else
    display('Run 2 is already unzipped; doing nothing')
end

if exist(['/home/wshao/BART_spm/sub-' subject '/func/sub-' subject '_task-
↳balloonanalogrisktask_run-03_bold.nii']) == 0
    display('Run 3 has not been unzipped; unzipping now')
    gunzip(['/home/wshao/BART_spm/sub-' subject '/func/sub-' subject '_task-
↳balloonanalogrisktask_run-03_bold.nii.gz'])
```

(continues on next page)

(continued from previous page)

```
else
    display('Run 3 is already unzipped; doing nothing')
end

if exist(['/home/wshao/BART_spm/sub-' subject '/anat/sub-' subject '_T1w.nii']) == 0
    display('Anatomical image has not been unzipped; unzipping now')
    gunzip(['/home/wshao/BART_spm/sub-' subject '/anat/sub-' subject '_T1w.nii.gz'])
else
    display('Anatomical image is already unzipped; doing nothing')
end
```

you need to change the directory `/home/wshao/BART_spm` to your subject directory, you can `cd` to the subject directory and type `pwd` to know the working directory

The above code does the following:

the variable `user` takes the value returned from the command `getenv('USER')`. This should return the username of the current user of the computer - in the current example, “ajahn”.

We then begin a for-loop that is initialized with the code for `subject=subjects`. This means that a new variable, “subject”, will assume the value of each consecutive entry in the array “subjects”. In other words, the first instance of the loop will assign the value “01” to `subject`; on the second instance, it will assign the value “02”, and so on, until the loop reaches the end of the array.

Since an array will strip any leading zeros, and since we need to convert the numbers in our array to a string, the “subject” variable is converted using the `num2str` command. The text “%02d” is string-formatting code indicating that the current value being converted from a number to a string should be zero-padded with as many zeros as needed until the number is two characters long.

The conditional statements look for whether the unzipped functional and anatomical files exist, and if they don’t, the files are unzipped using Matlab’s `gunzip` command.

Concatenating strings

Throughout the rest of the code that was generated when we saved the Batch module as a Matlab script, we will need to replace each instance of 03 with the string `subject`, and each instance of `wshao` (or whatever your username is) with the variable `user` that was defined above. This can be done using search and replace.

In the example code above, we used brackets to horizontally concatenate strings with variables. A line of code like the following:

```
['/home/wshao/BART_spm/sub-' subject '/anat/sub-' subject '_T1w.nii']
```

will concatenate the strings surrounded by single apostrophes with the variables. If the variable “subject” contains the value “03”, then the above code would expand to the following:

```
‘/home/wshao/BART_spm/sub-03/anat/sub-03_T1w.nii’
```

You will need to perform these substitutions for the rest of the script, taking care to use single apostrophes to set off the strings from the variables. Brackets will be required for this concatenation, even within the cells denoted by curly braces. (Cells are arrays that can contain several different data types, such as strings and numbers.)

Loading the Onset Files

The last part of the script we have to edit is the onset times. In this experiment, each subject had different onset times for each condition. If the timing files have already been converted to a different format, then you can create a variable that contains the timing information and insert it into the “onset” field for the stats module. For example, the following code found around line 134 of the Matlab script can be changed from this, which contains onset times specific to sub-03:

```
matlabbatch{9}.spm.stats.fmri_spec.ssess(1).cond(1).onset = [55.013
78.631
96.264
109.377
209.904
247.183
253.506
272.949
289.177
296.349
305.967
325.969
429.214
476.099
525.129
535.041
556.235
572.240];
```

To change like this:

```
data_cash_run1 = load(['/home/wshao/BART_spm/sub-' subject '/func/cash_run1.txt']);
matlabbatch{9}.spm.stats.fmri_spec.ssess(1).cond(1).onset = data_cash_run1(:,1);
```

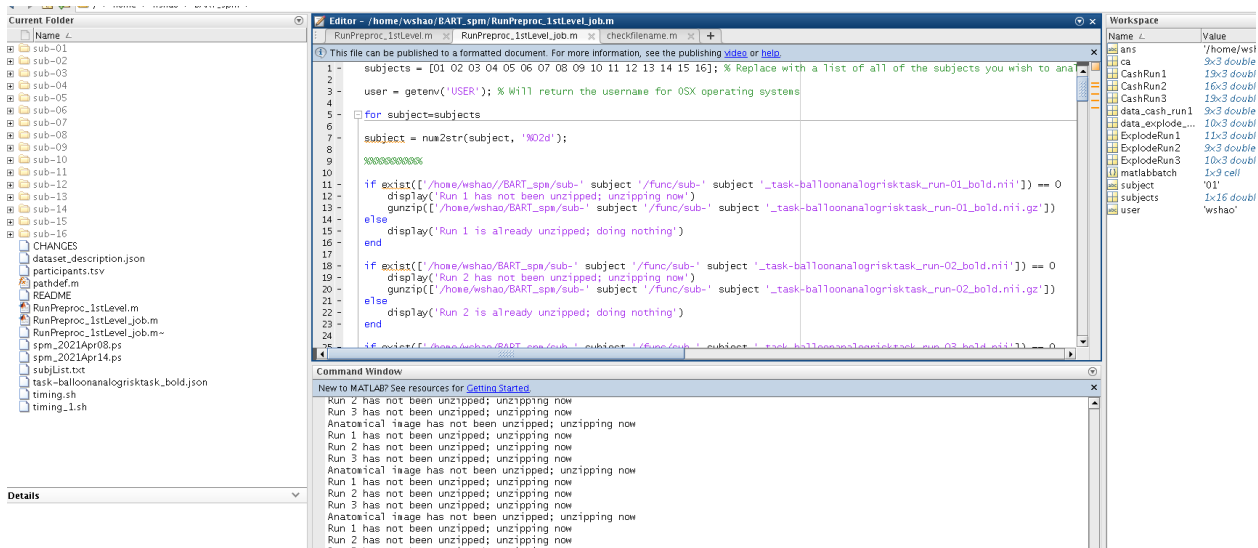
In which the variable `data_cash_run1` stores the onset times for the subject in the current loop, and then enters those numbers into the onset field. Note that the code `(:,1)` indicates that only the first column of the variable should be read, which contains the onset times. What's more, you need to do this 6 times, the cash and explode 3 times each.

Running the Script

When you have finished editing the script, save it and return to the Matlab terminal. You can then execute the script by typing:

```
BART_runprproc
```

You will then see windows pop up as each preprocessing and statistical module is run, similar to what you would see if you executed each module manually through the GUI.



Next Steps

The script will take a while(depends on the performance of the machine) to run for all the 16 subjects, When you are finished, we will examine the output.

you can find a copy of this full script from github page [here](#).

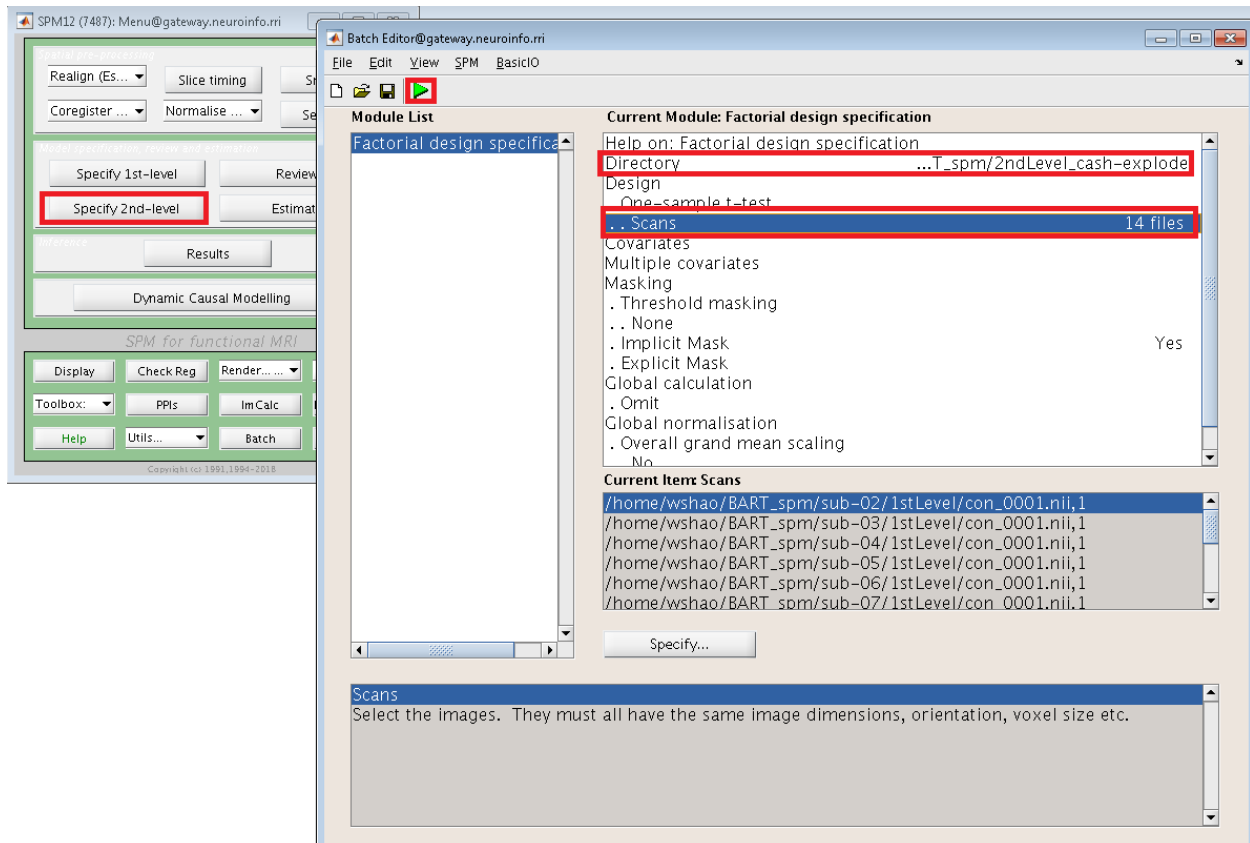
Group analysis

Since our aim in interpreting this dataset is to extrapolate the findings to the entire population from which the sample was taken. To put it another way, if we see changes in brain function in our study, can we assume that these changes will be observed in the general population? We'll run a group-level review to see if this is true (second-level analysis). This means that in SPM, we compute the standard error and mean for a contrast estimate before determining if the average estimate is statistically important. This group-level study will be conducted using a summary statistic method that ignores parameter estimate uncertainty and conducts a t-test on the mean parameter estimates from each subject.

Specifying the 2nd-Level Analysis

We need to create a new directory to store the 2nd-level findings after you've completed all of the 1st-level analyses. Navigate to the BART directory containing all of your subjects and type `mkdir 2ndLevel cash-explode` from the terminal.

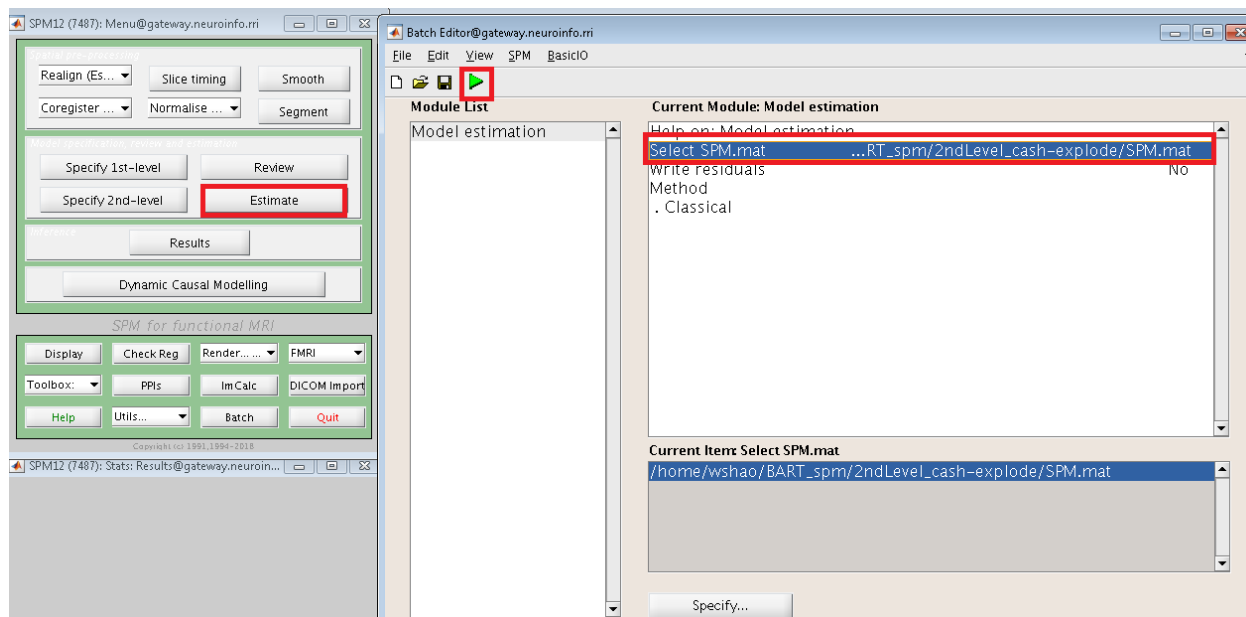
Select the button Specify 2nd-level from the SPM GUI. The default test will be a one-sample t-test, and there are only two fields to fill in: the output directory for the results, and the scans you'll be testing on - in other words, the contrast images generated during each 1st-level study. Select the 2ndLevel cash-explode folder we just created from the Directory field. Go to the 1stLevel directory of sub-02 and select the cash-explode contrast image, `con_0001.nii`, for the Scans sector. Select the `con_0001.nii` for each subject from the 1stLevel folders of the rest of other subjects. After you select all the images, click the green "Go" button when you've done selecting the `con_0001.nii` for all 14 subjects(exclude the sub-01 and sub-09).



Note: Remember we have set the 4 different contrasts from cash-explode, explode-cash, cash and explode. If you forget the contrast conditions, you can load the SPM.mat file for a sample subject via the Results button from SPM GUI and see which number contrast corresponds to the con images produced in each subject's folder.

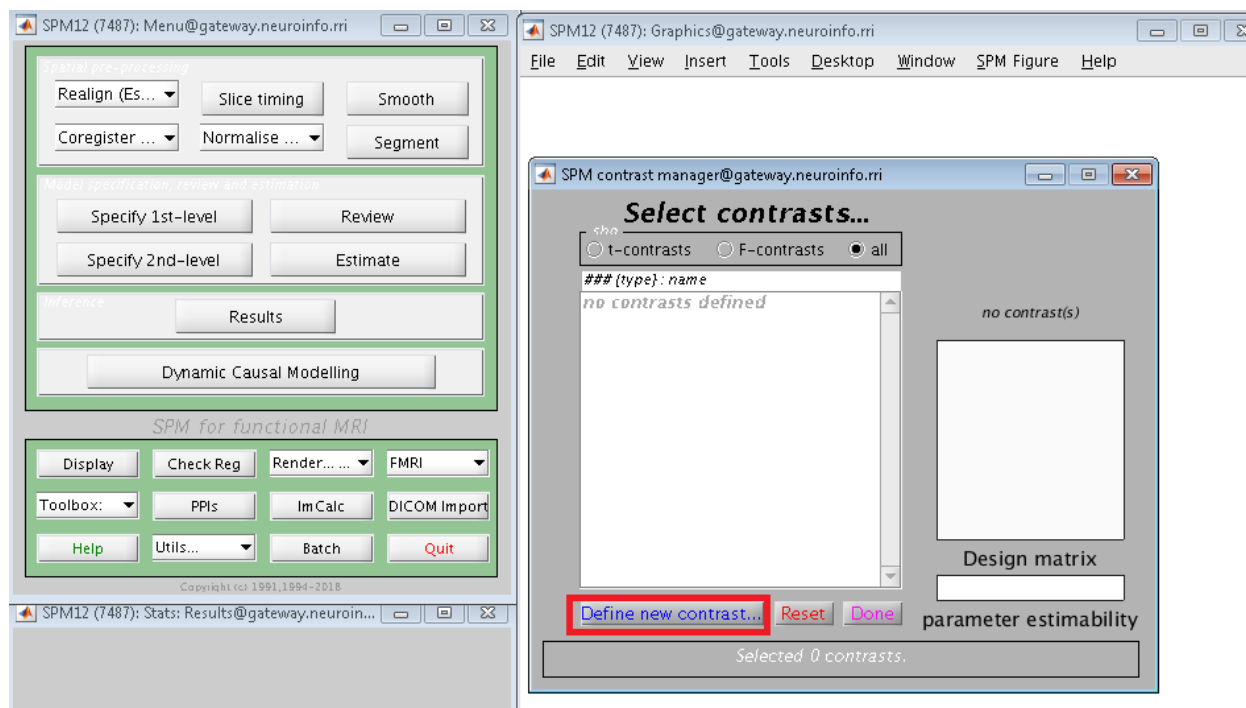
Estimating the 2nd-Level Analysis

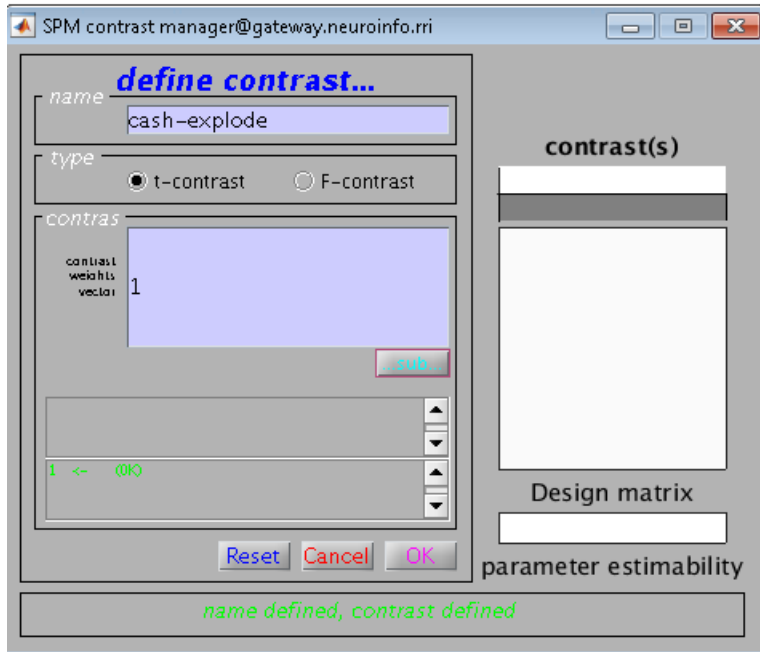
It will only take a second to specify the model. You'll need to estimate the model after it's done, just as we did with the first-level analyses. Click the Estimate button in the SPM GUI and the green "Go" button after selecting the SPM.mat file from the 2ndLevel Flanker directory you created.



Viewing the Results

We can now display the findings by selecting the **Results** button from the SPM GUI, just as we did with the 1st-level analyses. Click **Done** after selecting the SPM.mat file from the 2ndLevel cash-explode directory. Another contrasts window will appear, but with a small difference: This design matrix resembles a white box, while the first-level studies had a design matrix that included all of the regressors in the model. That means there's only one regressor to test: the mean activation over all of the individual contrast images used in the model. Click **Define new contrast**, name the contrast "cash-explode", and give it a contrast weight of 1. When you are finished, it should look like this:





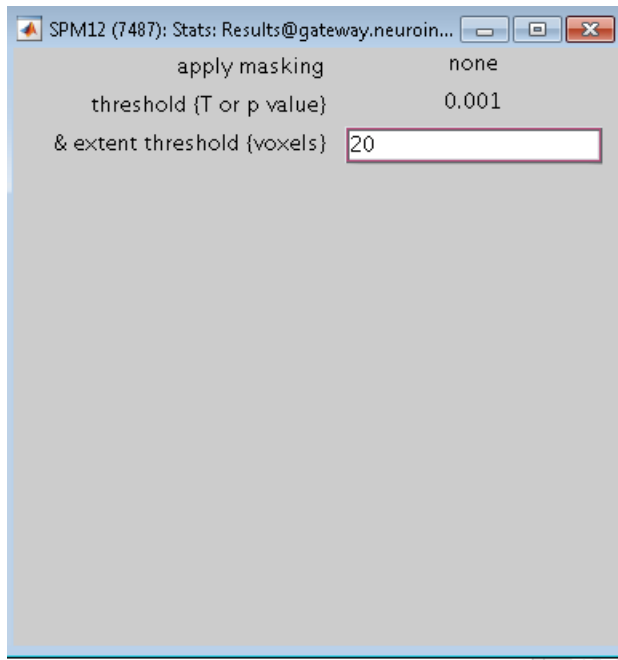
Click OK, and then click Done. You will be asked the same questions as before about masking, cluster-thresholding values, and cluster extent. For this group analysis, select the following:

