func2roi-sess, roisummary-sess

Comments or questions: analysis-bugs@nmr.mgh.harvard.edu
$Id: func2roi-sess.tex,v 1.1 2005/05/04 17:00:48 greve Exp $

1 Introduction

func2roi-sess is an FS-FAST program for averaging functional time courses across select spatial voxels, the Region of Interest, or ROI. Currently, there is only support for averaging the results created by **selxavg-sess**. The key to understanding **func2roi-sess** is to understand the many ways to select the voxels which constitute the ROI. Broadly speaking, the ROI can be constrained both structurally and functionally. For example, the user may want to constrain the ROI to be within fusiform gyrus. However, there may be many voxels in this structure that are not activated by the task at hand, and averaging the non-active voxels with the task-related voxels will water down the significance of the ROI. Thus, **func2roi-sess** allows the user to specify a minimum level of functional activation before a voxel can be included in the ROI. *Note that care must be taken not to evaluate the significance of the ROI using the same functional criteria which which the ROI was chosen as such a statistic will have elevated false-positive rates.* In what follows, the structural constraint will be referred to as the *label*. The functional constraint will be referred to as the *functional mask*. The intersection of these two regions will be referred to as the *final mask*.

2 Usage

Typing func2roi-sess at the command-line without any options will give the following message:

```
USAGE: func2roi-sess
Options:
   -roidef   name : name of ROI definition
   -analysis name : source is averaged data from analysis
   -raw           : source is raw data
   -rawfsd        : functional subdirectory for raw data (bold)
   -motioncor     : use motion corrected raw data (with -raw)
   -rawstem       : stem of raw data (f or fmc)
   -rawrlf        : run list file for raw data
   -noraw         : don't process raw data
   -sesslabel name : use name.label found in sessid/labels
   -anatlabel name : use name.label found in SUBJECTS_DIR/subject/label
   -labelfile file : give full path of label file
   -labelspace space
   -labelspacexfm xfm file found in mri/transforms (talairach.xfm)
   -labelfillthresh thresh: fraction of voxel that must be filled with label points
   -maskcontrast contrast: contrast to use for mask
   -maskthresh   threshold
   -masktail     tail : thresholding tail (<abs>, pos, neg)
   -maskmap      map     : map to use for mask <sig>
   -maskframe    frame   : 0-based frame number in map <0>
   -masksessid   sessid : sessid of mask (default is that of source)
   -maskanalysis name : analysis of mask (default is -analysis)
   -maskspace    space : space of mask (<native> or tal)
```

```
-maskisxavg   effect: fixed or random (when masksessid is a group)
-float2int method: method = <tkreg>,round,floor
-sf sessidfile  ...
-df srchdirfile ...
-s  sessid      ...
-d  srchdir     ...
-umask umask   : set unix file permission mask
-scriptonly    : don't run, just generate a script
-version       : print version and exit
```

2.1 Specifying the Structural Constraint

The structural constraint is specified using a "label file". This file can be created by **tkmedit** or **tksurfer** and always has the extension '.label". The label file is just a list of the xyz coordinates of 1 mm^3 voxels to include in the label. These coordinates can be defined in either a subject's native anatomical space or in talaiarch space, the latter being more convenient when the same label will be used across multiple subjects. **func2roi-sess** is designed to easily accommodate either method.

A per-subject label file can be specified in one of two ways, depending upon where the file is stored. It can be stored within the functional session directory in a subdirectory called "labels"; this type is specified with *-sesslabel* command-line flag followed by the label name. The label file can also be stored in the subject's FreeSurfer anatomical directory in a subdirectory called "label"; this type is specified with *-anatlabel* command-line flag followed by the label name. Note that the label name SHOULD NOT include a ".label" extension.

If a single label is to be applied across all sessions, use the *-labelfile* command-line flag followed by the full path of the label file (WITH the ".label" extension).

If the label was constructed in talairach space, then include the *-labelspace tal* option. Note also that a label does not have to be purely structural. For example, one can display a functional map when choosing label voxels with **tkmedit** or **tksurfer**, and only tag those voxels that are active. If using this method, take care not to evaluate the significance of the ROI with the same criteria used to select the voxels. Note also that a structural criteria need not be specified at all.

