Mediation Toolbox

The Multilevel Mediation and Moderation (M3) Toolbox is a joint project of Tor D. Wager, currently Assistant Professor of Psychology at Columbia University, and Martin A. Lindquist, currently Assistant Professor of Statistics at Columbia.

Dr. Wager heads the Cognitive and Affective Control Lab, a member of the multi-laboratory Social, Cognitive, and Affective Neuroscience Unit in the Department of Psychology at Columbia.

A) Installation and requirements

You will need to download two SCNlab packages, each with several subdirectories, and place them on your Matlab path. You will also need SPM installed and on your Matlab path. The most extensive testing of the current M3 toolbox version has been done with SPM5. Most or all of it should work with SPM99 or SPM2 as well.

The first SCNlab toolbox is the mediation_toolbox package. The second is the SCN_Core_Support package. Both are available on Tor's lab website.

Optional: If you want to take full advantage of the results visualization of the SCNlab tools, you will also need the 3DheadUtility folder (not on the web currently; ask the authors if interested).

To install these, place them in a folder on your hard drive, and type

```
pathtool
```

at the Matlab command prompt. Select each of the folders named above, and select “add with subfolders”

Use of SPM and other toolboxes

The robust regression toolbox is not an SPM toolbox per se. It uses SPM image manipulation (data I/O) functions, but does not rely on any SPM functions for statistics. It does, however, use the Matlab Statistics Toolbox, so you will need that as well.

There is an SPM5 GUI, which you can use to set up analyses. However, at this point, the GUI is “very” beta… actually, we all just run things from the command line and I've never used the GUI. I'd like to make the GUI work well, but it might take a bit of time. In the meantime, the command line functions are very easy to run.

Using the M3 toolbox to run analyses: A 3-variable analysis

NOTE: This is not intended as a comprehensive guide to mediation analysis, but as a way to provide at least a minimal description of the M3 toolbox functions. We hope that this will develop and expand over time into more comprehensive documentation, and we hope that you will contribute to this effort!

Performing a single mediation test on data from any source

mediation.m is a function that will run single- or multi-level mediation analyses on any kind of data. You first need to define variables in Matlab with the data in them. type » help mediation at the command line for a list of options.

You need to enter 3 variables:

- X, the initial variable
- Y, the outcome variable
- M, a potential mediator

You can also specify:

- Covariates, which are controlled for in all regressions in the mediation
- Additional mediators, which are controlled for only in M→Y regressions
- Moderators (but not yet! COMING SOON, we hope.)

Here are some examples using brain imaging data, though the tools are generic and can be applied to any kind of data.

Single-level example

For a single-level analysis, you might begin with data from a contrast of interest, so you have one image per subject. You might then extract average data for each subject from a brain region of interest (ROI). This becomes the X variable. If you have 20 subjects, you will have X = a [20 x 1] vector of average ROI scores.

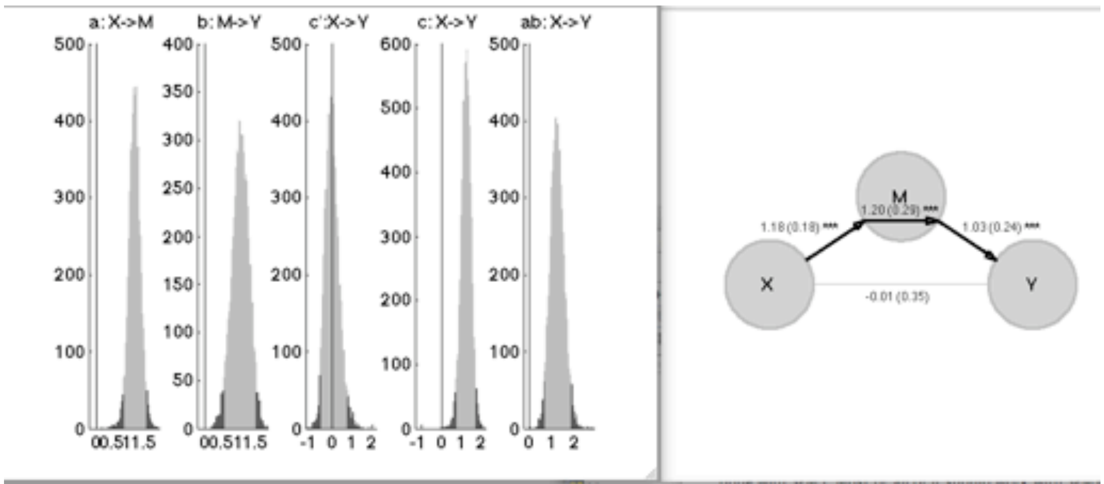
You might then specify a performance variable as the Y variable. It would be one number per subject representing the outcome of interest. So Y = a [20 x 1] vector of average performance scores.

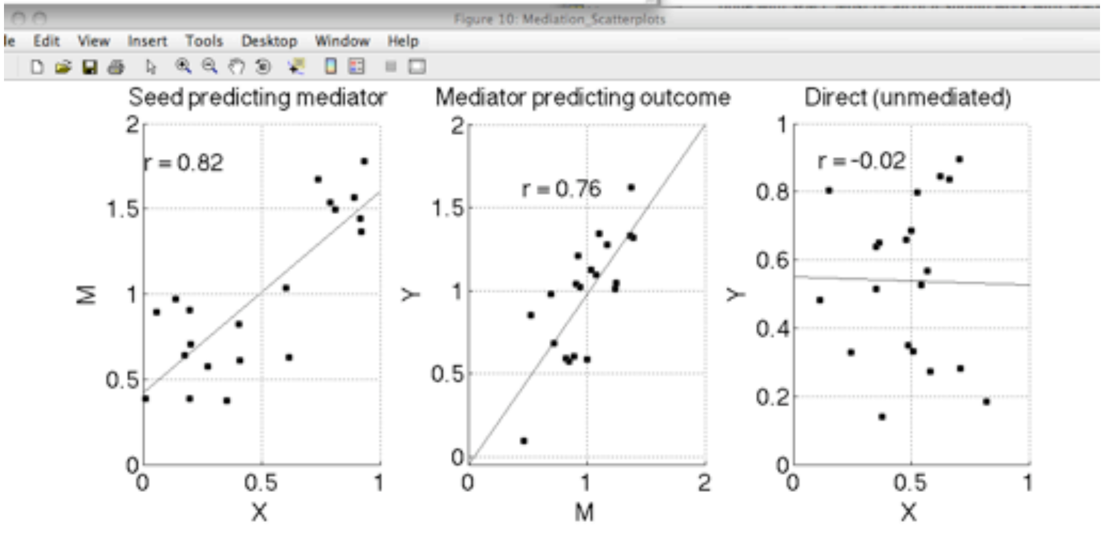
M might be data from another ROI that might mediate the relationship between X and Y. M = a [20 x 1] vector of average contrast values from ROI 2.