```

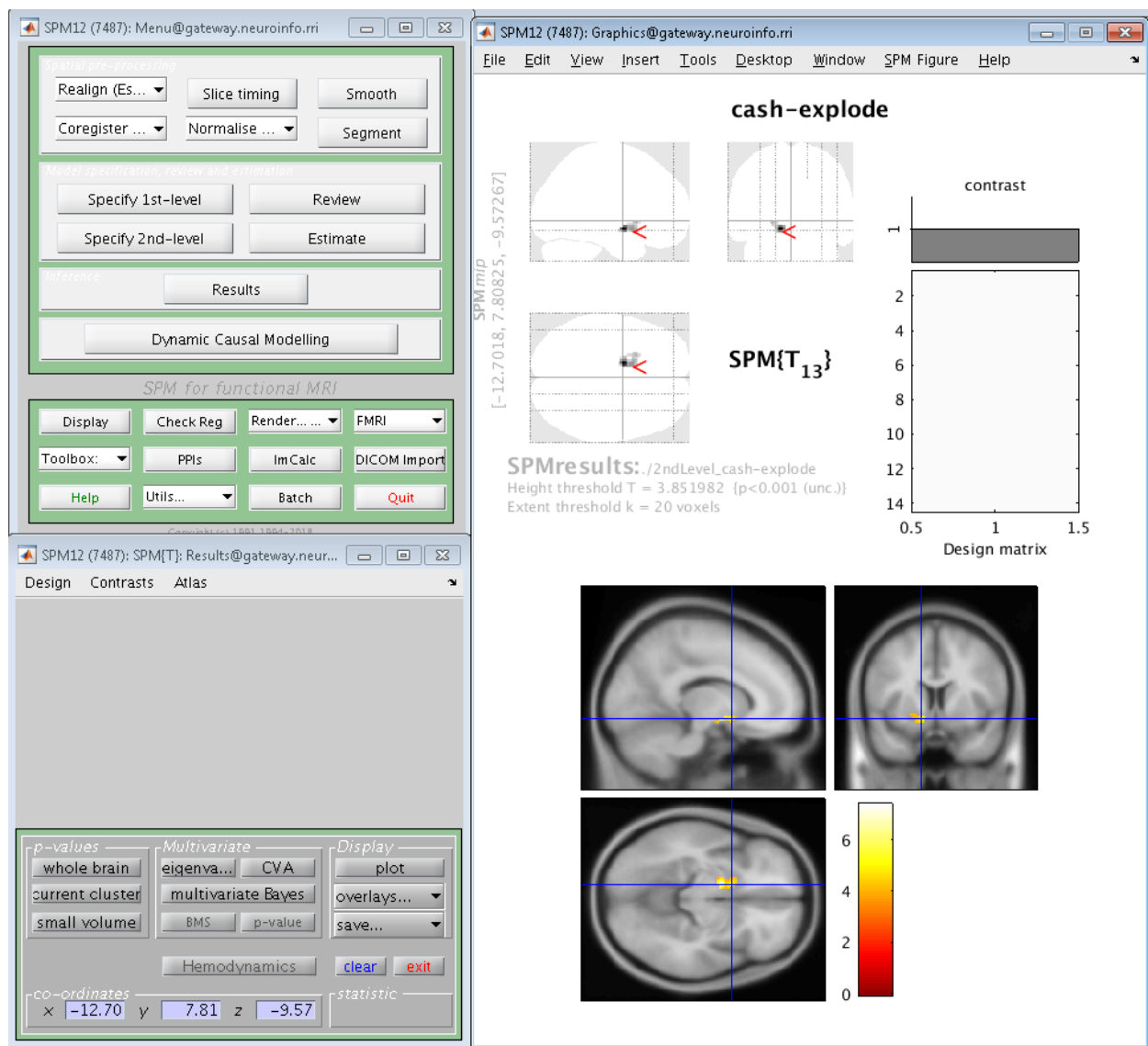
apply masking -> none
p value adjustment to control -> none
threshold {T or p value} -> 0.001
& extent threshold {voxels} -> 20

```

The image will be thresholded to only display clusters made up of individual voxels that each pass a 0.001 threshold.



When you are finished, you should see output like this, showing a significant cluster:



2nd-Level Results for cash and explode

If you're only interested in the areas where the cash and explode contrasts are vary significantly, the measures above are all you'll need to do. As an exercise, create a second-level result for the explode-cash, cash and explode contrasts respectively. If you examine the cash and explode results at the same threshold, do you see what you would expect given the cash-explode contrast that you viewed above?

Note: Just a hint for you, the cash contrasts are located in the con 0003.nii file for each subject, and the explode contrasts are located in the con 0004.nii file for each subject, using the same method as

above for deciding which contrast is located in the SPM.mat file. Starting with the cash contrast image, go to the SPM GUI's Specify 2nd-Level button and select the 2ndLevel cash folder for the Directory input. Select the con 0003.nii images for each subject using a method similar to the one described above.

When you have finished creating all of the second-level analyses, try the remaining exercises to test your understanding of what you have just learned.

ROI analysis

As we just completed a group-level analysis, and identified some regions of the brain show a significant difference under the condition of the experiment. Now, we are going to continue our learning with **region of analysis(ROI)**. This is called a **whole-brain** or **exploratory analysis**. When we doesn't have a hypothesis to test, these types of studies are beneficial.

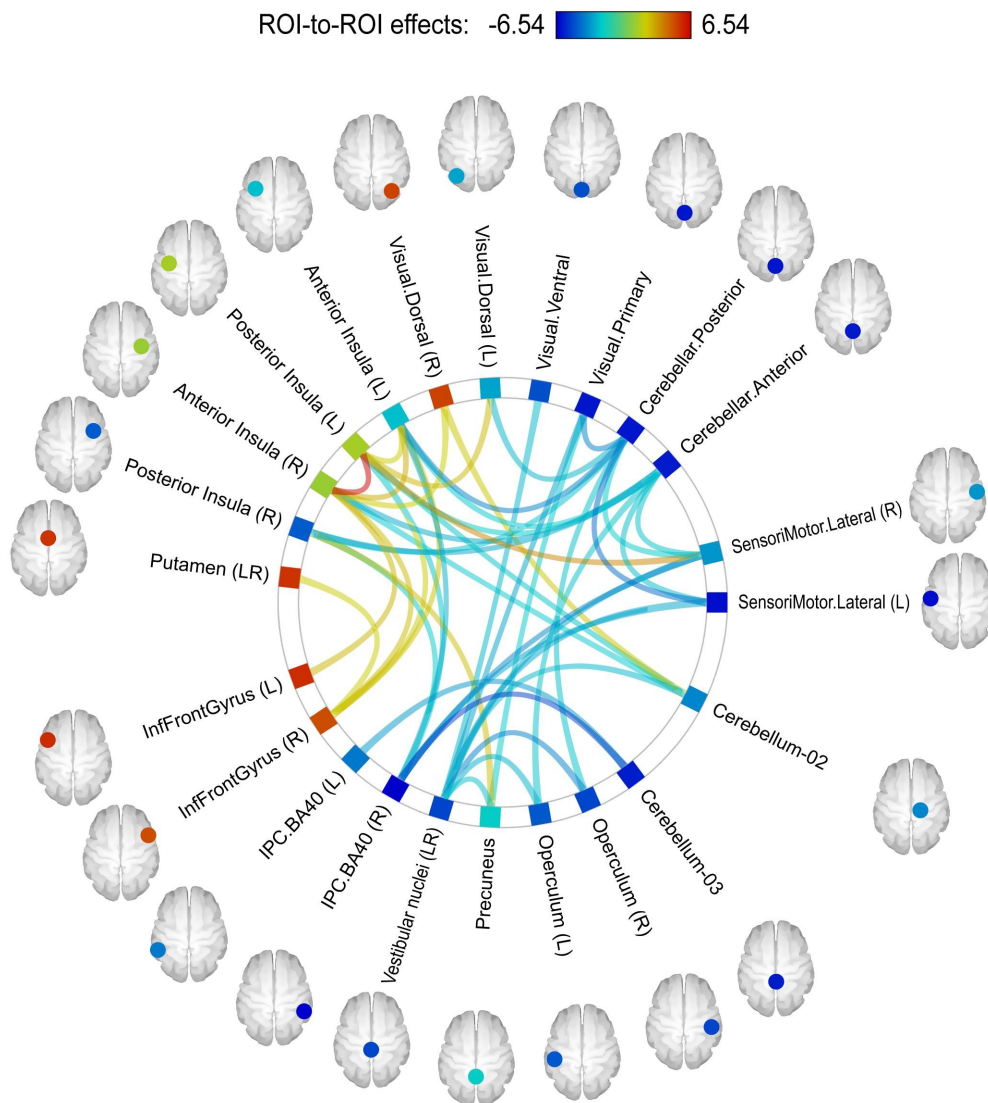
While a large number of studies have been run about a specific topic, we can begin to make more specific hypotheses about where we should find our results in the brain images. For instance, memory has been studied for many years, and many fMRI studies have been published about it using different paradigms that compare different memory tasks. In general, significant increases in the BOLD signal during various memories conditions are seen in a region of the brain known as the Hippocampus and medial temporal lobe. For this BART study, then, we could restrict our analysis to this region and only extract data from voxels within that region. This is known as a ROI analysis. In short, a general name for an analysis in which we choose to analyze a region selected before look at whole-brain results is called a confirmatory analysis.

In terms of BART study, Whole-brain maps can hide important details about the effects that we're studying. As you may find a significant effect of BART conditions, but the reason the effect is significant could come from a greater change of cash than explode, or because explode is much more negative than cash, or some combination of the two. The only way to determine what is driving the effect is with ROI analysis, and this is especially important when

Atlas

1.21 CONN introduction

CONN is a Matlab-based cross-platform imaging software for the computation, display, and analysis of functional connectivity in fMRI data in the resting state and during task. The CONN is known as a SPM toolbox, a suite of commands that is designed to be used as an add-on to SPM. you can find more details in [here](#). You can do load the The CONN from [here](#). Click on the download button to begin downloading the latest version.



This module will focus on the functional connectivity with the SPM-depend tool name CONN. In general, the correlation in BOLD signal between two distinct regions of the brain. This correlation can be analyzed when the subject is doing a task (i.e., task-based connectivity), or when the subject is at rest - relaxed and alert, but not doing any particular task (i.e., resting-state connectivity).

In the following tutorials, you will learn how to perform resting-state connectivity analyses on a dataset. We will use the CONN toolbox to run the analyses, which includes both creating correlation maps for each voxel of the brain, and generating connectomes that visualize the strength of the connectivity between different regions.

1.21.1 CONN with SPM

Before going on, you may want to work through the [SPM](#) if you are not familiar with SPM. You can check the previous main course SPM, it will introduce you to Matlab and SPM, which you will need to run the CONN toolbox. Note that SPM is a prerequisite for the CONN toolbox, meaning that you will need to download and install SPM before you can use CONN. And the rest of content in the SPM module aren't required, although a better understanding of how SPM preprocesses fMRI data will prepare you to use the CONN toolbox.

In order to download the SPM, please [click](#).

You can go [there](#) for more details about CONN

After you have successfully installed Matlab, SPM and CONN toolbox, now we can play it with a new dataset.

1.22 Language dataset

1.22.1 Exploring the Resting State Neural Activity in language

Individuals who speak more than one language have been found to enjoy a number of benefits not directly associated with the use of the languages themselves. One of these benefits is that bilingual individuals appear to develop symptoms of dementia 4-5 years later than comparable individuals who speak just one language. Studies on this topic, however, do not necessarily account for factors including if the individual learned their second language as a child or later in life, or their language proficiency. In an attempt to more carefully examine these variables, this study looks at structural and resting-state functional MRI scans of the default mode network, English and Spanish (where applicable) proficiency, language background, and demographics of young healthy adults who fall into one of three groups: early bilinguals, late bilinguals, and monolinguals.

