

Using the Volume Rendering Plugin

- Accessed via [Define Datamode](#) → [Plugins](#) → [Render \[new\]](#)

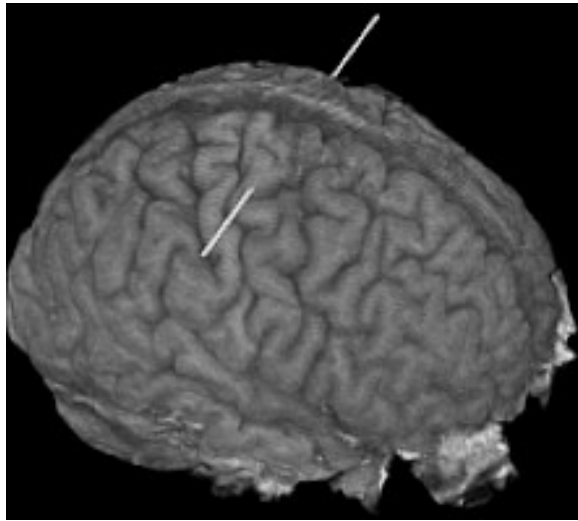
The screenshot shows the AFNI Renderer [A] window with the following controls and annotations:

- Pick new underlay dataset:** Points to the "Choose Underlay Dataset" button.
- Name of underlay dataset:** Points to the "Brick" text field.
- Sub-brick to display:** Points to the "Bot" and "Top" spinners.
- Open color overlay controls:** Points to the "Overlay" button.
- Range of values in underlay:** Points to the "Min=0 Max=255" text.
- Range of values to render:** Points to the "Bot" and "Top" spinners.
- Change mapping from values in dataset to brightness in image:** Points to the "Brightness" graph.
- Mapping from values to opacity:** Points to the "Opacity" graph.
- Histogram of values in underlay dataset:** Points to the "Sqrt Histogram" plot.
- Maximum voxel opacity:** Points to the "Opacity Factor" spinner.
- Menu to control scripting (control rendering from a file):** Points to the "Scripts" button.
- Cutout parts of the 3D volume Compute many images in a row:** Points to the "Cutouts" and "OR" buttons.
- Render new image immediately when a control is changed:** Points to the "Compute" button.
- Accumulate a history of rendered images (can later save to an animation):** Points to the "Accumulate" checkbox.
- Show 2D crosshairs:** Points to the "See Xhairs" checkbox.
- Control viewing angles:** Points to the "Roll", "Pitch", and "Yaw" spinners.
- Detailed instructions:** Points to the "Help" button.
- Force a new image to be rendered:** Points to the "Draw" button.
- Reload values from the dataset:** Points to the "Reload" button.
- Close all rendering windows:** Points to the "done" button.

- Volume rendering concepts:

- ◇ Goal is to create a 2D image consisting of pixels

- ◇ Each 2D pixel is obtained from data looking down line of sight into 3D volume:



If we looked directly from the subject's right to left, all the data along the white line would contribute to one image pixel

- ◇ Each 3D voxel contains one numerical value

- ◇ Voxel value determine the brightness (or color) of that voxel—if it is visible

- ◇ Voxel value determines the opacity of that voxel:

- ↳ Opacity = 0 \Rightarrow Transparent (brightness does not contribute to image)

- ↳ Opacity = 1 \Rightarrow Opaque (nothing behind it along the line will be seen)

- ↳ Intermediate values are translucent:

- ↳ Opacity = 0.5 \Rightarrow 50% of voxel brightness is added to pixel; voxels farther down the line will contribute to other 50% of pixel result

