1. Set up computer
   1. Install Matlab
   2. Set up CANlab tools from Github
   3. Add CANlab tools to Matlab Path
   4. Other helpful resources
2. Locate files to analyze on your computer
   1. Download Wager2008\_EmotionReg\_Sampledata if you don’t have it yet
   2. Add Wager2008\_EmotionReg\_Sampledata to your Matlab Path
3. Basics of object-oriented tools
4. Simple data loading, visualization and regression walkthrough

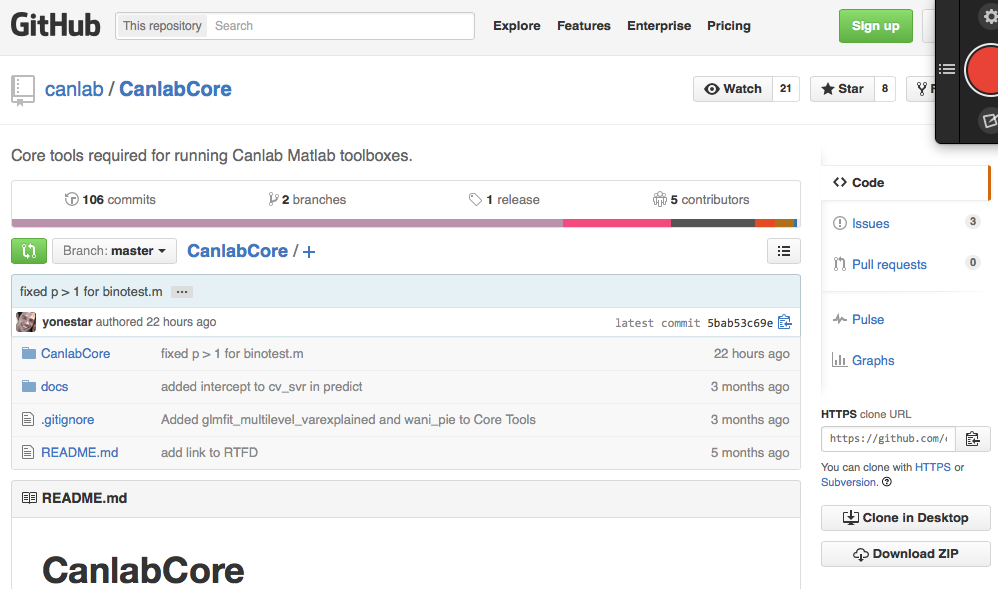
**1a. Install Matlab**

**1b. Set up CANlab tools from Github**

There are two options: 1) Download a .zip file with the code, or 2) install Github on your computer and Clone the respository. With the 2nd option you will get updates, but it requires more steps and some knowledge of how Github works.

First, go to the Github CANlab tools site:

•<https://github.com/canlab/CanlabCore>



In the bottom right corner you can either “Download ZIP” or “Clone in Desktop” (options 1 and 2 above).

If you “Clone in Desktop”, you need to:

1. Download the Github application to your computer and run as administrator. On your computer it has the little cat icon. You need admin privileges. On Windows, right click and “run as administrator.”