In the example below, we'll run mediation.m on simulated data with 20 "subjects," using a bootstrap test.

```
X = rand(20, 1); M = X + rand(20, 1); Y = M + rand(20, 1);
[paths, stats] = mediation(X, Y, M, 'plots', 'verbose', 'boot', 'bootsamples', 10000)
```

With plotting options on, the output I get to the screen looks like this:





The output I get in the command window shows me the stats for each relationship:

Mediation analysis

Observations: 20, Replications: 1 Predictor (X): X, Outcome (Y): Y: Mediator (M): M

Covariates: No

Single-level analysis. Options:

```
Plots: Yes
Bootstrap: Yes
Robust: No
Bootstrap or sign perm samples: 10000
```

Bootstrapping: Min p-value is 0.000100. Adding 0 samples Done in 2 (s) Done. Mediation scatterplots: Replications: 1 Covariates controlled for in all regressions: 0 predictors Additional mediators controlled for in outcome predictions: 1 predictors

Single-level model

```
a       b       c'      c       ab
```

```
Coeff 1.18 1.03 -0.01 1.19 1.20
STE 0.18 0.24 0.35 0.24 0.29
t (~N) 6.65 4.29 -0.04 4.88 4.11
Z 3.58 3.67 -0.11 3.52 3.71
p 0.0003 0.0002 0.9132 0.0004 0.0002


Total time: 3 s
```

The output is tab delimited so you can paste it into excel etc.

Multi-level example

For a multi-level analysis, let's stick with the same example. We have X = ROI1, Y = performance, M = ROI2 But now, consider the case where I want to analyze relationships across trials, within subjects. Now, X, Y, and M will be time-series data. Let's say you have 200 observations across time for each of 20 participants. You'll need to define X, Y, and M vectors for EACH SUBJECT that are each 200 x 1 elements long.

NOTE: if x is performance data from a series of trials or events, you'll need trial-by-trial data for X and Y as well. Another example where you'd be most likely to have this kind of data readily available from fMRI imaging is when you're trying to predict scan-to-scan timecourses of physiological responses (e.g., heart rate) across time with brain ROIs.

For a multi-level analysis, you enter data in cells. X{1} is X data for subject 1, M{1} is M data for subject 1, Y{1} is Y data for subject 1, X{2} s X data for subject 2, and so on. You should have X = a 1 x 20 cell array, where each cell is a [200 x 1] vector. Same for M and Y.

You can enter these variables into mediation.m in the same way as for the single-level model.

Running a whole-brain analysis

The idea of a whole-brain mediation analysis (or search through mask or ROI voxels) is to search for regions that show statistical mediation of the relationship between a known initial variable (X) and a known outcome variable (Y). You can search for voxels that satisfy the multiple constraints that make the strongest case for mediation, or search for maps of each effect in the 3-variable x-m-y mediation equation.

In a search, you specify data for 2 of the 3 variables (X, M, Y) and a set of images as the other variable. So, for a single-level analysis, X could be a 20 x 1 vector of data from the prefrontal cortex, Y could be a 20 x 1 vector of outcome scores, and M could be a string matrix of 20 image names (one subject's image per row, one image per subject).

For a multi-level analysis, let's say you have t = 200 observations on each of 20 subjects. X is 1 x 20 cell array, and each cell contains a 200 x 1 vector of observations on each subject. Ditto for Y. M is a 1 x 20 cell array, where each cell contains a string matrix of 200 image names for one subject (4-D files are OK as well; the key is to have 200 volumes for each subject).

The main functions are below. All of them write out images in the current directory and write a mediation_SETUP.mat file. There are many options for how to use them, which are documented in their individual help files.

mediation_brain - Search using a set of images for either X, M, or Y.

mediation_brain_multilevel – Search using a set of images for M, using a two-level model of observations nested within subjects**igls** – Two-level hierarchical regression with cariance component estimation using both restricted and normal Iterative Generalized Least Squares

NOTE: right now, you have to enter X, M, and Y data or images. There is not yet facility for handling interposed design matrices. If you want to work with “betas”, you're best off running a first-level analysis on each subject and then doing a single-level mediation on betas or contrast images, one per subject. You might want to use the multi-level search option if you: a) Have a series of trial-by-trial response estimates, or b) are using timeseries data and can specify the timeseries observations for X, M, and Y. An example of where you DON'T satisfy (b) is if you have 100 trials and 500 images per subject, and you're interested in a performance measure assessed on each trial (t = 100), but you have 500 fMRI time points. Getting a trial-by-trial estimate would mean estimating responses for each of the 100 trials. An example of where you DO satisfy (b) is if you're looking for brain-physiology relationships both sampled at the same time resolution.

