DOCUMENTATION FOR

Multi-Level Kernel Density Analysis (MKDA):

TOR WAGER'S

META-ANALYTIC PROCEDURES

FOR

NEUROIMAGING FINDINGS

Documentation by:

Lisa Feldman Barrett
Boston College

Kristen Lindquist
Boston College

Eliza Bliss-Moreau

Lauren Kaplan
Columbia University

Table of contents

NOTE:

The original KDA analysis toolbox is located in the folder ***densityUtility***
The MKDA toolbox is located in the folder ***densityUtility3***
These folders contain the top-level functions you're most likely to want to run.

There is also a graphical interface for MKDA analysis that you can launch with the command "Meta_Analysis" at the Matlab prompt

Other useful functions you may want to run are in the root folder or in other subfolders.

Citations

MKDA methods:

Wager, T. D., Lindquist, M. A., Nichols, T. E., Kober, H., & van Snellenberg, J. X. (in press). Evaluating the consistency and specificity of neuroimaging data using meta-analysis. *Neuroimage*.

Wager, T. D., Lindquist, M., & Kaplan, L. (2007). Meta-analysis of functional neuroimaging data: Current and future directions. *Social, Cognitive, and Affective Neuroscience, 2*(2), 150-158.

MKDA application:

Etkin, A., & Wager, T. D. (2007). Functional Neuroimaging of Anxiety: A Meta-Analysis of Emotional Processing in PTSD, Social Anxiety Disorder, and Specific Phobia. *Am J Psychiatry, 164*(10), 1476-1488.

Wager, T. D., Barrett, L. F., Bliss-Moreau, E., Lindquist, K., Duncan, S., Kober, H., et al. (2008). The Neuroimaging of Emotion. In M. Lewis, J. M. Haviland-Jones & L. F. Barrett (Eds.), *Handbook of Emotions* (3rd ed., pp. 249-271). New York: Guilford Press.

Kober, H., Barrett, L. F., Joseph, J., Bliss-Moreau, E., Lindquist, K., & Wager, T. D. (2008). Functional grouping and cortical-subcortical interactions in emotion: A meta-analysis of neuroimaging studies. *Neuroimage, 42*, 998-1031.

CHAPTER I. OVERVIEW

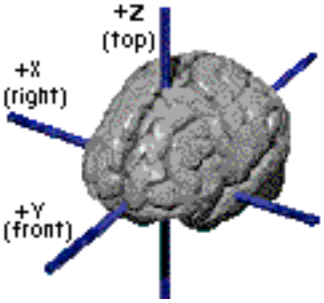
Background

The image data are a series of 3D volumes that we put into matrix format and analyze. The data of interest are peak activation coordinates that come from contrasts that are nested within studies. Multiple peaks are nested within a contrast and multiple contrasts nested within a study. Contrasts are treated as independent maps.

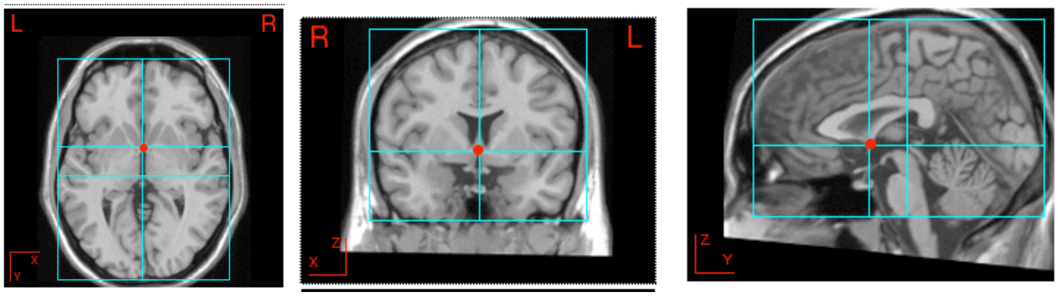
For example:

Study	Contrast	Description	x	y	z
1	1	Positive vs neutral	X1	Y1	Z1
1	1		X2	Y2	Z2
1	1		X3	Y3	Z3
1	2	Negative vs neutral	X4	Y4	Z4
1	2		X5	Y5	Z5
1	2		X6	Y6	Z6
2	3	Positive vs neutral	X7	Y7	Z7
2	3		X8	Y8	Z8
2	4	Negative vs neutral	X9	Y9	Z9

Where x, y, z refers to either Talairach or MNI coordinates (in mm from anterior commissure)



x = Lateral/Medial (L-, R+)
y = Anterior/Posterior (P-, A+)
z = Inferior/Superior (I-, S+)



Peak activation coordinates (x, y, z) are nested within contrasts (e.g., anger vs neutral) that are nested within studies. This produces a distribution of recorded peaks. Do these peaks show any pattern (i.e., are they related to certain kinds of conditions) or are they randomly distributed across the entire brain?

Prior Meta-Analyses

Prior meta-analyses have divided the brain into voxels and plotted peak coordinates. Then count how many peaks within each voxel (the observed frequency count). Compare this to the number expected by chance if peaks were distributed randomly throughout the brain (the expected frequency count). Have to establish threshold (using Monte Carlo method).

Problems with this method: This is a fixed effect procedure which ignores the fact that points are not independent of one another (as they are nested within contrasts within studies). An important consequence is that any single study that has a large number of peaks (due to differences in reporting, voxel size, thresholding) can overly influence the analyses. In fact, the Damasio et al. (2000) PET study of emotion experience may have biased previous meta-analyses given that they reported a large number of peaks.

Tor's Methods

There are several analyses that one can use to answer the question of whether peak activations are distributed randomly throughout the brain (the null hypothesis) or whether they show some pattern that is related to an experimental condition (or type of contrast):

1. Density Analyses
2. Chi-square Analyses (on voxels, or on parcels defined by SOM)
3. Connectivity Analyses
    - self organizing maps
    - dimensional analyses (such as MDS, factor analysis, cluster analysis)
4. Classifier Analyses
5. Specificity Analyses

Since we use an SPM template, all coordinates are converted into MNI. [AFNI uses Talairach coordinates; Matthew Brett (http://imaging.mrc-cbu.cam.ac.uk/imaging/CbuImaging) has equations for translating between MNI and T that are distinct for top and bottom parts of brain, but not clear if these are really any better than the standard translations that just use one set of equations for the entire brain)

Peak Density Method

1. The brain is divided up into a set of three-dimensional cubic volumes (called voxels; each is 2 x 2 x 2 mm)

2. Every point is replaced with a sphere or a Gaussian kernal [in fMRI lingo: Convolving peak activations with a kernal (e.g., a sphere or a Gaussian curve)]. For example, replace every point with a 10 mm sphere, and then do analysis on convolved data. As a result, multiple peaks that are close together will be captured by the same sphere (if they are all from the same study, this reduces the extent to which any single study can bias the overall results). For the convolution, set a

maximum threshold at 1; each activation counts equally. There will be a spatial overlap, but the activation will not exceed 1 (i.e. the contrast will not count anymore than other contrasts with fewer reported peak activations). What you end up with is a contrast (row) by voxel (column) matrix with 1s and 0s.

3. This procedure respects the multi-level aspect of the data (treat peaks as random rather than fixed). This means modeling the variability across peaks within a contrast, rather than just counting all the peaks as if they were all from the same study or contrast. To do this, create one map for each contrast. **Note: we are treating contrasts as if they are independent, rather than modeling the fact that multiple contrasts can come from the same study.

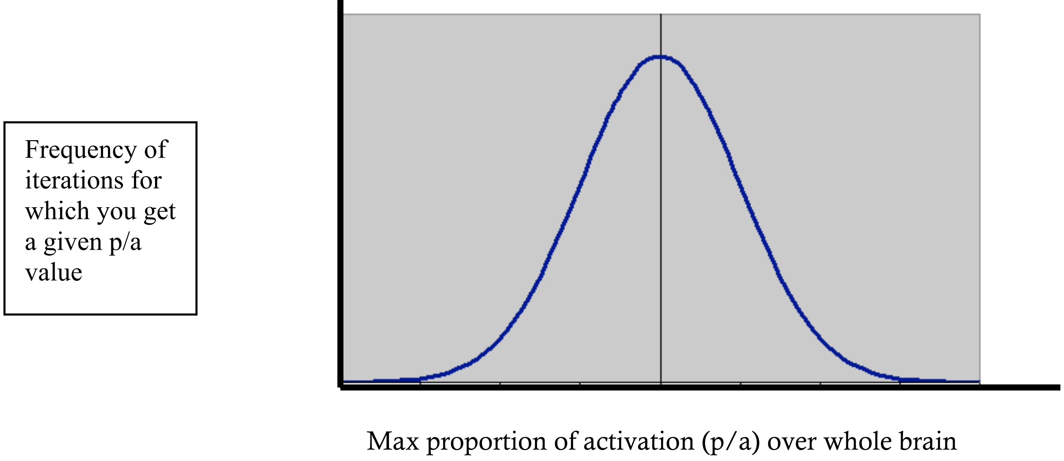
4. Produce an overall summary density map with the proportion of contrasts that report activation within 10 mm of a given voxel by taking a weighted average of contrast activation maps. The weight for each contrast was the square root of the sample size, multiplied by an adjustment weight for type of the analysis used for population inference (1 for random, .75 for fixed). This is the observed frequency map. Essentially, we are averaging each column (or voxel) across rows (or contrasts) to determine the proportion of contrasts that activate a given voxel. ** Note. We don't weight by z score because it makes it more difficult to construct the data indicator matrix. Weights are normalized

5. Then ask: which proportions are greater than what we would expect by chance. To answer this, the observed map is thresholded, meaning it is compared to proportions of studies that activate that voxel by chance. The density map is then compared to Monte Carlo simulations to identify voxels with activations that exceed the frequency expected by chance (i.e., a uniform distribution of activation coordinates across the brain's gray matter). Using a Monte Carlo simulation in which the observed number of activation coordinates from the contrast maps are placed at random locations throughout the brain—and repeating this process 5000 or more times—allows us to determine which locations in the brain show a greater-than-chance number of nearby activation coordinates, providing a stringent familywise error rate correction for search across the locations within the brain.

So, on each iteration, the MC procedure takes the activations in the original contrast map and shuffles them and puts them in random places, and then computes the maximum proportion of activation across the brain, or p(a) (the proportion of activations is computed for each voxel, and then the maximum is computed). This information is saved for each iteration. The result is the frequency with which max prop of a given size are seen in the brain by chance. This is a sampling distribution under the null, corrected for multiple comparisons (for however many voxels are included in the analysis).

It is also possible to compute the maximum proportion of activation conditionalized on a particular statistical or task contrast (such as anger vs disgust, e.g. p(a/anger) – p(a/sad)). The difference in activation is computed for each voxel, and then the max is computed and saved after each iteration, thereby producing a sampling distribution for the max prop that would be seen by chance for this contrast.

Also, a sampling distribution for max cluster size is also computed. So it is possible to say at $p < .05$, or .01, or .001, how many contiguously activated voxels do you need to say that a cluster of activation is significantly different from chance.



Voxels are considered significant anywhere in the observed summary activation map (or a statistical contrast map comparing two tasks such as anger and disgust) where the max proportion of activation is greater than what would be expected by chance (e.g., p < .05, or p < .01, or p < .001).

Similarly, MC simulations will estimate the number of contiguously activated voxels than differ from chance at different levels of significance:

e.g., 30 voxels at .001, 900 voxels at .01, or 5000 voxels at .05.

In the above example, the cluster size at .05 is so large that it would be difficult to say where it is located.

6. When you compare maps for different kinds of contrasts (e.g., experience vs perception of emotion), this tells you about the relative difference in the distribution (and size) of peaks across task conditions. The sampling distributions produced by the MC simulation are not centered on zero. For example, let the task contrasts be anger vs disgust. If there are more data contrasts available for anger, then this will make D > A in any given voxel less likely (since the MC simulation contains more activations that are derived from anger vs disgust to begin with).

To examine absolute differences between brain activity for different tasks, you must use a voxel-based chi-square. This will tell you the absolute proportion of activation associated for any task condition. But in cases where there is a massive difference between activation levels for different conditions (e.g., negative affect always produces more activity than positive affect), then you will always see absolute differences in a given direction (e.g., always N > P). Weaker task contrasts (e.g., P > N) will never give significant results (whereas density analysis would be more sensitive to this).

[Activation Likelihood Estimation (or ALE, http://brainmap.org/ale/): convolve peak activations with Gaussian curve and treat as probabilities; hard to interpret]

What is Really Happening

All analyses (density and otherwise) produce an **data indicator matrix** where columns are voxels in the brain (strung out like a vector) and rows are contrasts:

	V1	V2	V3	V4	V5	V6	V7	…	V200,000
C1	1	0	0						
C2	0	0	0						
C3	0	1	1						
C4	1	0	1						

Compute probability of activation:

	.5	.25	.5						

Can take any single voxel and analyze it. Can take overall summary of whole brain and analyze it.

Can test hypotheses. Let C2 and C4 be +(positive) vs N(neutral), C1 and C3 be – (negative) vs N. Look at the proportion of activation at a given voxel for each type of contrast and compare (see point 6 above).

<u>**Chi-Square (χ2) Analyses**</u>

This is a univariate method in that you are looking at frequency voxel by voxel (or parcel by parcel if you use the output of SOM as the input to this analysis). Get a frequency matrix for each voxel.

For example, at voxel 1, observed matrix is:

	+	-		marginal
Yes (activation) 1	0	1		1
No 0	2	1		3

marginal	2	2

Ho = proportion of activation (p/a) do not differ across + and – contrasts

p/a = .25

Expected matrix is

	+	-
Yes (activation) 1	(2*1)*.25=0.5	(1*2)*.25= 0.5
No 0	(3*2)*.25= 1.5	(3*2)*.25 = 1.5

Then compute χ2 = S (o-e)2 / e = [(0-0.5)2/.5] + [(1-0.5)2/.5) + ………

This is a χ2 test for independence (is activation/no activation independent of + vs -

Conditions could be fear vs anger vs sad (and so on) rather than + vs –

This tells you the absolute (rather than relative) difference in the distribution of peak activations across conditions.

Can do this for every voxel in the brain. But some cells in the whole brain table will have very low frequency counts, and χ2 is not robust if expected cells are less than 5, so analysis is not meaningful if do a voxel by voxel analysis under these conditions (will be too liberal). To counteract this, Tor has developed an alternative procedure: Chi2test.mat checks to see if counts are less than 5 and if p < .2 and if so, then runs an alternative (nonparametric) version. Need more information about this.

Normally, for a 2 by 2 matrix, can do a Fischer's exact test, but the extension of this for larger tables is computationally infeasible. Solution: permute one column with respect to another. Under the null, there is no relation between whether there is an activation or not and whether the contrast represents + or -, so can change the 1s and 0s and calculate χ2, then do it again, and again, and so on. Do a number (5000) MC simulations and build a χ2 sampling distribution under the null. Each permuation, keep number of 1s and 0s in a column the same (i.e., column and row marginals must be the same; this is called conditionalizing on those variables), but change their placement randomly.

p(a/+) and p(a/-) is what is tested in the χ2

χ2 is a direct spatial test that can be easily interpreted (using ANOVA on the conditionalized probability matrices or on the data indicator matrix (which?) will tell you that there is a significant difference in the pattern activity somewhere, but it won't tell you where (this needs more explanation).

<u>Connectivity Analyses</u>

Data reduction to promote an understanding of what the brain is doing

	V1	V2	V3	V4	V5	V6	V7	…	V200,000
C1	1	0	1	1					
C2	0	0	0	0					
C3	0	0	0	0					
C4	1	0	1	1					
C5	1	1	0	1					
C6	0	0	0	0					
C7	0	0	0	0					

Two kinds of connectivity: <u>Self-organizing maps or parcellation</u> (where you indentify contiguous groups of voxels with similar patterns of activation; e.g., V1 and V4 group together, and maybe V3); <u>Network analysis</u> (where you want to see if several ROIs or parcels that are distributed in space can be grouped together into a network; this uses data reduction techniques such as MDS, principal components or factor analysis, and cluster analysis).

