xjView 4 Manual

Xu Cui
Human Neuroimaging Laboratory
Baylor College of Medicine

6/24/2006 created
2/20/2007 updated

“... a VERY useful viewing program that is small and easy to use. p-value slider, id of anatomy with a single mouse click, etc.”

from IMAGING AND UNIX GLOSSARY @ CNL @ UNIVERSITY OF ARIZONA

Contents

PREFACE

xjView was written by Xu Cui and Jian Li, both graduate students at Human Neuroimaging Laboratory (HNL) directed by Dr P. Read Montague at Baylor College of Medicine located in Houston, Texas, United States. The database used in xjView is from MNI Space Utility written by Sergey Pakhomov.

xjView works under MatLab 6.5/7.0 and SPM2/5 both in Windows XP Service Pack 2 and Red Hat Linux 7.3. xjView may not work under MatLab 6.1 or below.

xjView is downloadable at
http://people.hnl.bcm.tmc.edu/cuixu/xjView/

You can choose to download only xjview.m and TDdatabase.mat if you only use xjView as a viewer. If you also want to do some data analysis using xjView, you need to download all the files listed.

For bugs or comments, please don't hesitate to send a message to cuixu at hnl.bcm.tmc.edu.

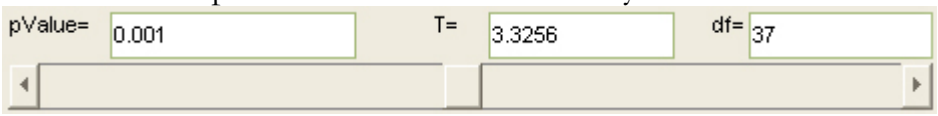
Chapter

0

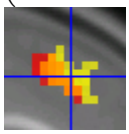
xjVIEW AT A GLANCE

xjView is mainly a viewing program for SPM. With xjView version 4 you can:

† Easily change p-Value (or T, F-value) by sliding a scroll bar and display the resulted supra-threshold voxels instantly.



† Display multiple images at the same time and find the common regions. (Common regions are displayed in intermediate colors.)



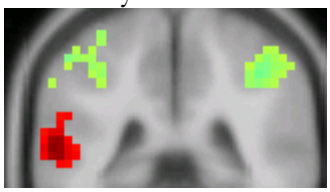
† Pick up one or more clusters and save them as a mask image for future ROI analysis.

† Find out the anatomical name of the selected voxel(s) or cluster(s).

† Search the selected brain region in external databases (xBrain, google scholar or pubmed) and find out what the function of this region is.

† Display the images in glass view, section view and 3-D render view.

† Display a T-test image with positive T coded by hot color and negative T coded by cold color.



† Overlay a known structure (e.g. Brodamnn 10) to the current hot spot.

- Set the scale of the colorbar in the section view so that you can use the same colorbar for multiple T/F test images.
- Data preprocess, GLM estimation, ROI analysis, behavior analysis, etc
- And more ...

Chapter

1

VIEWING PART

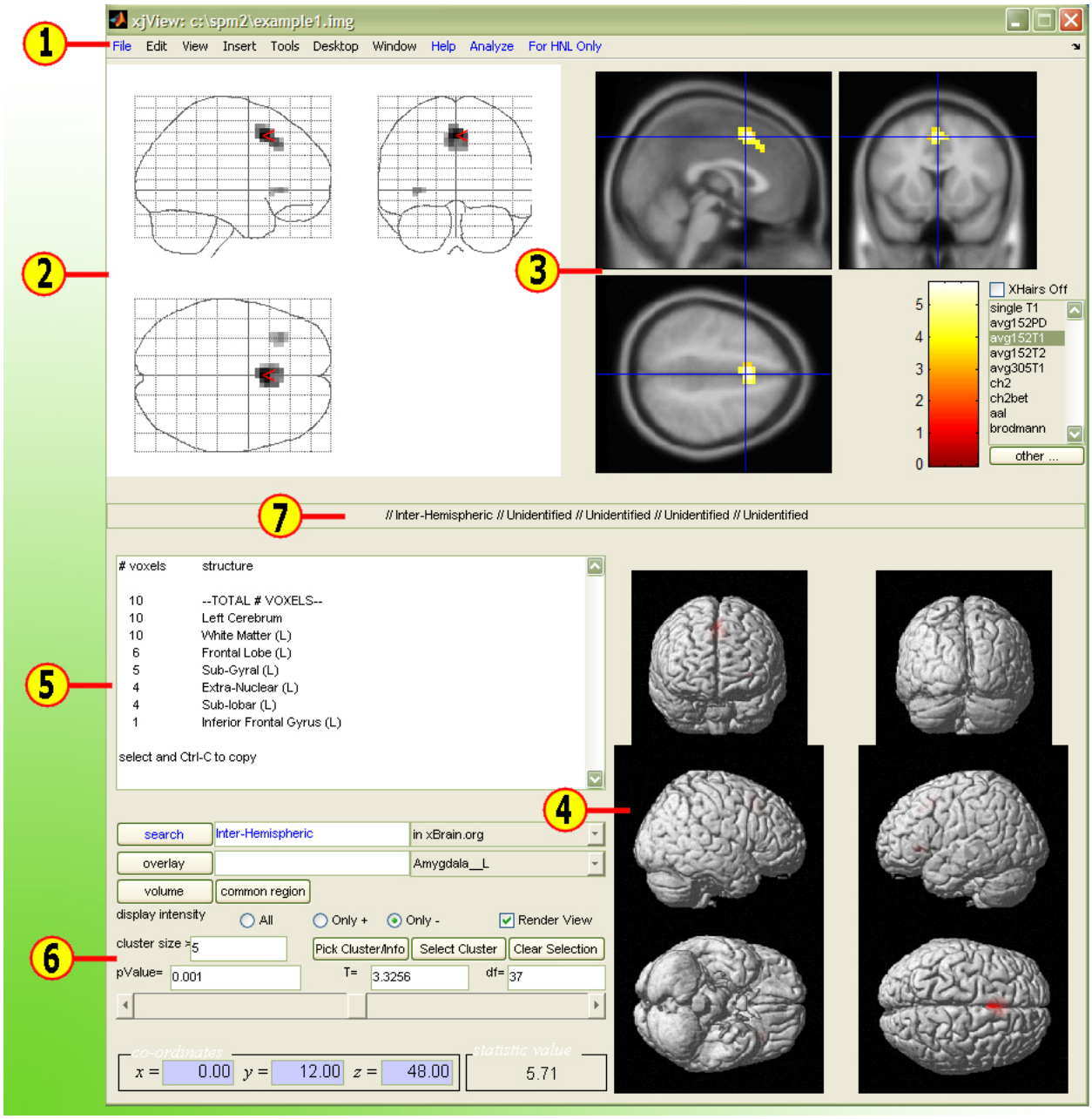
Before using xjView, please check if xjView flipped the left/right of your images. You may want to open a T-test image in spm, pay attention to the left/right, and then exit spm and open the same T-test image in xjView. If xjView flipped the left/right of your image, you need correct this by changing the value of a variable called `leftrightflip` to 1 in the beginning of xjview.m. You don't need to worry about the left/right issue any more as long as you are displaying images generated from the same lab.