Thresholding and visualizing results

Once you've estimated a model with one of the functions above, the next step is to visualize the results. The main gateway function for this is:

mediation_brain_results

Using this, you can display results from mediation_brain and other SCNlab toolboxes on brain “orthoviews”, extract data and save clusters (blobs) with useful info for visualization and ROI analysis

mediation_brain_results will display orthviews with one or more effects in separate viewer panes. There are many options. For example, if you enter 'all', it will show you the 'a' (X→M), 'b' (M→Y), 'c' (X→Y), 'c-prime' (direct) and 'ab' (mediation) effects from a mediation search analysis.

It also returns clusters, a structure with a special format that is widely used in our lab. Clusters are a series of structures in a vector, each element of which contains information about a contiguous blob. By default (if output variables are requested and it can find valid images), mediation_brain_results returns clusters for both positive and negative effects for the *last* image in the series of effects shown with image data extracted for each region.

Once you have a clusters structure, you can do lots of things with it:

1. Extract data from any images you want and do ROI analysis: see extracting_timeseries_data
2. Plot blobs on orthviews, with many options for control
3. Make a montage with montage_clusters
4. Make a table of coordinates, etc. with cluster_table or mediation_brain_print_tables
5. Make a surface image plot with cluster_surf
6. Plot on flexible, custom surfaces by combining use of addbrain and cluster_surf

Here are some links to other pages on our WIKI that cover results visualization with mediation_brain_results and other tools:

visualizing_results

igls_getting_results

interactive_data_viz

Index of functions

Top-level functions many users will want to run from the command line

mediation - Examines 3 timeseries to determine if one of them acts as a mediator between the other two. Works for both single-level and multi-level (multiple subjects/observations) data

mediation_brain - Given two variables of the mediation formula, searches over functional MRI data for candidates for the third variable

igls - Two-level hierarchical regression with cariance component estimation using both restricted and normal Iterative Generalized Least Squares
igls_mediation - Use (R)IGLS to compute mediation paths NOTE: NEEDS TO BE UPDATED BEFORE READY FOR USE

mediation_brain_results - Display results from mediation_brain and other SCNlab toolboxes on brain “orthoviews”, extract data and save clusters (blobs) with useful info for visualization and ROI analysis

Plotting and output support functions

These are called within the main top-level functions when plotting options are turned on. Many of them can be called from the command line with output from top-level functions as input, so you can display plots for analyses you've already run.

igls_plot_slopes - Display (R)IGLS results

mediation_brain_print_tables - Print out info on clusters found by a mediation search across the brain

mediation_brain_results_detail - Display detailed results for a particular cluster found by mediation_brain

mediation_brain_surface_figs - Display visually clusters found by mediation_brain on a 3-D rendering of the brain

mediation_path_diagram - Displays a traditional mediation plot of the path results

mediation_scatterplots - Display scatterplots of mediation output

mediation_sim_output_figs - Display simulation results

Computation support functions: High-level

These are functions that are called within top-level functions that provide computational support. Like all toolbox functions, they can also be used as stand-alone tools from the command line. These, however, are ones that I think are likely to be useful to run as stand-alone tools.

matrix_direct_effects - Given a set of pairs of correlated data, examines all possible mediators amongst the set.

mediation_X_search, mediation_M_search, mediation_Y_search, mediation_search - Holds two variables (X, Y, or M) of the mediation formula constant while comparing a set of possibilities for the third to see how well it fits

mediation_region_stepwise - Examines a set of ROIs to see if they are mediators

Computation support functions: Low-level

These are functions that are called within top-level functions that provide computational support. It's unlikely that you would want to run these on your own unless you're programming your own tools.

bootbca_ci - Computes confidence interval from corrected and accelerated percentile bootstrap

bootbca_pval - Computes p-values from corrected and accelerated percentile bootstrap

mediation_X_search_mask - Generates a mask based on mediation path p-vals less than .05.

mediation_latent - compute mediation on data after deconvolving the hemodynamic response function, to compare purported neural activity rather than BOLD activity

mediation_latent_sse - computes OLD mediation on HRF-deconvolved data and returns sums of square errors (SSE)

mediation_path_coefficients - Returns single-level mediation path coefficients and standard errors using subfunctions optimized for efficiency

mediation_results_p2z - Backend function for converting images from p-values to z-scores

mediation_shift - Compute mediation paths on X, Y, and M while shifting them back and forth in time for better fits

mediation_shift_sse - Compute mediation paths using OLS only on X, Y, and M while shifting them back and forth in time for better fits and returning sums of squared errors (SSE)

optimal_delay_mediation_search - Compute optimal shift for mediation on a given set of timeseries

Simulation and performance evaluation scripts

These are sort of haphazard at the moment. We haven't yet tried to provide a comprehensive set of tools for testing these functions within this package itself.

mediation_dcm_sim1 - Runs a DCM simulation designed to compare DCM as much as possible with mediation
mediation_power - Tor's simulation to determine mediation power; not well documented or developed
mediation_problem1 - script to illustrate possible conceptual flaws in mediation
mediation_sim1 - Simulation for power and false positive rates for mediation analysis
mediation_sim2_igls - Simulation for power and false positive rates for mediation analysis using (R)IGLS

Other functions of ambiguous status in the current toolbox

mediation_results - Threshold mediation_search results and write out thresholded images

mediation_toolbox.txt · Last modified: 2008-11-06 02:39 by 98.14.165.6

Visualizing results of fMRI statistical models

These functions can help you get and visualize results easily: mediation_brain_results.m (works for robust reg directories, too; my favorite) robust_results_threshold.m robust_results3.m

mediation_brain_results.m

Works with mediation directories, robust regression directories, and IGLS directories Go to an analysis directory and run the function from within that directory. Features:

- Multiple thresholds in different colors
- Multiple orthviews windows to view different effects

(e.g., diff. covariates) at the same time

- Positive and negative effects (activations/deactivations) on the same slices

Enter separate keywords for different effects:

- 'all', 'a', 'b', etc. for mediation
- 'rob', 'rob0', etc. for robust regression
- 'igls', etc. for igls

See the help (type: help mediation_brain_results) for more details.

robust_results_threshold

Go to an analysis directory and run the function from within that directory. Features:

- Orthviews and montage slice plots of a single effect
- Flexibly specify any input image(s), so you can use this with any type of results t or p map
- Positive and negative effects (activations/deactivations) on the same slices

See also iimg_threshold and threshold_imgs for basic t/p/etc. image thresholding functions.

robust_results3

Go to an analysis directory and run the function from within that directory. Features:

- Montage slice plots (both axial and saggital) of robust regression images
- Positive and negative effects (activations/deactivations) on the same slices

Example: Using mediation_brain_results to get robust regression maps

This example uses the CORT study SEAT task. We have one covariate: Group status, so the first image we'll see (top) will show us the overall effect across subjects, controlling for Group, and the second image (bottom) will show us results for the Group difference effect.