<u>SOM</u>: started by Kohonen's work on how a neural net comes to simulate or carry information about a group of categories (or nodes). You begin with a number of categories that you specify (4, 9, 16) and weights between categories begin at zero. (More nodes, more coherent parcels; fewer nodes, less coherent parcels as may be grouping together too many voxels that only look mildly the same; but too many parcels defeats the purpose). As you present voxels to the classification algorithm, it finds the closest match and updates the weights (not clear). Do this across many iterations during training. A node starts to look like a frequently occurring pattern in the data so that each node captures a canonical pattern in the data (so what is the business about connection weights between nodes if one node is a pattern of activity?) . Impose a contiguity constraint (areas of activity must be

next to one another). Average over them (?) So, can look at which voxels make up a given node, and then this tells you what the parcel is.

If one category has multiple distributed regions, then this tells you something about how the brain is organized as a distributed network

?is this capturing a brain mode or a psychological construct

Network analyses: Once you have groups of voxels (anatomically defined ROIs or ROIs defined by peak activations, or SOM) then have an indicator matrix where columns are REGIONS (not voxels) and columns are contrasts.

Compute phi correlations for Regions (across contrasts). If phi > 0 for regions 1 and 2, then contrasts that activation region 1 also activation region 2. Compute matrix of phis (below the diagonal, a correlation matrix).

Then subject this resulting proximity matrix to either a PC (principal components analysis), a nonparametric MDS (since brain is curved and don't only want to model linear relations), or hierarchical cluster analysis. Identifies the functional networks distributed across the brain.

Permute phi coefficient matrix and do again and again (MC, 5000 times) to get a sampling distribution to determine which solution to choose (can do this for PC or HCA). (needs more explanation)

Classifier Analyses

(from old LFB notes)

1. Begin with a matrix of contrasts (rows) by voxels (columns); for example, in our overall emotion analysis, we begin with 437 rows by ? voxels (all voxels in brain? Or all active voxels in summary activation map?)

2. Select those contrasts you want to use (e.g., like the meta-select step). For example, in the discrete emotion analysis, we would choose only those contrasts from papers dealing with discrete emotions.

3. Feature Selection: don't compute classifier analysis using all voxels in the brain.

- Eliminate voxels with no variance: Eliminate columns that have no variance across rows. This restricts the analysis to a set of voxels to those that have predictive value. restrict analysis to set of voxels that will yield good results (don't compute ron whole brain): first, from matrix eliminate voxels (columns) have no variance

- Compute p_task: The overall goal of this step is to compute posterior probabilities (given activation in a certain voxel, what is the probability that this activation came from a study/task of a certain type). For example, given the pattern of activity in a voxel (a column), what the probability that the activation at that voxel was associated with anger (or fear or what have you). So, this is equivalent to the proportion of columns (or contrasts) in that vector that came from studies of a certain type (e.g., fear).

- Final selection: select voxels (columns) that have a high-enough p_task (selectivity cut off; e.g., p _task must = .6) AND those that have some minimum number of activations (activation feature cutoff; e.g., significant activation for at least 2 contrasts or rows). Voxels with too few activations (e.g., 1 activation across all contrasts) are not informative for this analysis.

- Result: p_task matrix: task (e.g., 5 emotions) x some number (v) of voxels

4. Cross-Validation Procedure – Can this p_task matrix predict which task generated new data? Recompute p_task for combinations of n-1 studies and try to predict from it the final study (like a jackknifing procedure). Right now, this use same set of features each time. This is similar to an item-total correlation in internal consistency analyses. Chance classification is 1/number of tasks (e.g. for discrete emotion analyses, chance classification is 1/5 or .20). Probably should do feature selection on each iteration.

5. Plot classification accuracy for p_task matrices computed using different selectivity and activation feature cutoffs.

Specificity Analyses

TBA

CHAPTER II. MATLAB FILES AND COMMANDS

Analysis Folder

- The analysis software consists of a set of matlab scripts and functions. The main functions are called Meta_*.m. (.m or M-files are executable in Matlab).
- In order to use them, you need to make sure the folders with the relevant M-files are in your Matlab path. You can add them by typing pathtool from the matlab prompt. (click add with subfolders)
- If you type Meta_Setup get an error like this: ??? Undefined function or variable 'Meta_Setup', it means that you do not have the function Meta_Setup.m in the matlab path.

Function Help Files

- Each Matlab function (M-file) has a set of notes that gives information about how the function works and what specific variables it either requires or produces.
- All function notes are accessed with the following command. Type, e.g., help Meta_Activation_FWE to get help for the function Meta_Activation_FWE.m

Function Overview

- The main functions you might want to run are divided into setup and analysis functions. The setup functions to run, in order, are:
    - read_database : read a .txt database into matlab
    - Meta_Setup : set up analysis
    - Meta_Select_Contrasts : prepare a subset of coordinates in the database for analysis

- The analysis functions are as follows.
    - Meta_Activation_FWE : This is the density analysis

- To make figures and tables, use the following:
    - Meta_Activation_FWE ('results'): Using FWE in the results mode allows you to make figures (blobs on a standardized brain) to summarize the density analysis; it uses the more general matlab procedure cluster_orthviews in a specified way
    - Meta_Study_Table: Allows you to make tables to summarize the activations in the density analysis; it uses the more general matlab procedure cluster_tables in a specified way
    - cluster_orthviews: allows you to plot center of significant clusters in slices
    - cluster_surf: allows you to look at surface maps
    - cluster_tables: allows you to make a table of the size and location of significant clusters
    - montage_clusters: allows you to depict significant clusters in a series of slices
    - mask2clusters: uses the specified img file which is a mask to create a new cl variable. Lets you plot z scores or p values.

- Additional functions
    - Meta_cluster_tools: set of tools for working with clusters

- Meta_analysis_table: allows you to look at all existing meta-analyses databases for other study contrasts that activate at a given point in the brain where there is a significant cluster in the meta-analysis of interest

Matlab Commands

clear – clears workspace (or clear all)
whos – lists variables
DB.emotion (where DB is a data structure with the field "emotion) – list for every contrast in the DB the value for this field
which - tells you which file is in the workspace (e.g., which DB)

A Note on Matlab Language

```
varargout = Meta_cluster_tools(meth,varargin)
```

```
>> MC_Setup = Meta_Activation_FWE('setup', DB)
```

This indicates that a variable called MC_Setup is being created with the function Meta_Activation_FWE and we are passing in the DB structure (which is a variable) to make MC_Setup (as the input argument 'setup' is applied)

CHAPTER III. SET-UP

STEP 1: MAKE NEW DIRECTORY OR FOLDER

Make a new directory for the analysis. Do this by making the folder in the Finder and dragging it into to the Matlab window, or by using the matlab command

```
>> mkdir foldername
```

and change the directory to the folder in which the database is located using the command

```
>> cd foldername
```

Any time you want to change directories, you can do so by dragging the wanted folderinto the Matlab window

STEP 2: PREPARE EXCEL DATABASE

Database can have no spaces or special characters. Fill all blank cells with NaN. (edit, go to, special, blank)

Count the columns and insert the number of columns into the first cell [the remainder of the row should be filled with NaN].

Save the database as a text file [If working on a Mac, save the file as ‘Text (Windows) Format’].

Some database variable (column) names have special meaning in the program. See below for a list of these.

STEP 3: READ DATABASE INTO MATLAB

In the COMMAND WINDOW, read in the database using the command read_database

```
>> read_database
```

When prompted for the database filename, enter it [or, you can supply the file name before the read_database command by typing matlab command: dbname=’filename.txt’]

e.g., `Enter name of input file: meta_061106.txt`

read_database translates the xxx.txt database into matlab format-- xxx.mat—and gets the file and variables into Matlab workspace

You get this output:

```
Converting T88 coordinates to MNI (M. Brett transform): 1724 Transformed
Converting MNI coordinates to T88 for XYZtal array: 557 Transformed
```

When prompted, save the variables in a .mat file:

```
Enter name of file to save (without .mat extension): meta061106
```

This saves a file called meta061106.mat.

Look at the help file from read_database which tells you about the database. Type help read_database

STEP 4: LOAD THE DATABASE

If you begin the process with read_database, the database (e.g., meta061106.mat) will already be loaded and ready to go.

If you begin the process at this stage using a database that you have formatted in a prior session, you will need to load the database (in COMMAND WINDOW) using the matlab command

```
>> load database.mat
```

e.g., `>> load meta061106`

STEP 5: CHECK THE DATABASE

Before running any analyses, check and make sure that the all the needed variables are in the file, and that the values are OK within each variable. Do this by using a series of unique(var) commands. For example

```
>> unique(Emotion)
```

produces

```
ans =

    'aff'
    'anger'
    'disgust'
    'fear'
    'happy'
    'mix'
    'sad'
```

STEP 6: META_SETUP

In COMMAND WINDOW, run command Meta_Setup using matlab command:

```
>> DB=Meta_Setup (DB, 10);
```

Where DB is the database structure, specifying a bubble with radius of 10 mm, and the semicolon suppresses some output to screen

More specifically, the parameters required are as follows:

```
>> DB = Meta_Setup(DB,[radius_mm],[con_dens_images])
```

This command creates two files: SETUP.mat (which contains the DB structure). You can load this file for later analyses. To see what is in SETUP.mat, type `whose`. The main purpose of the SETUP file is to hold DB. DB is a data structure (with fields). To see what is in DB type `DB`. DB just stores the data base information. ANALYSIS_INFORMATION.txt contains information about this stage in the analysis. This stage counts the imaging contrasts and applies the weights. It also makes the mask, sets up the radius of the kernal, and checks for needed fields in the database. Note: this stage used to create contrast images for each unique contrast, but no longer does so.

If run correctly, you get the output:

```
Radius is 10 mm.
Setup. Cannot find study field.  Copying from Study.
No xyz field in DB.  Creating from x, y, z.
Scaling by SubjectiveWeights
DB structure saved in SETUP.mat
Contrast image writing is OFF.  Use Meta_Activation_FWE for analysis.
```

STEP 7: CREATE NEW FOLDER (optional)

Before you continue with the analysis, you might want to make a new directory (especially if you are selecting contrasts or pairing down the DB structure to certain contrasts, excluding others, e.g., selecting only discrete emotion experience contrasts)

e.g.,

```
>> mkdir exp_discrete_emotion
>> cd exp_discrete_emotion
```

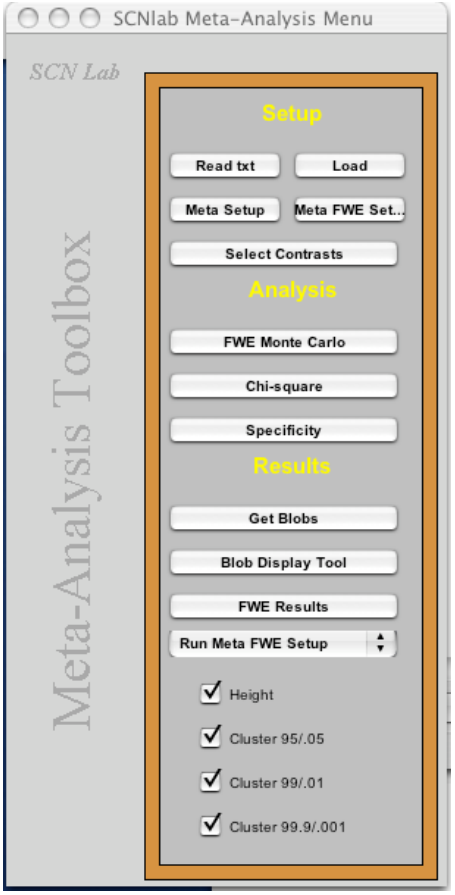
or just make a new folder and drag into the matlab window

CHAPTER IV. DENSITY ANALYSIS STEP-BY-STEP

It is possible to run the density analyses in the CONTROL WINDOW by typing commands, or in the GUI (which is launched by typing “Meta_Analysis”).

```
>> Meta_Analysis
```

When you type this, GUI will appear:



NOTE: whatever files you create by running analyses in the GUI, they are not available to the command window unless you read them in (meaning, you either run the analyses in the GUI, or in the command window, but not both). Also, if you use the GUI, it declares the DB structure as a “global variable” (state what this mean).

This manual will first discuss how to run the analyses in the COMMAND WINDOW, and then using the GUI.

STEP 1: SELECTING CONTRASTS FOR DENSITY ANALYSIS (optional)

You might want to select only certain contrasts for analysis (e.g., you might want to only analyze experience contrasts, or only perception contrasts, etc.). This is the step for selecting those contrasts (or pairing down the DB structure). The SETUP.mat file must be loaded for this step.

In Command Window, type

```
>> DB=Meta_Select_Contrasts(DB)
```

After which matlab will list the database field names (the columns in the database that contain the variables that code the contrasts) and you are asked to enter field name

```
Enter field name or return if finished:
```

e.g., `Mode`

Then you are provided with unique values for that variable and asked to enter a vector of levels to use:

```
Unique values of this variable:
  1       exp
  2       mix
  3       per
Enter vector of levels to use:
```

Here I will enter just one level (e.g., 1) to select only contrasts pertaining to experience

The number and name of the studies to be included is then listed.

Then you are prompted again

```
Enter field name or return if finished:
```

It appears here that you can enter another selection to pair the DB structure even further but that is not so. Just hit return.

If you want to pair the DB structure even further, you must run `DB=Meta_Select_Contrasts(DB)` again.

e.g.,

```
>> DB=Meta_Select_Contrasts(DB)
Enter field name or return if finished: Emotion
```

Then you are provided with unique values for that variable and asked to enter a vector of levels to use:

```
Unique values of this variable:
  1   aff
  2   anger
  3   disgust
  4   fear
  5   happy
  6   mix
  7   sad
Enter vector of levels to use:
```

You can enter just one level (e.g., 3) or a vector of levels (e.g., [2 3 4 5 7])

After selected contrasts have printed on the screen, hit return to finish. When you hit return, you will get a summary of the points, contrasts, and studies selected.

To save the DB structure for future use, type

```
>> save SETUP DB
```

This saves the DB structure in the SETUP.mat file. Note: this SETUP file (which is in the folder exp_discrete_emotion) is different than the original SETUP file. You can load either one for later analysis, so be careful that the correct version is loaded)

STEP 2: DENSITY ANALYSIS

You now need to load SETUP.mat in the COMMAND WINDOW (make sure you load the correct one)

```
>> load SETUP.mat
```

This is the step where you create the indicator matrix, create the summary density map, threshold it (using Monte Carlo simulations), and view the results using the procedure Meta_Activation_FWE (FEW stands for family wise error rate, meaning that the chance of making a Type 1 error is corrected across the whole brain, for all voxels). There are two ways to run the density analysis in the COMMAND WINDOW:

1. If you type `Meta_Activation_FWE (all, DB, number of interations for MC simulation)`, then this will take you, in sequence, through the entire setup (which included building the summary map), the Monte Carlo simulation (which allows you to threshold the summary map and determine what is significant), and viewing the results.

2. You can type each command separately for set up, MC simulation, and viewing results (and described further below). The overall structure of the command is MC_Setup=Meta_Activation_FWE(“mode”, other stuff)

```
>> MC_Setup = Meta_Activation_FWE('setup', DB)
```

This runs FWE in “setup” mode and creates the summary activation map (activation_map.img) and MC_Setup structure (which is saved in MC_Info.mat). This is where the data indicator matrix is made.

```
>> Meta_Activation_FWE('mc', iterations)
```

This runs FEW in “Monte Carlo” mode and creates the sampling distributions to threshold activation_map.img. MC_Setup is a variable that contains a data structure (a structure with multiple fields for values) and this is where information about the Monte Carlo simulation is stored. The output of the MC iterations is appended to the MC_Info.mat file. One sampling distribution is created for max proportion of activations (to be able to assess the max prop of activation that would be greater than would be expected by chance 1/1000 times (p < .001), 1/100 times (p < .01), and 5/100 times (p < .05). A set is a sampling distributions of max cluster size are computed at three prespecified thresholds (p < .001, p < .01, p < .05). All this info is added to MC_Info.mat.

```
>> Meta_Activation_FWE('results')
```

This runs FEW in “results” mode to get and plots results. There are lots of options here. Type `help Meta_Activation_FWE`

STEP 2a: FWE SETUP

```
>> MC_Setup = Meta_Activation_FWE('setup', DB)
```

If run correctly, you get the output:

```
Compute contrasts across conditions also? (1/0)
```

You type ‘1’ for yes because you want to be able to test the proportion maps for different (experimental or test) contrasts such as positive/negative, anger/sadness/fear, etc.