- **Start xjView**

There are 4 ways you can start xjView in MatLab command window (type help xjview):

```
usage 1: xjview (no argument)
          for displaying a result img file, or multiple image files,
          (which will be loaded later) and changing p-value or t/f-value
usage 2: xjview(imagefilename)
          for displaying the result img file and changing p-value or
          t-value
          Example: xjView spmT_0002.img
                   xjView('spmT_0002.img')
                   xjView mymask.img
usage 3: xjview(imagefilename1, imagefilename2, ...)
          for displaying the result img files and changing p-value or
          t/f-value
          Example: xjView spmT_0002.img spmT_0005.img spmT_0007.img
                   xjView('spmT_0002.img', 'spmT_0003.img', 'spmT_0006.img')
                   xjView myMask1.img myMask2.img myMask3.img
usage 4: xjview(mnicoord, intensity)
          for displaying where are the mni coordinates
          mnicoord: Nx3 matrix of mni coordinates
          intensity: (optional) Nx1 matrix, usually t values of the
          corresponding voxels
          Example: xjView([20 10 1; -10 2 5],[1;2])
                   xjView([20 10 1; -10 2 5])
          Note: to use xjview this way, you may need to modify the value
          of M and DIM in the begining of xjview.m
```

After you start xjView (and open an image file), the main screen will look like the figure below:



1 Menu bar. This menu bar is a standard MatLab figure menu bar, with some changes in blue. If you downloaded xjview.m and TDdatabase.mat only, you won't see the Analyze and For HNL Only menu.

2 Glass view. Right click glass view and you will see a context menu by which you can change the cursor to the (local or global) maximum.

③ Section view. If there is an error when you choose some target images in the section view (such as ch2, ch2bet, aal and brodmann), that's because those images are not provided by SPM. You may want to check out MRIcro. Copy the corresponding image files (unzipped, with header files) to your spm canonical directory. You can also set the max of the colorbar. If you want SPM automatically determine the max, reset the max to 'auto' and press enter.

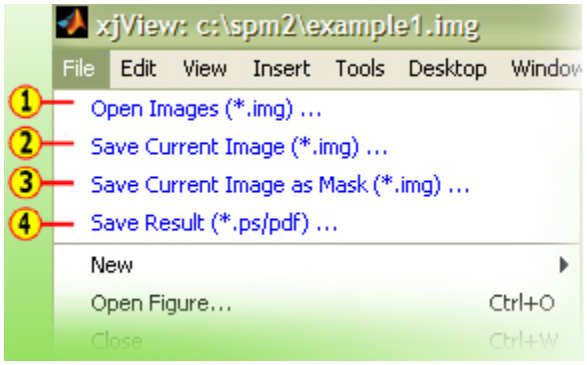
④ Render view. You have to check "Render View" to display. Note: displaying render view takes considerably longer time than glass view and section view and thus you might want to check the render view only when necessary.

⑤ Information window. Updates, warnings, errors or status will be shown here.

⑥ Control panel.

⑦ Anatomical description (in Talairach Atlas Labels) of the pointed voxel.

- **File Menu**



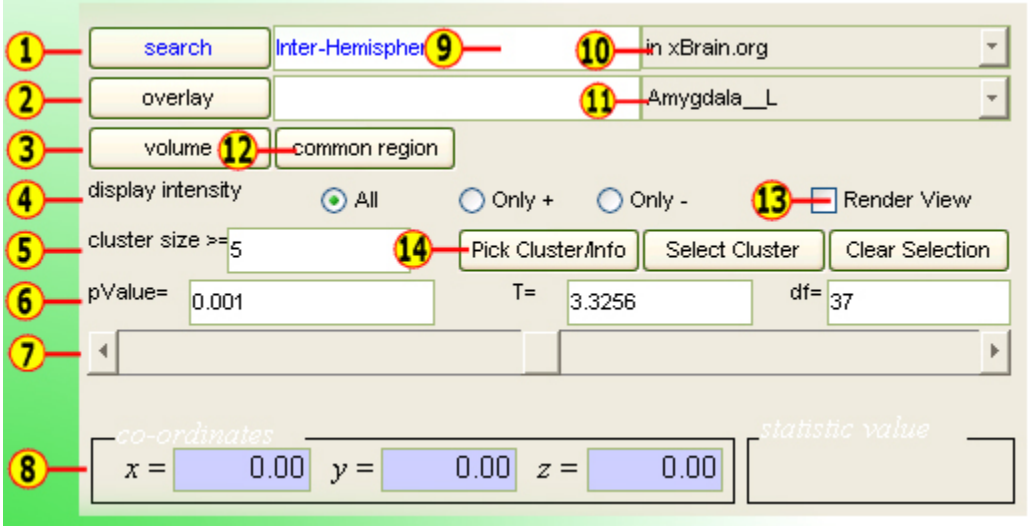
① Open a single image file or multiple image files. For example, T-test image files. **Tip: You can also double click mouse on any blank space in xjView window to open image files.**

② Save the currently displayed image to an image file. The intensity values (i.e. the T values, F values, or other user-defined values of each displayed voxel) are also saved.