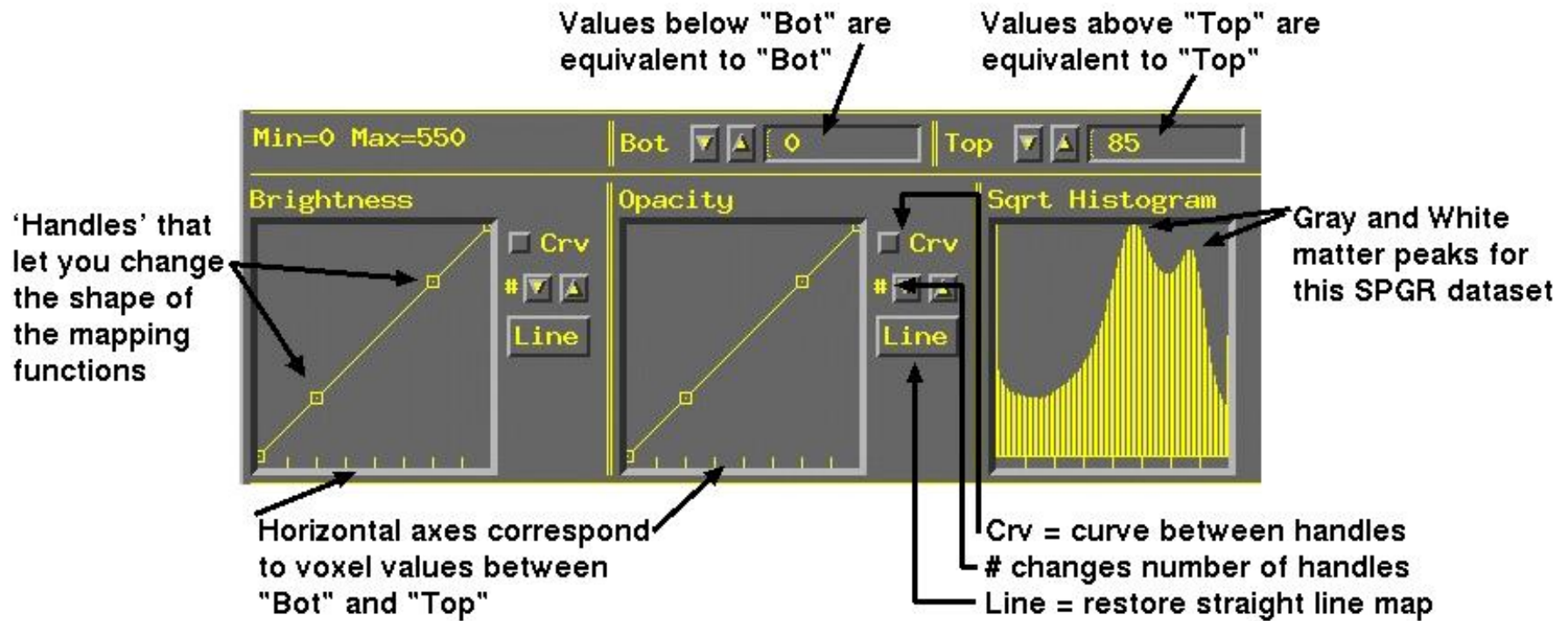
- ◇ 3D viewing angles:
 - Roll = angle about I-S axis
 - Pitch = angle about L-R axis (after roll rotation)
 - Yaw = angle about A-P axis (after roll and pitch)
- ◇ Rendering is CPU and memory intensive—a fast computer is very desirable
- Utility program [3dIntracranial](#) can be used to strip the scalp off a T1-weighted anatomical volume. In some cases, this may need to be done with the orig dataset, which may then be written out in Talairach coordinates:

```
3dIntracranial -anat anat+tlrc -prefix airstrip
```