In the robust regression output, rob_t_0001.img and rob_p_0001.img correspond to the INTERCEPT, which is the overall effect across all subjects, controlling for any covariates (i.e., Group status in this example.) This is only true if the covariates are mean-centered! Covariates are automatically mean-centered in robfit. One exception, however, is if you enter CONTRAST CODES with 1 and -1 values only, indicating a group assignment (i.e., two-sample t-test case), then the covariate is NOT centered. If you centered it, you get the equiv of Type II sums of squares, which is less appropriate for a group difference, and if you don't center you get Type III SS. If the groups have equal N's, they're identical, but if they don't Type III is usually what you want, and is usually more conservative.

We'll look at the contrast PLACE - FACE First, get results at p < .001 and 5 voxels, and show also contiguous regions down to p < .01

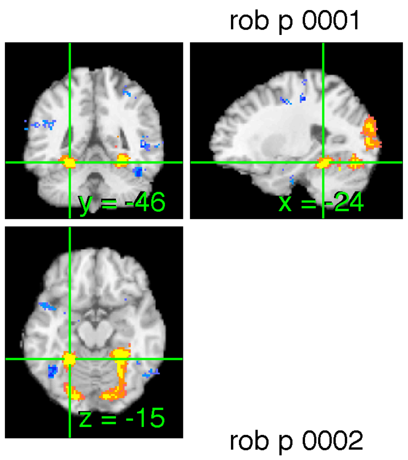
```
mediation_brain_results('rob', 'thresh', [.001 .005 .01], 'size', [5 1 1], 'prune');
```

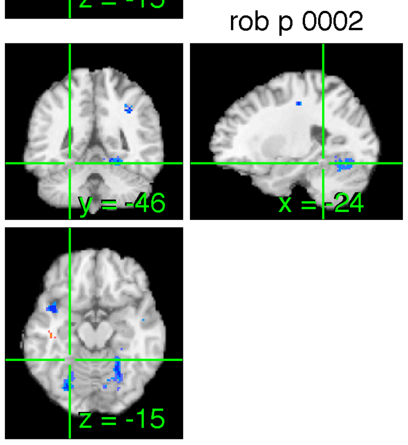
Now some maps come up. For PLACE - FACE, the parahippocampal place area should show positive effects (yellow/orange), and the fusiform face area should show negative effects (blues)

Saving images

We find nice activity, so we want to save an image. I'm going to click on the PPA and then run the following: saveas(gcf,'orth_PPA1', 'png');

Here's what was saved:





Looking at the covariate maps

Now we'll look at some group differences. We have a covariate with 1 for Stress Group, and -1 for Control group. That means in the bottom (covt) image, Stress > Control should appear in yellow, and Control > Stress should appear in blue

We see activity in PPA that's blue, i.e., Control > Stress Putting the pieces together, PPA shows Place > Face, but less Place > Face for the Stress group.

Getting clusters: Gateway to more stuff

I can request "clusters" outputs from mediation_brain_results that lets me do lots of stuff. For more information on clusters: clusters_structure

```
[clpos, clneg, clpos_data, clneg_data] = mediation_brain_results('rob', 'thresh', [.001 .005 .01], 'size', [5 1 1], 'prune');
```

clpos and clpos_data contain blobs with positive effects, and clneg / clneg_data have negative effects.

clpos (and clneg) contains a cell array with one cell per threshold. each cell contains a clusters structure with one element per contiguous "blob" The "blobs" are defined exclusively for easy visualization purposes. The blobs significant at the highest threshold (cell {1}) are excluded from the blobs significant at lower thresholds (cells {2}, {3}, etc.)

In this example, I get: clpos =

```
   [1x6 struct]    [1x14 struct]    [1x33 struct]
```

clneg

clneg =

```
   [1x20 struct]    [1x58 struct]    [1x96 struct]
```

This means that there are 6 distinct significant POSITIVE blobs at .001, 14 at .005, and 33 at .01.

Since mediation_brain_results always returns clusters (blobs) for the LAST image in a set – i.e., the image that appears at the bottom right position in the orthviews display–these blobs apply to the COVARIATE 1 (Group) effect in this example.

If I want blobs for the INTERCEPT (Overall average effect), I can run mediation mediation_brain_results only the intercept. In the line below, 'rob0' specifies that we're looking for the intercept results only. [clpos, clneg, clpos_data, clneg_data] = mediation_brain_results('rob0', 'thresh', [.001 .005 .01], 'size', [5 1 1], 'prune'); If we entered 'rob1', we'd get covariate 1 results only. For more help and a full list of options, see » help mediation_brain_results

What's even more useful is clpos_data and clneg_data, which contain clusters structures for contiguous blobs across all thresholds.

clpos_data

clpos_data =

1×6 struct array with fields:

```
  title
  threshold
  Z_descrip
  voxSize
  M
  name
...etc
```

Here, there are 6 contiguous blobs, where each blob is distinct even at the lowest statistical threshold (p < .01 in this example).