Now, let's go [there](#) to download the data and save it as language in our home directory.

1.23 CONN_GUI

Now we have set up the software and the dataset, it is time for us to analyze the data!

Open the Matlab from the home directory, and type `conn`, wait for a few minutes, you will see the CONN_GUI

1.24 Models for Neuroimage

In this chapter, I'm going to introduce some statistical (machine) models that have been developed decades ago and have been applied in neuroimage. More importantly, I strongly encourage you to read this article for a brief historical view of [machine learning](#).

1.24.1 Computational Models overview

The goal of this page is to go focus on the key question of how to figure out what a model is telling us about the mind. The rest content of this chapter is to implement the models, the mechanics of our brain and behaviour.

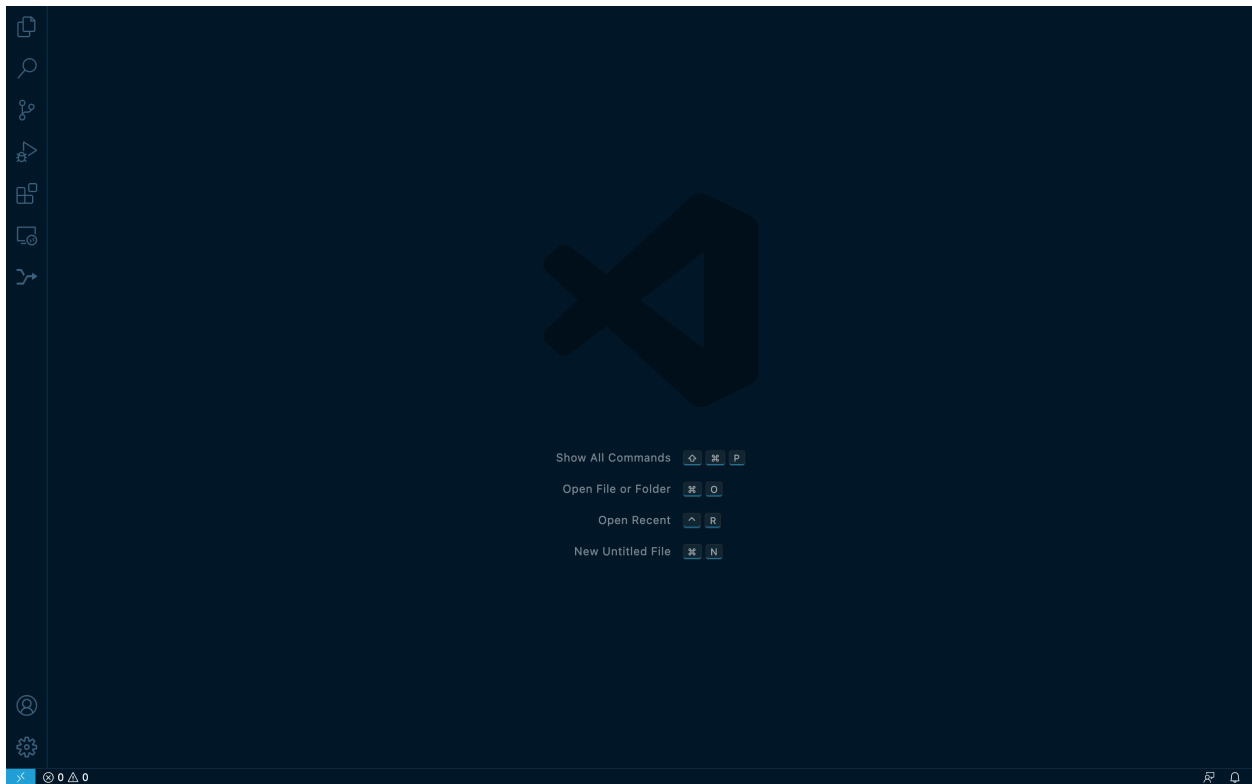
1.24.2 Visual studio code

What is visual studio code(VSC)? Acoording to Wikipedia, VSC is a source-code editor made by Microsoft for Windows, Linux and macOS. It has many features including debugging,syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. You can also install all kinds of extensions that allow additional function. VSC Visual Studio Code was ranked the most popular developer environment in Stack Overflow survey 2021.

You cna go to [HERE](#) to download the right version.

Click [HERE](#) for **set up** and **get started** as you follow the instructions

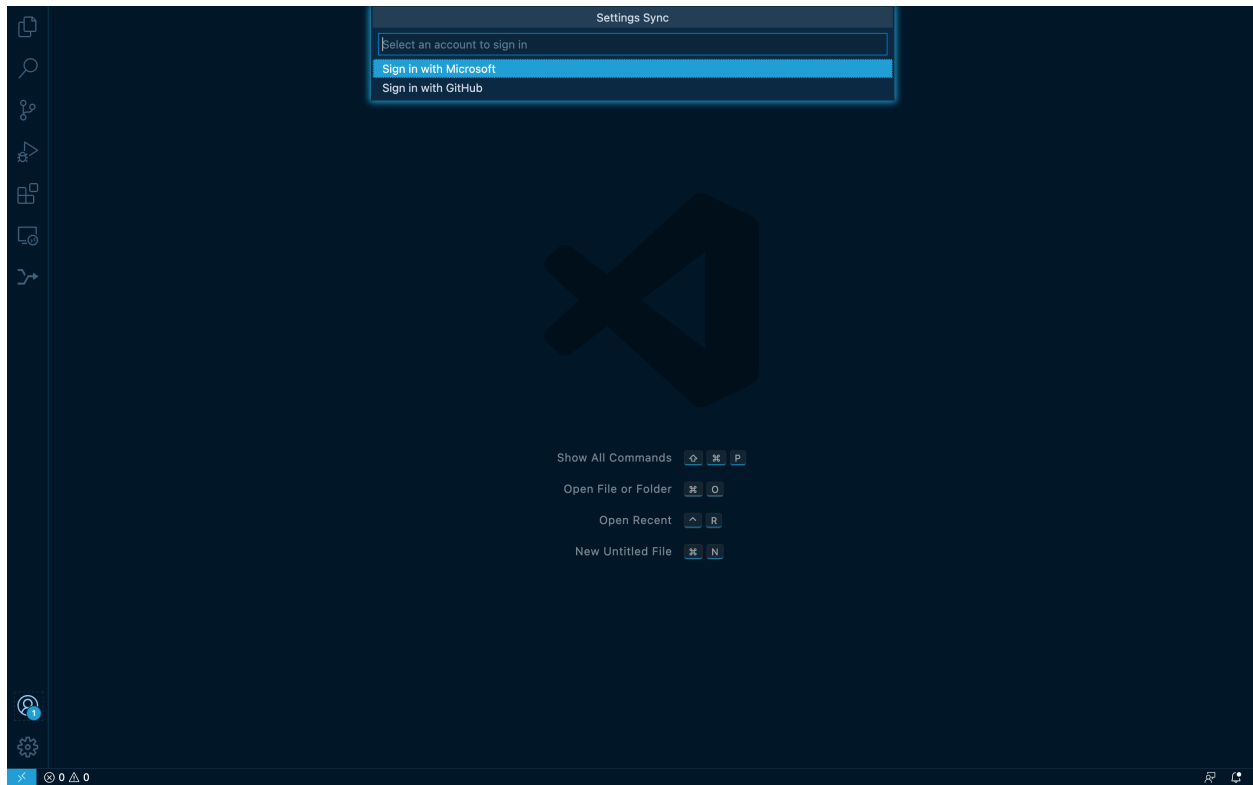
After everything is set up, you are supposed to see this when you click the VSC icon on your laptop.



Now, let's do some prework before we actually use VSC to analyze the data. One of the best functions of VSC is that you can connect VSC with you GitHub, therefore, you can get rid of what you have learned from **Appetizer 3** and combine git add/commit/push into simpler operation

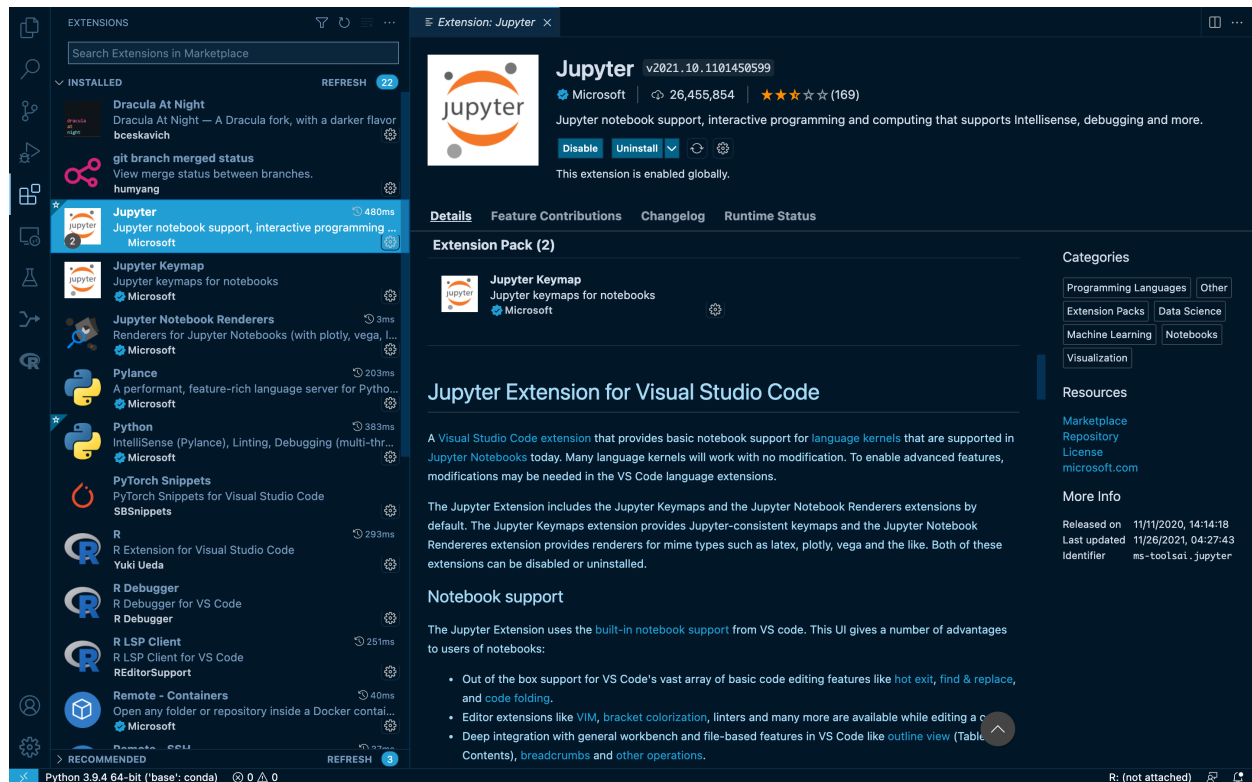
Sync VSC with your GitHub

Click the left bottom corner sign in to your Github account in order to Sync the information.



Add the extensions on VSC

You might want to add some extensions such as Python, R and jupyter notebook in your VSC so that we can implement the code under these languages.



1.24.3 Running GLM on python

Since we have the basic knowledge of the tool. Let's start the fun with running fMRI analysis code on python.

HCP dataset and Gambling project

HCP

The Human Connectome Project (HCP) is a data sharing initiative to address fundamental questions about human anatomy and variation related to neuron connectivity. It is based at Washington University and includes collaborations between University of Southern California, Harvard and Massachusetts General Hospital. The HCP collected scanned the brains of more than 1200 young men and women, mostly in their 20s, participants also completed questionnaires about their health and lives, an aerobic fitness walking test and cognitive tests.

The HCP dataset comprises task-based fMRI from a large sample of human subjects. The dataset includes time series data that has been preprocessed and spatially-downsampled by aggregating within 360 regions of interest. In order to use this dataset, please electronically sign the HCP data use terms at [ConnectomeDB](#). Although we only use the gambling sub dataset for this chapter, you can apply the code with different dataset once you learned the code.

Instructions for this dataset are on pp. 24-25, 36 and 50-51 of the [HcP Reference Manual](#). you are more than welcome to study this manual before the analysis

Gambling project

This task was adapted from the one developed by Delgado and Fiez (Delgado et al. 2000). Participants play a card guessing game where they are asked to guess the number on a mystery card (represented by a “?”) in order to win or lose money. Participants are told that potential card numbers range from 1-9 and to indicate if they think the mystery card number is more or less than 5 by pressing one of two buttons on the response box. Feedback is the number on the card (generated by the program as a function of whether the trial was a reward, loss or neutral trial) and either: 1) a green up arrow with “\$1” for reward trials, 2) a red down arrow next to “-\$0.50 for loss trials; or 3) the number 5 and a gray double headed arrow for neutral trials. The “?” is presented for up to 1500 ms (if the participant responds before 1500 ms, a fixation cross is displayed for the remaining time), following by feedback for 1000 ms. There is a 1000 ms with a “+” presented on the screen. The task is presented in blocks of 8 trials that are either mostly reward (6 reward trials pseudo randomly interleaved with either 1 neutral and 1 loss trial, 2 neutral trials, or 2 loss trials) or mostly loss (6 loss trials pseudorandomly interleaved with either 1 neutral and 1 reward trial, 2 neutral trials, or 2 reward trials). In each of the two runs, there are 2 mostly reward and 2 mostly loss blocks, interleaved with 4 fixation blocks (15 seconds each).

Some Credits of this chapter go to [Neuromatch Academy](#).

Running the analysis

Now, let's open the VSC and the jupyter book. You also can see and run all the code from Colaboratory [here](#). each of them has its own advantage, depending on your preference.

Once you are familiar with the dataset, we can start by loading the libraries:

```
import os
import numpy as np
import matplotlib.pyplot as plt
import nibabel as nib
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

Next, let's setting the figure for the visualization:

```
#title Figure settings

%matplotlib inline

%config InlineBackend.figure_format = 'retina'

plt.style.use("https://raw.githubusercontent.com/NeuromatchAcademy/course-content/master/
↪nma.mplstyle")
```

Once we set up the figure, set up with data and environment would be the next:

```
# The download cells will store the data in nested directories starting here:
HCP_DIR = "./hcp"
if not os.path.isdir(HCP_DIR):
    os.mkdir(HCP_DIR)

# The data shared for NMA projects is a subset of the full HCP dataset
N_SUBJECTS = 339
```

(continues on next page)

(continued from previous page)

```

# The data have already been aggregated into ROIs from the Glasser parcellation
N_PARCELS = 360

# The acquisition parameters for all tasks were identical
TR = 0.72 # Time resolution, in seconds

# The parcels are matched across hemispheres with the same order
HEMIS = ["Right", "Left"]

# Each experiment was repeated twice in each subject
N_RUNS = 2

# There are 7 tasks in the dataset. Each has a number of 'conditions'. and we are only
↪ use the gambling data

EXPERIMENTS = {
    'GAMBLING' : {'runs': [11,12], 'cond':['loss','win','neut']}
}

# Load all subjects all subjects:
subjects = range(N_SUBJECTS)

```

For more information about the files in the gambling project, go 45-47 page of [HCP Reference Manual](#).

Download the data, the task data are shared in different files, but we will unpack them into the same directory structure:

```

fname = "hcp_task.tgz"
if not os.path.exists(fname):
!wget -qO $fname https://osf.io/s4h8j/download/
!tar -xzf $fname -C $HCP_DIR --strip-components=1

```

Loading region information. Downloading this dataset will create the `regions.npy` file, which contains the region name and network assignment for each parcel:

```

regions = np.load(f"{HCP_DIR}/regions.npy").T
region_info = dict(
    name=regions[0].tolist(),
    network=regions[1],
    hemi=['Right']*int(N_PARCELS/2) + ['Left']*int(N_PARCELS/2),
)

```

Loading the time series from a single subject and a single run, and one for loading an EV file for each task. An EV file (EV: Explanatory Variable) describes the task experiment in terms of stimulus onset, duration, and amplitude. These can be used to model the task time series data:

```

def load_single_timeseries(subject, experiment, run, remove_mean=True):
    #Load timeseries data for a single subject and single run.

Args:
    subject (int):      0-based subject ID to load
    experiment (str):    Name of experiment
    run (int):          0-based run index, across all tasks

```

(continues on next page)

(continued from previous page)

```

    remove_mean (bool): If True, subtract the parcel-wise mean (typically the mean BOLD
    ↪ signal is not of interest)

Returns
    ts (n_parcel x n_timepoint array): Array of BOLD data values

bold_run = EXPERIMENTS[experiment]['runs'][run]
bold_path = f"{HCP_DIR}/subjects/{subject}/timeseries"
bold_file = f"bold{bold_run}_Atlas_MSMA11_Glasser360Cortical.npy"
ts = np.load(f"{bold_path}/{bold_file}")
if remove_mean:
    ts -= ts.mean(axis=1, keepdims=True)
return ts

def load_evs(subject, experiment, run):
#Load EVs (explanatory variables) data for one task experiment.

Args:
    subject (int): 0-based subject ID to load
    experiment (str) : Name of experiment

Returns
    evs (list of lists): A list of frames associated with each condition

frames_list = []
task_key = 'tfMRI_'+experiment+'_'+['RL', 'LR'][run]
for cond in EXPERIMENTS[experiment]['cond']:
    ev_file = f"{HCP_DIR}/subjects/{subject}/EVs/{task_key}/{cond}.txt"
    ev_array = np.loadtxt(ev_file, ndmin=2, unpack=True)
    ev = dict(zip(["onset", "duration", "amplitude"], ev_array))
    # Determine when trial starts, rounded down
    start = np.floor(ev["onset"] / TR).astype(int)
    # Use trial duration to determine how many frames to include for trial
    duration = np.ceil(ev["duration"] / TR).astype(int)
    # Take the range of frames that correspond to this specific trial
    frames = [s + np.arange(0, d) for s, d in zip(start, duration)]
    frames_list.append(frames)

return frames_list

```

OK, let's load the timeseries data for the GAMBLING experiment from a single subject and a single run:

```

my_exp = 'GAMBLING'
my_subj = 0
my_run = 1
data = load_single_timeseries(subject=my_subj, experiment=my_exp, run=my_run, remove_
    ↪ mean=True)
#print the data shape
print(data.shape)

```

As you can see the time series data contains 284 time points in 360 regions of interest (ROIs). Now in order to understand how to model these data, we need to relate the time series to the experimental manipulation. This is described by the

EV files. Let us load the EVs for this experiment:

```
evs = load_evs(subject=my_subj, experiment=my_exp, run=my_run)
# lets visualize the loss regressor
los_reg = np.zeros(253)
win_reg = np.zeros(253)
net_reg = np.zeros(253)
res_reg = np.ones(253)

for id in range(0, len(evs[0])):
    los_reg[evs[0][id]] = 1
# lets visualize the win regressor
for id in range(0, len(evs[1])):
    win_reg[evs[1][id]] = 1
# lets visualize the neut regressor
for id in range(0, len(evs[2])):
    net_reg[evs[2][id]] = 1
#let screate the resting phase regressor
for id in range(0, len(evs[0])):
    res_reg[evs[0][id]] = 0
for id in range(0, len(evs[1])):
    res_reg[evs[1][id]] = 0
for id in range(0, len(evs[2])):
    res_reg[evs[2][id]] = 0
```

Let's take a look at the regressor:

```
fig, axs = plt.subplots(2, 2, figsize=[15, 6])
axs[0,0].plot(los_reg, 'k')
axs[0, 0].set_title('Loss Regressor')
axs[0,1].plot(win_reg, 'g')
axs[0, 1].set_title('Win Regressor')
axs[1,0].plot(net_reg, 'r')
axs[1, 0].set_title('Neutral Regressor')
axs[1,1].plot(res_reg, 'b')
axs[1, 1].set_title('Resting Regressor')
```

Next, one of the most important functions in fMRI, general linear model:

```
def glm(data, reg):
    constant = np.ones(253)
    X = np.vstack((reg, constant)).T
    y = data

    # Calculate the dot product of the transposed design matrix and the design matrix
    # and invert the resulting matrix.
    tmp = np.linalg.inv(X.transpose().dot(X))

    # Now calculate the dot product of the above result and the transposed design matrix
    tmp = tmp.dot(X.transpose())

    # Pre-allocate variables
    beta = np.zeros((y.shape[0], X.shape[1]))
    e = np.zeros(y.shape)
```

(continues on next page)

(continued from previous page)

```

model = np.zeros(y.shape)
r      = np.zeros(y.shape[0])

# Find beta values for each voxel and calculate the model, error and the correlation_
↪coefficients
    for i in range(y.shape[0]):
        beta[i] = tmp.dot(y[i,:].transpose())
        model[i] = X.dot(beta[i])
        e[i]     = (y[i,:] - model[i])
        r[i]     = np.sqrt(model[i].var()/y[i,:].var())

    return beta, model, e, r

```

OK, now, let's apply the function into our data for one example:

```

X = np.vstack((los_reg, win_reg, net_reg, res_reg)).T
y = data
constant = np.ones(253)
c = np.vstack(constant)

# Calculate the dot product of the transposed design matrix and the design matrix
# and invert the resulting matrix.

tmp = np.linalg.inv(X.transpose().dot(X))

# Now calculate the dot product of the above result and the transposed design matrix

tmp = tmp.dot(X.transpose())

# Pre-allocate variables
beta = np.zeros((y.shape[0], X.shape[1]))
e     = np.zeros(y.shape)
model = np.zeros(y.shape)
r     = np.zeros(y.shape[0])

```

So far so good, let's apply the model for all the subjects, all runs and all condition:

```

# Lets bring together the previous steps all in one for running through all subjects,
↪all runs
# Create the beta for 4 conditions
betas_los = np.zeros((2, 360, 2, 339))
betas_win = np.zeros((2, 360, 2, 339))
betas_net = np.zeros((2, 360, 2, 339))
betas_res = np.zeros((2, 360, 2, 339))

# Create R for 4 conditions
r_los = np.zeros((1,360,2,339))
r_win = np.zeros((1,360,2,339))
r_net = np.zeros((1,360,2,339))
r_res = np.zeros((1,360,2,339))
for sub_id in subjects:

```

(continues on next page)

(continued from previous page)

```

my_exp = 'GAMBLING'
my_subj = sub_id
for run in [0,1]:
    my_run = run
    #load data
    data = load_single_timeseries(subject=my_subj,experiment=my_exp,run=my_run,
    ↪remove_mean=True)
    # load the evs and create regressors
    evs = load_evs(subject=my_subj, experiment=my_exp,run=my_run)
    los_reg = np.zeros(253)
    win_reg = np.zeros(253)
    net_reg = np.zeros(253)
    res_reg = np.ones(253)
    #visualzie the loss regressor
    for id in range(0,len(evs[0])): los_reg[evs[0][id]] = 1
    # lets visualize the win regressor
    for id in range(0,len(evs[1])): win_reg[evs[1][id]] = 1
    # lets visualize the neutral regressor
    for id in range(0,len(evs[2])): net_reg[evs[2][id]] = 1
    #let create the resting phase regressor
    for id in range(0,len(evs[0])): res_reg[evs[0][id]] = 0
    for id in range(0,len(evs[1])): res_reg[evs[1][id]] = 0
    for id in range(0,len(evs[2])): res_reg[evs[2][id]] = 0
    #let create the model structure for all
    betas_los_tmp, model_los_tmp, e_los_tmp, r_los_tmp = glm(data, los_reg)
    betas_win_tmp, model_win_tmp, e_win_tmp, r_win_tmp = glm(data, win_reg)
    betas_net_tmp, model_net_tmp, e_net_tmp, r_net_tmp = glm(data, net_reg)
    betas_res_tmp, model_res_tmp, e_res_tmp, r_res_tmp = glm(data, res_reg)

# transfer the r data strucrture
r_los[:, :, run, sub_id] = r_los_tmp
r_win[:, :, run, sub_id] = r_win_tmp
r_net[:, :, run, sub_id] = r_net_tmp
r_res[:, :, run, sub_id] = r_res_tmp

# transfer the beta data strucrture
betas_los[:, :, run, sub_id] = betas_los_tmp.T
betas_win[:, :, run, sub_id] = betas_win_tmp.T
betas_net[:, :, run, sub_id] = betas_net_tmp.T
betas_res[:, :, run, sub_id] = betas_res_tmp.T

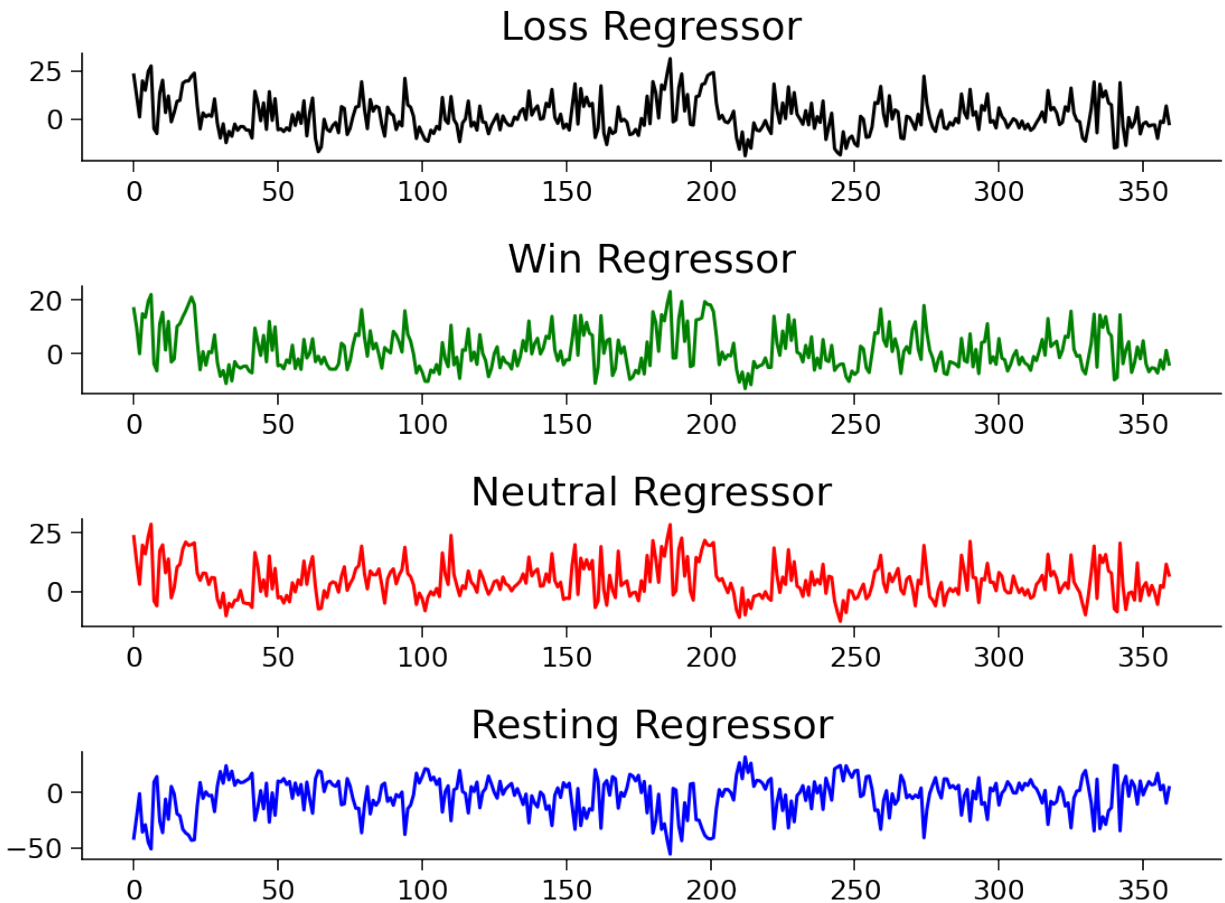
# mean value of beta
betas_avg_sub_run_los = betas_los.mean(axis = 2).mean(axis = 2)
betas_avg_sub_run_win = betas_win.mean(axis = 2).mean(axis = 2)
betas_avg_sub_run_net = betas_net.mean(axis = 2).mean(axis = 2)
betas_avg_sub_run_res = betas_res.mean(axis = 2).mean(axis = 2)

# mean value of r
r_avg_sub_run_los = r_los.mean(axis = 2).mean(axis = 2)
r_avg_sub_run_win = r_win.mean(axis = 2).mean(axis = 2)
r_avg_sub_run_net = r_net.mean(axis = 2).mean(axis = 2)
r_avg_sub_run_res = r_res.mean(axis = 2).mean(axis = 2)