Then, you see a list of eligible fields for contrast (variables from the database). You enter the field name for the contrast(s) of interest.

e.g.,

```
Enter field name or return if finished: Emotion
```

Next enter vectors of levels to be used

e.g.,

```
Unique values of this variable:
  1   anger    6 Contrasts
  2   disgust      22 Contrasts
  3   fear    19 Contrasts
  4   happy   15 Contrasts
  5   sad     30 Contrasts
Enter vector of levels to use: [1 2 3 4 5]
```

The FWE density analyses will be run for imaging contrasts with these values only.

Then enter the task or statistical contrasts that you want (keep track of the number)

e.g.,

```
001  anger
002  disgust
003  fear
004  happy
005  sad
Enter contrast across 5 conditions in [], return to quit: [1 -1 0 0 0]
```

Then name the task/statistical contrast:

```
Enter short name for this contrast, no spaces or special chars: AngerDisgust
```

You can either end here or enter multiple contrasts.

e.g.,

```
001  anger
002  disgust
003  fear
004  happy
005  sad
Enter contrast across 5 conditions in [], return to quit: [4 -1 -1 -1 -1]
Enter short name for this contrast, no spaces or special chars: AngervsDFHS
```

When you are done, hit return.

e.g.,

```
Enter contrast across 5 conditions in [], return to quit: Hit return.
```

Then you will see

```
Enter field name or return if finished:
```

You can select another variable and add more statistical contrasts for that variable if you want to, or hit return to finish.

This produces a design matrix (in DESIGN_REPORT.txt), and an indicator matrix for task conditions (in MC_Setup) which is saved in a new data structure called MC_Info. This modifies the DB structure in SETUP.mat and is automatically saved.

Note: when you include multiple statistical contrasts, you may end up with a non-orthogonal set of contrasts. If you included all contrasts in a multiple regression, you would have a multicollinearity (degree of collinearity indexed by the "condition" number in FEW setup output. If you're testing the contrasts one at a time, as we are here, then it won't be a problem. Just remember that some contrasts overlap with others.

This is what you see in the matlab window

```
Created new MC_Info.
Appended DB to existing SETUP.mat.

MC_Setup =

    unweighted_study_data: [231202x92 double]
```

%voxel by contrast data matrix (above called the indicator matrix; this is the matrix of 1s and 0s that constitute the observed data)

```
                  volInfo: [1x1 struct]
```

%info needed to get from vector format back to 3D image of brain

```
                        n: [1x92 double]
```

%number of peak coordinates

```
                      wts: [92x1 double]
                        r: 5
```

%radius of sphere in mm?

```
                contrasts: [15x5 double]
                 connames: {1x15 cell}
```

%names of statistical contrasts

```
                       Xi: [92x5 double]
```

%task indicator matrix for statistical contrasts – contrasts by condition – contrast is contrast across columns in indicator matrix**

```
Xinms: {'anger'  'disgust'  'fear'  'happy'  'sad'}
ctxtxi: [15x92 double]
```

%matrix products; apply to every voxel in the brain to give you the differences between statistical contrasts**

```
con_data: [231202x15 double]
```

%differences in proportions of activation at each voxel for levels in a statistical contrast (e.g., anger vs all other emotions)

```
cl: {1x92 cell}
```

%cl is a generic name for any set of clusters. Here, the cl variable represents the contiguous “blobs” from each study contrast. Iti s the location of these blobs that is shuffled in the Monte Carlo simulation in the next step.

You can now load MC_Info.

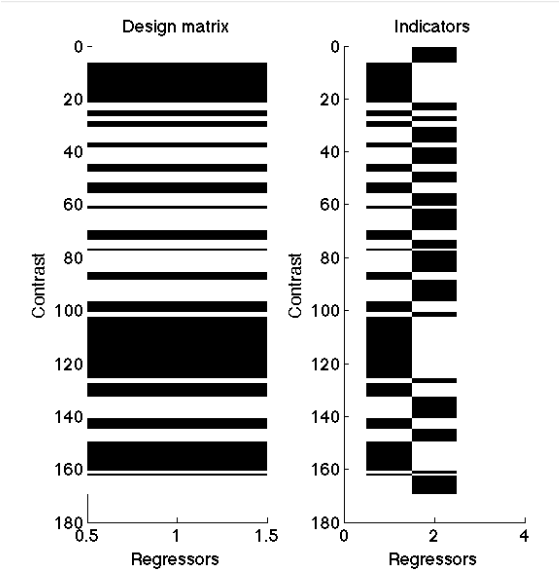
To see what is in MC_Info, type `whos`

Matlab will give you a graphic depiction of the design matrix, the task indicator matrix (for the statistical contrasts to compare different tasks that you have set up, where the levels of a task =1 are depicted with white bars and -1 with black bars), and a figure that show the indicator matrix with white dots as significant cells in the data matrix (where rows are voxels in the brain and columns are contrasts, white=1, black=0):

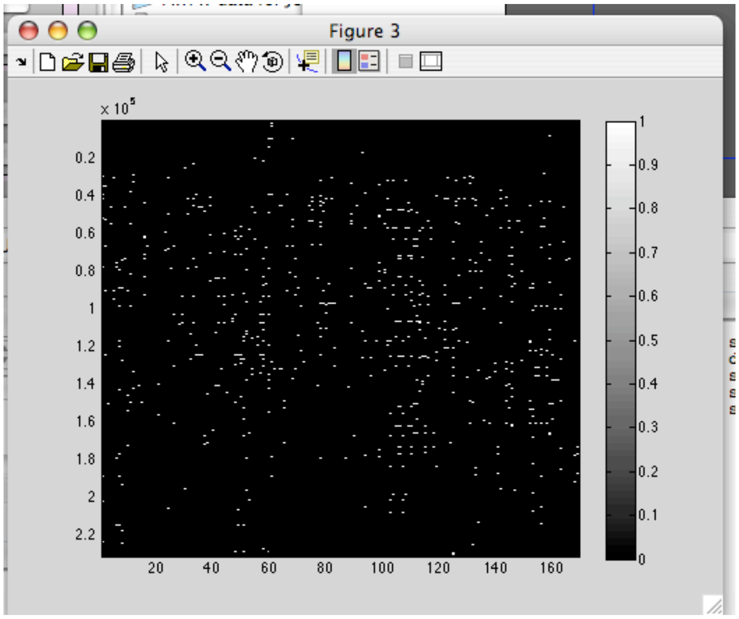
Type

```
>> figure; plot(MC_Setup.wts)
```

Here is an example of what you would get



```
>> figure; imagesc(MC_Setup.Xi)
```



If you want to look at the cl variable, type

```
>> MC_Setup.cl
```

STEP 2b: FWE MONTE CARLO

This procedure thresholds the activation maps (both the overall map and the map for each statistical contrast).

```
>> Meta_Activation_FWE('mc',15000)
```

This step adds information to MC_Info. It saves the results after every 10 iterations. You can quit it (type `Cntrl+C`) and start it up again later to add more iterations (which will just append). To append 300 iterations, type

```
>> Meta_Activation_FWE('mc',300)
```

Make sure to load MC_Info first. The observed activation maps are thresholded at different levels of significance, .05, .01, and .001, and for the maximum cluster size at these levels. This thresholds the summary activation map (and the summary maps for each task/statistical contrast) to show you anywhere in the brain where the observed p(a) (or the prob associated with a task contrast) is higher (or larger) than what would be expected by chance (corrected for comparisons across the whole brain).

Monte Carlo information added to MC_Info.mat includes:

- MC_Setup
- activation_proportions = overall proportion of activations in vector format
- maxcsize = maximum cluster size in MC simulation (overall and for all statistical contrasts)
- maxprop = contain information about each Monte Carlo simulations that are about to be conducted

- uncor_prop = stores proportions to use in thresholding

If you want to look at the value of maxcsize, type

```
>> maxcsize
```

and you will see

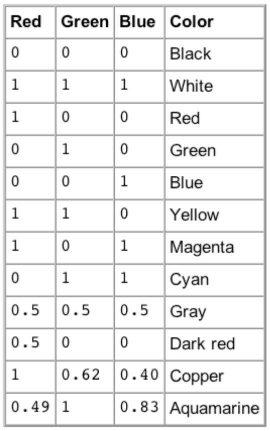
```
maxcsize =
```

`act: [18000x3 double]` Max cluster sizes in the MC simulation for the overall activation map
`poscon: {1x15 cell}` what is this?
`negcon: {1x15 cell}` what is this?

STEP 2c: FWE RESULTS

The MC simulation gives you a 3Dimage of significant voxels (that fall above threshold). This is written as a mask (which file?) and as a cluster (cl) variable in a cluster.mat file. cl is an arbitrary variable name that refers to any set of clusters. It is an arbitrary way of storing brain maps and results.

A note on color: [R G B]. color [0 1 1] (cyan) is reserved for the overlap color btwn cluster sets. Here are some representative RGB color definitions from matlab 3-D visualization help:



Red	Green	Blue	Color
0	0	0	Black
1	1	1	White
1	0	0	Red
0	1	0	Green
0	0	1	Blue
1	1	0	Yellow
1	0	1	Magenta
0	1	1	Cyan
0.5	0.5	0.5	Gray
0.5	0	0	Dark red
1	0.62	0.40	Copper
0.49	1	0.83	Aquamarine

The results mode has many different options. Let's begin with the simplest, which is plotting the observed reference space (overall areas of activity that are significantly different from chance):

```
>> Meta_Activation_FWE('results', 1)
```

You are running FEW in "results" mode and 1 indicates plot the data. This step produces a number of different files, which you can see by typing `ls`

e.g.,

Activation_FWE_height.img

This is the summary activation map thresholded for significance, meaning that it is a masked image with 1 in each voxel where the proportion of contrast activations (from the database) are greater than what you would expect by chance (1/20 times, or p < .05) across the entire brain. This is a "height-based" threshold and corresponds to "heigh-based" corrected results. Basically, these are voxels that are have proportions of contrasts that exceed the maximum expected over the entire brain by chance (or where the chance of making a Type I error at any single voxel is 1/20).

Activation_FWE_extent.img

It is also possible to estimate which individual voxels show greater activation (prop of contrasts) than would be expected at various levels of chance (p < .001, p < .01, p < .05). These would be uncorrected values (not corrected across all voxels in the brain). So, it is possible to ask "how many contiguous voxels would I need to have to consider this cluster significant at p < .001 whole brain corrected?" Or we could ask the same questions for p < .01 or p < .05. That is what this summary activation map contains. This is the summary activation map thresholded for size of clusters. This uses "extent-based" thresholds and corresponds to "extent-based" corrected results.

Activation_FWE_all.img

This is the union of height and extent.

Activation_clusters.mat

This saves clusters of significant activations (activations that pass a threshold) as a cl variable in a clusters.mat file.

Matlab also tells you the number and size of significant clusters at lenient (p < .05), medium (p < .01), and stringent (p < .001) thresholds.

e.g.,

```
Thresh = 0.0330, size threshold = 2995: 21710 sig. voxels.
Cluster sizes range from 21710 to 21710
```

Thresh .033 means that you need 3% of contrasts to be active for a blob (or cluster) to be significant at p < .05. Size threshold is the number of contiguous voxels needed to be significant. At p < .05 there is just one cluster with 21710 voxels

```
Thresh = 0.0458, size threshold = 656: 13881 sig. voxels.
Cluster sizes range from 1168 to 10998
```

Thresh .0458 means that you need approx 5% of contrasts to meet criteria for a blob (or cluster) to be significant at p < .01. Size threshold is the number of contiguous voxels needed to be significant.

```
Thresh = 0.0614, size threshold = 163: 6949 sig. voxels.
Cluster sizes range from 211 to 5025
```

Thresh .0614 means that you need approx 6% of contrasts to meet criteria for a blob (or cluster) to be significant at p < .001. Size threshold is the number of contiguous voxels needed to be significant.

This information is then summarized:

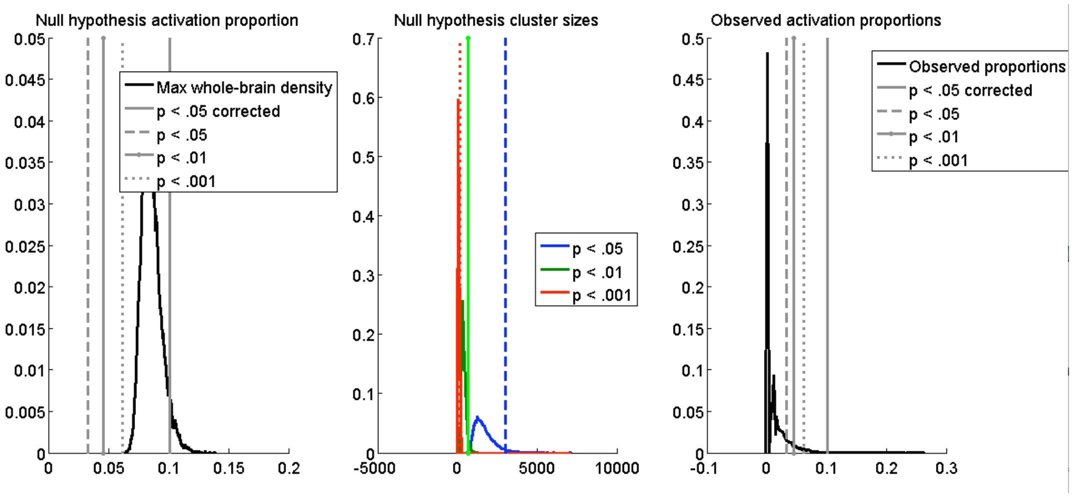
```
iimg_multi_threshold viewer
=====================================================
Showing positive or negative: both
Overlay is: /fMRI Analysis
Software/3DheadUtility/SPM2_brains/spm2_single_subj_T1_scalped.img
Height thresholds:0.1012 0.0614 0.0458 0.0330
```

%Proportion of contrasts that must be significant at any given voxel for a given significance level (p < .05 height-based, p < .001 extent-based, p < .01 extent-based, p < .05 extent-based).

```
Extent thresholds:   1 163 656 2995
```

%Number of contiguous voxels that are needed to say that a cluster is significant at p < .05 (meaning that the Type 1 error is held at p < .05 across the whole brain).