③ Save the currently displayed image to an image file, with all displayed voxels having intensity 1 and undisplayed voxels, 0.

④ Save the whole figure in ps/pdf format. (Not recommended. I myself use 'Print Screen' on the keyboard and paste to Microsoft Paint for further editing.)

- **Control Panel**



(1) Search a brain region (in edit box (9)) in an online database ((10) xbrain, google scholar, or pubmed).

(2) Overlay a known brain structure (in the right edit box). You can choose the known structure from the list ((11)).

(3) List supra-threshold voxels. You will be asked to select the SPM.mat file which generated the T or F image you are displaying. If you don't specify any SPM.mat file, xjView will list the voxels anyway but the values may not be accurate.

(4) Select positive, negative or all values to display. For example, a T-test image usually have both positive (activation) and negative (inactivation) values and you may want to display them at the same time.

(5) Cluster size threshold.

(6) pValue/T/F threshold and degree of freedom.

(7) Slider bar. You can change pValue/T/F continuously with this slider bar.

(8) Current voxel coordinate in MNI space.

(12) If multiple images are displayed, click 'common region' button to display the common regions (of the currently displayed regions) only.

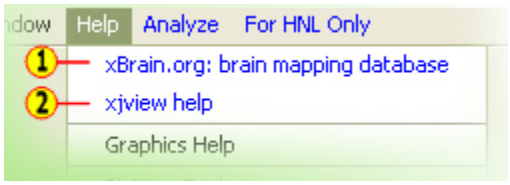
(13) Check to display render view.

(14) Select and Pick cluster. "Select Cluster" button allows you to select multiple clusters. If a cluster (a multiple clusters) is picked, the cluster information will be displayed in information box. The number on the left is "the number of voxels". For example,

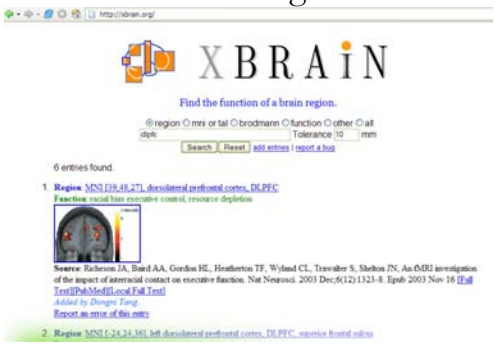
```
# voxels structure
```

```
10 --TOTAL # VOXELS--
10 Left Cerebrum
10 White Matter (L)
6 Frontal Lobe (L)
5 Sub-Gyral (L)
4 Extra-Nuclear (L)
4 Sub-lobar (L)
1 Inferior Frontal Gyrus (L)
```

- **Help Menu**



1 Go to xBrain.org website.



xBrain.org is a growing database where you can find the functions of a certain brain region. You can search by anatomical name or coordinates. With full access, you can see the pictures of that region and the full-text source paper, etc.

2 xjView help

Chapter

2

ANALYZE PART

Before you read the manual below, you may want to download the sample data and play with them while reading. **The sample data are fake and are only for demonstration purpose.** Download sample data at (http://people.hnl.bcm.tmc.edu/cuixu/xjView/sampledata.zip) (42M). Also, you need to change the value of TR in the beginning of xjview.m.