```

Now, let's plot all the output! Start with the mean beta value:

```
# plot the mean beta
fig, axs = plt.subplots(4)
axs[0].plot(betas_avg_sub_run_los[0,:], 'k')
axs[0].set_title('Loss Regressor')
axs[1].plot(betas_avg_sub_run_win[0,:], 'g')
axs[1].set_title('Win Regressor')
axs[2].plot(betas_avg_sub_run_net[0,:], 'r')
axs[2].set_title('Neutral Regressor')
axs[3].plot(betas_avg_sub_run_res[0,:], 'b')
axs[3].set_title('Resting Regressor')
```



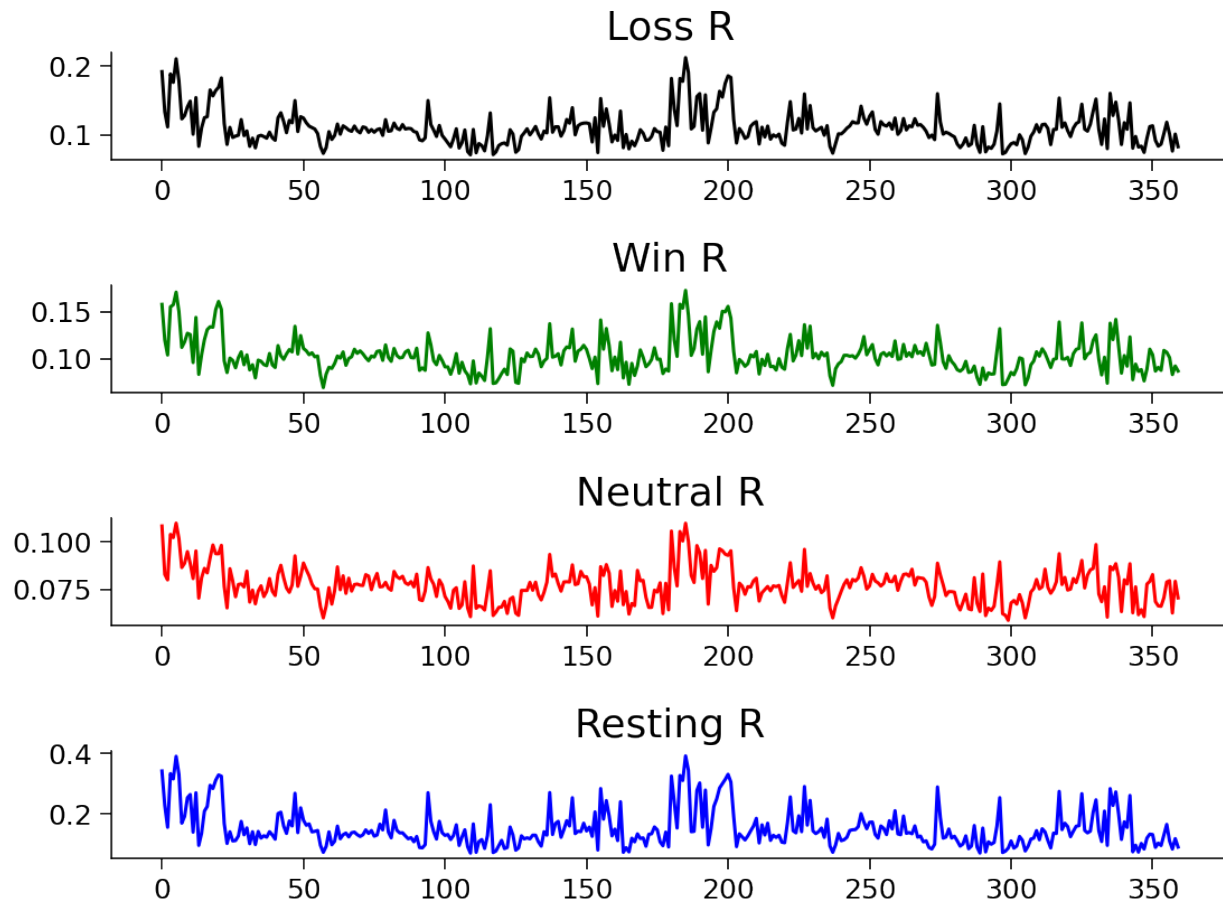
Then, mean value of R:

```
# plot the mean r
fig, axs = plt.subplots(4)
axs[0].plot(r_avg_sub_run_los[0,:], 'k')
axs[0].set_title('Loss R')
axs[1].plot(r_avg_sub_run_win[0,:], 'g')
axs[1].set_title('Win R')
axs[2].plot(r_avg_sub_run_net[0,:], 'r')
axs[2].set_title('Neutral R')
axs[3].plot(r_avg_sub_run_res[0,:], 'b')
```

(continues on next page)

(continued from previous page)

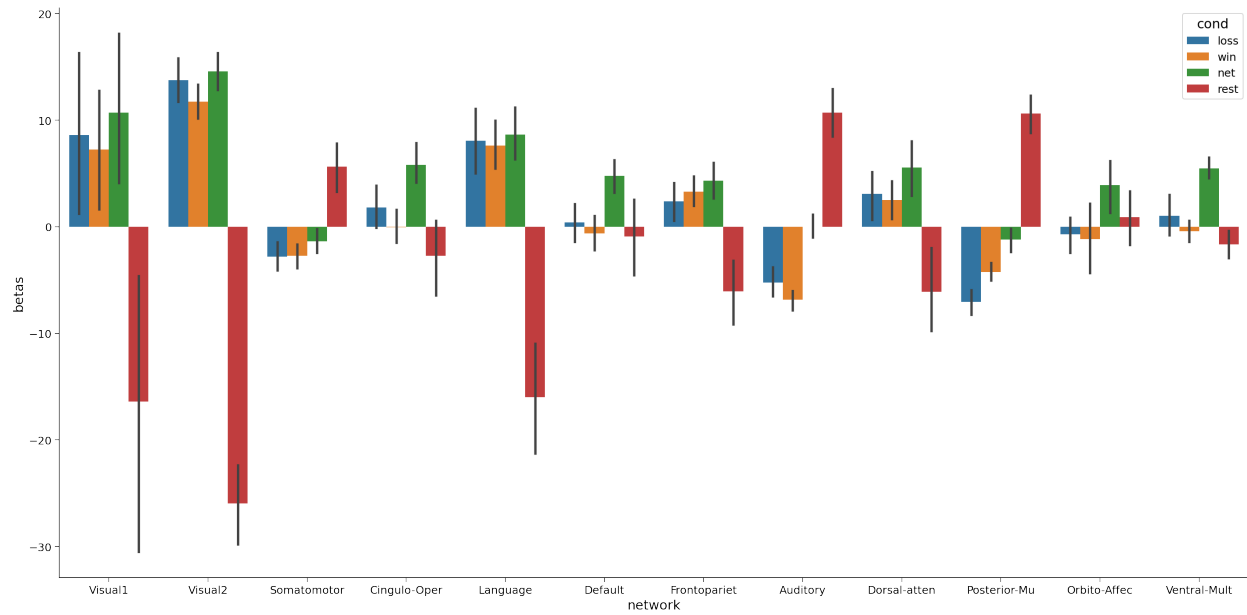
```
axs[3].set_title('Resting R')
```



Remember that we have 360 ROI and these ROIs become 12 networks, let's add the network component:

```
# plot the mean beta based on the network and compare with 4 conditions
df_beta_2 = pd.DataFrame({'betas' : np.hstack((betas_avg_sub_run_los[0,:], betas_avg_
↳ sub_run_win[0,:], betas_avg_sub_run_net[0,:], betas_avg_sub_run_res[0,:])),
    'cond' : np.hstack((['loss']*360, ['win']*360, ['net']*360, ['rest
↳ ']*360)),
    'network': np.hstack((region_info['network'], region_info['network'],
↳ region_info['network'], region_info['network'])),
    'name' : np.hstack((region_info['name'], region_info['name'],
↳ region_info['name'], region_info['name'])),
    'hemi' : np.hstack((region_info['hemi'], region_info['hemi'],
↳ region_info['hemi'], region_info['hemi']))
})

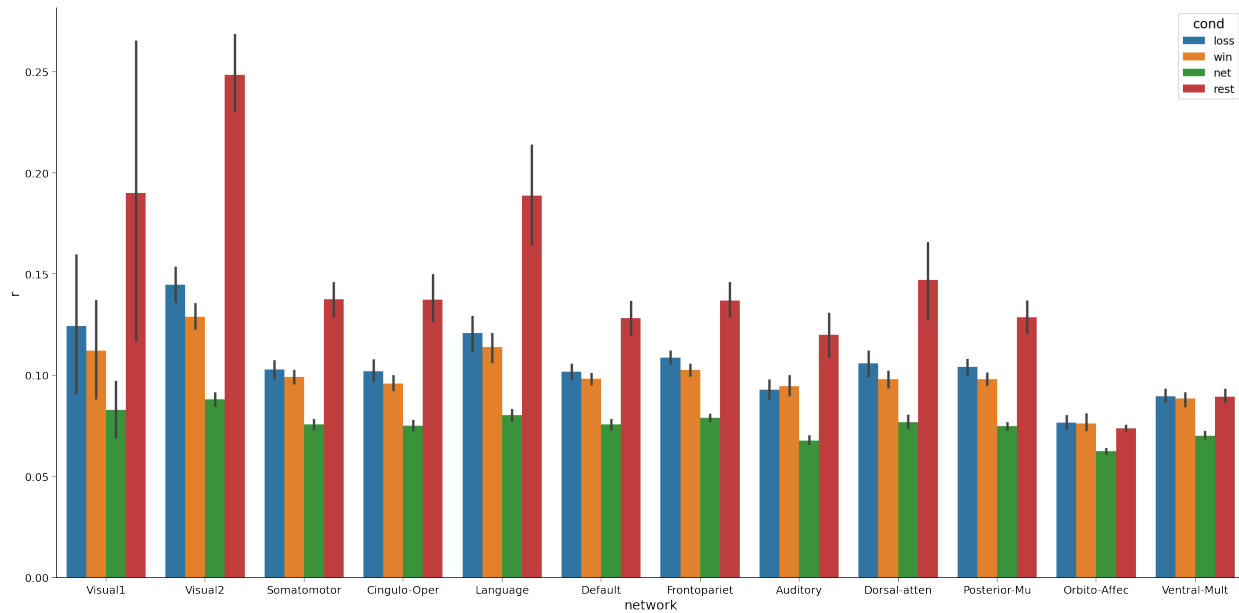
fig, (ax1)= plt.subplots(1,1, figsize = (20,10))
sns.barplot(x='network', y='betas', data=df_beta_2 , hue='cond', ax=ax1)
#sns.barplot(x='network', y='betas', data=df_beta_2 , hue='hemi', ax=ax2)
```



R value with network:

```
# plot the mean r based on the network and compare with 4 conditions
df_r = pd.DataFrame({'r' : np.hstack((r_avg_sub_run_los[0,:], r_avg_sub_run_win[0,:], r_avg_sub_run_net[0,:], r_avg_sub_run_res[0,:])),
                    'cond' : np.hstack((['loss']*360, ['win']*360, ['net']*360, ['rest']*360)),
                    'network': np.hstack((region_info['network'], region_info['network'], region_info['network'], region_info['network'])),
                    'name' : np.hstack((region_info['name'], region_info['name'], region_info['name'], region_info['name'])),
                    'hemi' : np.hstack((region_info['hemi'], region_info['hemi'], region_info['hemi'], region_info['hemi']))})

fig, (ax1)= plt.subplots(1,1, figsize = (20,10))
sns.barplot(x='network', y='r', data=df_r, hue='cond', ax=ax1)
#sns.barplot(x='network', y='r', data=df_r, hue='hemi', ax=ax2)
```

Now, let's make a group contrast with beta and r value so we can really know the brain activity on different condition:

```
def average_frames(be, evs, experiment, cond):
    idx = EXPERIMENTS[experiment]['cond'].index(cond)
    return np.mean(np.concatenate([np.mean(data[:, evs[idx][i]], axis=1, keepdims=True) for
    i in range(len(evs[idx]))]), axis=-1), axis=1)

loss_activity = average_frames(data, evs, my_exp, 'loss')
win_activity = average_frames(data, evs, my_exp, 'win')

#change the data structure and calculate the contrast map to fit in the brain image
loss_beta = betas_avg_sub_run_loss[0,:]
win_beta = betas_avg_sub_run_win[0,:]

contrast_beta = loss_beta - win_beta # difference between loss and win in average_
beta

los_r = r_avg_sub_run_loss.T
win_r = r_avg_sub_run_win.T

# contrast_r = los_r - win_r # difference between left and right hand movement
contrast_r = win_r - los_r
```

Create group contrast map:

```
group_contrast = 0
for s in subjects:
    for r in [0,1]:
        data = load_single_timeseries(subject=s, experiment=my_exp, run=r, remove_mean=True)
        evs = load_evs(subject=s, experiment=my_exp, run=r)

        loss_activity = average_frames(data, evs, my_exp, 'loss')
        win_activity = average_frames(data, evs, my_exp, 'win')
```

(continues on next page)

(continued from previous page)

```

contrast      = loss_activity-win_activity
group_contrast      += contrast

group_contrast /= (len(subjects)*2) # remember: 2 sessions per subject

