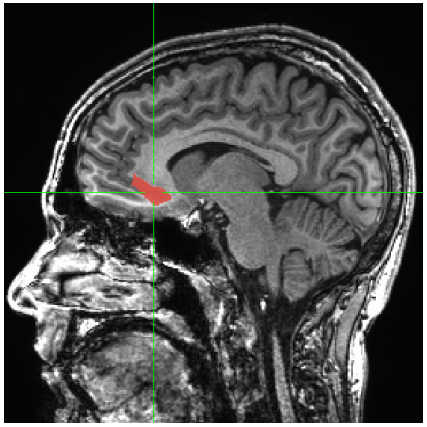
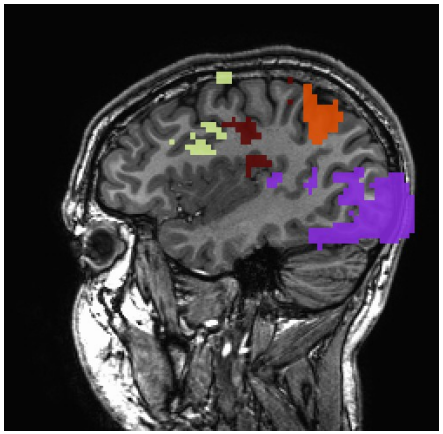
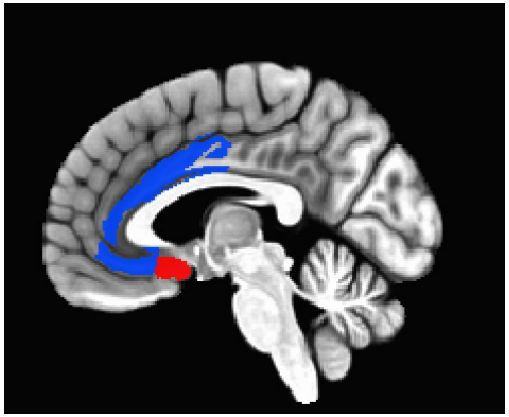
Regions of Interest (ROIs)
for those interested in regions

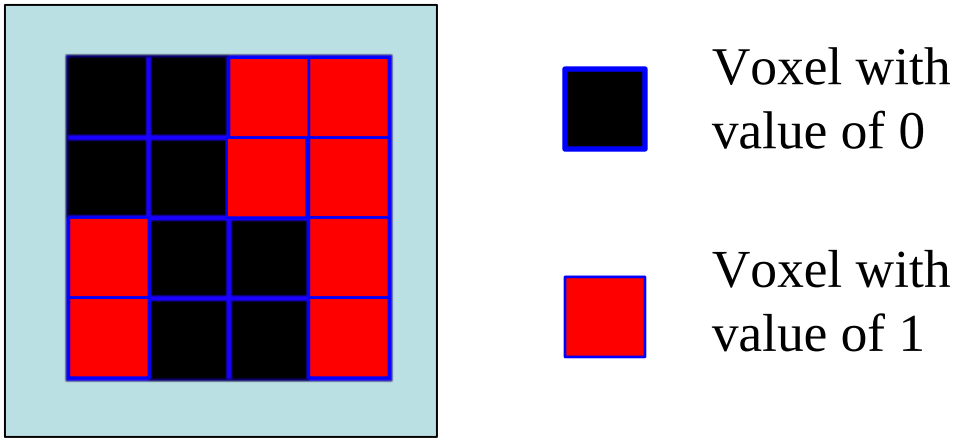






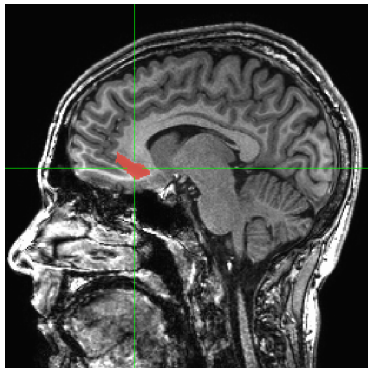
What is an ROI?

- An ROI is a "region of interest" usually used as a mask of voxels
- In AFNI, ROIs are stored as any other dataset (.HEAD/.BRIK, .nii, ...) , typically with positive integer values for voxels to consider.
  - Zero values are outside the mask.
  - Positive values are inside the mask

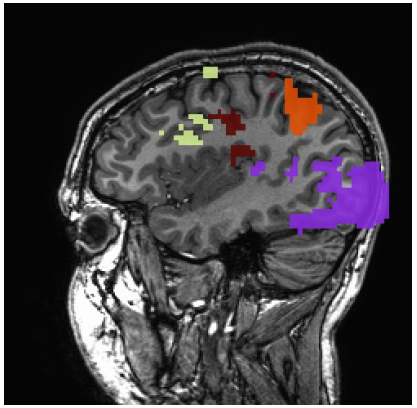


- Note any dataset can be considered as an ROI if it is non-zero in areas in which you are interested.
- Usually stored as byte integers to save disk space and memory