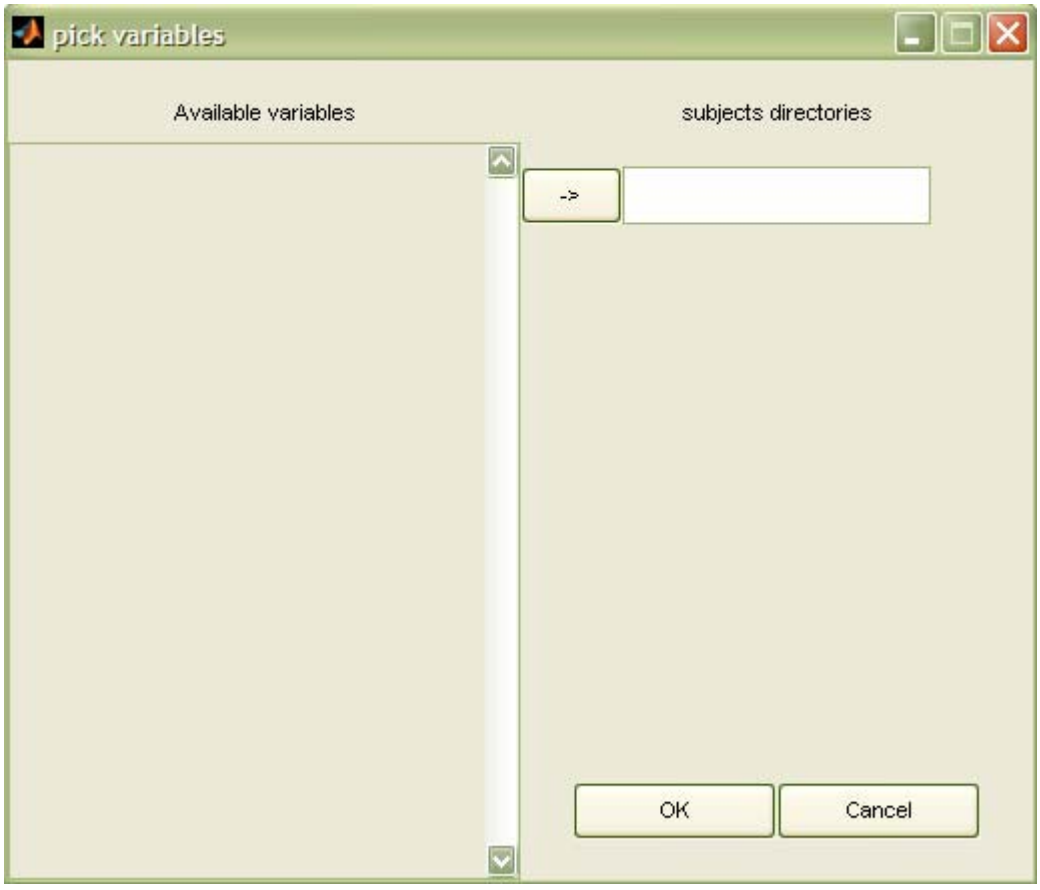
The sample data are preprocessed analyze files. It contains one folder, `sampledata`. Inside this folder, you will find three more folders, called `subject1`, `subject2`, and `subject3`. You will also find several mat files. `s?.mat` pointing the imaging data for each subject, `e?.mat` containing the event timing and `c?.mat` containing the event modulator for each subject. `p.mat` contains three variables, `ps`, `pe` and `pc` which point to the above mat files. Please load each mat file to see what is inside. Under each subject's folder, you will see a directory called `image` which contains the preprocessed imaging files.

You will find that the path I used in the variables (e.g. `ps`) is my local linux path, which is probably different from that in your local computer. It's easy to change using spm_get or spm_select – see Appendix (Others).

The task associated with this sample data is: flashes and beeps are presented briefly to each subject in alternation. The timing of events (saved in `e?.mat`) are not necessarily evenly spaced. Brightness of flashes and intensity of beeps vary and they are saved in `c?.mat`.

Preprocess: Preprocess data. (Converting IMA to analyze files (img and hdr files), motion correction, coregister, slice-timing, normalization, spatial smooth etc). I don't recommend people from other labs to use this function for two reasons: (1) The original imaging data may not be in DICOM format and (2)

You might have different preference such as voxel size (xjView produces 4x4x4mm voxels).



To preprocess data, select (or type) the variable (in the base workspace) whose content is the directory of each subject. This variable is usually obtained by using `spm_get`. For example, `P=spm_get([-40:0]);` This variable is a string array, with each row the directory name of each subject. For example:

```
P =
/mnt/nfs/proj/timing/subjects/AC081704
/mnt/nfs/proj/timing/subjects/BD071304
```

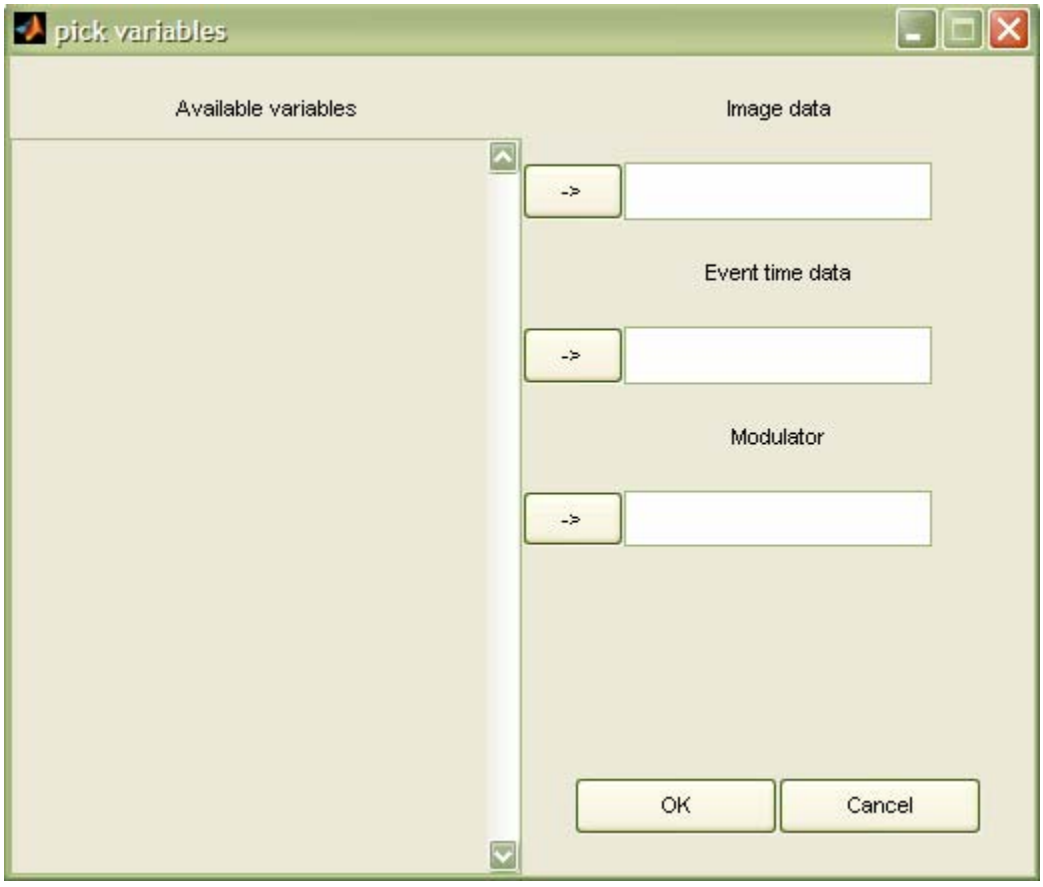
Please note: xjView assumes that you have a `"data"` directory, which contains the raw DICOM files, under each subject's directory. xjView will automatically create a directory called `"preprocessing"` under each subject's directory to hold the preprocessed image files. xjView will also save the head movement data to `headmovement.mat` and the 4-D image data matrix(s) (for example, `0003.mat`. The names of these files are session number.) in the `preprocessing` directory. These mat files can be used in future ROI analysis to save time. You may want to copy these mat files into a single directory later.