Saving clusters for later

It's a good idea to save your clusters to disk so they can be used later. They also contain extracted data for each blob (see below) … another reason to save them. I like to use a descriptive name that capures all the relevant choices: Which effect it is (rob1, covariate), thresholds, extent thresholds, and pruning. If you use an external mask (see help mediation_brain_results), you might include that in the name as well.

save cl_rob1_001_005_01_k5_1_1_prune clpos clneg clpos_data clneg_data

Making montages of slices

Now we can create montages of all the slices showing significant results.

One way is to use the graphics from the orthviews you've already created, and make a montage of slices showing the center of each blob. This is done with cluster_orthviews_showcenters. See its help file for options (type help cluster_orthviews_showcenters)

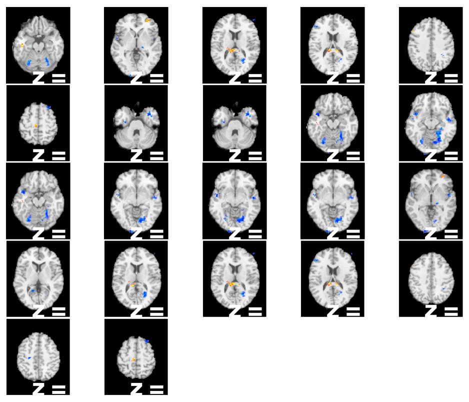
slices_fig_h = cluster_orthviews_showcenters([clpos_data clneg_data], 'axial', [], 0);

Now I'll save this. There are many options for manipulating graphics in matlab to make things look nicer. Here are a couple of things:

```
scn_export_papersetup(1000); % format paper size; higher values make text smaller relative to graphics
h = findobj(gcf, 'Type', 'text'); delete(h)
saveas(gcf,'montage_group_axial', 'png');
```

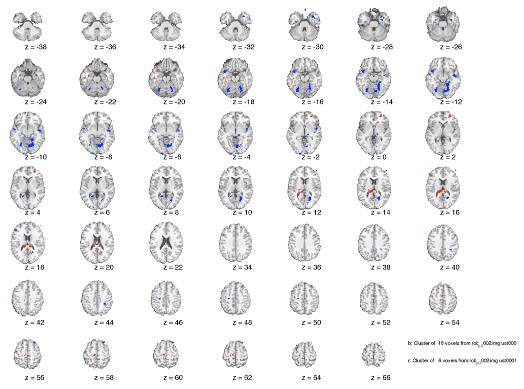
NOTE: BUG: if you close these montage figure windows it can mess up SPM, requiring you to restart/clear variables.

Here's what it produced.



Here's an alternate function that will produce montages from any set of clusters:

```
montage_clusters([], clpos_data, clneg_data, {'r' 'b'});
```



Scatterplots of results in individual regions

Each element of a clusters structure is a separate blob. Data averaged across voxels for each subject is saved in clusters(x).timeseries. Timeseries is a misnomer here (sorry), because it really stores average contrast data for each subject.

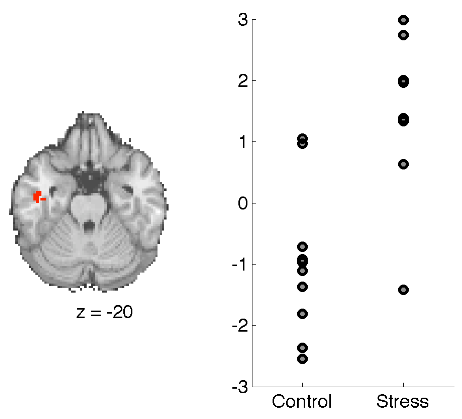
In our example, the contrast was PLACE - FACE, so clpos_data(3).timeseries, for example, contains average Place - Face data for each subject, averaged over voxels.

The code below shows us the blob on the left side of a figure, and a scatterplot of Group vs. Brain on the right side. Plots like those below can be very study-specific, so it makes sense to learn how to do a bit of coding to create exactly what you want.

```
create_figure('Scatterplot', 1, 2);
montage_clusters_maxslice([], clpos_data(1), {'r'});
load SETUP
x = SETUP.covariates
y = clpos_data(1).timeseries;

subplot(1, 2, 2);
plot(x, y, 'ko', 'MarkerSize', 8, 'MarkerFaceColor', [.5 .5 .5], 'LineWidth', 3);
set(gca, 'XLim', [-2 2], 'XTick', [-1 1], 'XTickLabels', {'Control' 'Stress'})
```

Here's the result:



Naming clusters: Which elements are which regions?

The function cluster_names.m is a convenient way of naming clusters. Names are assigned in the field clusters.shorttitle, which is used automatically by some other functions (e.g., cluster_table.m)

```
clneg_data = cluster_names(clneg_data, 1);
```

Enter short name for this cluster: L TC Enter short name for this cluster: R TC Enter short name for this cluster: L PPA Enter short name for this cluster: R PPA Enter short name for this cluster: LTC2 Enter short name for this cluster: LOC Enter short name for this cluster: RTC2 Enter short name for this cluster: LTC3 Enter short name for this cluster: R_Thalamus? Enter short name for this cluster: LOC2 Enter short name for this cluster: ROC Enter short name for this cluster: out Enter short name for this cluster: LFC Enter short name for this cluster: RParietal Enter short name for this cluster: L_PostCentral? Enter short name for this cluster: RFrontal

Now that we've entered all those names, we should save our hard work:

```
save cl_rob1_001_005_01_k5_1_1_prune clpos clneg clpos_data clneg_data
```

I can see the names using: char(clneg_data.shorttitle)

Making a table of results

Now that we have named clusters, we can make a table. You don't actually need names, but it helps to create a useful table!

See help cluster_table for more options. The command below produces a row with a cluster name for each contigous blob. Arrows beneath each name indicate “sub-peaks” or local maxima as defined using your version of SPM's algorithm. The tables are tab delimited, so they look good in Excel (for example).