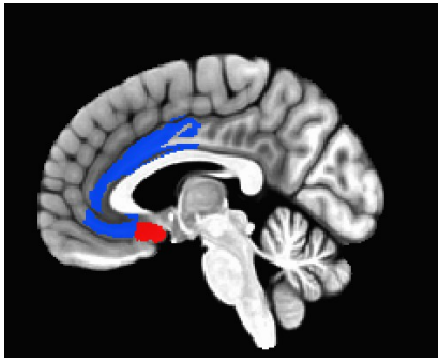
Region of Interest Creation in AFNI



- Method 1: **Draw.** Manually select Regions of Interest (ROIs) based on anatomical structures, then analyze functional datasets within these regions.



- Method 2: **Cluster.** Analyze functional dataset for *entire* brain first, then focus on geometrically connected clusters of 'activity' (supra-threshold voxels in some functional statistical map). **3dClusterize** or Clusterize plugin in afni GUI.

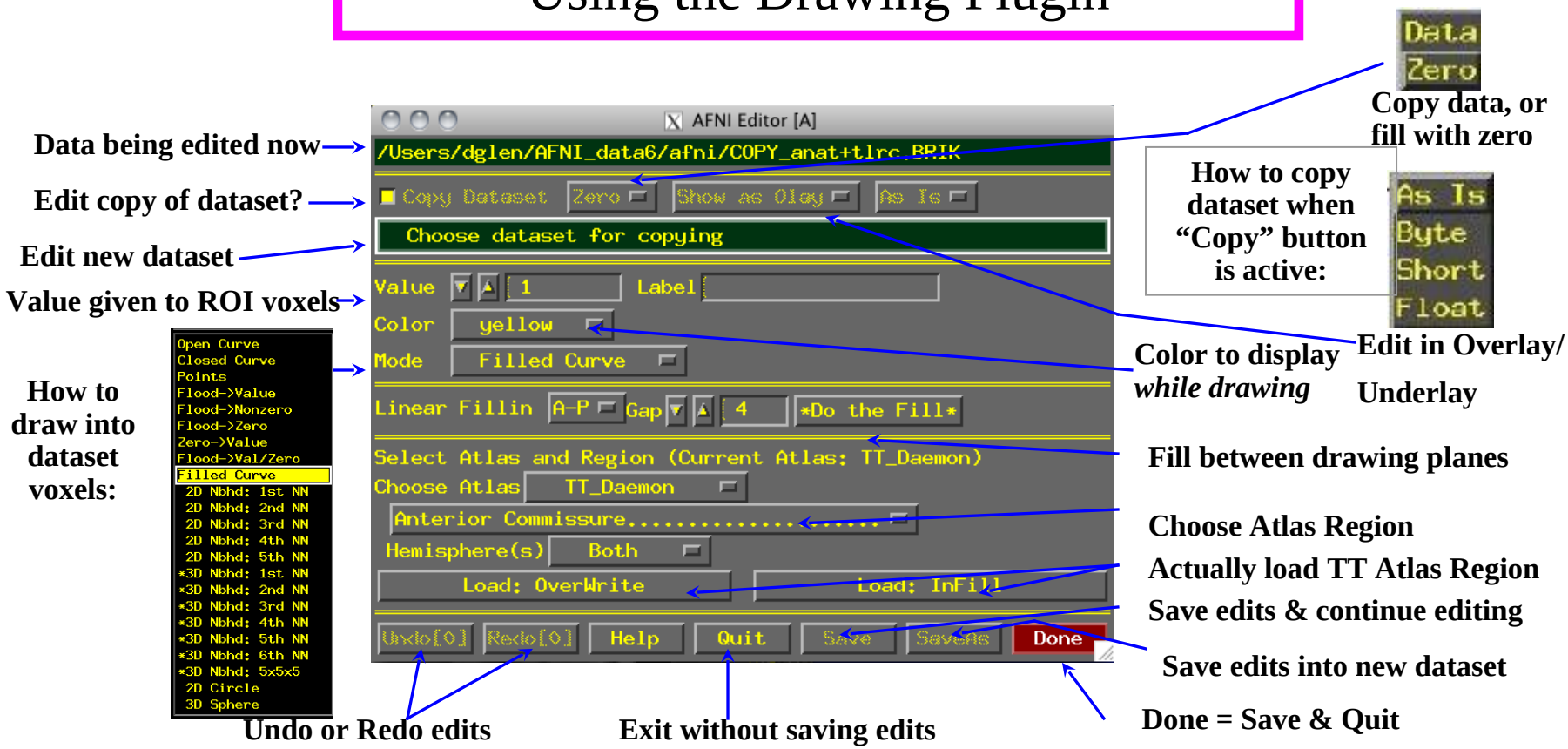


- Method 3: **Atlas**. Use atlases to select your ROIs regions
  - Use atlas datasets, transform to output space (resampling or warping) by using the **`whereami/3dcalc/...`** programs, symbolic notation or simple dataset indices to create masks on the command line.