The head movement file contains 6 columns, corresponding to the head movement in each translational and rotational direction.

Unfortunately the sample data were already preprocessed so you can't play this function using the sample data.

Process: Simple GLM estimation on detrended BOLD signals. This function allows you to quickly get the beta values of each voxel. In the end I still recommend you to do SPMProcess (below) to confirm your findings.

Note: If you have more than one independent variable, this function doesn't do multi-variant GLM estimation. Instead, it does multiple uni-variant linear regressions. We do regression this way for the following reason: if multi-variant linear regression were done where two of the independent variables are correlated, then one of which will absorb the variance and the other get little. Then it's unfair to say that the second variable doesn't explain the data -- it does.



Select (or type) the 3 variables (`ps,pe,pc,` for example. You may load `p.mat` under `sampledata` directory to get examples of these variables) whose contents are the mat file names of the image data (resulted from preprocessing), the event onset time, and the modulator (or correlator or regressor, you may call it), respectively. The third variable is optional. If you don't have modulator, leave the field blank. For example:

```
load p.mat;
ps =
/mnt/nfs/proj/gang/xjview/sampledata/s1.mat
/mnt/nfs/proj/gang/xjview/sampledata/s2.mat
...
pe =
/mnt/nfs/proj/gang/xjview/sampledata/e1.mat
/mnt/nfs/proj/gang/xjview/sampledata/e2.mat
...
pc =
/mnt/nfs/proj/gang/xjview/sampledata/c1.mat
/mnt/nfs/proj/gang/xjview/sampledata/c2.mat
...
```

By loading `s1.mat`, you will get a string matrix with each row an imaging file (e.g. `001f.img`). By loading `e1.mat`, you will get a structure variable (e.g. `e`); each component is the onset times of a type of event (For example, `e.flash=[5 23 43 ...], e.beep=[17 34 ...]`. In this example, you have two types of events (flash and beep), event flash has onset time `5, 23`, etc.). The onset time should be in the unit of second. By loading `c1.mat`, you will get a structure variable (e.g. `c`), each component is again a structure. For example

```
c.flash.brightness = [3 3 1 4 ...]
```

Now you see, the first order component of `c` has to be the same with the event type. The size of each modulator should be the same as the size of event onset times.

If you don't have a modulator for event flash (for example), then don't specify it in `c`.

Note: If your events is not pulse-like and they last for a period of time (e.g., `8` seconds), then you can specify the duration in variable `c`. For example,

```
c.flash.duration = [8 8 7.5 4 ...]
```

Again, the length of this vector has to be exactly the same with the length of event. Here `duration` (case sensitive) is a special name. Other names such as '`dur`', '`Duration`' or '`interval`', etc. will not be recognized as the duration of the events; xjView will treat them as to modulate the height of hrf, just like `brightness`.

Then select the directory where you want to put the result beta files. The output of "process" is beta value in imaging files (for example, `beta_flash_s1.img`) for each subject.

In the sample data, I saved the variables `ps`, `pe` and `pc` to `p.mat` and I also saved the beta images to directory `beta_process` under `sampledata` directory. You may want to check them out.

SPMProcess: Multi-variant GLM estimation (not multiple uni-variant) using standard SPM procedure.