You also get this information in distribution form

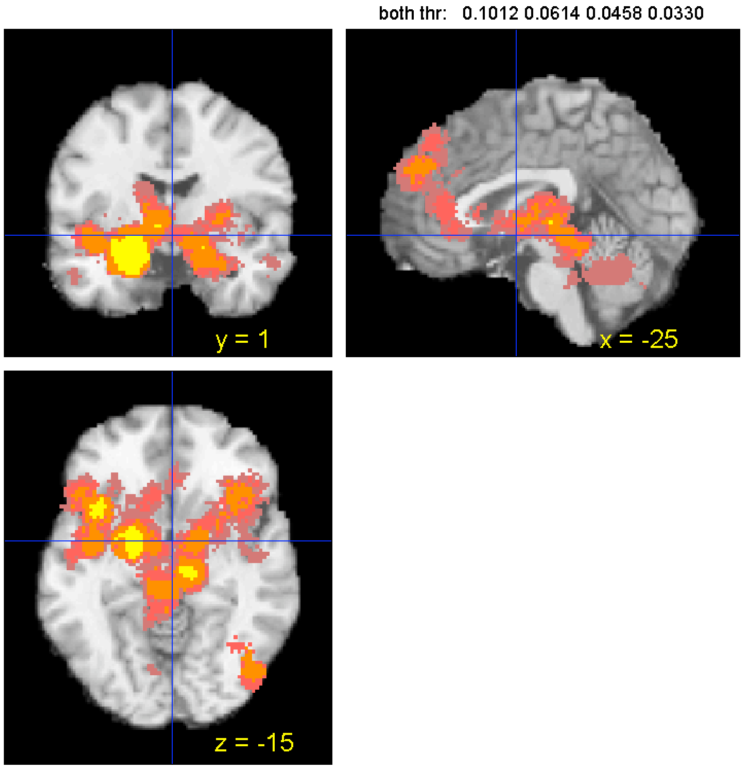


Tor, can you insert discussion

You also get a table with local maxima within 10 mm (where -> refers to subclusters within the significant clusters) and you get a brain figure with significant blobs. The first table corresponds to the clusters depicted in yellow; these are the clusters that surpass the height-corrected threshold. The second table corresponds to the clusters depicted in orange; these are incremental clusters that pass the most stringent extent-based threshold (p < .001) that are not within 10 mm of the clusters for the height-based threshold. The third table corresponds to the clusters depicted in red; these are incremental clusters that pass the medium extent-based threshold (p < .01) that are not within 10 mm of the clusters for the height-corrected and stringent extent-corrected thresholds. The fourth table corresponds to the clusters depicted in purple; these are incremental clusters that pass the lenient extent-based threshold (p < .05) that are not within 10 mm of the clusters for the height-corrected, as well as the stringent and medium extent-corrected thresholds. Maxstat is the maximum of the z field.

```
Height
Getting local maxima within 10 mm for subcluster reporting
index x      y      z     corr  voxels      volume_mm3  maxstat
  1   -24     0    -16    NaN   888    7104  0.26
->    -38    14    -16    NaN   185    0.14593
->    -20    -4    -16    NaN   597    0.26259
->    -32     4    -10    NaN   106    0.15718
  2    40    10    -24    NaN     6     48    0.11
  3    26    -2   -20   NaN     1     8   0.11
  4   -26    30   -14   NaN    27   216   0.11
  5    20    -4   -12   NaN     3    24   0.11
  6    12   -20    -4   NaN    53   424   0.12
  7    -2   -30    -8   NaN     1     8   0.11
  8    10   -26    -8   NaN     1     8   0.11
  9   -46    18     0   NaN     9    72   0.12
 10    48   -68     0   NaN     2    16   0.12
 11    48   -64     0   NaN     2    16   0.10
 12    -6    -4     0   NaN     2    16   0.11
 13     2   -22     2   NaN     1     8   0.11
 14    -2   -12     2   NaN     1     8   0.10
 15    -4   -24     6   NaN    10    80   0.11
 16    -6   -12     4   NaN     3    24   0.11
 17    -4   -16     6   NaN     2    16   0.10
 18    50   -70     8   NaN     5    40   0.11
 19   -10   -14     8   NaN     2    16   0.11
 20    -6    52    28   NaN    25   200   0.11

Additional regions at Extent: Stringent and size >=  10
index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -14     2    -8   NaN   4021  32168 0.10
  2    42    18   -14   NaN   706   5648  0.10
  3    30     4     2   NaN   211   1688  0.09

Additional regions at Extent: Medium and size >=  10
index x     y     z     corr  voxels      volume_mm3  maxstat
  1    46   -58    -8   NaN   1065  8520  0.08
  2     0     4    -4   NaN   5023  40184 0.07
  3    -4    42    22   NaN   816   6528  0.08

Additional regions at Extent: Lenient and size >=  10


index x     y     z     corr  voxels      volume_mm3  maxstat
  1    -2    -4    10   NaN    24   192   0.04
```

Here is an example of the brain slice that depicts activations when viewed from the center of the first cluster in the first table.



The important information for results is contained in the cl variable in the Activation_clusters.mat file.

For example, type

```
>> load Activation_clusters
>> cl

cl =

    [1x20 struct]    [1x5 struct]    [1x22 struct]    [1x41 struct]
```

This gives you a variable with 4 fields or cells, and within a cell is information about the regions (clusters/brain blobs) that pass the height threshold (referred to as cl{1}), the p < .001 threshold (referred to as cl{2}), the p < .01 threshold (referred to as cl{3}), and the p < .05 threshold (referred to as cl{4}), respectively.

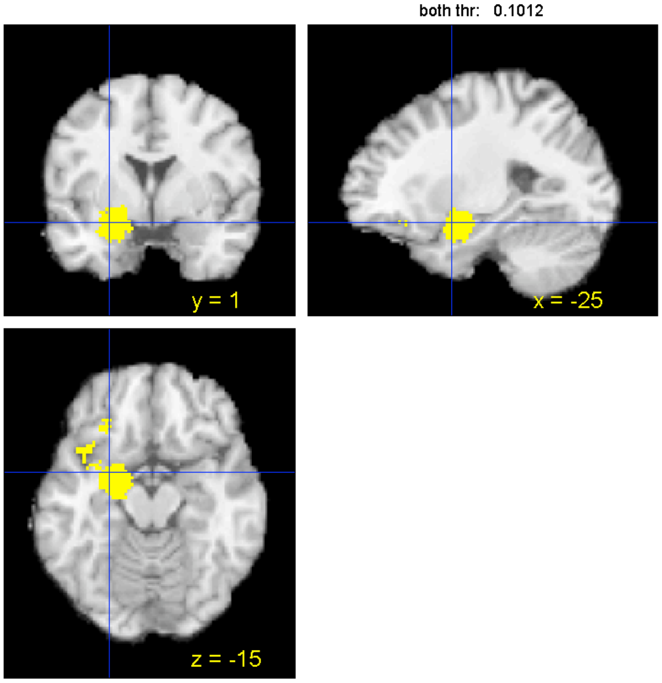
To some extent, the information contained in these cl fields is the same as you would see in the table of activations printed to the matlab window after you run `Meta_Activation_FWE('results', 1)`, but this is not completely true. The height table printed to the matlab window (above, labeled `Height` with 20 activations) will always be the same as cl{1}. But cl{2} might contain activations that are within 10 mm of those in cl{1}, and so will not appear in the table printed to the matlab window with three activations, entitled `Additional regions at Extent: Stringent and size >= 10.` And so on. You can very this by looking at the contents of the cl variable. The first field has 20 columns (for the 20 locations that you see in the height table). The second field has 5 columns (even though we only see 3 entries in the stringent extent-based table). And so on.

You can use cluster_orthviews to map individual clusters, or you can use FWE in results mode (which uses cluster_orthviews in a specific way). cluster_orthviews function uses spm_orthviews to display activation blobs from a clusters structure on a canonical structural brain image. Multiple clusters may be plotted in multiple colors, and blobs may be added to an existing orthviews figure.

So, lets say we want to map of the blobs that pass the height threshold. Type

```
>> Meta_Activation_FWE('results', 1,'height')
```

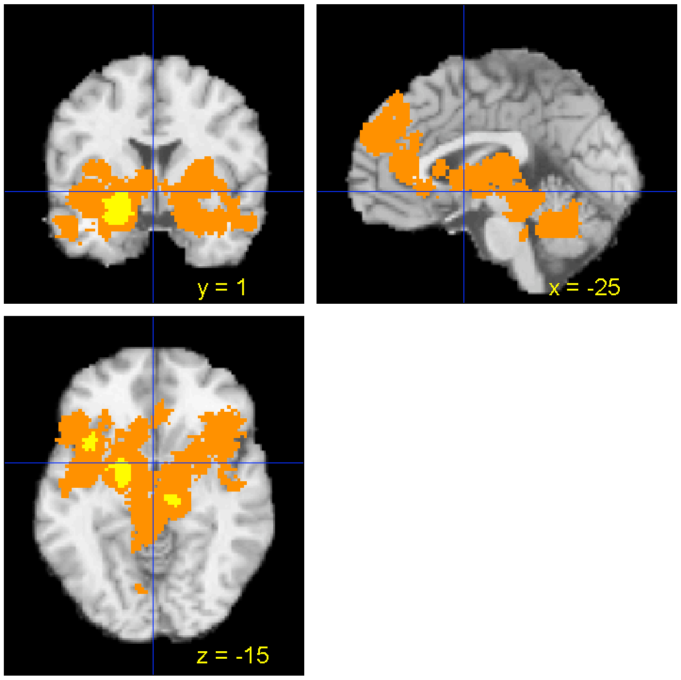
And this gives me graphs, a table, and a brain with blobs. Here is the brain slice when viewed from the center of the location for the first cluster



Now lets add in the voxels that make a cluster size (the number of continguous voxels significant at p < .001 uncorrected) that would be significant at p < .05, whole brain corrected.

```
>> Meta_Activation_FWE('results', 1,'height','stringent')
```

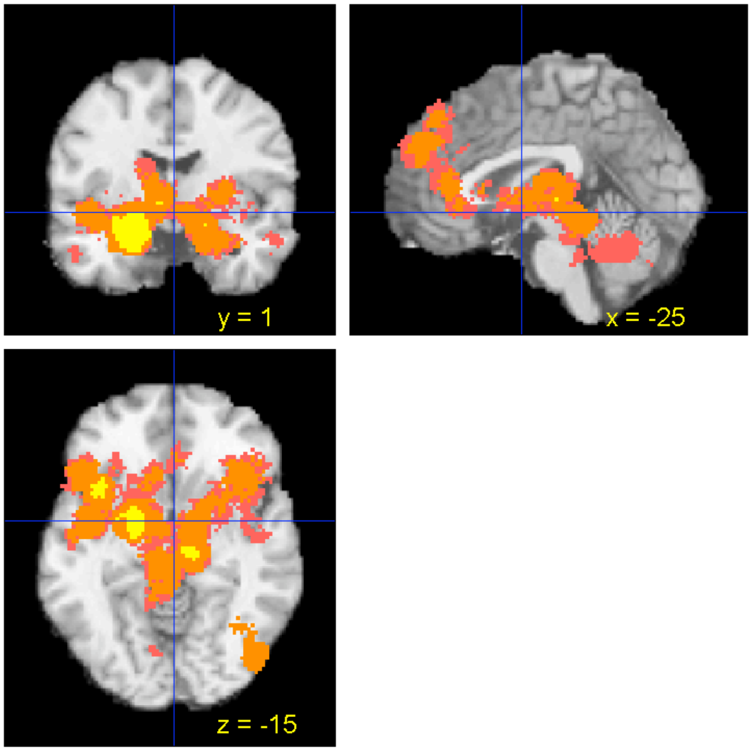
both thr: 0.1012 0.0330



Now lets add in the voxels that make a cluster size (the number of continguous voxels significant at p < .01 uncorrected) that would be significant at p < .05, whole brain corrected.

```
>> Meta_Activation_FWE('results', 1,’height’,’stringent’,’medium’)
```

both thr: 0.1012 0.0458 0.0330



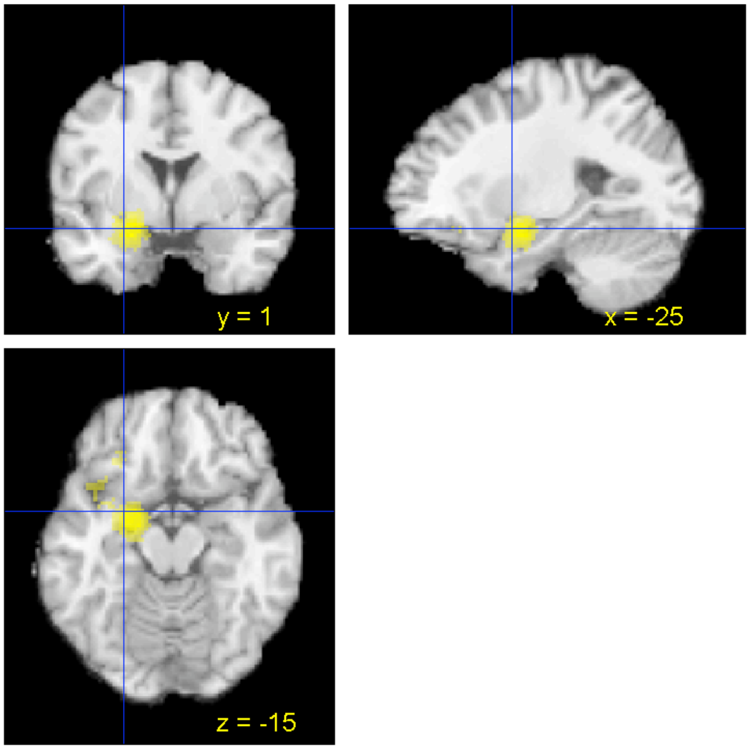
If you want to work further with the results of the MC analysis, you need to load the relevant clusters.mat file which contains the cl variable with the results. For example

```
>> load Activation_clusters
```

Now, lets say that I want to just view which areas are significant at the height-based threshold (that is, the first element in the cl structure)

```
>> cluster_orthviews(cl{1},{[1 1 0]});
```

This displays the blob 1 in yellow.



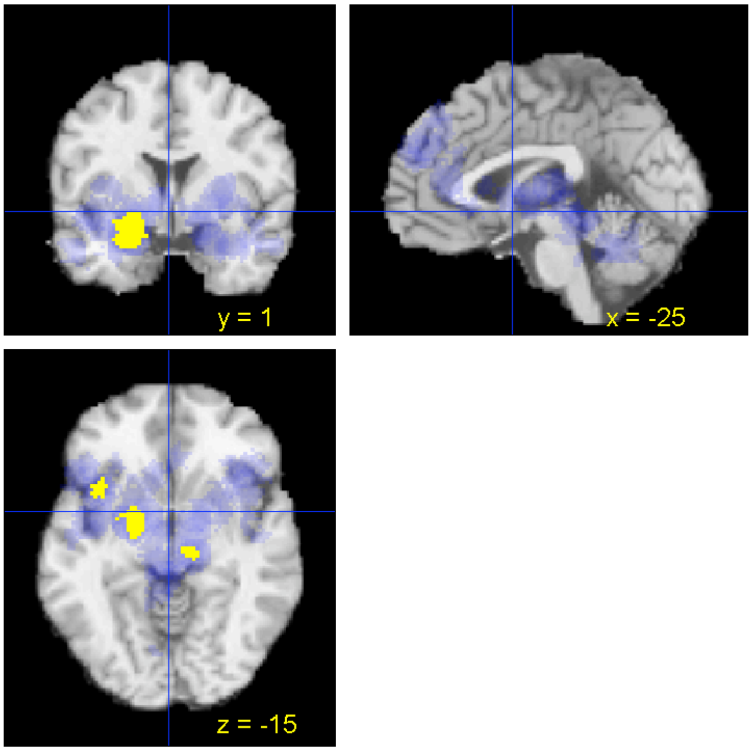
**Tor, what does the intensity of yellow signify here?

You can make the blobs solid by typing

```
>> cluster_orthviews(cl{1},{[1 1 0]},'solid');
```

or you can add the second field (and suppress making a new orthviews figure) by typing ‘add’ and you can make the blobs transparent by typing ‘trans’

```
>> cluster_orthviews(cl{2},{[0 0 1]},'add','trans');
```



It is also possible to display results for specific statistical contrasts. Remember to load the correct clusters.mat file (which has the needed cl variable). When you pass this cl variable into the workspace (by loading the clusters.mat file), it will replace the current cl.

```
>> Meta_Activation_FWE('results', 1, 'poscon')
```

This saves files to disk but does not return them to the workspace. You must do so with the load command. When you load a new clusters.mat file, the new cl will replace the old one.

```
load Emotion_AngerDisgust_Pos_clusters.mat
```

This will run FWE in "results" mode, indicate that you want to make plots ("1") for positive contrasts (e.g., Anger > Disgust when anger was coded 1 and disgust -1 in the original statistical contrasts).

If you have more than one statistical contrast, matlab will ask:

```
Choose contrast number (1 thru 15):
```

There are 15 choices because I entered 15 statistical contrasts in FWE setup.

Or you can just type

```
>>  Meta_Activation_FWE('results', 1, 'poscon', 'contrast', 1)
```

And then you get images, plus a clusters.mat file, that correspond to this task contrast that you included in FWE setup.

e.g.,

Emotion_AngerDisgust_Pos_FWE_all.img
Emotion_AngerDisgust_Pos_FWE_extent.img
Emotion_AngerDisgust_Pos_FWE_height.img
Emotion_AngerDisgust_Pos_clusters.mat

These maps will display Anger > Disgust thresholded activations (because anger was coded 1 and disgust -1 in FWE Setup, this is the positive version of the task contrast).

```
>> Meta_Activation_FWE('results', 1, 'negcon')
```

This will run the same thing for negative contrasts (e.g., Disgust > Anger)

You can indicate the degree of thresholding you want with Meta_Activation_FWE('results', 1, 'height', 'stringent', 'medium', 'lenient')

```
>> Meta_Activation_FWE('results', 1, 'height', 'stringent')
```

Creates a figure with for stringent (p < .001) height threshold

Or

```
>> Meta_Activation_FWE('results', 1, 'negcon', 'height', 'stringent')
```

Creates a figure of negative contrasts with most stringent height values.