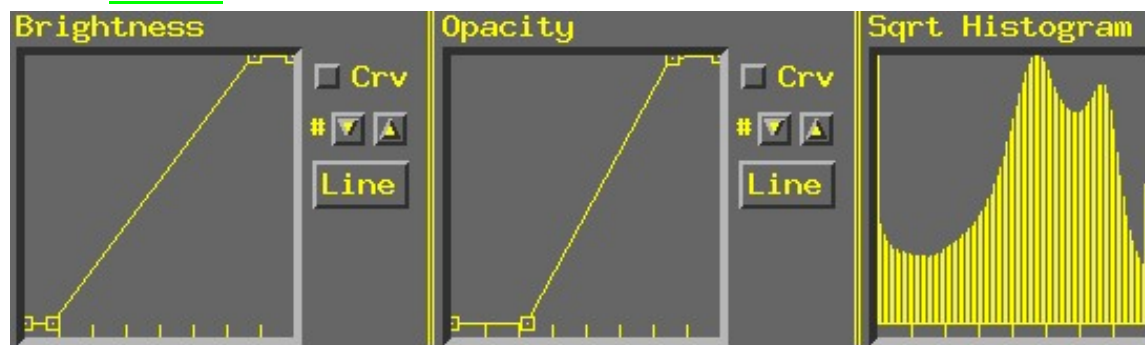
(using the datasets from the Talairach transform lecture)
- AFNI can now render datasets that are stored with an arbitrary orientation and voxel size. However, the orientation of cuts still (incorrectly) assumes the axial slice order.
 - ◇ So making a cut "Anterior to X" might actually result in a cut "Left of X"
 - ↪ A fix for this is coming soon to an AFNI distribution near you
 - ◇ Note that axial slice order is the standard for 'warped' datasets written out to disk in +acpc or +tlrc coordinates

- In Talairach View, open the rendering plugin, and choose astrip as the underlay dataset
 - ◇ Plugin will load the voxel values, build the histogram, and then be ready to render
 - ◇ Press Draw to make your first image
 - ◇ Press Accumulate, then DynaDraw, then Roll ▼ a few times
 - ↪ Will generate renderings from different view angles (lines of sight)
 - ▷ If DynaDraw is off, then you must press Draw to get a new rendering
 - ↪ Accumulate on ⇒ rendered images are saved, and can be reviewed by using the image viewer slider
 - ▷ This slider does not move you through slices, as it does in the 2D image viewing windows
 - ▷ It just moves you backward and forward in the history of saved rendered images
 - ▷ If you turn Accumulate off, then creating the next rendered image will erase the history
 - ▷ By default, the plugin's controls ('widgets') do not change as you move around in the rendering history
 - ▷ Selecting Script->Load widgets will make the widgets display the settings they had when the currently display image was rendered

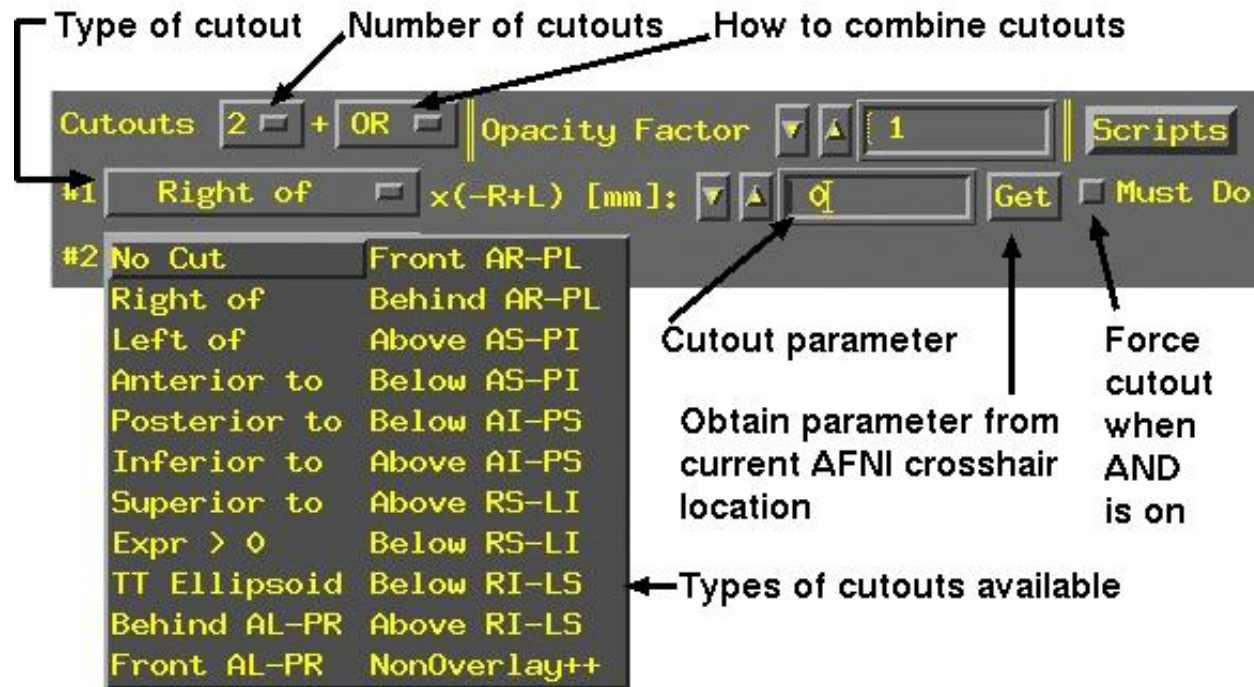
- Controlling the mappings from voxel value to brightness and opacity:



- Probably want to make white matter be fully white
Drag #3 Brightness handle up to top, over to white matter value
- Probably want to reduce Opacity to 0 for all low intensity voxels
Drag #2 Opacity handle to bottom, over to histogram trough value
Then press [Draw](#) to force a re-rendering



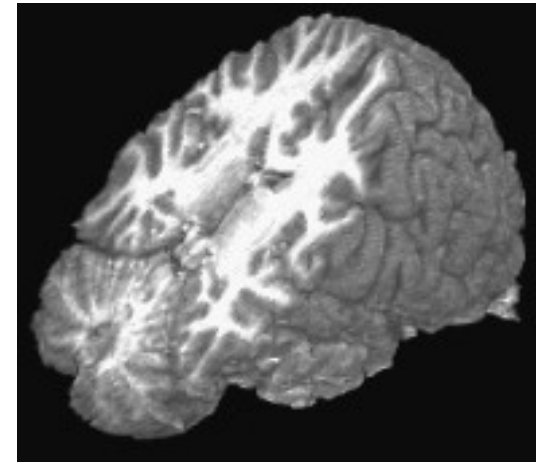
- Cutouts are for removing parts of the volume so you can see the parts you want:



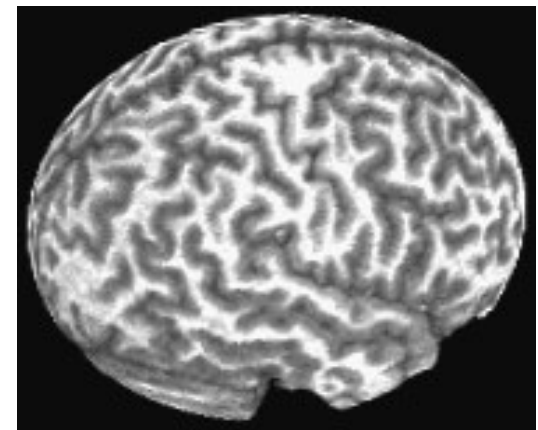
- ◇ Each cutout specifies a sub-volume in space that will be removed from the dataset before rendering (done by setting voxel opacity to zero inside the cutout)
- ◇ Multiple cutouts can be combined in two different ways:
 - ↳ OR ⇒ all voxels in all cutouts will be removed
 - ↳ AND ⇒ only voxels that are in every cutout sub-volume will be removed
 - ▷ Must Do can be used to force the removal of cutout voxels even if AND is active
 - ▷ OR is equivalent to Must Do for all cutouts

- ◇ Most cutout types are controlled by a single numerical parameter determining the position of the cutout
 - ↪ Right of 'x' means to cut out all voxels to the right of the given x-coordinates ($-x$ is Right, $+x$ is Left)
 - ▷ Similarly, can cutout everything Anterior to, or Posterior to, or Superior to, or Inferior to, or Left of a given coordinate position
 - ↪ Behind..., Below..., Front..., Above... cut out 45° diagonally slanted half-spaces, with respect to the listed planes:

For example, Above AS-PI is above a plane that slants from the Anterior-Superior front of the brain downwards to the Posterior-Inferior back of the brain—that is, halfway between a coronal and axial slice

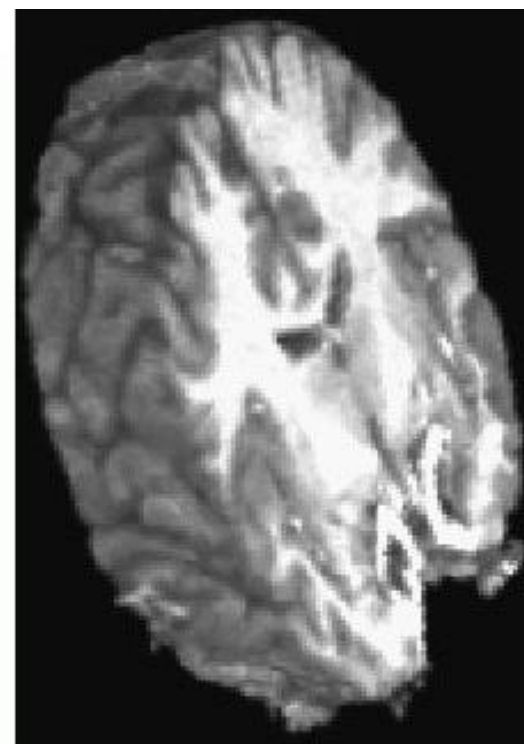
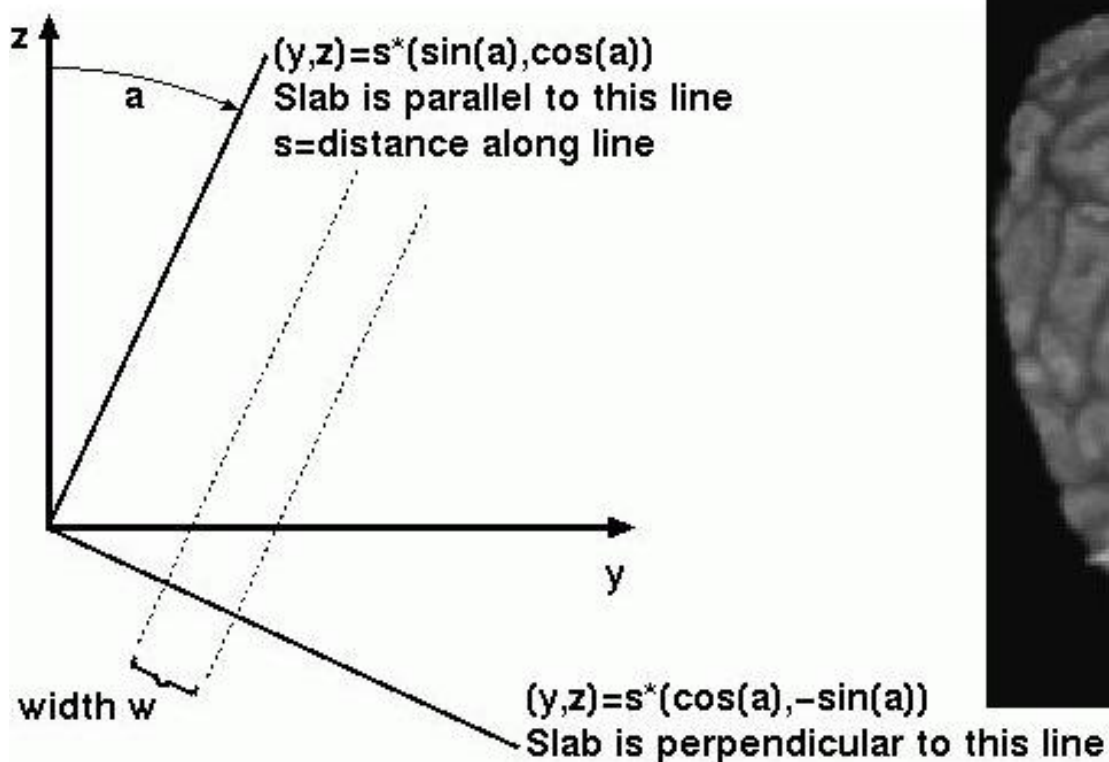