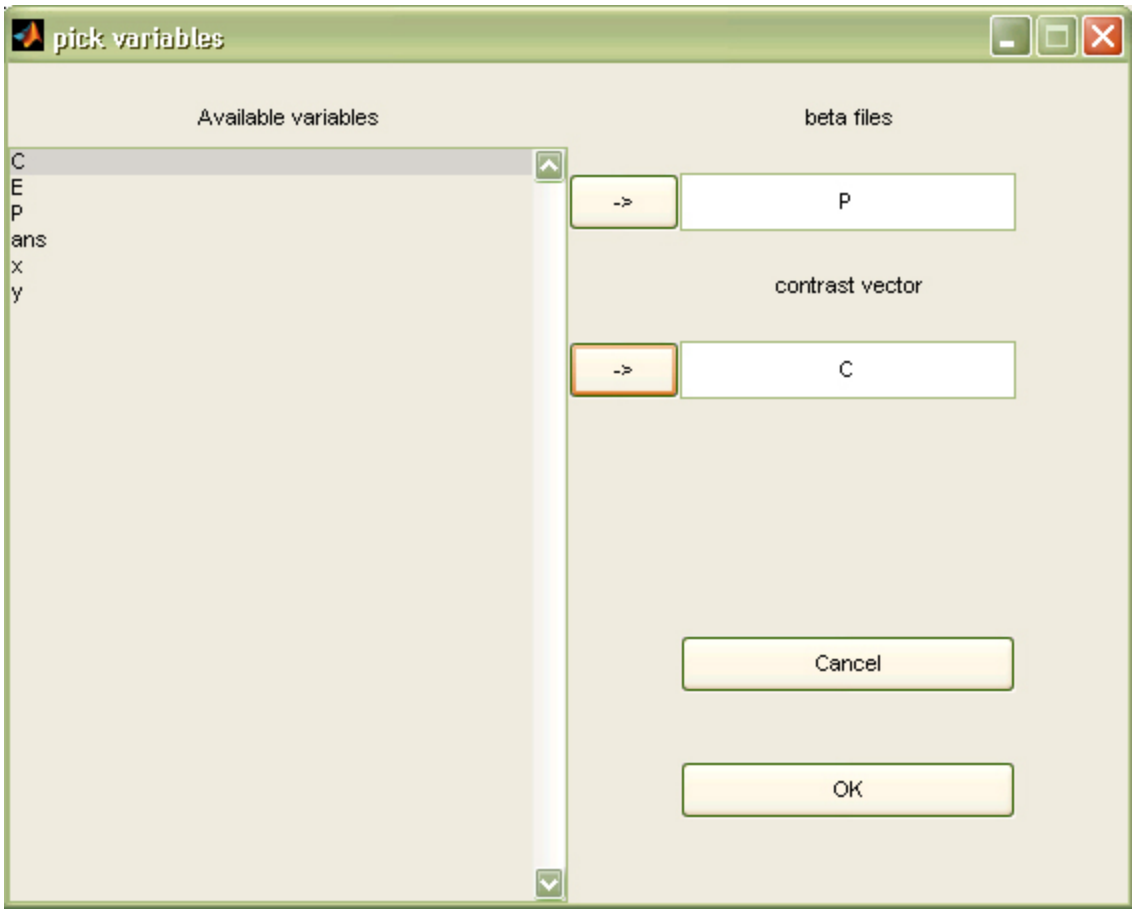
The input requirement is almost identical to Process, except you can also put head movement data there. The headmovement data is of the same format as the event data. For example, h.x = […], h.y=[…]. The length of each vector should be the same as the length of imaging data. The output is standard SPM output files such as beta images, SPM.mat, etc. The output files are saved a directory whose name is given by you under each subject's directory. In the sample data, the output files are saved in a directory called '`FixedEffect`'.

GLM on peak BOLD: This function is designed to find which regions in the brain correlates with the modulator of your events rather than the events themselves. For example, in some circumstances you might be interested in "which region correlates the brightness of flash?" instead of "which region is activated during flash?". Again, the function allows you to quickly find brain regions which may interest you.

xjView assumes the peak of hrf is at 6s after an event onset. It averages the BOLD signal at 4s and 6s after an event onset to get peak BOLD. Then it correlates this peak BOLD with the modulator to get the betas. It saves the betas as imaging files.

The input requirement of this function is the same with "Process" except that you have to input the 3rd variable, modulator.

Contrast: Contrast between betas and t-test



Select (or type) the 2 variables (`pcontrast` and `c`, for example. You may load `p.mat` under `sampledata` directory to get examples of these variables) whose contents are the beta files and contrast vector, respectively.

`pcontrast` is a cell array, with each component a list of beta files: e.g.

```
pcontrast{1}=
/mnt/nfs/sampledata/beta_process/beta_flash_s1.img
/mnt/nfs/sampledata/beta_process/beta_flash_s2.img
/mnt/nfs/sampledata/beta_process/beta_flash_s3.img
pcontrast{2}=
/mnt/nfs/sampledata/beta_process/beta_beep_s1.img
/mnt/nfs/sampledata/beta_process/beta_beep_s2.img
/mnt/nfs/sampledata/beta_process/beta_beep_s3.img
```

I usually use `spm_get` to get the beta files. For example, `pcontrast{1}=spm_get;` You can see that each row is for one subject. If you only have one subject, then there is no point to do a t-test.

`c` is a vector: e.g.

```
c = [1 -1]
```

The length of `pcontrast` and `c` should be the same. In the above example, a one sample T test will be done on the difference of `pcontrast{1}` and `pcontrast{2}`. This contrast will allow you to find which region is more activated during flash than during beep (and vice versa).

Then xjView asks you where to save the resulted T-test image. After you are done, xjView automatically loads the T-test images.

In the sample data, I saved the resulted t-test image to `flashbeep.img` under `sampledata` directory.

Component of `c` is allowed to be 0. For example, `c=[1 0 0 -1]` (of course, the length of `pcontrast` should be 4 in this case) means doing a t-test on the difference of `pcontrast{1}` and `pcontrast{4}`.

FDR: (false discovery rate) xjview will calculate the FDR-corrected p-value. First, you input an FDR value (e.g. 0.05) in the pValue edit box, then you click Analyze|FDR. You will find the both the displayed images and the pValue change. These are FDR-corrected images and p-value.

ROI: retrieve signal: Retrieve BOLD from the currently displayed voxels.