When you have multiple task contrasts, you specify which one you want to view with

```
>>  Meta_Activation_FWE('results', 1, 'negcon', 'contrast', 3)
```

Negative con values for contrast 3 so you have to know the order in which you entered contrasts in FWE Setup.

You can also change the color of the figures.

```
>>  Meta_Activation_FWE('results','colors',[1 1 0;1 .5 0;1 .3 .3;.8 .4 .4])
```

Colors are specified for each threshold (separated by a semi-colon).

```
>> Meta_Activation_FWE('results','grayscale');
```

You can also plot activations as they appear on the surface of the brain:

```
>> Meta_Activation_FWE('results','surface');
```

Finally, you can save slices showing each cluster center:

```
>> Meta_Activation_FWE('results','slices');
```

This makes slices for summary map

```
>> Meta_Activation_FWE('results', 1, 'negcon','contrast',8,'slices');
```

This makes slices for task contrast 8.

For example, lets display the results for the anger vs disgust contrast:

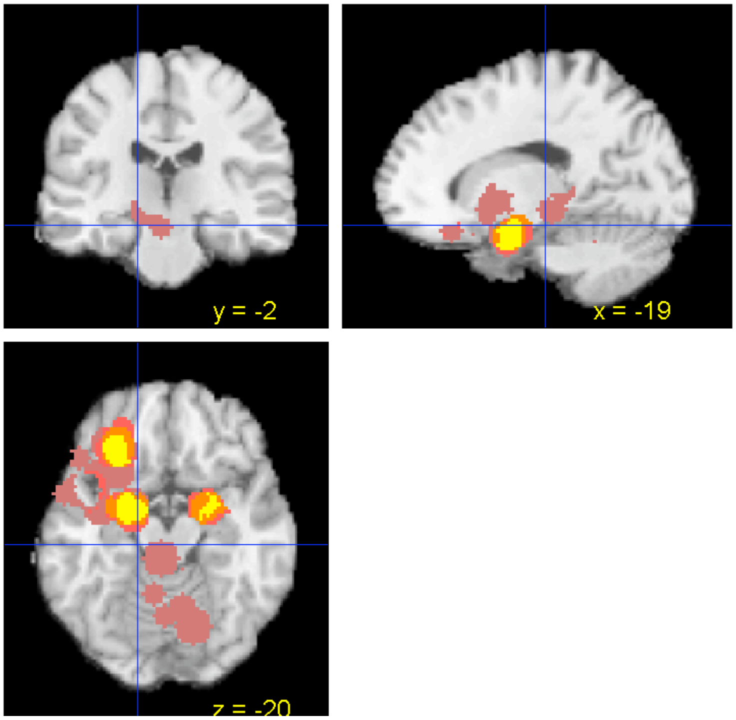
```
>> Meta_Activation_FWE('results',1,'poscon','contrast',1)
```

```
p < .05 Corrected:   0 voxels
Writing image: Emotion_AngerDisgust_Pos_FWE_height.img
Writing: Emotion_AngerDisgust_Pos_FWE_height.imgWarning: Cant get default Analyze
orientation - assuming flipped
```

comment on what this means

```
Thresh = 0.1543, size threshold = 5546: 6331 sig. voxels.
Cluster sizes range from 6331 to 6331
Thresh = 0.1881, size threshold = 5106:   0 sig. voxels.
Thresh = 0.2822, size threshold = 515:   0 sig. voxels.
=====================================================
Showing positive or negative: both
Overlay is: /fMRI Analysis
Software/3DheadUtility/SPM2_brains/spm2_single_subj_T1_scalped.img
Height thresholds:0.4964 0.2822 0.1881 0.1543
Extent thresholds:   1 515 5106 5546
Show only contiguous with seed regions: No

A windowbtnup function already exists; not showing position coordinates.
Saving clusters as cl variable in Emotion_AngerDisgust_Pos_clusters.mat
Height
No results to print.

Additional regions at Extent: Stringent and size >=  10
Additional regions at Extent: Medium and size >=  10
Additional regions at Extent: Lenient and size >=  10


index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -14   -24   -14   NaN   6331   50648 0.42
```

So in this analysis, the only one cluster is significant at p < .05.

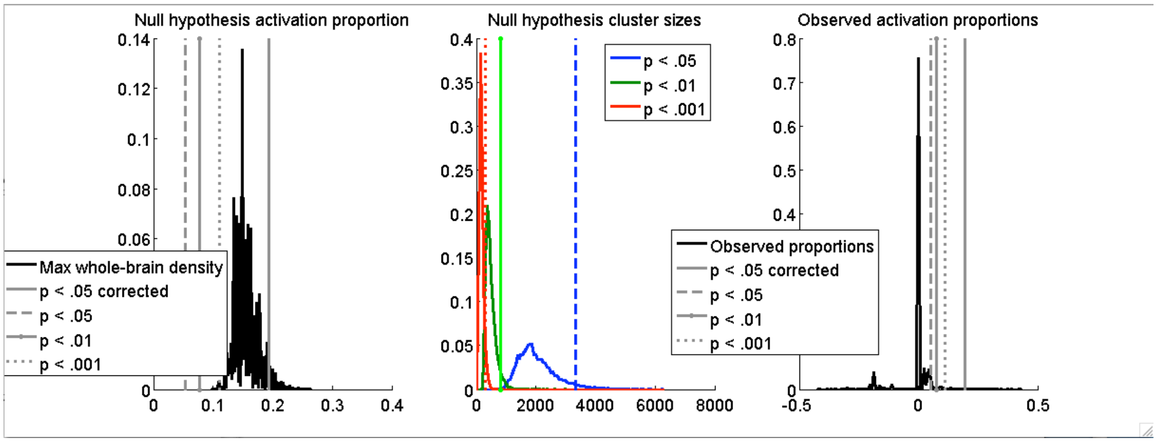


We can confirm this by looking at the cl structure:

```
>> load Emotion_AngerDisgust_Pos_clusters.mat
>> cl

cl =

     []      []      []     [1x1 struct]
```

Now lets view the negative contrast (Disgust>Anger):

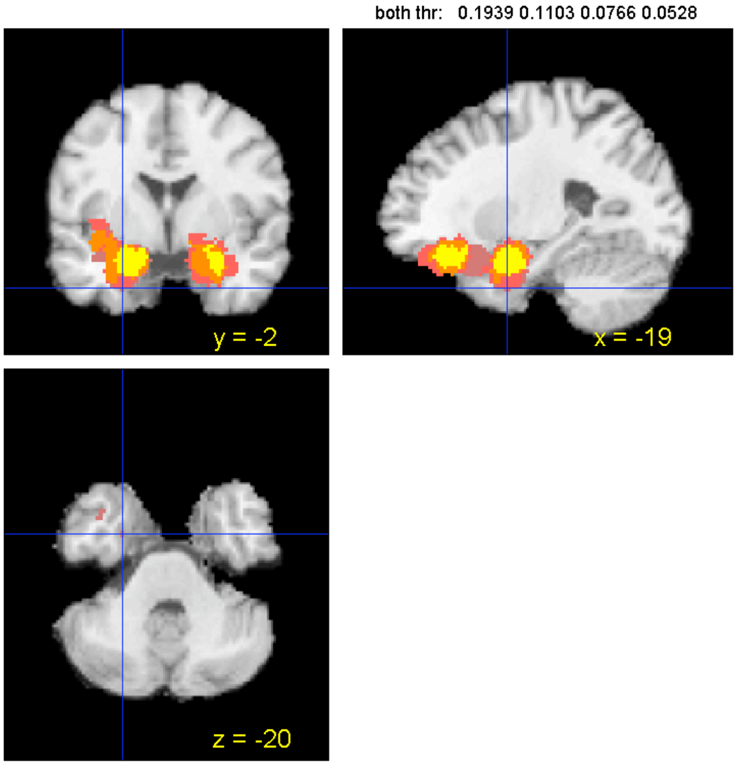
```
>> Meta_Activation_FWE('results',1,'negcon','contrast',1)
```



```
p < .05 Corrected: 825 voxels
Writing image: Emotion_AngerDisgust_Neg_FWE_height.img

Thresh = 0.0528, size threshold = 3330: 3465 sig. voxels.
Cluster sizes range from 3465 to 3465
```

```
Thresh = 0.0766, size threshold = 806: 3729 sig. voxels.
Cluster sizes range from 966 to 1561

Thresh = 0.1103, size threshold = 318: 2101 sig. voxels.
Cluster sizes range from 571 to 937

iimg_multi_threshold viewer
=====================================================
Showing positive or negative: both
Overlay is: /fMRI Analysis
Software/3DheadUtility/SPM2_brains/spm2_single_subj_T1_scalped.img
Height thresholds:0.1939 0.1103 0.0766 0.0528
Extent thresholds:   1 318 806 3330

Saving clusters as cl variable in Emotion_AngerDisgust_Neg_clusters.mat
Height
Getting local maxima within 10 mm for subcluster reporting

index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -20    -2   -20   NaN   415   3320  0.43
  2    26     0   -20   NaN   164   1312  0.33
  3   -28    30   -18   NaN   286   2288  0.32
  4   -20    38   -14   NaN     1      8  0.23

Additional regions at Extent: Stringent and size >=  10
No results to print.

Additional regions at Extent: Medium and size >=  10
No results to print.

Additional regions at Extent: Lenient and size >=  10

index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -26   -10   -30   NaN    37   296   0.05
  2   -28    26    -4   NaN    29   232   0.06
```

And we get this orthviews slice (at the location of the first cluster in the table above). This figure displays both the height (in yellow) and the extent (orange, red, purple) based results.



If you want to know which contrasts contribute to this particular cluster (within 10 mm of the center of the activation, you type

>> Meta_interactive_table

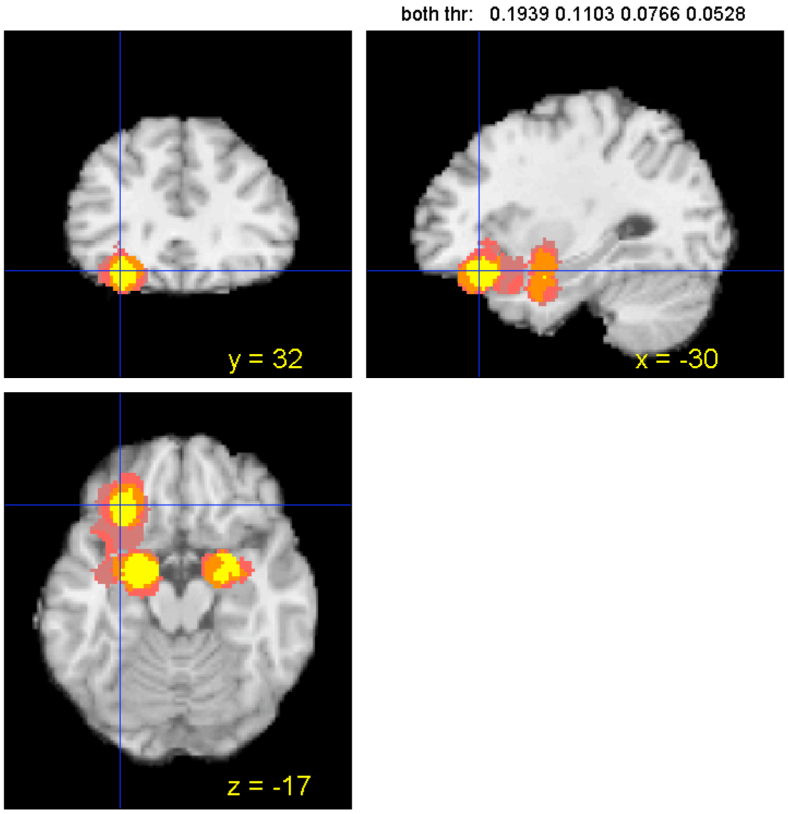
And matlab will as you for the field in the DB structure to display. We type Emotion because that is the relevant data contrast field. Note: If you click on a point, Meta_interactive_table will list all the studies that show activity within 10 mm of that point according to whatever DB is loaded. You can check to see which DB is loaded by typing which DB. If you select contrasts first (using Meta_Select_Contrasts) then you won't have to select a contrast field.

```
>> Meta_interactive_table

Warning: Variable 'testfield' not found.
> In Meta_interactive_table at 21
Cannot load testfield from SETUP.  Type name of field in DB to display: Emotion

testfield =

Emotion

Studies activating within  10 mm of       -20    -1   -21
Study x     y     z     Emotion     N     Contrast #  point index dist
Goldin2005  -16    -6   -13   sad    13    99   282   9.65
Hutcherson05      -18    -8   -15   sad    28   141   321   9.29
Lane97c     -26    -1   -14   sad    11   178   354   9.12
Levesque2003      -24    -3   -18   sad    20   198   402   5.66
Liotti00    -24     5   -16   fear    8   211   414   8.56
Schafer2005 -18    -6   -18   disgust      40   323   505   6.16
Schafer2005 -24     0   -27   disgust      40   324   514   7.28
Schafer2005 -24    -3   -27   disgust      40   327   522   7.48
```

```
Schafer2005 -18     0   -21   disgust      40   328   532   2.24
Schienle06  -24    -3   -18   disgust      12   331   546   5.39
Schienle06  -24     0   -15   disgust      12   332   551   7.28
Schienle2002      -18    -3   -21   disgust      12   335   561   2.83
Schienle2002      -18    -3   -21   disgust      12   337   571   2.83
Stark2003   -18    -6   -18   disgust      19   360   574   6.16
Whalen1998  -15     1   -14   fear    8   399   598   8.45

Level Count TotalContrasts    %Activating Weighted    Weighted%
disgust       9    22   40.91 0.13  49.08
fear    2    19   10.53 0.02  7.59
sad     4    30   13.33 0.05  16.97
```

Notice that this table includes imaging contrasts that are not related to disgust or anger (even though the task contrast is disgust vs anger). This is because this function lists all contrasts that are significant at that location, even if they are not in the specific task contrast you are looking at. This function works off the DB of ALL contrasts in either the saved DB.mat or SETUP.mat, not just the ones you selected.

We can also move around in the brain space to see another slice (cluster 2)



To save a particular orthviews slice (for a manuscript), type

```
>> scn_export_papersetup
```

This sets the figure size.

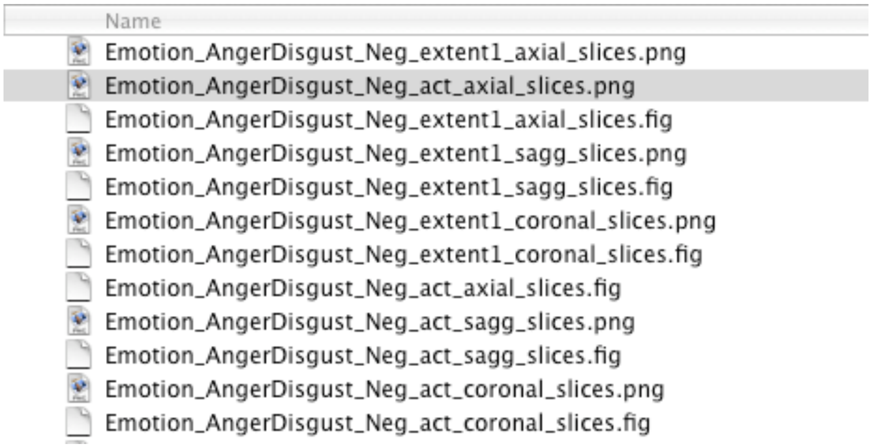
```
>> saveas(gcf,'test','png')
```

This saves the slice in a file named “test” as a .png format. You can also save as a tif.