```

Finally, let's plot the brain:

```

# This uses the nilearn package
!pip install nilearn --quiet
from nilearn import plotting, datasets

# loading the atlas
fname = f"{HCP_DIR}/atlas.npz"
if not os.path.exists(fname):
    !wget -qO $fname https://osf.io/j5kuc/download
with np.load(fname) as dobj:
    atlas = dict(**dobj)

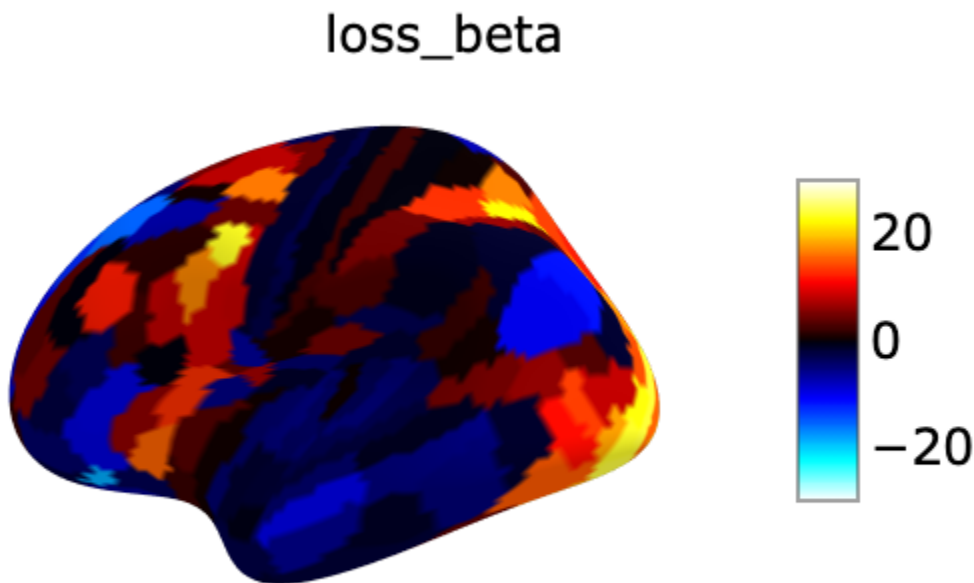
```

los_beta:

```

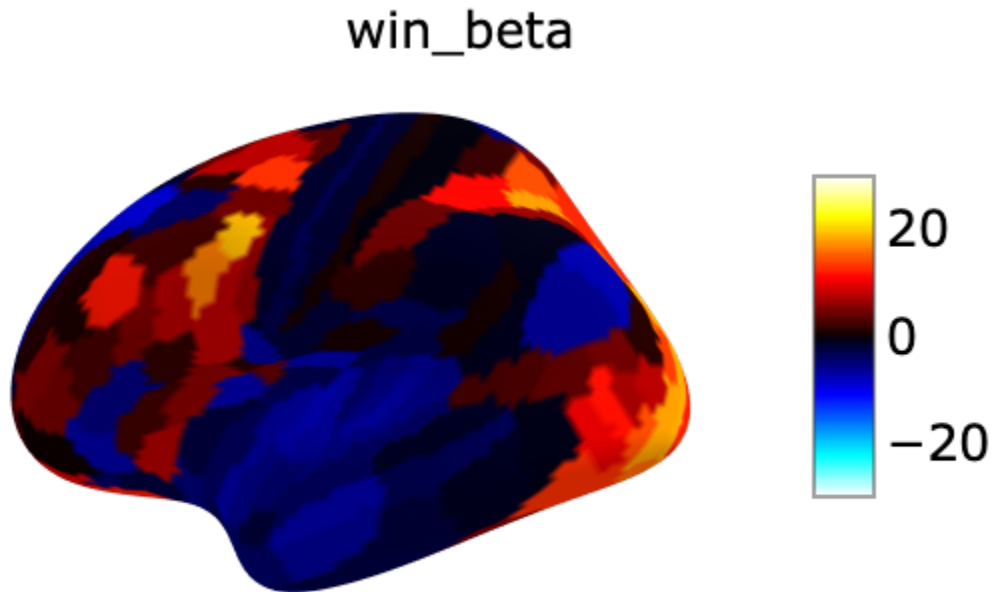
fsaverage = datasets.fetch_surf_fsaverage()
surf_contrast = betas_avg_sub_run_los[0,:][atlas["labels_L"]]
plotting.view_surf(fsaverage['infl_left'],
                   surf_contrast,
                   vmax=30,title='loss_beta')

```



win_beta:

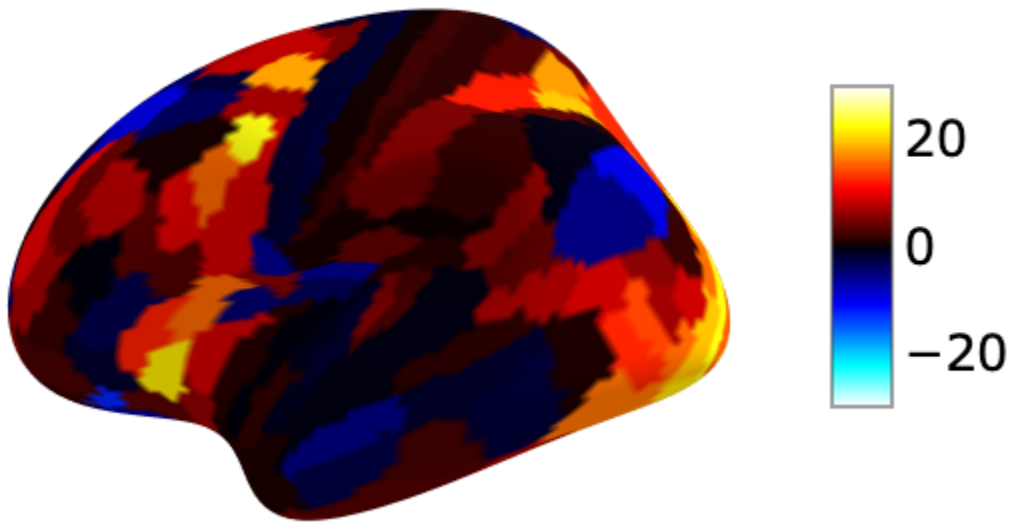
```
fsaverage = datasets.fetch_surf_fsaverage()
surf_contrast = betas_avg_sub_run_win[0,:][atlas["labels_L"]]
plotting.view_surf(fsaverage['infl_left'],
                   surf_contrast,
                   vmax=30,title='win_beta')
```



net_beta:

```
fsaverage = datasets.fetch_surf_fsaverage()
surf_contrast = betas_avg_sub_run_net[0,:][atlas["labels_L"]]
plotting.view_surf(fsaverage['infl_left'],
                   surf_contrast,
                   vmax=30,title='neutral_beta')
```

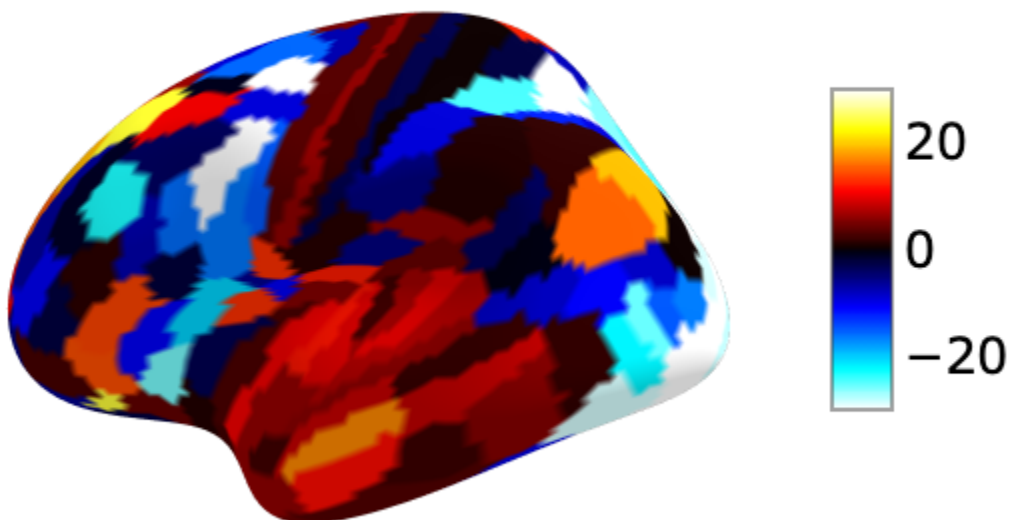
neutral_beta



res_beta:

```
fsaverage = datasets.fetch_surf_fsaverage()
surf_contrast = betas_avg_sub_run_res[0,:][atlas["labels_L"]]
plotting.view_surf(fsaverage['infl_left'],
                   surf_contrast,
                   vmax=30, title='resting_beta')
```

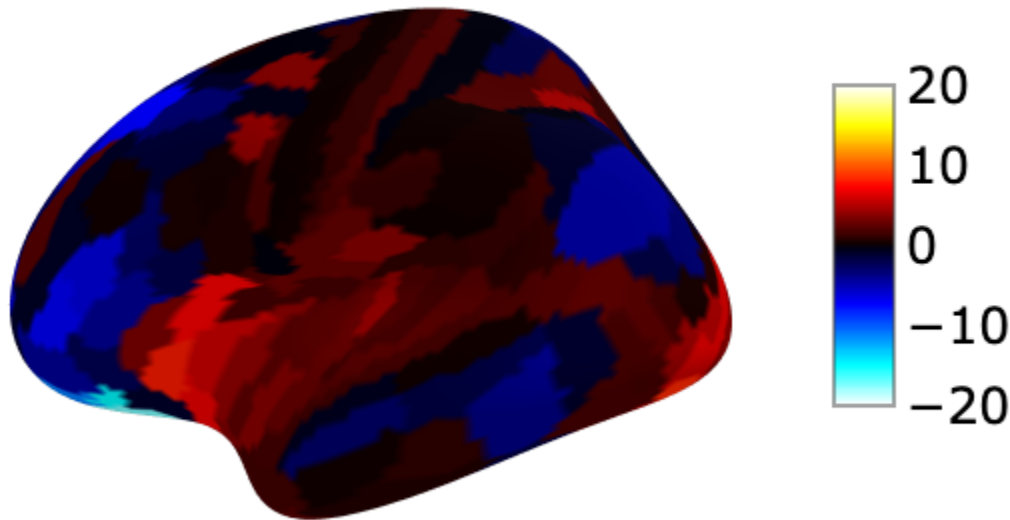
resting_beta



let's see the group contrast,beta_contrast:

```
fsaverage = datasets.fetch_surf_fsaverage()
surf_contrast = contrast_beta[atlas["labels_L"]]
plotting.view_surf(fsaverage['infl_left'],
                   surf_contrast,
                   vmax=20,title='beta_contrast for loss-win')
```

beta_contrast for loss-win



Congratulations! you made it. It's time to take a break and have a cup of coffee.

1.24.4 Logistic regression

After you have familiar with the idea that running GLM on python, let's try more models on the HCP dataset. If you already forget the dataset, please go to the Dataset instructions on pp. 24-25,36 and 50-51 of the [HcP Reference Manual](#). In order to use this dataset, please electronically sign the HCP data use terms at [ConnectomeDB](#).

First question first, what is the logistics model? In general, the logistic model has been applied for the probability of binary classes or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object would be assigned a probability between 0 and 1. In regression analysis, logistic regression is estimating the parameters of a logistic model. If you want to know more, here is a good video for [logistic regression](#).

Now, as we can see the brain activation under different conditions as a group, we can use the **Logistic regression** for decoding the data. In the context of HCP dataset and gambling project, It means that we can predict the brain activation of different conditions based on the output of GLM we analyzed before:

```
X = data
X_1 = data.T
Y = res_reg
Y_1 = Y.T
```

(continues on next page)

(continued from previous page)

```
# Define the model
log_reg = LogisticRegression(penalty="none")

# fit the model
log_reg.fit(X_1, Y_1)
y_pred = log_reg.predict(X_1)
```

Now we need to evaluate the model's predictions. We'll do that with an accuracy score. The accuracy of the classifier is the proportion of trials where the predicted label matches the true label:

```
def compute_accuracy(X, y, model):
    #Compute accuracy of classifier predictions.

    Args:
        X (2D array): Data matrix
        y (1D array): Label vector
        model (sklearn estimator): Classifier with trained weights.
    Returns:
        accuracy (float): Proportion of correct predictions.

    y_pred = model.predict(X)
    accuracy = (y == y_pred).mean()
    return accuracy

# Compute train accuracy
train_accuracy = compute_accuracy(X_1, Y_1, log_reg)
print(f"Accuracy on the training data: {train_accuracy:.2%}")
```

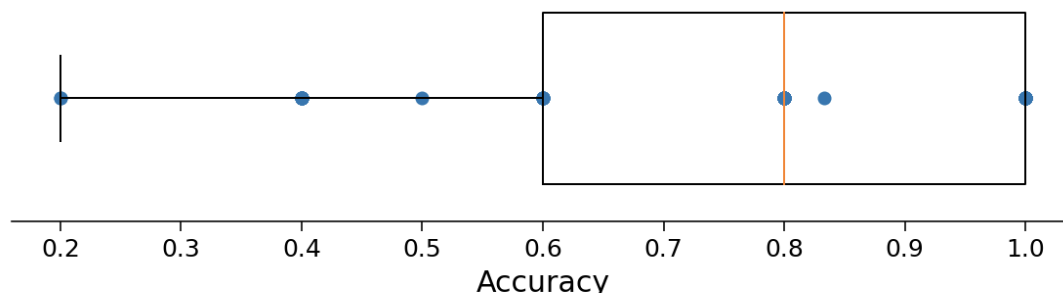
It seems like that the Classification accuracy on the training data is 100%! That might sound impressive, but there is a concept called overfitting: the classifier may have learned something idiosyncratic about the training data. If that's the case, it won't have really learned the underlying data->decision function, and thus won't generalize well to new data. To check this, we can evaluate the cross-validated accuracy:

```
accuracies = cross_val_score(LogisticRegression(max_iter=5000,penalty='none'), X_1, Y_1,
↪cv=50) # k=50 crossvalidation
```

Let's plot the accuracy on the test data:

```
#markdown Run to plot out these `k=50` accuracy scores.
f, ax = plt.subplots(figsize=(8, 3))
ax.boxplot(accuracies, vert=False, widths=.7)
ax.scatter(accuracies, np.ones(50))
ax.set(
    xlabel="Accuracy",
    yticks=[],
    title=f"Average test accuracy: {accuracies.mean():.2%}"
)
ax.spines["left"].set_visible(False)
```

Average test accuracy: 74.27%



1.24.5 Random Forest

According to Wikipedia, Random forests(RF) is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees from training data. For classification tasks, the prediction of the random forest is the class selected by most trees. Random decision forests correct for decision trees' habit of overfitting to their training set. In general, although the data characteristics may affect the performance RF, Random forests outperform decision trees. More more information, please see this [Random forest](#).

To demonstrate RF, we are going to use another dataset named [ADNI](#).

ADNI

The Alzheimer's Disease Neuroimaging Initiative (ADNI) unites researchers with study data as they work to define the progression of Alzheimer's disease (AD). ADNI researchers collect, validate and utilize data, including MRI and PET images, genetics, cognitive tests, CSF and blood biomarkers as predictors of the disease. Study resources and data from the North American ADNI study are available through this website, including Alzheimer's disease patients, mild cognitive impairment subjects, and elderly controls.

First of all, let's use FreeSurfer to preprocess the data to get the volume assessment from 106 mild impaired cognition(MCI) and 84 from healthy control people(NC), and put all of the data into a single file. Next, select the 6 ROI(regions of interest) volume with target label from the left and right hemisphere and organize a volume matrix into a CSV file named ROI_data.