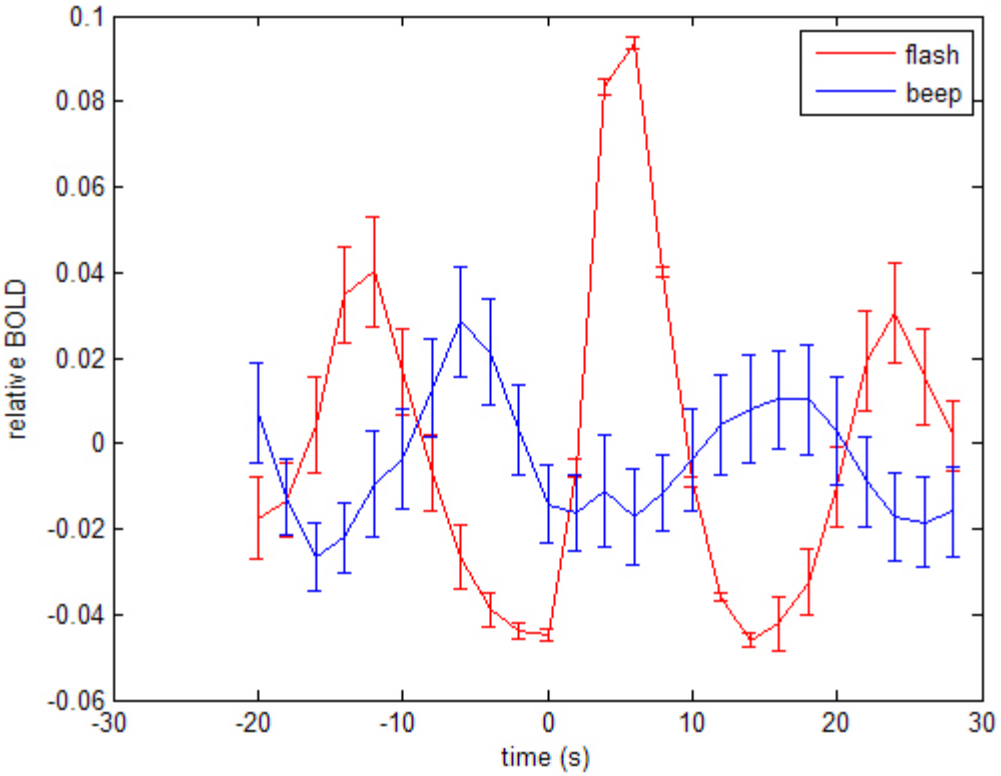
Almost the same with "Process", except that you don't input modulator, but input correlator. (Indeed modulator and correlator are the same.) This correlator is optional. You can also remove the unwanted effects of head movement, for example. Note, signal from the current displayed voxels will be retrieved. The result will be saved in a mat file.

In the sample data, you will find a file called `1.img`. Open it with xjView. Then run **ROI:retrieve signal** and input `ps`, `pe` and `pc` to the dialog box. Then give a name of the result file (I use `ROI_region1`). After you finish, you will find xjView produces a figure of time series. Upon your selection, xjview will also plot your the peak BOLD signal against the correlator (e.g. BOLD against flash brightness).

You will find a file called `ROI_region1.mat`. You are encouraged to load this file and see what is inside.

You may also repeat with the region in `2.img`.

ROI: plot: Plot the result produced by "ROI: retrieve signal". For example, click ROI: plot and select `ROI_region1.mat`, you will get



You may find that the BOLD signal is really huge. That's because the data are fake. In real data, a BOLD signal of height 0.005 is fairly big.

ROI: individual plot: Similar to ROI: plot but this is done to each individual subject. This allows you to investigate the individual variance of your data.

ROI: individual plot with behavior: For each subject, the raw BOLD signal is plotted in black. On top of it, the timing of each event is plotted. If you input the head movement data, the head movement is also plotted in gray. This helps you to find out the abnormality of your data.

ROI: correlation plot: Plot correlogram between regions. The input should be the result produced by "ROI: retrieve signal".

For example, click "ROI: correlation plot" and select `ROI_region1.mat` and `ROI_region2.mat`, you will find the correlogram of the signals in the two regions. If the correlogram peak is at negative time shift, it means region1 is leading region2.

If you input only one ROI mat file, xjView will produce an auto-correlogram. If you input more than two ROI mat files, xjView will produce correlogram between every two regions.

Behavior analysis: For each subject, plot timing of each event, the Fourier power of the events, and the correlogram of the intervals between events. This helps you to check if there is periodicity in the event train. If head movement data is input, the head movement is also plotted on top of the event timing plot.

If you input the correlator data, xjview will plot two correlators (your choice) against each other. For example, you have one type of event called flash. Flash has two properties (or correlators), intensity and trial number. Then xjview can plot how intensity changes with the trial number.

Head movement analysis: For each subject, plot head movement data. This helps you to find subjects who moved a lot during the experiment.

For HNL Only Menu: Functions under "For HNL Only" menu are highly dependent on the hardware of Human Neuroimaging Laboratory (HNL) so people outside HNL might not find them useful.

Format convert: Change filename of data so that xjView can preprocess it. If you find your image files are collected in several directories with each directory the session number, you should use this function. Files are directly exported from scanner don't need to converted.

Preprocess, Process and SPM Process: Same with the menus in Analyze Menu, except that the jobs will be done in HNL cluster. After you specify the variables, you will be asked to input your password to log in cluster.hnl.bcm.tmc.edu. Type your password, and then press enter **twice**. You will find your job is being submitted.

xjView creates a temporary folder called "`xjviewtmp`" under your current working directory. Inside the folder xjView stores the script files which are used to submit jobs. The status of the jobs (OU files, ER files and log files) will also be saved here. OU files are output files, ER files are error files. You need to check *.ER files and *.log files to see if your jobs are finished successfully or terminated by an error.

You should also log in to cluster.hnl.bcm.tmc.edu and run `pbstop` to see the status of your jobs. Jobs which are terminated by errors usually end within several seconds. So if a job stays there for more than one minute, you job should be fine.

Jobs may not run successfully in cluster. The most common error is "`permission denied`" error which for some reason can't be solved. In that case, you need to submit the corresponding job again. Fortunately you don't need xjView to submit it. First, identify the unsuccessful jobs. For example, you may find job 1, 2, 4 and 9 are failed. Then go to `xjviewtmp`, you will find two *.pbs files. One of them ends with *ALL.pbs. Edit this file using e.g. `emacs`: Change the for loop line

```
for ((s=1; s<?; s++))
```

to

```
for s in 1 2 4 9
```

Then run command `bash ./????ALL.pbs` (you should replace `????` with the actual file name).

Chapter

3

FREQUENTLY ASKED QUESTIONS

† I have a mask file (`mymask.img`) and I want to know where the masked region is. What should I do?

type `xjview mymask.img` in the command window of MatLab. Then put the cursor in glass view (or section view) to the displayed region.