You can also plot the significant activations in slices showing each cluster center

```
>> Meta_Activation_FWE('results',1,'negcon','contrast',1,'slices')
```

and you get a series of new files which show slices in all 3 dimensions:



Name
Emotion_AngerDisgust_Neg_extent1_axial_slices.png
Emotion_AngerDisgust_Neg_act_axial_slices.png
Emotion_AngerDisgust_Neg_extent1_axial_slices.fig
Emotion_AngerDisgust_Neg_extent1_sagg_slices.png
Emotion_AngerDisgust_Neg_extent1_sagg_slices.fig
Emotion_AngerDisgust_Neg_extent1_coronal_slices.png
Emotion_AngerDisgust_Neg_extent1_coronal_slices.fig
Emotion_AngerDisgust_Neg_act_axial_slices.fig
Emotion_AngerDisgust_Neg_act_sagg_slices.png
Emotion_AngerDisgust_Neg_act_sagg_slices.fig
Emotion_AngerDisgust_Neg_act_coronal_slices.png
Emotion_AngerDisgust_Neg_act_coronal_slices.fig

Meta_Activation_FWE (‘results’,’surface’): Using FWE results with this argument allows you to plot surface figures to summarize the density analysis; it uses the more general matlab procedure cluster_surf in a specified way

?? >> Meta_Activation_FWE('results',1,'negcon','contrast',1,’surface’) – need to debug

STEP 3: ADDITIONAL WAYS OF DISPLAYING RESULTS

a. montage_clusters

montage_clusters allows you to display results in a series of axial slices in the brain. The general format is

montage_clusters(ovl, clusters, varargin)

where ovl is the brain image you overlay on the significant clusters and varargin (in any order) = additional clusters structures or a cell array of colors (text format), must be ROW vector {'r' 'g'} etc.. (if length of color string is longer than number of clusters inputs, additional colors are interpreted as '2-intersection' and 'all-intersection' colors, in that order). For more information, type help montage_clusters.

e.g., lets take a look at Sad vs Anger task contrast.

```
>> Meta_Activation_FWE('results',1,'negcon','contrast',4)
```

Here are the results (for height only)

```
index x     y     z     corr  voxels      volume_mm3  maxstat
```

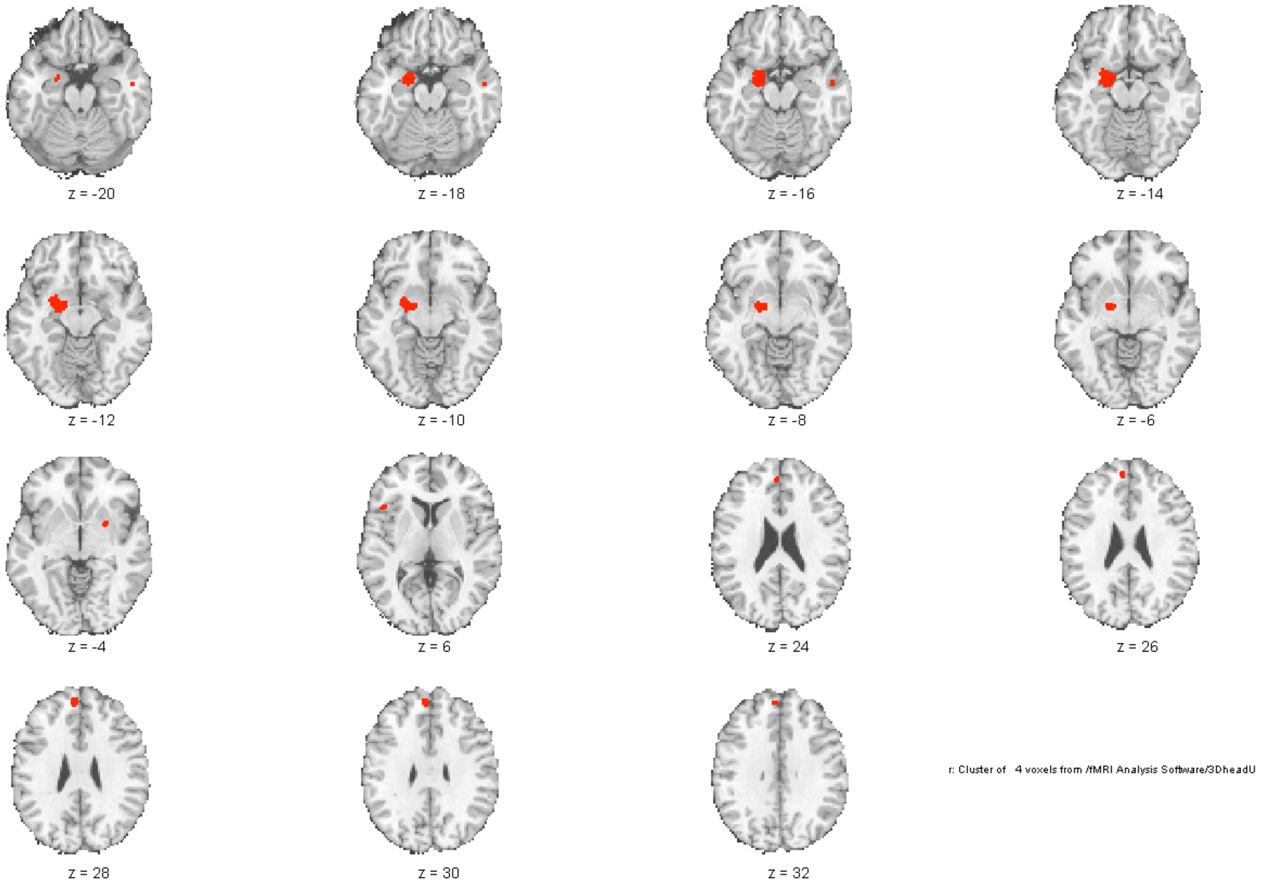
```
  1    52   -10   -16   NaN     4    32    0.26
  2   -22    -4   -14   NaN   205   1640   0.36
  3    26     4    -4   NaN     1     8    0.24
  4   -46    20     6   NaN     4    32    0.27
  5    -2    48    24   NaN     1     8    0.22
  6    -6    52    28   NaN    21   168    0.22
```

Now, to display results as a montage, first load the relevant clusters.mat file so that we have the cl variable that we need

```
>> load Emotion_AngerSad_Neg_clusters.mat
>> montage_clusters([],cl{1})
```

This says use the default brain image to overlay blobs on, and display the data from the height field of the cl variable in Emotion_AngerSad_Neg_clusters.mat.

Here is the montage of those results

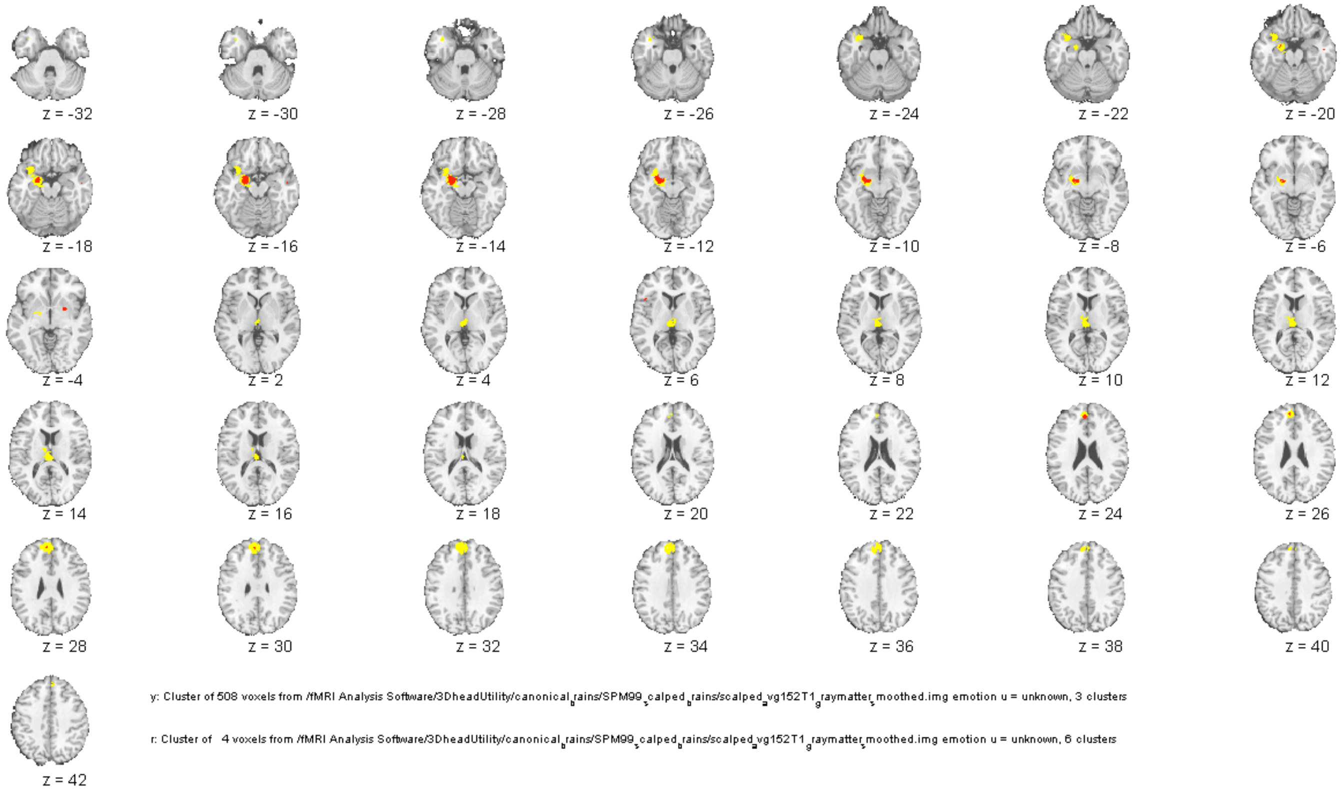


Now lets say we also want to look at the second cl field, which is the extent for p < .001 (stringent):

```
Additional regions at Extent: Stringent and size >=  10


index x      y      z      corr   voxels      volume_mm3   maxstat
  1    -2   -20     10    NaN    244    1952   0.21
```

```
>> montage_clusters([],cl{1},cl{2},{'y' 'r'})
```

```
2 Clusters found. %two cl fields found
Two cluster overlap: m, found 0 voxels.
All cluster overlap: y, found 0 voxels.
```


z = -32
z = -30
z = -28
z = -26
z = -24
z = -22
z = -20
z = -18
z = -16
z = -14
z = -12
z = -10
z = -8
z = -6
z = -4
z = 2
z = 4
z = 6
z = 8
z = 10
z = 12
z = 14
z = 16
z = 18
z = 20
z = 22
z = 24
z = 26
z = 28
z = 30
z = 32
z = 34
z = 36
z = 38
z = 40
z = 42
y: Cluster of 508 voxels from /fMRI Analysis Software/3DheadUtility/canonical$_b$rains/SPM99$_s$calped$_b$rains/scalped$_a$vg152T1$_g$raymatter$_s$moothed.img emotion u = unknown, 3 clusters
r: Cluster of 4 voxels from /fMRI Analysis Software/3DheadUtility/canonical$_b$rains/SPM99$_s$calped$_b$rains/scalped$_a$vg152T1$_g$raymatter$_s$moothed.img emotion u = unknown, 6 clusters

It is not possible to plot different blobs within a cl field in different colors. Rather, different cl fields can be plotted in different colors.

b. cluster_orthviews

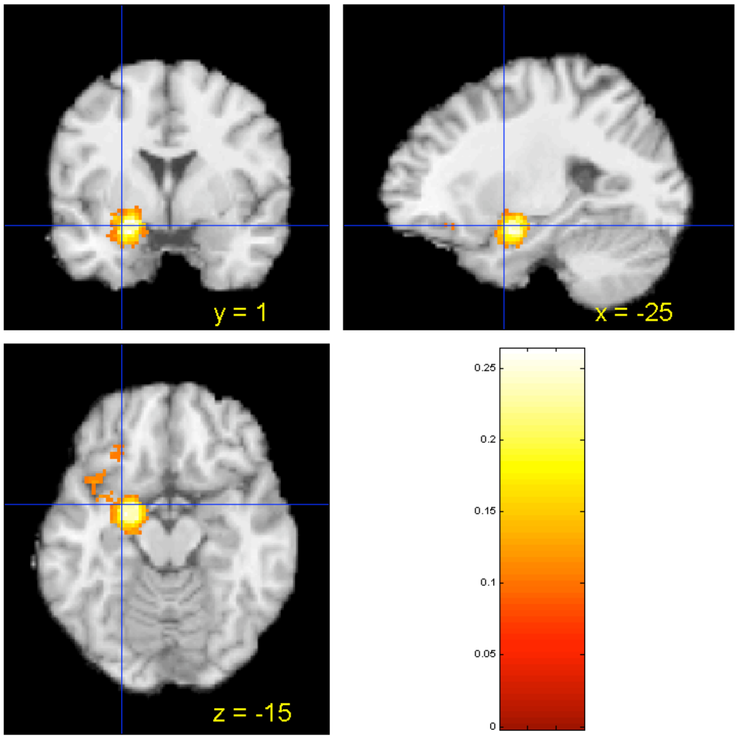
The cluster_orthviews function uses spm_orthviews to display activation blobs from a clusters structure on a canonical structural brain image. Multiple clusters may be plotted in multiple colors, and blobs may be added to an existing orthviews figure. If you do not specify colors, the function will use the proportions of activation to color map (or significant differences in proportions if the image relates to a task contrast; in other analyses it might use z scores or t scores to color map).

From the help file:
opt: 'add' to suppress making new orthviews
'copy' to copy to smaller subfigure with empty axis beside it
'unique' to display in unique colors for each cluster
'overlay', followed by name of image, to use a custom anatomical overlay
'bivalent', to plot increases in red and decreases in blue

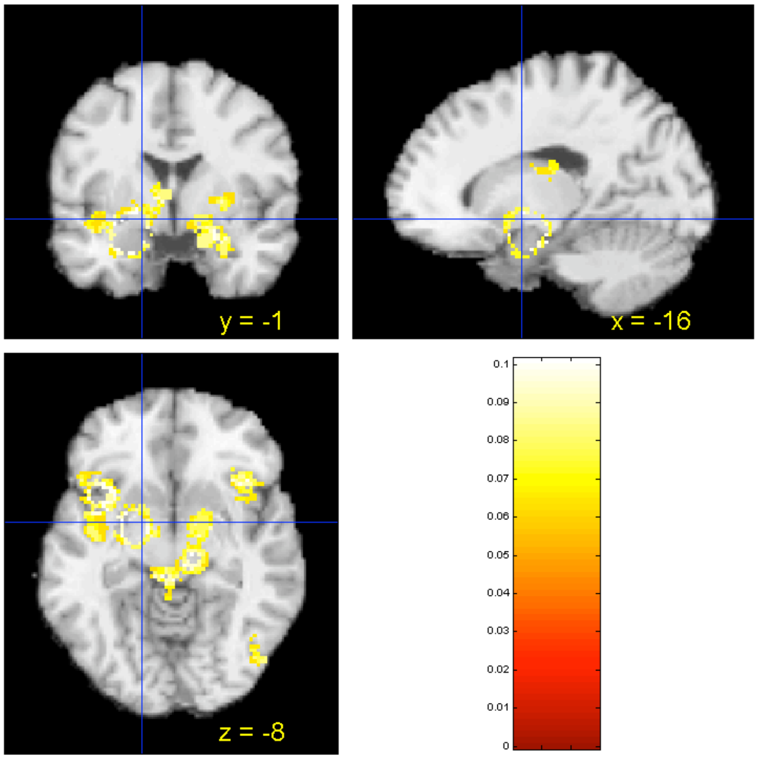
For example:

```
>> load Activation_clusters.mat
>> cluster_orthviews(cl{1});
```

This color map shows you the proportion of data contrasts activating (or the difference in the proportions for the conditions that make up a task contrast):



```
>> cluster_orthviews(cl{2})
```



The holes represent activations that appeared in cl{1} because cl{2} represents incrementally significant voxels.

c. cluster_tables

cluster_tables will allow you to print a table of MNI coordinates and cluster sizes for the clusters in any cell. You also have the option to print text labels from Carmack atlas (linked to MNI coordinates).

cluster_table(clusters, [opt] subclusters)