Here's the command to run for PLACE > FACE

```
cluster_table(clneg_data, 1, 0);
```

Here's the output for PLACE > FACE

```
Name    index   x       y       z       corr    voxels  volume_mm3      maxstat numpeaks        snr_avgts(d)    minsnr  maxsnr  numpos  power80
L TC      1     -32     -14     -30     NaN      27     -216    9.53      3     -0.66   -0.81   -0.35      5      39
R TC      2      38       6     -30     NaN      64     -512    14.96     5     -0.52   -0.78   0.14       8      60
->       36       8     -30     NaN      39     13.14442                -0.52   -0.78   0.14     8      60
->       44       4     -30     NaN      16     14.96046                -0.52   -0.78   0.14     8      60
->       30       2     -26     NaN       9     6.98315         -0.52   -0.78   0.14     8      60
L PPA     3     -26     -68     -16     NaN     228     -1824   10.59    12     0.76    0.28    1.13      14      29
->      -26     -68     -18     NaN     119     10.59174                0.76    0.28    1.13      14     29
->      -28     -72     -12     NaN      75     9.98357         0.76    0.28    1.13     14      29
->      -24     -62     -12     NaN      34     6.02796         0.76    0.28    1.13     14      29
R PPA     4      20     -64     -12     NaN     638     -5104   14.70    25     0.70    -0.64   1.30      11      34
->       24     -60     -18     NaN      92     9.34418         0.70    -0.64   1.30     11      34
->       24     -48     -14     NaN     127     10.14246                0.70    -0.64   1.30      11     34
->       12     -72      -8     NaN     193     13.15964                0.70    -0.64   1.30      11     34
->       22     -66     -10     NaN     150     14.69512                0.70    -0.64   1.30      11     34
->       28     -72     -18     NaN      35     9.05430         0.70    -0.64   1.30     11      34
->       14     -44     -12     NaN      19     7.26998         0.70    -0.64   1.30     11      34
->       28     -34     -12     NaN      22     7.13968         0.70    -0.64   1.30     11      34
LTC2      5     -42       8     -16     NaN      86     -688    13.41     5     -0.62   -0.96   -0.07      7      43
->      -42       8     -16     NaN      80     13.41234                -0.62   -0.96   -0.07      7     43
->      -38      12     -12     NaN       6     5.70712         -0.62   -0.96   -0.07     7      43
LOC       6     -16     -104     -6     NaN      94     -752    10.94    11     0.57    0.26    1.10      11      50
->      -16     -104      -8    NaN      65     10.94395                0.57    0.26    1.10      11     50
->      -14     -102      2      NaN      29     8.87523         0.57    0.26    1.10     11      50
RTC2      7      52      -6      -8      NaN     133     -1064   13.45     8      -0.54   -0.74   0.05      6      56
->       52      -8     -10      NaN      99     9.96481         -0.54   -0.74   0.05      6      56
->       50       0      -4      NaN      34     13.45201                -0.54   -0.74   0.05      6      56
LTC3      8     -54       0      -6      NaN      73     -584    9.60      7      -0.40   -0.53   -0.01     9     100
->      -52      -2      -8      NaN      42     8.21884         -0.40   -0.53   -0.01     9     100
->      -54       2       0      NaN      31     9.59626         -0.40   -0.53   -0.01     9     100
R_Thalamus?       9      18     -26      -2      NaN      20      -160    11.70     3      -0.27   -0.40   0.02      8     222
LOC2     10     -16     -52       6      NaN      42     -336    7.80      1      0.38    -0.03   0.85     12     111
ROC      11      24     -62      12      NaN     138     -1104   13.31     8      0.47    -0.51   1.10     12      73
->       24     -68      10      NaN      41     13.30817                0.47    -0.51   1.10     12      73
->       24     -58      14      NaN      80     9.40835         0.47    -0.51   1.10     12      73
->       26     -64      10      NaN      17     6.99441         0.47    -0.51   1.10     12      73
out      12      54      56      14      NaN      15     -120    9.49      3      0.18    -0.13   0.44     10     489
LFC      13     -48      36      18      NaN      38     -304    10.94     5      0.01    -0.24   0.83     10     473668
->      -46      36      18      NaN      22     10.94372                0.01    -0.24   0.83     10     473668
->      -54      36      18      NaN      16     5.91076         0.01    -0.24   0.83     10     473668
RParietal        14      36     -46      42      NaN      27      -216    14.25     5      -0.28   -0.45   -0.07     7     200
L_PostCentral?   15     -24     -20      48      NaN      25      -200    9.06      4      -0.40   -0.75   0.10      8     100
RFrontal         16      30      28      58      NaN      71      -568    12.90     2      0.03    -0.29   0.30     10     18925
-------------

Turn the local peak reporting off if you want:
>> cluster_table(clneg_data, 0, 0);

Cluster of  16 voxels from rob_p_0002.img_1_27_voxels
Z field contains: Log(1/p) (shown in maxstat)

Name    index   x       y       z       corr    voxels  volume_mm3      maxstat numpeaks        snr_avgts(d)    minsnr  maxsnr  numpos  power80
L TC      1     -32     -14     -30     NaN      27     -216    9.53      3     -0.66   -0.81   -0.35     5      39
R TC      2      38       6     -30     NaN      64     -512    14.96     5     -0.52   -0.78   0.14      8      60
L PPA     3     -26     -68     -16     NaN     228     -1824   10.59    12     0.76    0.28    1.13     14      29
R PPA     4      20     -64     -12     NaN     638     -5104   14.70    25     0.70    -0.64   1.30     11      34
LTC2      5     -42       8     -16     NaN      86     -688    13.41     5     -0.62   -0.96   -0.07     7      43
LOC       6     -16    -104      -6     NaN      94     -752    10.94    11     0.57    0.26    1.10     11      50
RTC2      7      52      -6      -8     NaN     133     -1064   13.45     8     -0.54   -0.74   0.05      6      56
LTC3      8     -54       0      -6     NaN      73     -584    9.60      7     -0.40   -0.53   -0.01     9     100
R_Thalamus?       9      18     -26      -2     NaN      20      -160    11.70     3      -0.27   -0.40   0.02      8     222
LOC2     10     -16     -52       6     NaN      42     -336    7.80      1     0.38    -0.03   0.85     12     111
ROC      11      24     -62      12     NaN     138     -1104   13.31     8     0.47    -0.51   1.10     12      73
out      12      54      56      14     NaN      15     -120    9.49      3     0.18    -0.13   0.44     10     489
LFC      13     -48      36      18     NaN      38     -304    10.94     5     0.01    -0.24   0.83     10     473668
RParietal        14      36     -46      42      NaN      27      -216    14.25     5      -0.28   -0.45   -0.07     7     200
L_PostCentral?   15     -24     -20      48      NaN      25      -200    9.06      4      -0.40   -0.75   0.10      8     100
RFrontal         16      30      28      58      NaN      71      -568    12.90     2      0.03    -0.29   0.30     10     18925
```