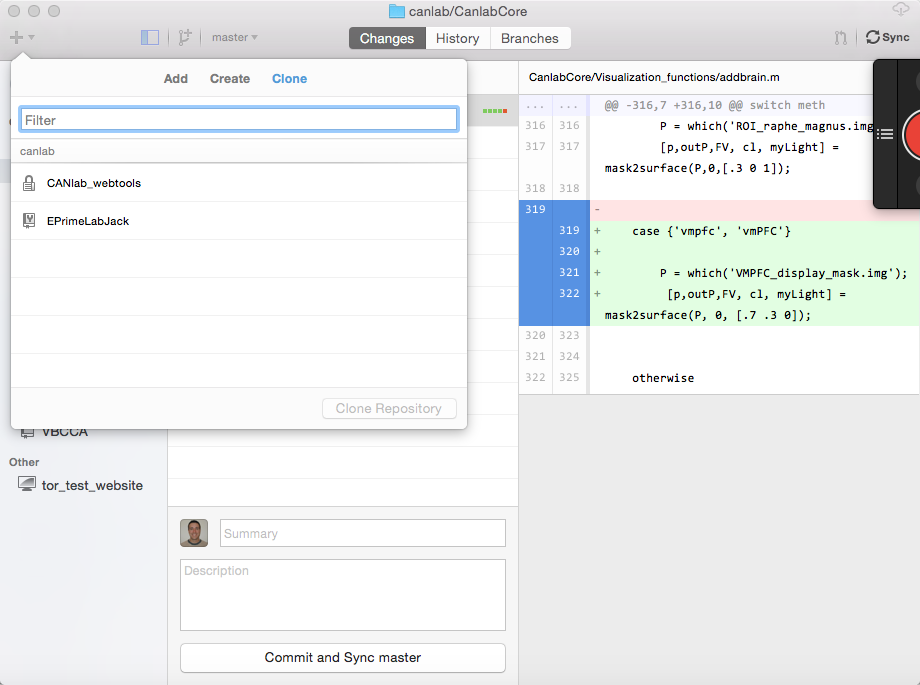


2. Go to CANlab tools (e.g., CANlab Core) on the web and click “Clone in Desktop”

3. Select a folder on your hard drive to store the code

4. It should start adding the CANlab Core repository to your Github application registry.

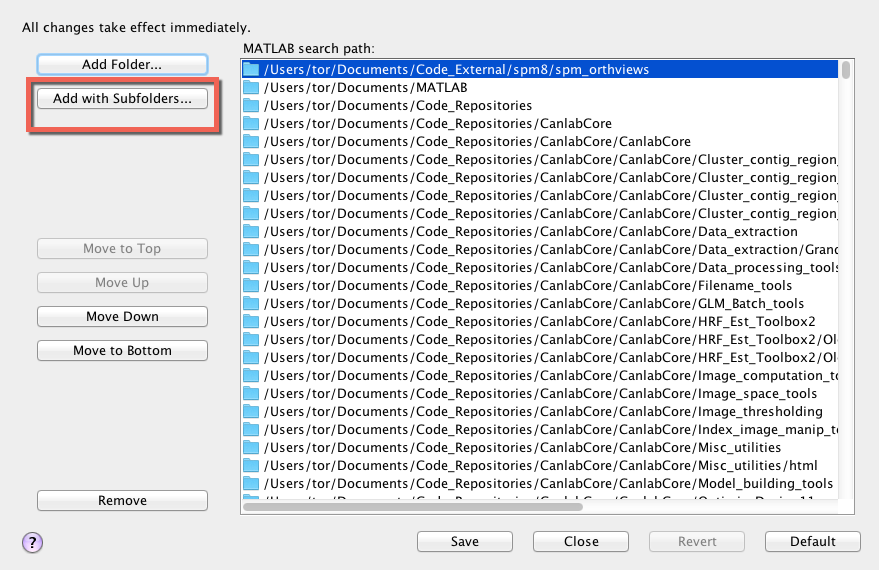
Your local Github application has a list of all your local repositories. The + sign in the top left will let you add, create, or clone repositories that Github already knows you have added. If you are a member of a group on Github, you should see associated repositories in that list.



**1c. Matlab path**

Open Matlab and type “pathtool” and then and then the “return” key.

It brings up the dialogue below. Click “add with subfolders” and add the repository code files you just downloaded through Github. Whether you download a ZIP or “clone” above, this step is the same.



**Check that it worked: Type**

**>> which fmri\_data**

It should show you the path where this function is located. If not, your paths are not set up right. This will be different for you depending on where you have stored the files:

/Users/tor/Documents/Code\_Repositories/CanlabCore/CanlabCore/@fmri\_data/fmri\_data.m % fmri\_data constructor

**1d. Other helpful resources**

**WIKI pages on fMRI help:**

<http://wagerlab.colorado.edu/wiki/doku.php/help/fmri_tools_documentation>

and specifically:

<http://wagerlab.colorado.edu/wiki/doku.php/help/core/object_oriented_fmri>

FAQs on GIThub:

•[http://wagerlab.colorado.edu/wiki/doku.php/github\_faq?s[]=github](http://wagerlab.colorado.edu/wiki/doku.php/github_faq?s%5B%5D=github)

Principles of fMRI YouTube Videos:

https://www.youtube.com/channel/UC\_BIby85hZmcItMrkAlc8eA

2. Locate files to analyze on your computer

* 1. Download Wager2008\_EmotionReg\_Sampledata if you don’t have it yet
  2. Add Wager2008\_EmotionReg\_Sampledata to your Matlab Path

type **pathtool** and then the “return” key at the Matlab prompt.

Navigate to the “Wager2008\_EmotionReg\_Sampledata” folder and add to your path.