From the help file:

```
cluster_table(cl);              % create subclusters on the fly, prompt for labels
cluster_table(cl, 0, 1);        % no subclusters, but labels
cluster_table(cl, 1, 0);        % create subclusters but no labels
cluster_table(..., 'tal_info', xyz, L3, L5);    % 3 input variables following
                                                'tal_info' are interpreted as xyz, L3, and
                                                L5 from talairach_info.mat.
  cluster_table(..., 'talairach'); % loads labels from taldata.mat
                                (Talairach database) instead of
                                talairach_info.mat. Note that you should
                                use the 'tal_info' call above if xyz, L3,
                                and L5 have already been loaded to the
                                workspace from taldata.mat. Also, if the
                                talairach database is being used, your
                                cl.XYZmm values MUST correspond to the
                                TALAIRACH, NOT MNI, database, or the
                                labels will be innaccurate.
```

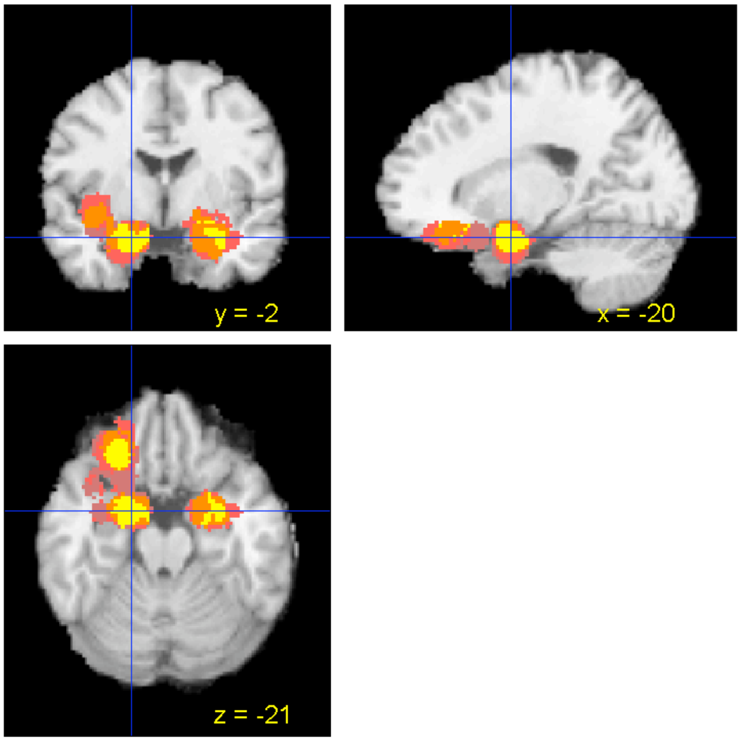
Tor does not remember the difference between talairach_info.mat and taldata.mat. To transform the coordinates in the cl variable (which are in MNI space) to get them into talairach space, you can use min2tal (or to go the other way use tal2mni)

For other info type help cluster_table

For example, lets take the Disgust vs Happy task contrast (remembering that height-based thresholded voxels are in yellow and extent-based thresholded voxels are represented by the other colors)

```
>> Meta_Activation_FWE('results',1,'poscon','contrast',6)
```

both thr: 0.1939 0.1094 0.0751 0.0509



```
Thresh = 0.0509, size threshold = 3264: 3358 sig. voxels.
Cluster sizes range from 3358 to 3358

Thresh = 0.0751, size threshold = 809: 3500 sig. voxels.
Cluster sizes range from 880 to 1635

Thresh = 0.1094, size threshold = 313: 1926 sig. voxels.
Cluster sizes range from 520 to 853

Height thresholds:0.1939 0.1094 0.0751 0.0509
Extent thresholds:   1 313 809 3264


Height
Getting local maxima within 10 mm for subcluster reporting

index x      y      z      corr   voxels       volume_mm3   maxstat
  1   -20    -2    -22     NaN    324    2592   0.43
  2    26    -2    -18     NaN    128    1024   0.33
  3   -26    30    -18     NaN    245    1960   0.32
  4   -16    -6    -14     NaN      1       8   0.20
  5   -20    38    -14     NaN      1       8   0.23

Additional regions at Extent: Stringent and size >=  10
index x      y      z      corr   voxels       volume_mm3   maxstat
  1   -30     0    -16     NaN    525    4200   0.19

Additional regions at Extent: Medium and size >=  10
No results to print.

Additional regions at Extent: Lenient and size >=  10
index x      y      z      corr   voxels       volume_mm3   maxstat
  1   -26   -10    -30     NaN     37    296    0.05
  2   -38    36    -16     NaN     70    560    0.05
  3   -32    -6     -2     NaN     17    136    0.05
```

Now we can look for subclusters using cluster_table. Notice there are no subclusters here because cl{1} looks the same when comparing the Meta_Activation_FWE output and the clusters_table output below:

```
>> load Emotion_DisgustHappy_Pos_clusters.mat

cl =

    [1x5 struct]    [1x5 struct]    [1x11 struct]    [1x22 struct]
```

There are 5 regions in cl{1}, 5 regions in cl{2}, 11 in cl{3}, and 22 in cl{4}.

```
>> cluster_table(cl{1},1,0)

Getting local maxima within 10 mm for subcluster reporting
index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -20    -2   -22   NaN   324   2592   0.43
  2    26    -2   -18   NaN   128   1024   0.33
  3   -26    30   -18   NaN   245   1960   0.32
  4   -16    -6   -14   NaN     1      8   0.20
  5   -20    38   -14   NaN     1      8   0.23
```

We turned subclustering on, but there are no subclusters reported. That may be because there are no subclusters, or because subcluster reporting does not work with cl{1}. Tor is not sure.

Now lets look at cl{2}. Notice it does not match the original Meta_Activation_FWE output because some of the clusters that are significant according to the extent-based p < .001 threshold are actually within 10 mm of the center of the clusters that are significant according to the height-based threshold. Only the incrementally significant voxels are printed as output from Meta_Activation_FWE but the cl variable contains all the information.

```
>> cluster_table(cl{2},0,0)

index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -30     0   -16   NaN   525   4200   0.19
  2    22    -2   -20   NaN   425   3400   0.19
  3   -18    -2   -30   NaN     2     16   0.19
  4   -30    30   -18   NaN   274   2192   0.19
  5   -24    -8   -18   NaN     1      8   0.17
```

Now we can look for subclusters.

```
>> cluster_table(cl{2},1,0)
Getting local maxima within 10 mm for subcluster reporting
index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -30     0   -16   NaN   525   4200   0.19
->    -28     0   -26   NaN   106   0.18921
->    -12     0   -24   NaN    56   0.18921
->    -14    -4   -18   NaN    49   0.18478
->    -36    -2   -12   NaN   314   0.18921
  2    22    -2   -20   NaN   425   3400   0.19
->     26     0   -26   NaN   177   0.18949
->     20    -2   -16   NaN   248   0.19216
  3   -18    -2   -30   NaN     2     16   0.19
  4   -30    30   -18   NaN   274   2192   0.19
->    -22    36   -18   NaN   185   0.18921
->    -32    32   -16   NaN    89   0.18921
```

```
  5   -24    -8   -18   NaN      1     8   0.17

>> cluster_table(cl{3},0,0)

index x      y      z      corr  voxels      volume_mm3  maxstat
  1   -34     4   -14   NaN   711   5688  0.12
  2    28    -6   -14   NaN   431   3448  0.13
  3   -14    -6   -28   NaN     1      8   0.08
  4   -12    -2   -28   NaN     2     16   0.08
  5   -24    40   -20   NaN   359   2872  0.11
  6   -16   -10   -20   NaN    65   520   0.11
  7    18   -10   -20   NaN     1      8   0.09
  8    -8    -2   -20   NaN     1      8   0.08
  9   -34    -2   -18   NaN     1      8   0.09
 10   -24    20   -18   NaN     1      8   0.11
 11   -36   -12   -12   NaN     1      8   0.09

>> cluster_table(cl{3},1,0)
Getting local maxima within 10 mm for subcluster reporting

index x      y      z      corr  voxels      volume_mm3  maxstat
  1   -34     4   -14   NaN   711   5688  0.12
->    -32    -4   -20   NaN   303   0.10935
->    -38     8   -12   NaN   408   0.12328
  2    28    -6   -14   NaN   431   3448  0.13
->     32     0   -26   NaN   208   0.12799
->     24    -4   -10   NaN   223   0.10902
  3   -14    -6   -28   NaN     1      8   0.08
  4   -12    -2   -28   NaN     2     16   0.08
  5   -24    40   -20   NaN   359   2872  0.11
->    -32    24   -20   NaN   155   0.10935
->    -24    40   -20   NaN   204   0.10935
  6   -16   -10   -20   NaN    65   520   0.11
->    -14   -10   -20   NaN    43   0.10492
->    -20    -6   -16   NaN    22   0.10695
  7    18   -10   -20   NaN     1      8   0.09
  8    -8    -2   -20   NaN     1      8   0.08
  9   -34    -2   -18   NaN     1      8   0.09
 10   -24    20   -18   NaN     1      8   0.11
 11   -36   -12   -12   NaN     1      8   0.09
```

We can manage which regions we want to report as distinct by defining subclusters as clusters (using subclusters_from_local_max) or by merging nearby clusters (using merge_nearby_clusters). The size of clusters to report depends on hypotheses and overall pattern of results.

We can define the subclusters as a new variable using subclusters_from_local_max

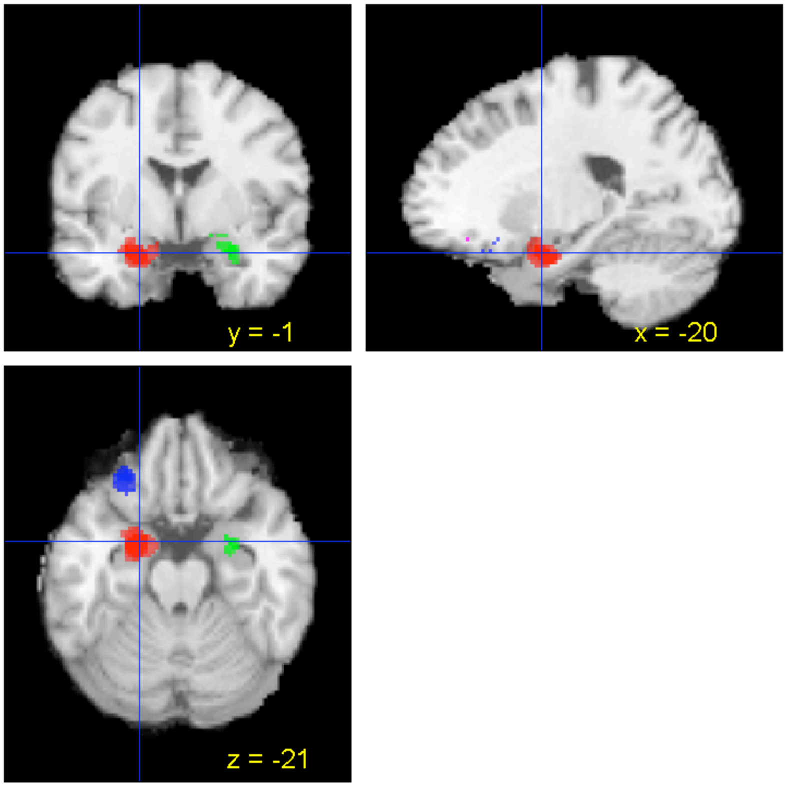
```
>> subcl = subclusters_from_local_max(cl{1}, 10);
>> cluster_table(subcl);

index x      y      z      corr  voxels      volume_mm3  maxstat
  1   -20    -2   -22   NaN   324   2592  0.43
  2    26    -2   -18   NaN   128   1024  0.33
  3   -26    30   -18   NaN   245   1960  0.32
  4   -16    -6   -14   NaN     1      8   0.20
  5   -20    38   -14   NaN     1      8   0.23
```

It appears as if there are no subclusters.

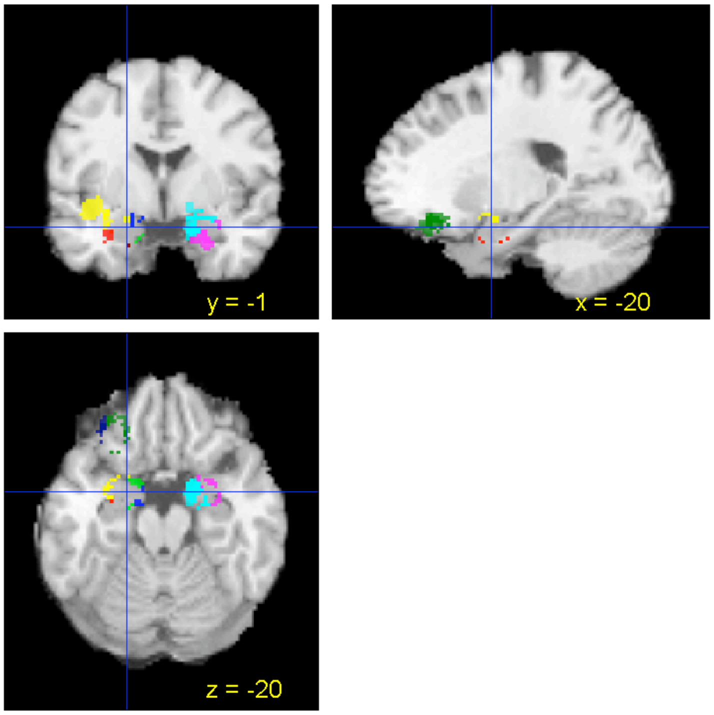
```
>> subcl = subclusters_from_local_max(cl{2}, 10);
>> cl{2} = subcl;
>> cluster_table(cl{2},0,0)
```

Now all the subclusters are defined as clusters, and there are 10 regions in the cl{2}field.

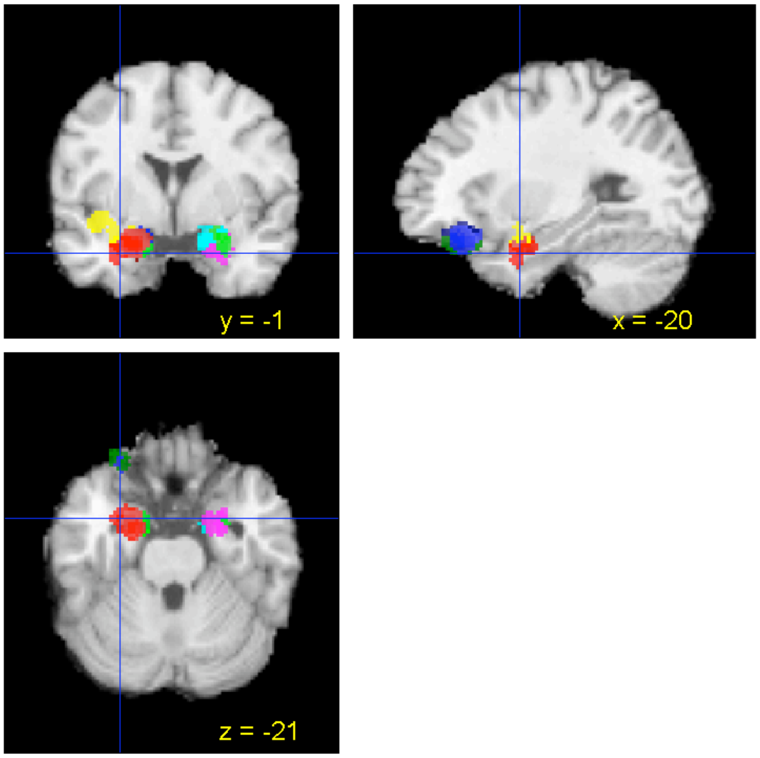
```
index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -28     0   -26   NaN   106   848   0.19
  2   -12     0   -24   NaN    56   448   0.19
  3   -14    -4   -18   NaN    49   392   0.18
  4   -36    -2   -12   NaN   314   2512  0.19
  5    26     0   -26   NaN   177   1416  0.19
  6    20    -2   -16   NaN   248   1984  0.19
  7   -18    -2   -30   NaN     2     16  0.19
  8   -22    36   -18   NaN   185   1480  0.19
  9   -32    32   -16   NaN    89   712   0.19
 10   -24    -8   -18   NaN     1      8  0.17

ans =

1x10 struct array with fields:
    title
    threshold
    M
    dim
    voxSize
    name
    numVox
    XYZ
    XYZmm
    Z
    mm_center
    from_cluster


>> cluster_orthviews(cl{1},'unique')
```



```
>> cluster_orthviews(subcl,'unique')
```