† I have a 23x3 matrix (called A) with each row the MNI coordinate of a voxel. How do I know where the region is?

type xjview(A) in the command window of MatLab. Then put the cursor in glass view (or section view) to the displayed region. You may need to change the presettings of xjView in the beginning of xjview.m.

† I have a image file which is the result of a GLM contrast. I want to make a mask of the activated region. How can I do it?

load the image file, change p-value (and cluster size threshold), put the cursor to a region of interest, click "Pick cluster/Info" and save the image by clicking menu “File|Save Current Image as Mask....”.

† How many image files can I load at the same time?

100. But we only have 6 different colors to code them. So it is advisable to load 6 images at the most.

† I loaded multiple images, can I change p-Values?

Yes. xjview will display the supra-threshold voxels for each image at the same time.

† Can I load a T test image, an F test image, a mask image, and another image file from an unknown source at the same time?

Yes. But don't confuse yourself before confusing xjView.

† There is an error when I choose some target images in the section view (such as ch2, ch2bet, aal and brodmann). What happened?

Those images are not provided by SPM. You may want to check out MRIcro. Copy the corresponding image files (unzipped, with header files) to your spm canonical directory.

† Are the coordinates in xjView MNI or TAL coordinates?

MNI.

† My image's left/right get flipped in xjView (compared to the display in SPM2). What happened?

You need to change the presettings in the beginning of xjview.m.

† My xjView window size is abnormal. How can I change it?

You need to change the variable called 'figurePosition' in xjview.m. The 3rd and 4th component controls the width and height of xjView.

† What is xBrain?

xBrain (http://www.xBrain.org) is a brain mapping database where you can search a brain region and find out what the function is. Everybody can contribute the database by adding entries.

APPENDIX: USEFUL FUNCTIONS

The following MatLab/SPM functions are frequently used to create the variables required by xjView. Familiarity with these functions allows you to use xjView efficiently. Here I list some examples which you may want to play with in MatLab command window.

† **spm_get** (in SPM2)

`spm_get` allows you to get a list of files (or folders) quickly and save the result as a string array with each row a file name or folder name. I myself think it's the most useful tool provided by SPM.

```
% select some files interactively and save the file names into a variable P
P = spm_get
% select 6 (not 5, not 7, exactly 6) files interactively
P = spm_get(6)
% select 2 to 4 files interactively
P = spm_get([2:4])
% select 0 to 8 files interactively
P = spm_get([0:8])
% automatically get all imaging files in the directory c:\abc
P = spm_get('Files', 'c:\abc', '*.img')
% automatically get all mat files starting with e in the current working directory
P = spm_get('Files', pwd, 'e*.mat')
% automatically get the mat files in the directory c:\abc which satisfy the
following condition: the file names end with f and another two characters. Pay
attention to the usage of * and ?
P = spm_get('Files', 'c:\abc', '*f??.mat')
% select some (<200) folders interactively. Now you see, negative numbers means to
select folder instead of files
P = spm_get([-200:0])
```

When you select files interactively, you can specify the filter in the 'spmget' window to filter out unwanted files. For example, if all your event onset data files start with character 'e', then type "e*" into the filter edit box and press enter. You will find only files with 'e' in the beginning are listed. If the listed files are exactly the files you need, you may click 'All' to select those files all in once instead of clicking them one by one. For this reason, when I create data files, I always prefix character 's' to imaging data files, 'e' to event onset files, and 'c' to correlator or modulator files, meaning 'signal', 'event', and 'correlator', respectively.

† **spm_select** (in SPM5)

For unknown reasons `spm_get` is no longer available in SPM5 (It's one of the reasons why I'm still using SPM2.). Instead, SPM5 has `spm_select` which does the similar job with `spm_get`.

```
% select some files interactively and save the file names into a variable P
P = spm_select
% select 6 (not 5, not 7, exactly 6) files interactively
P = spm_select(6)
% select 2 to 4 files interactively
P = spm_select([2 4])
% automatically get all imaging files in the directory c:\abc. Files are saved in
P and directories are saved in dirs
[P, dirs] = spm_select('list', 'c:\abc', 'img')
```

You will find both `spm_get` and `spm_select` return a string array with each row a file name. White spaces were added to the shorter file names so that the length of each row is identical.

† **save**

`save` allows you to save variables from your work space to hard disk for future use.

```
% save variable P to a file named eventfile.mat
save eventfile P
save('eventfile','P')
% save three variables, s, e, c to a file named datafile.mat
save datafile s e c
save('datafile','s','e','c')
% assuming you are using matlab 7, save three variables, s, e, c to a file named
datafile.mat which can be loaded by matlab 6 in the future
save datafile s e c –v6
```

† **load**

`load` allows you to load the variables from hard disk to your work space.

```
% load a data file called datafile.mat into workspace
load datafile
load datafile.mat
load('datafile')
```

† **deblank**

`deblank` removes the tailing white spaces of a string

```
% remove the tailing white spaces of a string
deblank('   abc d    ')            % the result is '   abc d'
% remove the tailing white spaces of a the 4th file in P
deblank(P(4, :))
```

† others

```
% remove the 3rd file from P
P(3,:)=[]
% remove the 3rd, 4th, 8th files from P
P([3, 4, 8],:)=[]
% remove the last files from P
P(end,:)=[]
% get the size of P
size(P)
% get the number of files in P
size(P, 1)

% replace / with \ in P (maybe you want to change the file names from linux
convention to windows?)
pos=find(P=='/')
P(pos)='\'
% remove the first 5 characters from each row of P
P(:, 1:5)=[]
% add c:\abc\ to the beginning of each row of P. You may also want to play with
repmat
P=[repmat('c:\abc\', size(P,1), 1), P]
```

The last 4 lines of commands allow you to change the linux path to windows path.