'left_subiculum-body','right_subiculum-body','left_subiculum-head','right_subiculum-head','left_CA1-head','right_CA1-head','left_CA1-body','right_CA1-body','left_CA3-hs

left_subiculum-body	right_subiculum-body	left_subiculum-head	right_subiculum-head	left_CA1-head	right_CA1-head	left_CA1-body	right_CA1-body	left_CA3-head	right_CA3-head	...	right_CA4-head
113.865949	110.274775	95.465680	84.260786	275.426088	230.024227	69.360573	83.038493	52.114655	47.266651	...	52.988783
172.868377	170.531035	120.331158	132.806578	379.293303	362.956476	57.413336	84.216403	95.201826	104.815295	...	96.756928
275.676821	238.395504	173.528869	223.882736	477.652537	560.811010	92.785046	118.973437	135.845207	147.463098	...	147.605026
173.211296	161.803118	140.820254	152.978856	419.968869	400.842401	88.132330	95.972669	81.050006	77.514157	...	87.256250
190.019095	175.045866	156.920350	157.346462	464.990769	563.058683	73.435507	104.918790	92.794529	111.523301	...	123.939776
...
189.607431	206.912644	182.637104	188.467054	459.956609	489.198400	125.404243	122.537497	119.862691	146.538209	...	145.722208
181.974948	177.358309	169.631965	166.934454	433.669350	453.177088	71.904769	81.753128	97.295889	118.679656	...	114.831740
249.274816	253.294162	183.578901	204.246382	522.791967	533.058609	131.300755	126.833292	127.663315	119.246587	...	129.433890
193.709054	200.963645	183.271694	187.207303	431.142627	484.123399	95.715707	107.990495	92.449742	98.856222	...	108.767954
201.321463	215.218025	139.721612	135.482501	425.775504	434.322246	110.089168	126.507066	108.101275	110.262975	...	105.199487

Next, divide the file into two datasets, the 6 ROI volume and the target label(1 for MCI, 0 for NC):

```
ROI_X = ROI_data.iloc[:, 0:-1] # Features
ROI_y = ROI_data.iloc[:, -1] # Labels
```

Next, let's split the dataset into two training and testing dataset with a random_state:

```
X_train, X_test, y_train, y_test = train_test_split(ROI_X, ROI_y, test_size=0.25, random_
↪state = 34)
```

As we specify the test_size = 0.25, which means that we are going to use 25% data for the test and 75% data for the training.

Since the task is a binary classification. Thus, we need to use the RandomForest Classifier. Fortunately, many RFs model have been developed on Python:

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier with 2000 decision trees
clf=RandomForestClassifier(n_estimators=2000)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

Next, let's evaluate the the accuracy of this RF model:

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

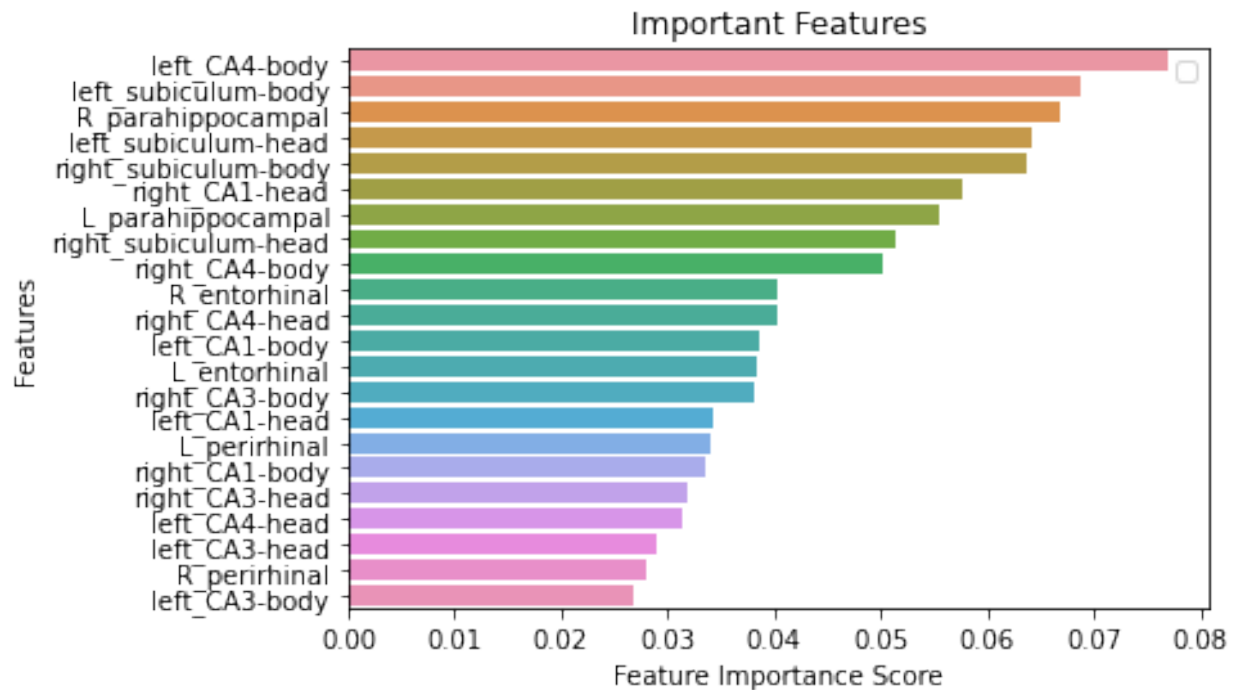
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

In order to further assess the accuracy of the RFs, we can use the recall and precision of the RF:

```
print(classification_report(y_test, y_pred, labels=[1,0]))
```

Given the nature of RF, we can visualize the importance of top features:

```
the # Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Important Features")
plt.legend()
plt.figure(figsize=(1000,1100))
plt.show()
```

1.24.6 Multilayer perceptron

Supervised learning

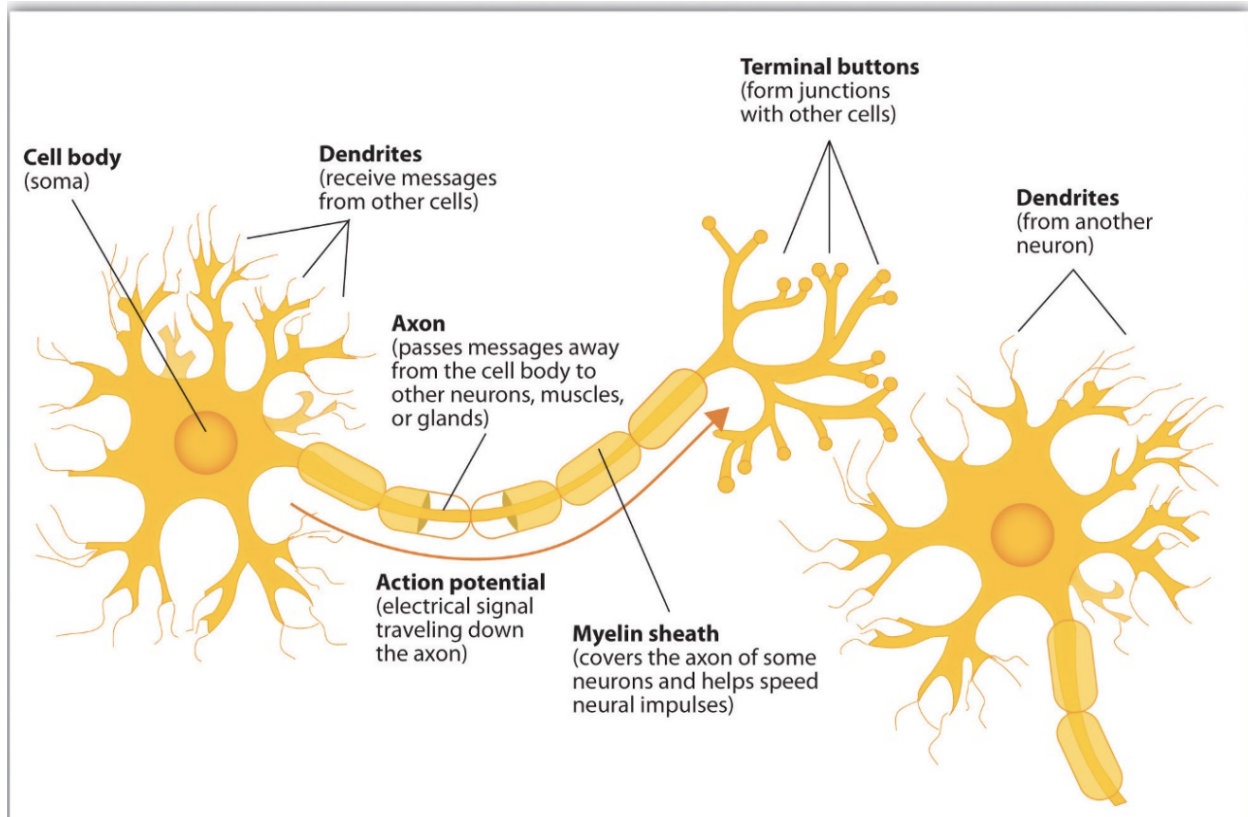
Let's start with the parent of the parent of MLP, **Supervised learning**.

machine learning was born - a computer was built that could approximate a function given known input and output pairs from it.

The key realization was that if the neural net neurons were not quite Perceptrons, but were made to compute the output with an activation function that was still non-linear but also differentiable, as with Adaline, not only could the derivative be used to adjust the weight to minimize error, but the chain rule could also be used to compute the derivative for all the neurons in a prior layer and thus the way to adjust their weights would also be known

The units are therefore stochastic - they behave according to a probability distribution, rather than in a known deterministic way

According to Wikipedia, Multilayer perceptron (MLP) is a kind of feedforward deep neural network where is become very popular in recent years. There are three layers of MLP: an input layer, a hidden layer and an output layer. MLP utilizes a supervised learning technique called **backpropagation** for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.



1.24.7 Naive Bayes classifier

1.24.8 Convolutional Neural Network

1.25 Functional connectivity

1.25.1 Resting-state fMRI

Resting-State Functional Connectivity (RSC) is a kind of study that indicated the significant correlation of signal between functionally related brain regions in the absence of any stimulus or task stimulus or task. This correlated signal arises from spontaneous low-frequency signal fluctuations (SLFs). A lot of the time series of voxels correlated significantly (after filtering the fundamental and harmonics of respiration and heart rates) while only a few voxel time courses ($< 3\%$) correlated with those in regions outside of the motor cortex. Subsequently, there are some evidence indicated that the presence of RSC in sensory cortices, specifically auditory and visual cortices. In a study, signal from visual cortex voxels during rest was used as a reference and correlated with every other voxel in the brain. A significant number of voxels from the visual cortex passed a threshold of 0.35, while only a few voxels from outside the visual cortex passed the threshold. some similar results also appear in the auditory cortex.

1.25.2 The history of Resting-State Functional connectivity

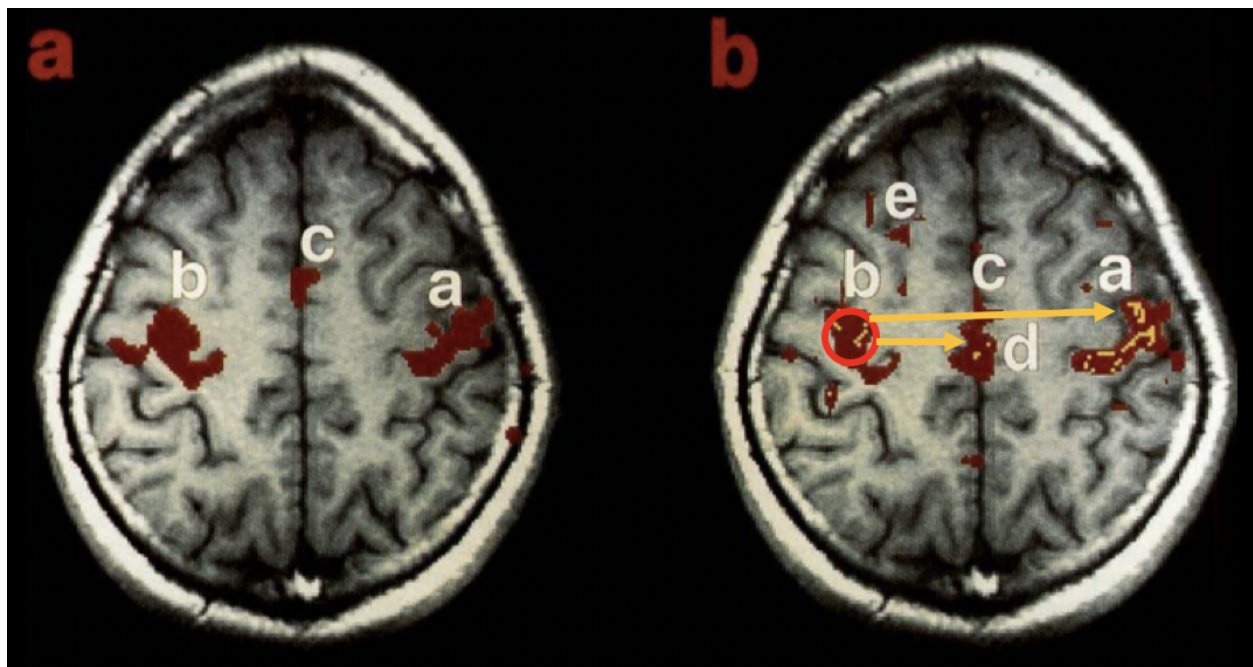
Experiments with fMRI focused primarily on mapping the BOLD response to sensory and motor stimuli as neuroimaging became more widely employed in the early 1990s. The reaction of the visual cortex to a flashing checkerboard, for example, was observed early on (Kwong et al., 1992), as were the responses of the primary sensory areas to auditory, tactile, and finger pressures. These tests were simple, but they were crucial for proving the validity of fMRI as a non-invasive imaging technology.

With these preliminary findings, researchers began looking into the source of the BOLD signal as well as the causes of noise. It was recognised that the signal obtained from the BOLD response was minimal in comparison to the noise that surrounds it; noise generated not only by the scanner itself (in the form of “scanner drift”), but also by physiological causes such as respiration and blood circulation throughout the brain. Bharat Biswal scanned people while they were undertaking a task and while they were not doing a task to better understand these physiological sources of noise.

Even after removing the physiological sources of noise from the resting-state data, Bharat Biswal was surprised to find that they did not account for all of the variance in the BOLD response. Biswal retrieved the time-series from the left motor cortex and correlated it with the time-series of all other voxels after noticing what appeared to be temporal correlations between distinct regions of voxels. There was a high association with the time-series of the opposite hemisphere's motor cortex, rather than the random correlations that one would predict if there were no regular BOLD fluctuations at rest. In other words, It suggesting that these two functionally equivalent regions generated identical patterns of activity even at rest, though the two areas being spatially separated.

Several other researchers looked into whether these resting-state patterns could be seen in other parts of the brain a few years following Biswal's finding. Because the motor cortices appeared to be highly associated, it was expected that other bilateral structures, such as the visual and auditory cortex, would also be highly connected. And if these functionally comparable regions were associated during a cognitively demanding task, it was claimed, then other functionally similar regions should be correlated during rest as well.