Each label point occupies 1 mm^3 of space whereas each functional voxel typically occupies about 45 mm^3. This creates a question as to how many label points must fall into a voxel for that voxel to be considered part of the label. By default, a voxel will be included in the label if any label point falls into it. This can be helpful when creating labels by hand because only a few points need to be specified (ie, it allows for sparse labels). On the other hand, if two labels are adjacent to each other, the the voxels on the border will end up belonging to both labels. To avoid this, users can specify the minimum fraction of a voxels volume that must be filled with label points for that that voxel to be considered part of the label (this is equivalent to setting a minium number of label points that must fall into a voxel). This fraction is specified with the *-labelfillthresh* flag followed by the fraction to use (eg, -labelfillthresh 0.5). The fraction should be between 0 and 1 (0 is equivalent to not having a label constraint). Setting the threshold to 0.5 assures that a voxel cannot belong to more than one label. Note, however, that this can have undesirable effects when the label is defined on the surface because there might not be enough surface points in a voxel to reach threshold.

2.2 Specifying the Functional Constraint

The basis of the functional constraint is usually a contrast map as computed by **stxgrinder-sess**. The contrast can be obtained from an individual session or the average of a group of

sessions. If a group is used, the contrast from a fixed or random effects averaging can be used, and the results from one analysis/contrast can be used to create an ROI for another analysis. The per-voxel threshold is chosen using the *-maskthresh* flag followed by $-log10$ of the significance threshold (eg, a threshold of .01 would require *-maskthresh 2*). A voxel can also be included or excluded based on the sign of the significance at that voxel using the *-masktail* flag followed by either *abs* for absolute value (ie, ignore sign), *pos* to include only voxels with positively signed significances, or *neg* to include only voxels with negatively signed significances. **stxgrinder-sess** typically produces three significance maps: *sig*, *minsig*, and *fsig*, any of which can be used for the functional mask using the *-maskmap* option followed by the name of the map. For event-related designs, the *sig* map will have multiple frames, and a particular frame can be chosen using the *-maskframe* option followed by the zero-based frame number. By default, **func2roi-sess** will search for the contrast within the analysis for which the ROI is being computed (ie, the argument of the *-analysis* flag). The contrast from a different analysis can be used by specifying the *-maskanalysis* flag. Finally, if the contrast is based on a group average, the group name (as supplied to **isxavg-fe-sess** or **isxavg-re-sess**) is specified using *-masksessid*. Note that it is not necessary to have a functional mask at all.

3 Output

When **func2roi-sess** is run, it creates a new directory in each session under the analysis subdirectory for that session. The name of this directory is the argument passed with the *-roidef* flag. There will be several files within this directory representing the results of the ROI analysis. There will be a file called *h_000.bfloat*. This is where the hemodynamic averages (as computed by **selxavg-sess**) averaged within the ROI are stored. There will also be *h-offset_000.bfloat* in which the average functional value within the ROI is stored. Note that these are the same types of files found in the analysis directory and in the resampled (ie, tal and sph) directories. In fact, one can think of ROI analysis as a type of resampling, and many of the FS-FAST programs that can be applied to resampled spaces (eg, **stxgrinder-sess**, **sliceview-sess**, **isxavg-fe-sess**, and **isxavg-re-sess**) can be applied to the ROI. The ROI "volume" is actually just a single voxel (ie, one slice with one row and one column). In addition to the ROI averages, there is a volume called "mask" which has the same dimension as the original functional volume. A voxel in this volume has a value of 1.0 if that voxel was included in the *final mask* and a value of 0.0 otherwise. Thus the final mask can be viewed with any FS-FAST program used to view a contrast just by specifying *-contrast "roidef"* ("roidef" is the name given to the ROI in) and *-map mask*. The minimum threshold should be set to 0.5 when viewing. Programs that can be used to view the mask are: **sliceview-sess**, **tkmedit-sess**, and **paint-sess**/**surf-sess**. Viewing the final mask can be helpful in assuring that the structural and functional constraints result in an ROI that is close to expectations.

4 Getting Results in ASCII Format

Often, one wants to pool all the ROI averages across sessions into a single table so that they can be read into a statistics package for more elaborate analysis. The results from **func2roi-sess** can be collected into such a table using **roisummary-sess**, which has the following options:

```
USAGE: roisummary-sess
Options:
   -sumfile filename : name of file in which to store the summary
   -roidef   name : name of ROI definition
```

```
-analysis name : source is averaged data from analysis
-transpose     : put sessid info on a row instead of a col
-excludesessid : do not include sess id in table
-sf sessidfile  ...
-df srchdirfile ...
-s  sessid      ...
-d  srchdir     ...
```

Use *-sumfile* to specify the name of the file in which to store the ascii results. The ROI is specified with *-roidef*. Without any other options, **roisummary-sess** will get the ROI results from each session and store them into the summary file. Each session will have its own column, and each row represents a different measurement according to the following scheme:

1. *sessionid* - text identifier of the session
2. *nlabel* - number of functional voxels in the structural label regardless of whether they met the functional criteria.
3. *nroi* - number of functional voxels in the final mask (ie, ones that met the structural and functional criteria.
4. $offset$ - mean functional baseline offset within the ROI (good for computing percent signal changes)
5. *eresstd* - standard deviation of the residual error. This is a measure of the variance left unexplained by the task.
6. DOF - degrees of freedom in *eresstd*. This can be used to asses whether there is a difference among the sessions in the DOF as could happen if some runs were excluded in some sessions and not in others.
7. TER - temporal resolution (in seconds) of the hemodynamic estimation. This is not a data value. This value is set at the time of **mkanalysis-sess** and is included here to facilitate plotting the results that follow. All sessions should have the same value.
8. $tPreStim$ - time (in seconds) of the prestimulus baseline used in hemodynamic estimation. This is not a data value. This value is set at the time of **mkanalysis-sess** and is included here to facilitate plotting the results that follow. All sessions should have the same value.
9. *nconditions* - number of conditions (excluding null condition). All sessions should have the same value.
10. *nestspercond* - number of estimates per conditions. For block analyses, this will have a value of 1. For event-related analyses, the value will equal to the number of points in the time window.
11. There will be $nconditions \times nestspercond$ numbers starting at this row. The first set of *nestspercond* numbers are the values for the first condition, the second set of *nestspercond* numbers are the values for the second condition, etc. Percent signal change can be obtained by dividing these numbers by $offset$ and multiplying by 100.

The format of this table can be modified slightly with the *-transpose* and *-excludesessid* flags. The *-transpose* flag creates the same table except that the data for the session is placed on a row instead of in a column. The *-excludesessid* flag will eliminate the *sessionid* row from the data. The *-excludesessid* is provided because some software does not allow data tables with text and numbers.

5 Examples

5.1 Example 1: Per-Session Label and Functional Mask

Let's say that you have two sessions, s1 and s2, for which you have already run individual event-related analyses (main2) as well as a contrast (allvfix). The analysis has 2 conditions, with $TER = TR = 2$sec $tPreStim = 4$sec, and $TimeWindow = 18$sec. The allvfix contrast gives you an indication of which voxels are activated by the overall task. A neuroanatomist has graciously labeled the fusiform and striate regions of each subjects' brain giving you a total of 4 label files. You create a directory in s1 called "labels" and put the fusiform.label and striate.label for subject 1 into this directory. You create a directory in s2 called "labels" and put the fusiform.label and striate.label for subject 2 into this directory.

5.1.1 Part 1 - Fusiform with Positive Activation

```
func2roi-sess -roidef fusi-avf-pos-2 -analysis main2
  -sesslabel fusiform -maskcontrast allvfix -maskthresh 2
  -masktail pos -maskmap minsig -s s1 -s s2 -d .
```

This will create an ROI named "fusi-avf-pos-2" which uses the fusiform.label in each subjects' labels directory along with the allvfix contrast as the functional constraint. Only voxels inside the pre-defined fusiform label that exceed a significance of 10^-2 in the minsig map and are positively correlated with the task will be selected. Once the final ROI is determined from the functional and structural constraints, **func2roi-sess** will average the ROI voxels from the main2 analysis together, and create a directory called "fusi-avf-pos-2" under main2 (in each session).

At this point, there are four things you can do next: (1) pool the data into a text file, (2) view the hemodynamic response in the ROI for each session, (3) view the final mask, or (4) inter-subject average.

Pooling the data into a text file. Here you will run **roisummary-sess** to create a text file of all the ROI data for both session 1 and 2:

```
roisummary-sess -sumfile fusi-avf-pos-2.dat -roidef fusi-avf-pos-2
  -analysis main2 -s s1 -s s2 -d .
```