Using clusters for visualization

Mostly all the visualization tools use the cluster format. All these functions allow you to see your clusters in solid colors, transparent colors, or heatmaps.

- *Q for Tor*: How is the heatmap calculated?

cluster_orthviews

- Check out the help. It's pretty useful.

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

- You can place these on your own images or the canonical brain (spm2 scalped 152 T1). If you don't specify your own image, AND don't have the canonical brain on your path, it won't work.
  - To put in your own overlay, use the flag 'overlay', followed by yourimagename.img

```
    which('spm2_single_subj_T1_scalped.img')
    cluster_orthviews(cl, {[1 0 0]}, 'solid','overlay', 'yourimagename.img')
```

- You can make clusters appear on a new spm_orthviews window, or use the 'add' flag to put it on the current image.

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

- You can use 'handle', 2 to put it on one of the images in the orthviews if you're displaying multiple windows.

```
    cluster_orthviews(cl, {[1 0 0]}, 'solid', 'add', 'handle', 2)
```

- You can specify specific colors {[1 0 0]} or have each blob be a unique color, using 'unique'.

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

Making region-specific visualizations

- Tor started with some atlases and made them match the spm2 anatomy and you can use these to define clusters iwthin a region.

```
    amy = mask2clusters('spm2_amy.img');
```

- To see amygdala clusters in transparent green

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

- You can make more masks of ROIs and add them to wherever the amygdala mask is.

montage_clusters

- You can put in multiple clusters, and you can decide whether to see the montage on your mean image or the canonical brain (replace the empty brackets with your own overlay).

```
     montage_clusters([], amy, {[0 1 0]});
```

- There's also montage_clusters_medial, which does the same thing for the medial surface of the brain.

cluster_surf

```
     create_figure('Surface');
     cluster_surf(cl, 5, 'right');
     cluster_surf(cl, 5, {[1 0 0]}, 'right');
```

- Check out help. There are lots of different structures on top of which to visualize your activations (basal ganglia, hires left.
- Use create_figure first.
- Define how many mm from surface to plot.
- Colors can be defined in different ways... See help.
- Internal matlab functions like 'lighting'

```
     lighting gouraud
     lighting phong
     camlight right
```

- Handles are used here too; *addbrain* has a whole bunch of stuff so that you can add transparent structures to the basic cortical surface.

```
     han = addbrain('brainstem');
     set(han, 'FaceAlpha',1);
```

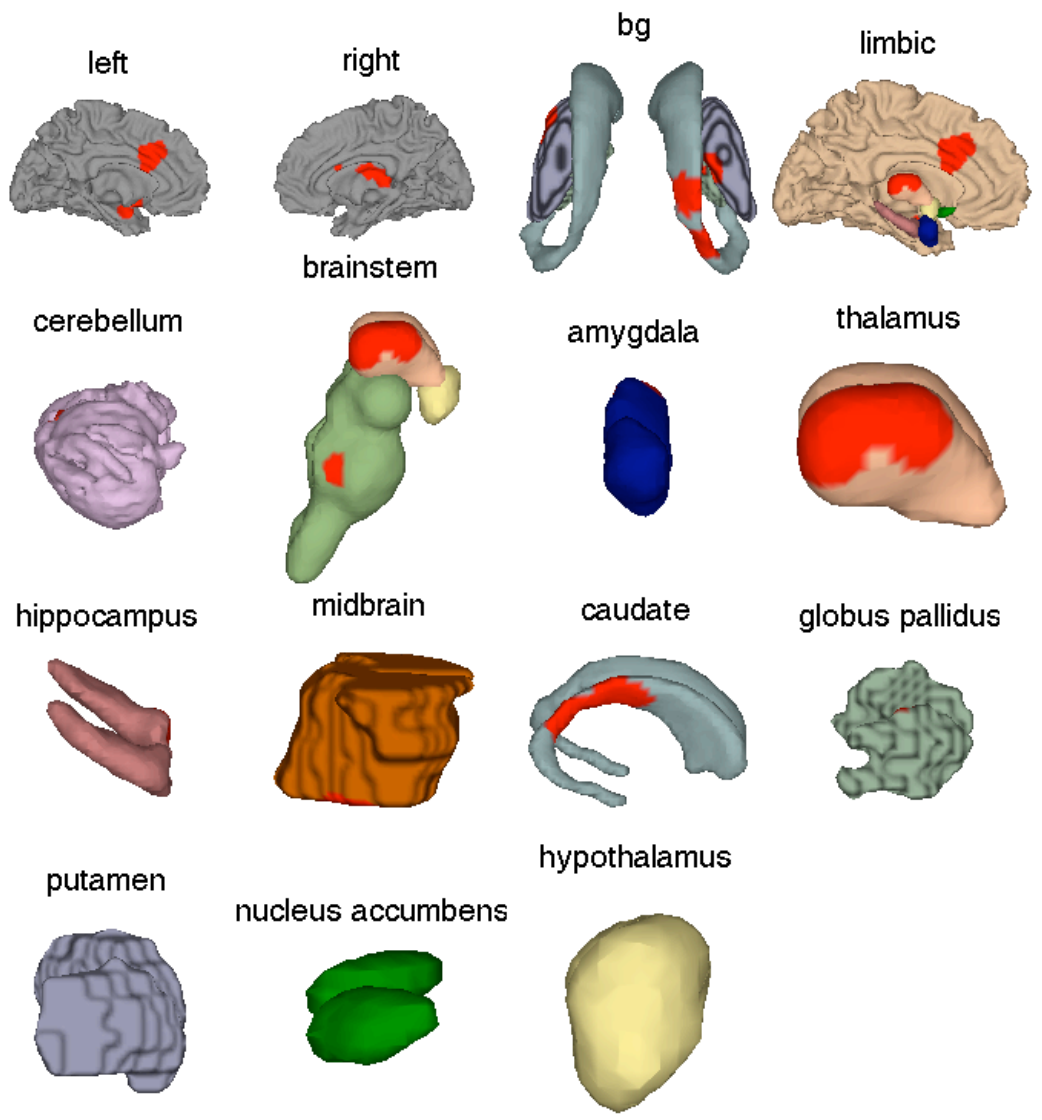
- You can plot stuff on the additional surface too

```
     cluster_surf(cl, 5, {[1 0 0]}, han);
     han = addbrain('bg');
     han = addbrain('caudate');
     set(han, 'FaceAlpha', 1);
```