Note: Biswal employed the left motor cortex as a seed region in this example, which was then linked with all of the other voxels in the brain - commonly known as a **whole-brain analysis**.



The original Biswal et al. 1995 paper yielded this result. Panel (a) depicts task-related activity in the motor cortices, whereas panel (b) depicts functional connectivity during a resting-state scan, with the left motor cortex serving as a

seed region. The task and resting-state images have a lot of overlap, as you can see.

1.25.3 Default-Mode networks

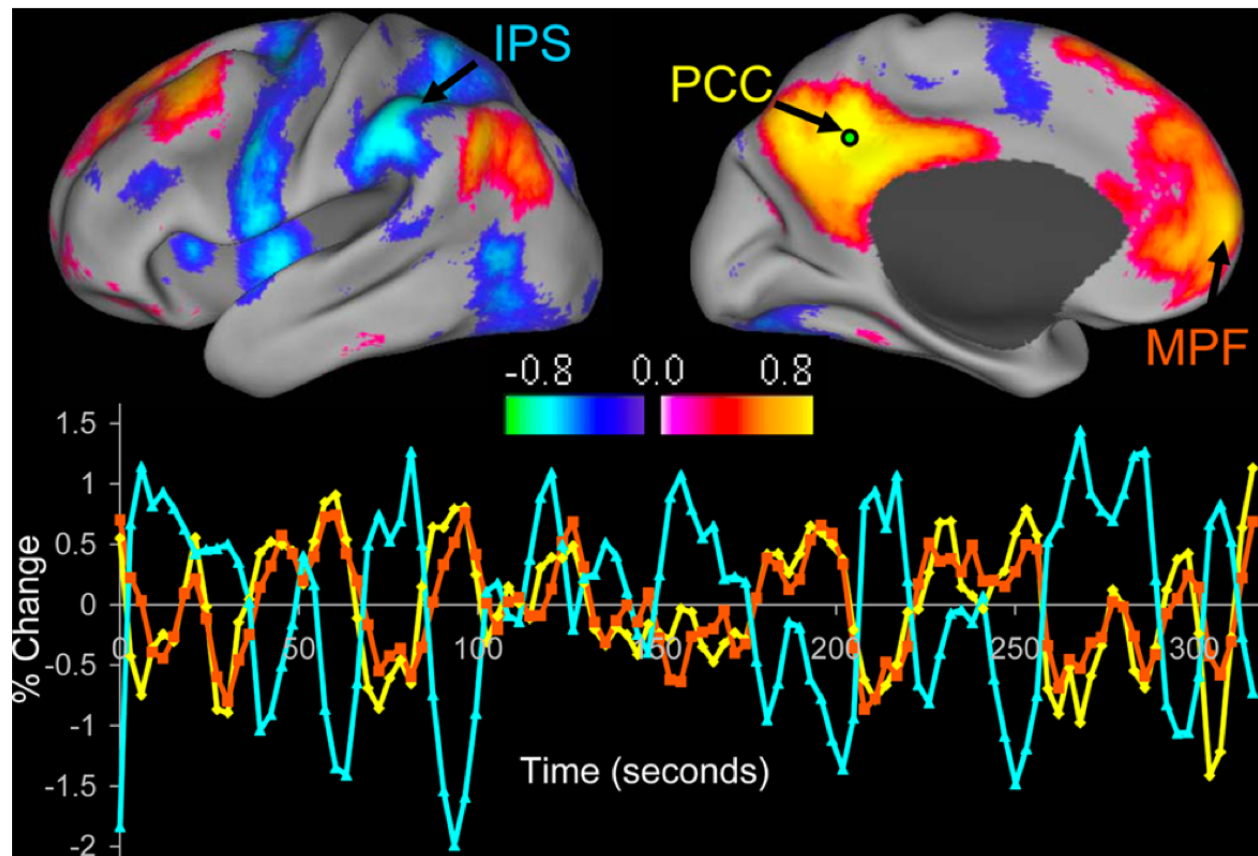
Biswal and other researcher's study paved the path for neuroimagers to investigate a series of novel theories, including: What would happen to these resting-state patterns if a different seed location was used? What would be the differences in these resting-state patterns amongst groups, particularly clinical populations? What did this mean in terms of treating prevalent mental illnesses? How did these relationships change through time and as people became older?

There was a search for a consistent resting-state pattern that could be related to how people performed during tasks before these questions could be answered. Behavioral disparities between schizophrenics and controls had been demonstrated in basic psychometric tests; however, was this due to, say, visual processing deficiencies or a difference in higher-level cognition?

A few years after the Biswal 1995 article, Gordon Shulman and a team of University of Washington researchers discovered that when some brain regions were more active during cognitively demanding tasks, blood flow to other brain regions decreased. This led Shulman to theorise that there may be two networks of brain regions that show inverse blood flow patterns depending on whether or not a task is being performed. To be more specific, the dorsal anterior cingulate and intraparietal sulcus were consistently active during tasks, whether verbal or nonverbal, but the ventromedial prefrontal cortex and posterior cingulate cortex had consistently decreased blood flow during activities, regardless of the type of activity.

Michael Fox and his colleagues didn't prove that these locations were anti-correlated with one another until nearly a decade later, in 2005: For example, an increase in activity in task-activated regions was linked to a reduction in activity in rest-related regions. The intraparietal sulcus, frontal eye fields, and middle temporal cortex were identified as the task-activated regions' three major nodes, whereas the medial prefrontal cortex, posterior cingulate cortex, and lateral parietal cortex were identified as the rest-activated regions' three major nodes. The task-positive and task-negative networks were named after the opposite groupings of nodes.

Those studies found that in order to do a task effectively, there required to be coordination between each of the task-positive nodes - as measured by the amount of correlation between the nodes - as well as a coordinated drop in activity in the task-negative nodes. Different mental diseases, such as schizophrenia and bipolar disorder, could be connected to disruptions in this synchronisation.



1.25.4 Artifacts

Motion Artifacts

As resting-state studies grew more common in the mid-2000s, the right methods for conducting them came under increased scrutiny. Despite Biswal's demonstration that the BOLD signal fluctuations that underpin resting-state signals are not artefacts, it became clear that resting-state data is particularly vulnerable to certain forms of distortions, particularly motion movement.

Although motion was recognised as an artefact and potential confound for task-related studies from the start, researchers later established that it could lead to erroneous correlations between distinct areas of the brain. In many resting-state investigations of elderly people, correlations between geographically distant nodes like the posterior and anterior parts of the cingulate cortex were shown to be weaker than in younger people. These changes were considered to reflect degradation of the older patients' brains, similar to how a machine's screws and pins wear out over time.

Despite the fact that these studies either accounted for motion as a covariate or only included subjects with low amounts of motion, Van Dijk and colleagues (2011) indicated that even little changes between groups could lead to large differences. Subjects were divided into groups based on how much they moved, and there were variations even across groups with mean motions of 0.044 and 0.048. Very modest changes in motion can become the primary source of variance when averaged over large groups of subjects.

of a millimetre.

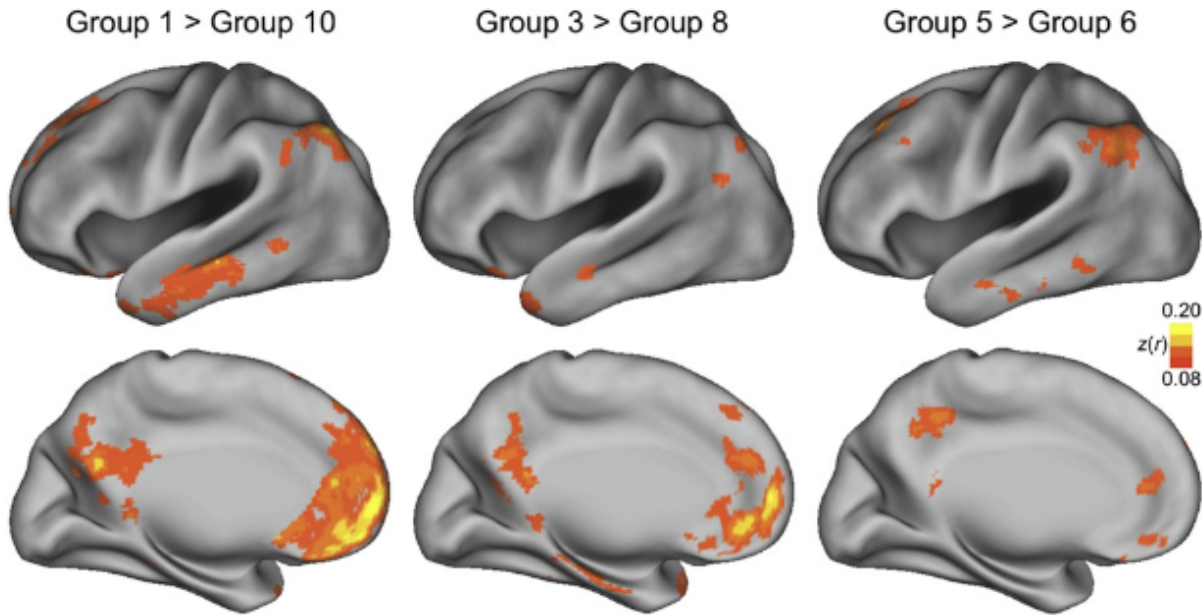


Fig. 24: Van Dijk et al. (2011) published this figure in their paper. The difference in average motion between the two groups on the right is only a few thousandths

Physiological Noise

The subject's voluntary motions, such as adjusting the head or scratching an itch, are generally assumed to be the source of motion artefacts. While those are the most obvious, other involuntary actions can have just as much of an impact and are typically more subtle. The two main causes of tiny, involuntary motions that can affect between-group differences are respiration and heart rate. Measuring and regressing these physiological signals helps account for spurious correlations that occur near the arteries or at the brain's borders, where these artifacts are most visible.

Global Signal Regression

Although movement artefacts have long been regarded a source of noise that must be accounted for, one approach of data analysis, Global Signal Regression, or GSR, has sparked debate since its inception. Fox et al. (2005, described above) investigated anti-correlations between the resting-state and task-positive networks, which was one of the first applications of GSR. Murphy et al. (2007) later demonstrated that GSR produces erroneous negative correlations in voxels. GSR was chosen because it catches noise that isn't related to neuronal activity, such as breathing, head movement, and scanner drift.

1.26 SuperComputer

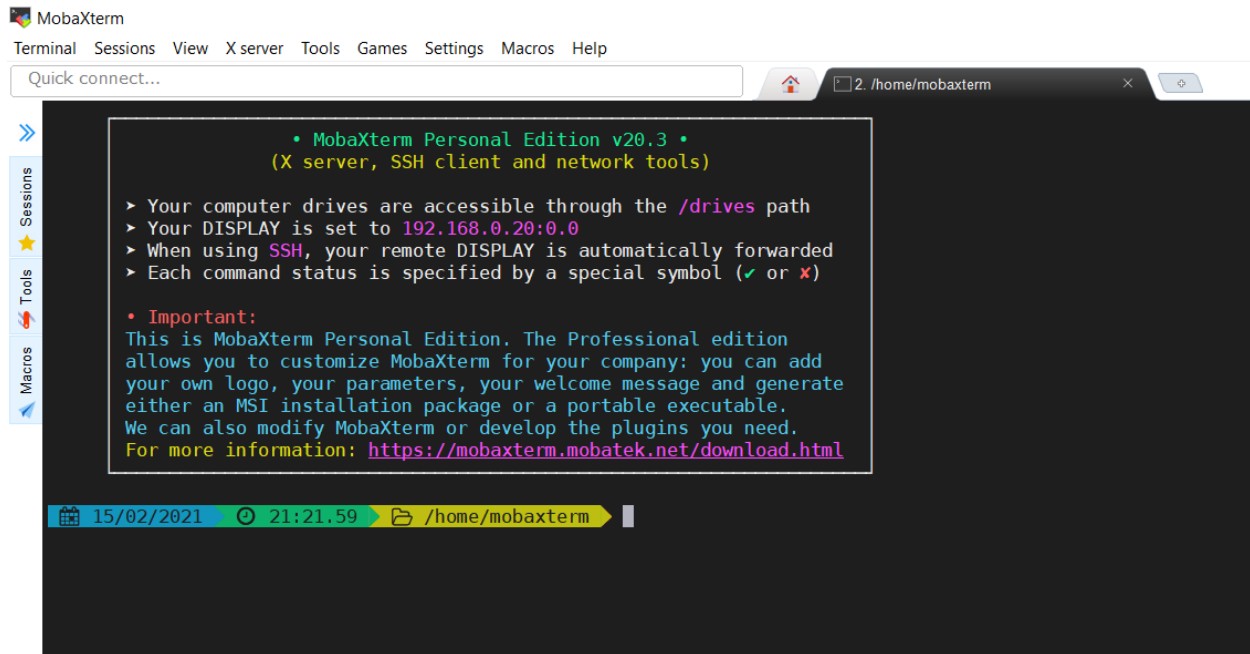
How to run analysis on a SuperComputer?

a supercomputer like central kitchen, everyone can get access to it and do some amazing jobs without actual buy all the expensive tools

Nowadays, every university or neuroimage research institution might have the right and we can take advantage of it, you can run the analysis on your own laptop or you can run the analysis on supercomputers!

first of all, you need a key to the door of central kitchen, and MobaXterm can do this job

Go to [MobaXterm](#) and find the right edition to set up.



if you have the access to any server/supercomputer, you can take the Cedar cluter from computecanada as a example:

```
ssh -Y (user id)@cedar.computecanada.ca
```

Secure Shell (ssh) is a widely-used standard to connect to remote servers in a secure way. SSH connection is encrypted. You can use ssh to execute commands, submit jobs, follow the progress of these jobs and in some cases, transfer files,etc.

The option -Y forwards X11 traffic which allows you to use graphical applications on the remote server. You also need to have an X11 server installed on your workstation. For windows, MobaXterm normally comes with an X11 server.

user id is the id you registered in ComputeCanada, @Cedar.computecanada.ca means I want to connect the Cedar cluster

Then, if you see a similar terminal like this, it means you are in

```
[weishao@cedar1 ~]$  
[weishao@cedar1 ~]$
```

In some cases, you can use `ssh -Y (user id)@(server IP)` as the command to connect the remote server.

Use `diskusage_report` to find the disk space on the server

1.26.1 transfer the data among different server

use `scp -r` to transfer the data

Or `sftp` to build the connection and `put` to upload the file or `get` to download the file

use `tar -czvf new.file original.file` to compress a file and use `tar -xzvf new_file` to decompress the file accordingly

`-c` :create a new archive `-z` :filter the archive through gzip `-v` :verbose output `-f` :use archive file `-x` :extract file

1.26.2 SGE

In the SGE server, you can use:

```
qsub -V -cwd -q parallel.q -pe mcore 8 script.sh
```

to submit a job with 8 cores in parallel computing

“IS WHAT EACH BEGINS AND PERFECTS ON HIMSELF”

–Thomas carlyle

if you have any questions/suggestions about this menu, please contact me by reclusedavid@gmail.com.