This will create a file called "fusi-avf-pos-2.dat" in which you will find two columns. The contents will look something like the following (not including the text after the #'s):

```
s1        s2          # session identifiers
480       467         # number of functional voxels in the fusiform
127       177         # number of active voxels
1117.06   1242.86     # average baseline activity
8.24      7.29        # std dev of residual error
574       574         # DOF
2.0       2.0         # TER = TR
4.0       4.0         # tPreStim
3         3           # number of conditions
9         9           # number of estimates per condition
 0.1746   -1.5937     # condition 1,  4 sec before stimulus onset
-0.1867   -1.4410     # condition 1,  2 sec before stimulus onset
 0.7258    0.5711     # condition 1,  stimulus onset
-0.2199   -0.2157     # condition 1,  2 sec after stimulus onset
```

```
 3.1832    1.1900     # condition 1,  4 sec after stimulus onset
 0.4660    1.1168     # condition 1,  6 sec after stimulus onset
 0.3595    0.8347     # condition 1,  8 sec after stimulus onset
 1.1501    1.3319     # condition 1, 10 sec after stimulus onset
 0.0847    0.6813     # condition 1, 12 sec after stimulus onset
-0.0956    1.1908     # condition 2,  4 sec before stimulus onset
-0.8323   -1.2025     # condition 2,  2 sec before stimulus onset
 0.2944   -0.0198     # condition 2,  stimulus onset
-0.9678    0.0275     # condition 2,  2 sec after stimulus onset
 1.7143   -1.1041     # condition 2,  4 sec after stimulus onset
 2.2259    0.5585     # condition 2,  6 sec after stimulus onset
-0.4462   -0.9337     # condition 2,  8 sec after stimulus onset
 0.9413    1.4568     # condition 2, 10 sec after stimulus onset
 1.2794   -0.7924     # condition 2, 12 sec after stimulus onset
```

View the hemodynamic response. To view the hemodynamic response for this ROI for session s1,

```
sliceview-sess -s s1 -d . -analysis main2 -space fusi-avf-pos-2 -slice 0
```

This will bring up what looks like one big voxel. Click on it and hit 'g' to view the hemodynamic response.

View the final mask. The final mask can be visualized with either sliceview, tkmedit, or tksurfer. To view in sliceview, type:

```
sliceview-sess -s s1 -d . -analysis main2 -contrast fusi-avf-pos-2
  -map mask -slice mos -thmin .5 -thmax 1
```

This will bring up a mosaic in which the base image will be the average functional value and the voxels selected for the ROI as a yellow overlay.

```
tkmedit-sess -s s1 -d . -analysis main2 -contrast fusi-avf-pos-2
  -map mask -fthresh .5 -fmid 1
```

This will bring up the ROI overlaid onto the 3D anatomical for that subject.

```
paint-sess -s s1 -d . -analysis main2 -contrast fusi-avf-pos-2
  -map mask
surf-sess -s s1 -d . -analysis main2 -contrast fusi-avf-pos-2
  -map mask -fthresh .5 -fmid 1
```

This will paint and display the final mask on the surface.

5.1.2 Part 2 - Fusiform with Negative Activation

The first part of example 1 showed how to obtain voxels in the fusiform label that were positively correlated with the task. Here, we show how to use the same label and contrast to define a new ROI in which the activity in the voxels is negatively correlated with the task:

```
func2roi-sess -roidef fusi-avf-neg-2 -analysis main2
  -sesslabel fusiform -maskcontrast allvfix -maskthresh 2
  -masktail neg -maskmap minsig -s s1 -s s2 -d .
```

Note the option *-masktail neg* instructs **func2roi-sess** to use the negatively correlated voxels, and we have changed the name of the ROI to "fusi-avf-neg-2". This will create a directory called "fusi-avf-neg-2" under main2 in each session. The remainder of Part 1 applies to Part 2; all you need to do is substitute "fusi-avf-neg-2" for "fusi-avf-pos-2".

5.2 Example 2: Cross-Session Label

Your friendly neuroanatomist has left you for some other brain-mapper because you insist on using sluggish, command-line-driven functional analysis tools. But you need one more ROI. You'll have to draw it yourself. You know about where Broca's area should be in the Talairach brain, so you pull up the Talairach brain in tkmedit:

```
tkmeidt talairach T1
```

You can create the label by pressing the middle mouse button. The marked voxels will appear green. When finished, pull down the File menu and select "Save Label As". Hit the little folder icon, and save the label in your study directory as "broca.label". Exit out of tkmedit. Cd to your study directory and make sure there is a file called "broca.label".

```
func2roi-sess -roidef broca-avf-pos-2 -analysis main2
  -labelfile broca.label -labelspace tal -maskcontrast allvfix
  -maskthresh 2 -masktail neg -maskmap minsig -s s1 -s s2 -d .
```

Note that *-sesslabel fusiform* has been replaced with *-labelfile broca.label* (also note the ".label" extension) and the addition of *-labelspace tal*.

6 Soon-To-Be Frequently Asked Questions

Do I have to have a structural mask and a functional mask ? No. You have to have one or the other, but both are not required. However, it probably won't make much sense to just have a functional mask as any and all areas in the brain that meet the functional critria will be included.