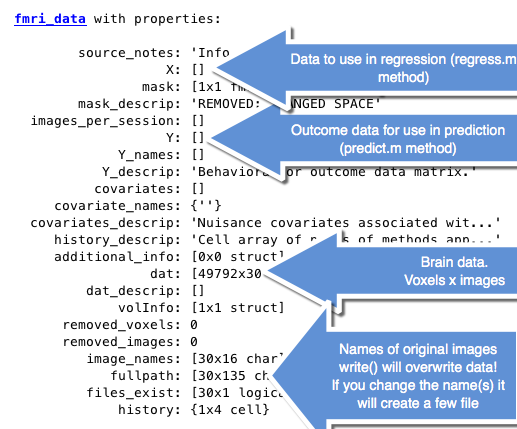
Save the path and close.

3. Basics of object-oriented tools:

Go to this link for basic info:

<http://wagerlab.colorado.edu/wiki/doku.php/help/core/object_oriented_fmri>

4. Explanation of basic fields in fmri\_data object



Let’s go through all the fields that are listed as attributes in the object above. There are four fields that are the most commonly used and that is the most important to know about as a user. They’re the ones of blue arrows above. If you are a programmer and interacting with the objects at a deeper level then the .volInfo attribute is important.

Here are the attributes grouped by the functionality.

**Data fields:**

***X: in*** a data set with N images, this attribute is an N x K matrix. It contains data that you would like to relate to Brain Data in the object via regression. The regress method he uses data in this field and the brain data.

**Y:** in a data set with N images, this attribute is an N x 1 vector. It contains data that you would like to relate to Brain Data in the object via a predictive model, with brain predicting the outcome (Y). The predict method uses data in this field and the brain data.

.dat: a v x N matrix of voxels x images. This is the main brain data. The matrix contains vectorized three dimensional images, with one image per column. The locations of each Vauxhall (row) in brain space are specified in the volInfo attribute as x, y, z coordinates. The methods ***reconstruct*** and ***write*** turn these vectorized images back into three dimensional bring images. The other important thing to know here is that vectorized brain images do not store every voxel in the entire image space. Rather, the only store data in Boxell’s within a predefined mask, to save space. The mask is defined in the volInfo field, which is needed to convert between vector space and brain space. In addition, empty voxels – which are defined as having values that are NAN or exactly 0 in all images – are also excluded to save space. An indicator vector (1s and 0s) for which voxels are excluded as empty is stored in .removed\_voxels attribute. Similarily, empty images (all 0 or NaN) are removed and the indicator vector for which are empty is stored in the .removed images attribute. Therefore, there are 3 subsets of voxels to consider, and 2 subsets that may be in the .dat field. 1: All voxels in image (not stored), 2: All voxels in mask , 3: All non-empty voxels in mask. The methods remove\_empty and replace\_empty switch back and forth between (2) and (3). Finally, Data are stored in single format to save space. In some versions of Matlab this causes problems with some operations, and we recommend recasting as double before operating on the data.

***.covariates***: ***in*** a data set with N images, this attribute is an N x M matrix. It contains data that is generally intended to be removed from imaging and other (X, Y) data before or during regression. The resid method uses data in this field and the brain data [I think!!!]. \*\*not sure which others use it, have to look up\*\*\*

.additional\_info: Other flexible data storage.

**Image, image space, and image filename info**

.mask: fmri\_data, not image\_vector, and it stores info about the mask that was applied during the loading of the image files into the object.

.image\_names = list of image names without full paths.

.fullpath = list of image names with full paths. This is a very important field because the ***write*** method reconstructs all images into analyze image files (.img) and uses these filenames to write them to disk. If you are planning on using the right method, you can define names in the full path attribute. This could be one name per 3-D image file, or one name for a four dimensional image file for the full data set.

.volInfo: a very important attribute for the internal operation of the toolbox. It contains multiple Fields it is a structure. This defines the mask image and the space (rotation, translation, position, voxel coordinates)

fname: 'REMOVED: CHANGED SPACE'

File name for the mask image used to define the mask. If the mask has been resampled (interpolated) to match the space of the functional image data, it will say removed changed space.

mat: [4x4 double]

SPM format Matrix of transformation. Diagonals are voxel sizes in XYZ. Off diagonals are related to rotations that have been defined in rigid body realignment operations. The fourth column contains values related to translation into “world space” relative to the x, y, z zero point of the anterior commissure. We usually don’t worry about those, but if you are using an SPM processing pipeline, then it will store Motion correction transformations in this matrix. When you use SPM to load an image from desk, It applies this transformation matrix upon loading.