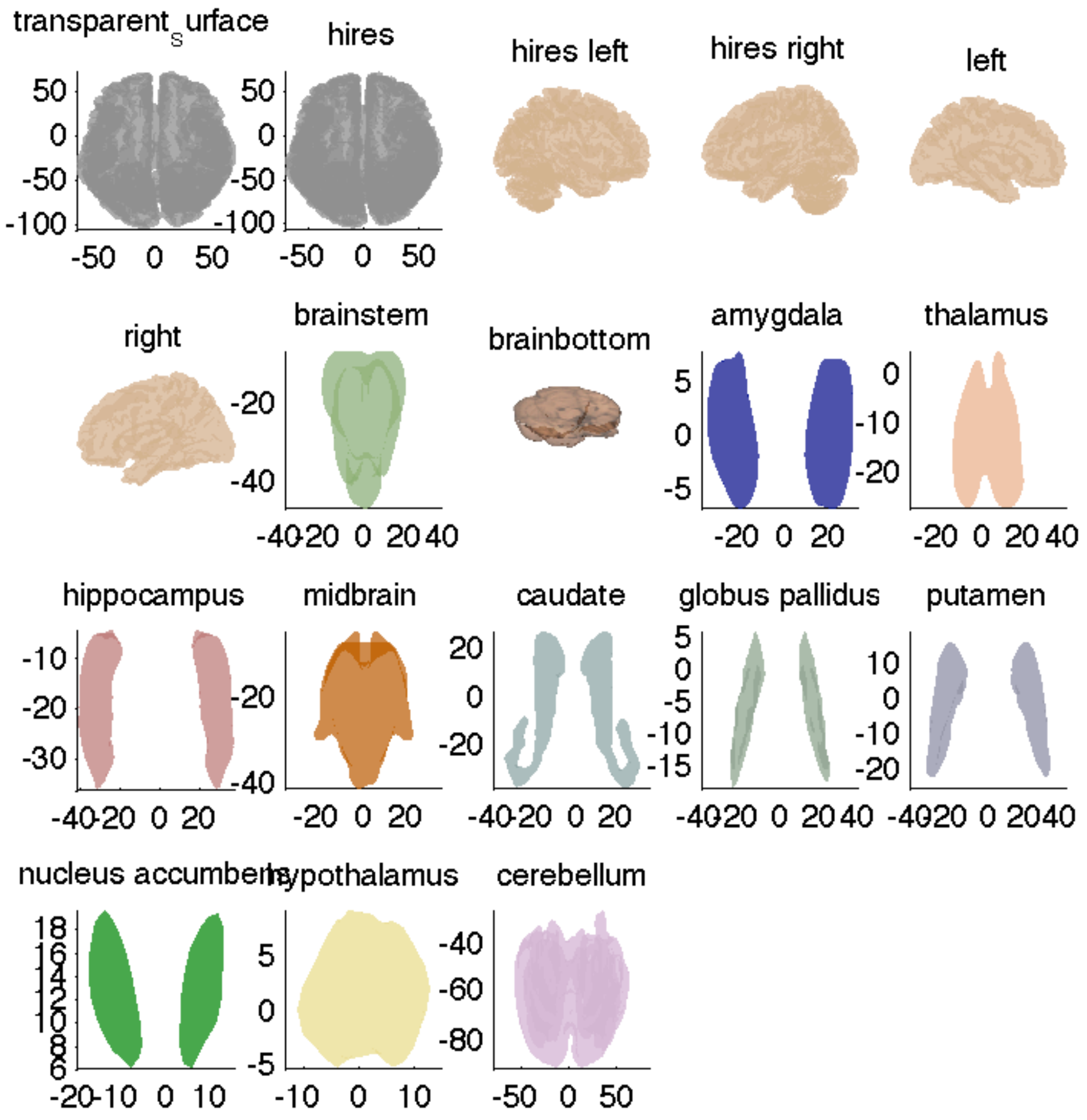
- Another way to plot clusters on surfaces is to use *imageCluster*. These blobs will be more 3-dimensional than the cluster_surf plotting.

```
     create_figure('Brain Surface')
     han = addbrain('brainstem')
     set(han, 'FaceAlpha',.3);
     [out,cl] = imageCluster('cluster', wh_pain2([1:2 4:5]), 'color', [1 1 0]);
```

- Here is an example of what each surface keyword looks like (the red stuff is overlaid blobs):



- These are the addbrain options:



spinn_scn_core_tools_/viz.txt · Last modified: 2008-11-06 02:48 by 98.14.165.6