- ↪ TT Ellipsoid cuts out the region outside an ellipsoid with the same proportions as the Talairach-Tournoux Atlas brain



This is fun, but not much use

- ◇ Cutout type Expr > 0 defines the region to be removed by a general mathematical expression, rather than a single parameter
 - ↳ The expression uses the same syntax as 3dcalc
 - ↳ Variables that can be used are 'x', 'y', and 'z', corresponding to spatial coordinates in the dataset
 - ▷ When using Automate (infra), variable 't' can also be used
 - ▷ The (x,y,z) locations where the expression evaluates to a positive number will be cut out
 - ↳ Example: rendering a slab tilted at an arbitrary angle between coronal (xz-plane) and axial (xy-plane):



↪ The set of points within the slab is described by the inequality

$$|y \cdot \cos(a) - z \cdot \sin(a) - s| < \frac{1}{2}w$$

for angle= a , slab center offset= s , and slab width= w . To render a slanted coronal slab 30 mm thick, tilted posteriorly from the vertical by 25° , we would use this for the cutout expression:

$$\underline{\text{abs}(y*\text{cosd}(25)-z*\text{sind}(25)-20)-15}$$

where the `sind()` and `cosd()` functions take arguments in degrees, and where I have chosen the offset to be 20 mm (you will have to alter this to get the exact position you want)

▷ By using Automate and setting the angle (25 above) and/or the offset (20 above) to depend on 't', we can make a sequence of images where the slab rotates downwards and/or moves backwards

- [Automate](#) lets you create a large number of renderings at once

◇ Note that most (but not all) number entry boxes have slightly raised borders:



◇ Such boxes can use an expression with the variable 't' when [Automate](#) is used:

↳ Turn [Automate](#) on

↳ Enter some small number in the [Frames](#) control (say 5)

↳ Enter $70+5*t$ in the [Roll](#) control, then press [Compute](#)

↳ The dataset will be rendered with the variable 't' set to 0, 1, 2, 3, 4 in turn

▷ That is, t will run from 0 to one less than the number of Frames, and all the raised-border boxes that use expressions with 't' will be evaluated prior to each frame being rendered

↳ In this example, this will result in a sequence of views of the dataset from different roll angles 70° , 75° , 80° , 85° , 90°

↳ Can also use 't' in cutout parameters to make cutouts depend on 'time'

▷ 2 cutouts, [Left of](#)= $10+3*t$ and [Right of](#)= $-10+3*t$ will produce a 20 mm thick slice that slides leftwards as t increases

↳ Can use 't' in more than one raised-border box simultaneously to make complex animations (e.g., [Roll](#) and [Cutouts](#) together)

↳ Put cursor in raised-border box and press [Enter](#) to have box reset to last numerical value used by [Automate](#)

- Color overlays (e.g., of functional activation maps)
 - ◇ Press the [Overlay](#) button to open up the panel that controls how functional overlays are generated:

The screenshot shows the 'Choose Overlay Dataset' panel in AFNI. The current dataset is 'sample/AFNI_sample_04/func+tlrc [fico] [*]'. The panel is divided into several sections:

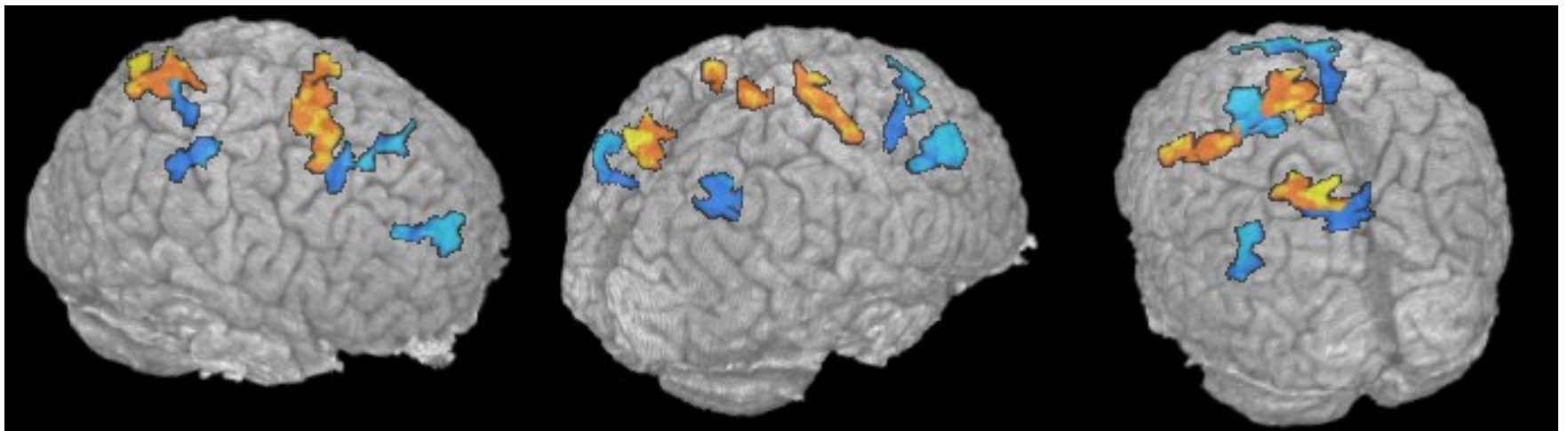
- Thresh Color:** A vertical color bar on the left shows a gradient from green (positive) to blue (negative), with a threshold line at 0.5000. Annotations indicate that voxels below this threshold will not be colored and that the colors above the threshold are determined by the 'Color' and 'Thr' settings.
- Color and Thresholding:** Sub-bricks for 'Color #0 % Change' and 'Thr #1 Correlation' are visible.
- Color Opacity:** Set to 0.5.
- ShowThru Mode:** Set to 75%.
- See Overlay:** Includes a checkbox for 'TT Atlas'.
- Cutout Overlay:** Includes a checkbox to turn overlay coloring on.
- Remove Small Clusters:** Includes a checkbox to apply cutouts to the overlay, with 'min' set to 1 and 'max' set to 200.
- Color Range:** Shows 'Color -33.26891: 44.87098' and 'Thr -0.6859: 0.7523'. The 'autoRange' is set to 44.87098.
- Other Settings:** Includes a 'Rota' dropdown and a 'Pos?' checkbox.

Annotations on the right side of the panel include:

- Sub-bricks for color and thresholding
- Opacity of colored voxels
- Select ShowThru mode and corresponding percentage
- Show regions from Talairach Atlas
- Turn overlay coloring on
- Apply cutouts to overlay
- 'Clean up' small blobs of above-threshold voxels (as in 3dclust)

- ◇ Controls are similar to [Define Function](#) for overlaying color on 2D image viewer windows
 - ↔ Will only discuss differences from 2D overlay control panel

- ◇ Color Opacity lets you select the opacity of colored voxels (those that are above the threshold)
 - ↳ Opacity of overlaid voxels is different from the opacity it would have from the underlay dataset at that location
 - ↳ Usually want this to be high (0.5 or above)
 - ↳ Two special values on this menu:
 - ▷ Underlay means that the colored voxel's opacity will be determined by the opacity that it would have from the underlay image
 - ▷ ShowThru means that colored voxels show through underlay voxels (the 'glass brain' effect), no matter how opaque the underlay is
 - Takes some practice to become accustomed to this type of image
 - But can be a very useful way to see lots of activation at once:



- Seeing this animated is especially helpful (but hard to publish)