dim: [91 109 91]

Dimensions of the original image in numbers of voxels and XYZ

dt: [2 0]

Related to datatype, used when using SPM to write to disk

pinfo: [3x1 double]

Related to SPM scaling factor and other stuff, used when writing info to disk using SPM

n: [1 1]

Related to number of volumes in the image that you loaded. Here this is defined based on the mask image and generally would just be one image.

descrip: 'Space of /Users/tor/Documents/Code\_Repositories...'

private: [1x1 nifti]

This contains some sort of pointers to the actual data on disk and is used by SPM when accessing and writing data. Related to NIFTI object. Values assigned here will directly change the values of the linked images on the hard disk.

nvox: 902629

Number of bottles in the entire image space

image\_indx: [902629x1 logical]

An indicator vector – a logical vector of ones and zeros–that lists which voxels have in mask values. The some of the specter is equal to the number of bottles that have not done zero, non-Nan values in the mask image

wh\_inmask: [352328x1 double]

Vector of element numbers corresponding to Image index. Stored for convenience.

n\_inmask: 352328

Number of voxels in the mask altogether.

xyzlist: [352328x3 double]

Foxholes by three list of the XYZ court nets of each bottle

cluster: [352328x1 double]

Subsets of the contiguous box goals defined by spm\_cluster.m. In-mask voxels only are considered, and contiguous vox are assigned an integer value. Voxels with the same integer are assigned to the same cluster.

obj.mask attribute (for fMRI data):

**Provenance/documentation:**

- Source notes is just a text drink to help you keep track.

- Mask descriptive describes the brain mask that is used for the object

* Why descriptive describes outcome data or other core that’s the you’ve attached to the object
* There are’s also descriptive text attributes for the covariates, Another place to attach payroll data, and the history and the data (.dat)
* The history attribute tracks what was done to the object. When some methods are run they attend a cell element to the end of the history attribute. history(obj) prints the history.
* Additional\_info is a cell array that is designed to be used flexibly so that you can attach other types of information or data.

5. Simple data loading, visualization and regression walkthrough (Actually, just go to walkthrough document and have everything there)

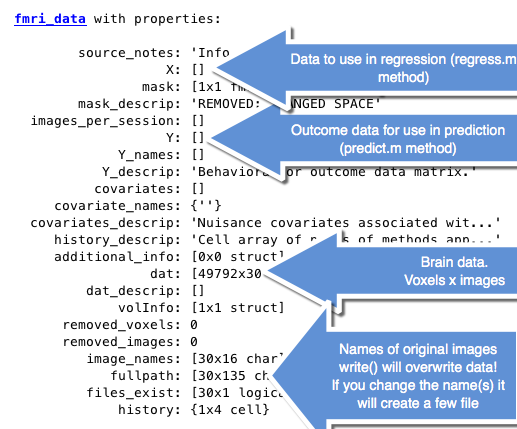
% Edit the code we will run - this file need to be on your matlab path:

edit regression\_walkthrough.m

The code is organized in blocks, and we’ll run it one block at a time. You can also copy and paste commands.

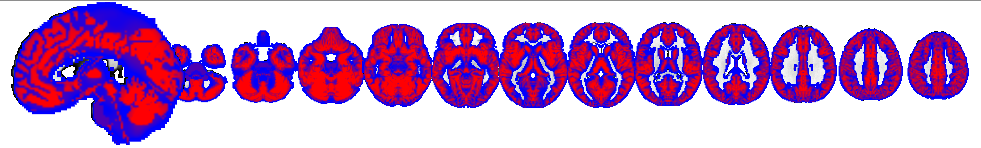
1. **RUN BLOCK 1**

If successful, this will create “dat”, an fmri\_data object. Let’s look at “dat” and a few of its key properties. You don’t necessarily need to know what all these fields are to use the tools, but you need to know some.



**b. Run BLOCK 2**

This simply makes a montage display of the mask we will use for analysis. It’s a gray-matter mask.



There are other plotting methods too, with lots of options.

To see a list of all the methods for fmri\_data, type >> methods(dat) at the Matlab >> prompt.

Here is a list of the other plotting methods:

plot(dat) This is perhaps the most useful for exploring a dataset.

orthviews(dat)

fastmontage(dat)

histogram(dat)

isosurface

montage

qchist

surface

slices

sagg\_slice\_movie

**c. Run BLOCK 3**

This uses the orthviews method, with some custom computation.