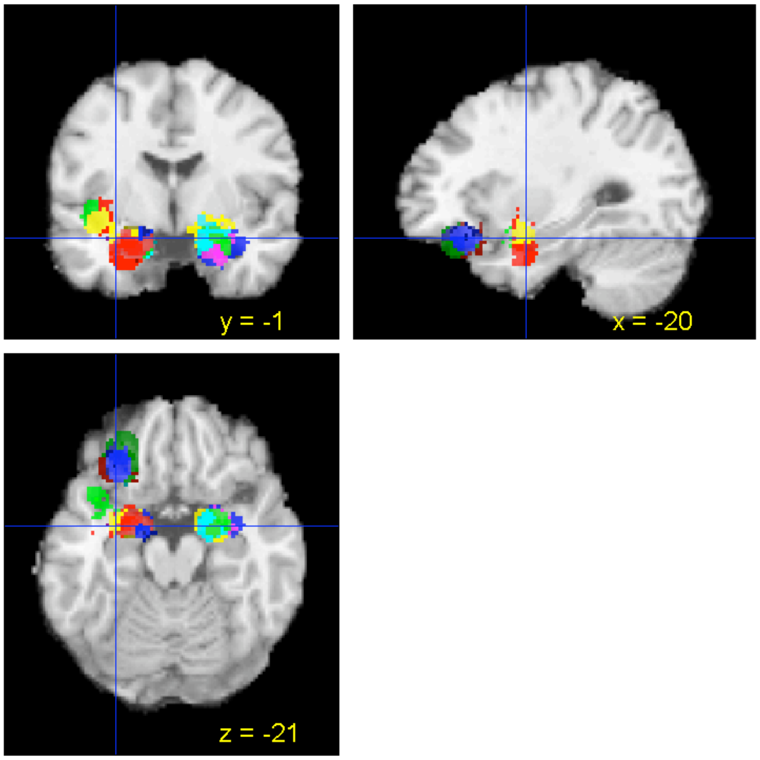
```
>> cluster_orthviews(subcl,'add','unique')
```



```
>> cluster_table(cl{3},1,0)

Getting local maxima within 10 mm for subcluster reporting
index x      y      z      corr   voxels      volume_mm3   maxstat
   1   -34     4    -14    NaN    711    5688   0.12
->     -32    -4    -20    NaN    303    0.10935
->     -38     8    -12    NaN    408    0.12328
   2    28    -6    -14    NaN    431    3448   0.13
->      32     0    -26    NaN    208    0.12799
->      24    -4    -10    NaN    223    0.10902
   3   -14    -6    -28    NaN      1       8   0.08
   4   -12    -2    -28    NaN      2      16   0.08
   5   -24    40    -20    NaN    359    2872   0.11
->     -32    24    -20    NaN    155    0.10935
->     -24    40    -20    NaN    204    0.10935
   6   -16   -10    -20    NaN     65    520    0.11
->     -14   -10    -20    NaN     43    0.10492
->     -20    -6    -16    NaN     22    0.10695
   7    18   -10    -20    NaN      1       8   0.09
   8    -8    -2    -20    NaN      1       8   0.08
   9   -34    -2    -18    NaN      1       8   0.09
  10   -24    20    -18    NaN      1       8   0.11
  11   -36   -12    -12    NaN      1       8   0.09

ans =

1x11 struct array with fields:
    title
    threshold
    M
    dim
    voxSize
    name
    Z
    XYZmm
    XYZ
    numVox
    mm_center

>> cluster_table(subcl,1,0)
```

Notice that this could be parsed even further into subclusters if we wanted to.

```
Getting local maxima within 10 mm for subcluster reporting
index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -32    -4   -20   NaN   303   2424  0.11
  2   -38     8   -12   NaN   408   3264  0.12
->    -26    10   -24   NaN    27   0.08335
->    -40     8   -10   NaN   381   0.12328
  3    32     0   -26   NaN   208   1664  0.13
->     22     0   -30   NaN    86   0.10492
->     36     0   -26   NaN    59   0.12799
->     34     2   -20   NaN    63   0.10492
  4    24    -4   -10   NaN   223   1784  0.11
->     14    -4   -20   NaN    38   0.10853
->     28    -2   -12   NaN    73   0.10902
->     18    -6   -12   NaN   112   0.10902
  5   -14    -6   -28   NaN     1     8   0.08
  6   -12    -2   -28   NaN     2    16   0.08
  7   -32    24   -20   NaN   155   1240  0.11
  8   -24    40   -20   NaN   204   1632  0.11
  9   -14   -10   -20   NaN    43   344   0.10
 10   -20    -6   -16   NaN    22   176   0.11
->    -22    -6   -16   NaN    18   0.10695
->    -18    -2   -12   NaN     4   0.09273
 11    18   -10   -20   NaN     1     8   0.09
 12    -8    -2   -20   NaN     1     8   0.08
 13   -34    -2   -18   NaN     1     8   0.09
 14   -24    20   -18   NaN     1     8   0.11
 15   -36   -12   -12   NaN     1     8   0.09

ans =

1x15 struct array with fields:
    title
    threshold
    M
    dim
    voxSize
    name
    numVox
    XYZ
    XYZmm
    Z
    mm_center

>> cluster_orthviews(subcl,'add','unique')
```



We can merge clusters using merge_nearby_clusters. Note that this relies on parcel_complete_sets.m which is experimental.

From help merge_nearby_clusters

newcl = merge_nearby_clusters(cl, thr)

Merge sets of clusters whose centers are all within thr mm of each other. You can run this function recursively until all clusters are > thr mm apart

newcl = merge_nearby_clusters(cl, thr, 'recursive')

subcl2 = merge_nearby_clusters(subcl,20,'recursive');

d. cluster_table_successive_threshold

cluster_table_successive_threshold uses cluster_table in a specific way to report clusters that are not already in the first set reported. So, it will plot blobs in cl{1}, and then only the unique blobs in cl{2} (not within 10mm of a previously printed cluster), and so on. This will reproduce what you get from FWE results. But if you manipulate the size of clusters before running this function, then you can manipulate how you parse the significant voxels into clusters.

From help cluster_table_successive_threshold

cl = cluster_table_successive_threshold(cl,[sizethr]) %sizethr = the number of contiguous voxels you have to have

cluster table of a cell array of clusters cl{1} cl{2} etc.
Prints table of cl{1} and then any additional regions in cl{2:n} that are not within 10 mm of a previously printed cluster; also merges clusters in set within 10 mm

e.g., using cl from Emotion_DisgustHappy_Pos_clusters.mat

```
>> load Emotion_DisgustHappy_clusters.mat
>> clnew = cluster_table_successive_threshold(cl,5);
```

```
Height
Getting local maxima within 10 mm for subcluster reporting


index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -20    -2   -22   NaN   324   2592  0.43
  2    26    -2   -18   NaN   128   1024  0.33
  3   -26    30   -18   NaN   245   1960  0.32
  4   -16    -6   -14   NaN     1      8  0.20
  5   -20    38   -14   NaN     1      8  0.23

Additional regions at Extent: Stringent and size >=   5


index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -30     0   -16   NaN   525   4200  0.19

Additional regions at Extent: Medium and size >=   5
No results to print.

Additional regions at Extent: Lenient and size >=   5


index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -26   -10   -30   NaN    37   296   0.05
  2   -38    36   -16   NaN    70   560   0.05
  3   -32    -6    -2   NaN    17   136   0.05
```

e. mask2clusters

To create a new variable cl that indicates the clusters or regions of significant activation, use the matlab command cl=mask2clusters('myfilename.img'). mask2clusters uses the specified img file which is a mask (an image of 1s for significant voxels and 0s for nonsignificant voxels) to create a new cl variable. Check help mask2clusters for important caveats when using this procedure.

To visualize image and produce a map where the activations are all non-random, use cluster_orthviews(cl)

f. cluster_surf – must debug. Currently not working

g. Meta_cluster_tools

This function contains multiple tools for working with clusters derived from Meta_Activation_FWE, Meta_Chisq_new and Meta_SOM tools. For example, it can extract data and print a table for a set of meta-analysis clusters. This function can do more specific work than cluster_tables. The tables generated by Meta_cluster_tools are similar, but not equivalent, to cluster_tables. Activations may be listed in a different order, and this table also lists activation proportions (that it gets from MC_Setup). This requires more explanation and additional detail about functionality (not much is in the help file).

For example:

```
>> load Emotion_DisgustHappy_Pos_clusters.mat
>> load MC_Info

>> cl1 = Meta_cluster_tools('make_table',cl{1},MC_Setup);
```

Anger_prop are the proportion of contrasts that have activation at this location

```
getting clusters for local maxima at least 10 mm apart
Getting studies that activated in each region.
Counting studies by condition
Adding field to cl: anger_prop
Adding field to cl: disgust_prop
Adding field to cl: fear_prop
Adding field to cl: happy_prop
Adding field to cl: sad_prop
Getting local maxima within 10 mm for subcluster reporting


index x     y     z     corr  voxels      volume_mm3  maxstat     anger_prop
      disgust_prop      fear_prop   happy_prop  sad_prop
  1   -20    38   -14   NaN     1     8   0.23    0   26.77   0     0     0
  2   -26    30   -18   NaN   245   1960  0.32    0   36.60 9.92  5.60  7.47
  3    26    -2   -18   NaN   128   1024  0.33    0   47.00 4.34    0   18.40
  4   -20    -2   -22   NaN   325   2600  0.43  19.17 55.64 11.93 20.89 43.93
```

Now compare this to what you get if you use cluster_table

```
>> cluster_table(cl{1})
Getting local maxima within 10 mm for subcluster reporting
Do text labels for clusters (current = Carmack labels)?0

index x     y     z     corr  voxels      volume_mm3  maxstat
  1   -20    -2   -22   NaN   324   2592  0.43
  2    26    -2   -18   NaN   128   1024  0.33
  3   -26    30   -18   NaN   245   1960  0.32
  4   -16    -6   -14   NaN     1     8   0.20
  5   -20    38   -14   NaN     1     8   0.23
```

Another example:

```
>> cl2 = Meta_cluster_tools('make_table',cl{2},MC_Setup);

getting clusters for local maxima at least 10 mm apart
Getting studies that activated in each region.
Counting studies by condition
Adding field to cl: anger_prop
Adding field to cl: disgust_prop
Adding field to cl: fear_prop
Adding field to cl: happy_prop
Adding field to cl: sad_prop
Getting local maxima within 10 mm for subcluster reporting

index x     y     z     corr  voxels      volume_mm3  maxstat     anger_prop
      disgust_prop      fear_prop   happy_prop  sad_prop
  1   -24    -8   -18   NaN     1     8   0.17    0   36.56   0   13.38 31.28
  2   -32    32   -16   NaN    89   712    0.19    0    36.60 13.67 5.60   5.66
  3   -22    36   -18   NaN   185   1480   0.19    0    36.60 5.31  5.60   7.47
  4   -18    -2   -30   NaN     2    16    0.19    0    27.92   0      0      0
  5    20    -2   -16   NaN   248   1984   0.19    0    51.58 4.34    0   18.40
  6    26     0   -26   NaN   177   1416   0.19    0    37.64 4.34  5.01   8.11
  7   -36    -2   -12   NaN   314   2512   0.19  47.42  56.93 23.81 29.40 49.13
->    -18     6   -16   NaN     8   0.17446
->    -36    -2   -10   NaN   306   0.18921
  8   -16    -2   -18   NaN   105   840    0.19  19.17  52.36 11.93 13.38 33.05
->    -12     0   -24   NaN    56   0.18921
->    -14    -4   -18   NaN    49   0.18478
  9   -28     0   -26   NaN   106   848    0.19    0    52.36 3.25    0   19.65
```

vs

```
>> cluster_table(cl{2})

Getting local maxima within 10 mm for subcluster reporting
Do text labels for clusters (current = Carmack labels)?0

index x      y      z      corr  voxels      volume_mm3  maxstat
  1   -30     0    -16    NaN   525    4200   0.19
->    -28     0    -26    NaN   106    0.18921
->    -12     0    -24    NaN    56    0.18921
->    -14    -4    -18    NaN    49    0.18478
->    -36    -2    -12    NaN   314    0.18921
  2    22    -2    -20    NaN   425    3400   0.19
->     26     0    -26    NaN   177    0.18949
->     20    -2    -16    NaN   248    0.19216
  3   -18    -2    -30    NaN     2     16    0.19
  4   -30    30    -18    NaN   274    2192   0.19
->    -22    36    -18    NaN   185    0.18921
->    -32    32    -16    NaN    89    0.18921
  5   -24    -8    -18    NaN     1      8    0.17

ans =

1x5 struct array with fields:
    title
    threshold
    M
    dim
    voxSize
    name
    Z
    XYZmm
    XYZ
    numVox
    mm_center
```

h. Meta_Study_Table % not clear how to use it or how it is different from the other table functions.

STEP 4: LINKING TO OTHER DATABASES

The function meta_analysis_table takes a list of XYZ mm points, and, for each point you specify, makes a table of other studies in other databases that report peaks within r mm (radius) of the point.

meta_analysis_table(XYZlist,r,[master file name]) where the master file name is a file that contains one or more meta-analysis database structures, named *DB*.

In my notes, it says we need a file called 'meta_analysis_master_file.mat – what does this contain?

```
>> meta_analysis_table(mm,10,'meta_analysis_mater_file.mat')
```

This loads the master file and generates a list of data contrasts from studies in these databases where you also see activation at the same point (within 10 mm)

Other Important Points

If you want to get the Talairach coordinates, you need to make sure that a file called mni2tal.m is in the matlab path, and then create a variable with the MNI xyz coordinates from the excel spreadsheet into one field, and type

```
>> xyz=[
                xyz
                xyz
                xyz
                xyz
                xyz
                        ];
>> mni2tal(xyz)
```

Copy resulting matrix of Talairach coordinates, etc., into excel spreadsheet.

CHAPTER V: A FEW KEY MATLAB COMMANDS

Basic Notes About MatLab

Matlab has excellent documentation, but learning to use Matlab takes some time. Matlab is a high-level programming language. Type **helpdesk** for a comprehensive help menu.

- Variables in the active file are displayed in the workspace. The workspace is a set of data organized in variables that's in RAM.
- Write a set of variables to disk using the **save** command.
- Use unix commands within matlab, such as **ls** (list files) and **pwd** (print working directory). (Do these work on PCs? I don't know.)
- To print a list of variables in the workspace use the Whos command.
  matlab command: **whos**
- To print a list of unique values in a given variable use the Unique command.
  matlab command: **unique(variable of interest)**
  This command can also be used in conjunction with structure fields to access data in particular fields. An example is below.
  matlab command: **unique(DB.gender)**

- To create a new variable pick a new variable name and set it equal to the value of interest.
  matlab command: **out= unique(variable of interest)**
- To determine the length of a variable (i.e., the number of items in the variable), use the following command.
  matlab command: **length(variable of interest)**
- To determine the maximum number in a particular variable, use the following command.
  matlab command: **max(variable of interest)**
- To make a figure or plot of a particular variable, use the following command.
  matlab command: **plot(variable of interest)**

Debugging functions

- If you get errors you don't understand, you can use the debugger to see what's happened (this will require some basic experience with how Matlab works, though.)
- Try **help debug** for help on how it works
- To break within a function if there's an error, type **dbstop if error**. Then re-run the function. You'll get a K>> prompt when you get the error, which means you're stopped within the function at the point of the error. Then you can examine the variables in memory, etc.
- To turn off the debugger, type **dbclear all**

```
Warning: Cant get default Analyze orientation - assuming flipped
> In spm_flip_analyze_images at 9
  In spm_vol_ana at 105
  In spm_vol>subfunc at 93
  In spm_vol>subfunc1 at 62
  In spm_vol>subfunc2 at 51
  In spm_vol at 37
  In pmap_threshold at 39
```

This means that it could be flipping your images Left-Right, which could throw things off.

edit spm_defaults
and set defaults.analyze.flip to 0
save spm_defaults