Method 1- Handmade ROI's from Anatomical Volumes

- **Quick outline of procedure**:
  1. On the main AFNI control panel, set the anatomical underlay dataset (with **[UnderLay]**) to be what you want to draw on -- usually a SPGR or MP-RAGE type of dataset
     1. i.e., the anatomical underlay will serve as our guide or template for drawing the ROI mask
  2. Start the **Draw Dataset** plugin (this is our ROI-creating plugin):
     1. **[Define Datamode]** → **[Plugins]** → **[Draw Dataset]**
  3. Create an *all zero* anatomical overlay dataset with the **Draw Dataset** plugin. This is the beginning of our ROI mask. At this point, the anatomical overlay is empty - i.e., all the voxel values are zero,
  4. To view and begin drawing an ROI mask (or several ROIs) on this blank anatomical overlay dataset, go to the main AFNI interface and **[Switch Underlay]** to the anatomical dataset you will be using as your template.
  5. Start drawing the ROI mask into this blank anatomical overlay dataset. Voxels inside the ROI mask will receive a non-zero value (you decide what value to give them). Values outside the ROI mask will remain zero
  6. Be sure to save the results by pressing **[Save]**, **[SaveAs]** or **[DONE]** in the ROI plugin GUI (**[Quit]** will exit the ROI plugin without saving your work)

Using the Drawing Plugin



- **Critical things to remember:**
  - You should have [See OverLay] turned on, and be viewing the same overlay dataset in AFNI as you are editing
    - Otherwise, you won't see anything when you edit!
  - When drawing, you are putting numbers into a dataset brick
    - These numbers are written to disk only when you do [Save], [SaveAs] or [Done]; before then, you can [Quit] (or exit AFNI) to get the unedited dataset back

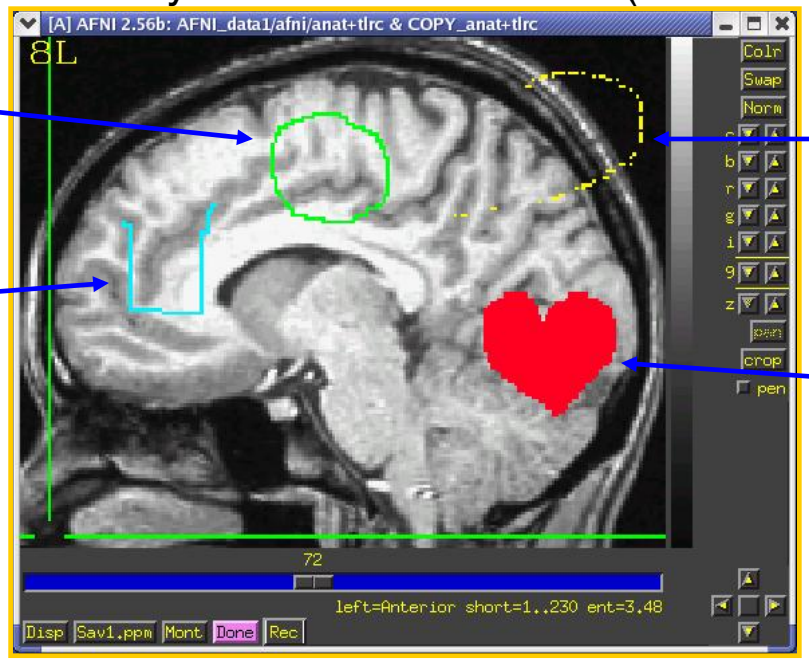
Keys 'o' and 'u' and scroll wheel come in handy here

- **Step 1: Load a dataset to be edited (for ROI creation):**
  - [Choose Dataset] button gives you a list of datasets that
    (a) Actually have brick data with only one sub-brick;
    (b) Are at the same voxel dimension, grid size, etc., as current anatomical underlay
  - When you are starting, you probably don't want to edit an existing dataset -- i.e., you don't want to write on the underlay itself; you just want to use it as a template and draw on a blank overlay that shares the same geometry as that existing underlay dataset
    - To do this, you must create an all-zero *copy* of the anatomical underlay (by "copy" we mean the all-zero dataset shares the same geometry as the underlay, but not the same voxel data values)
    - To create an all-zero copy, click the [Copy] button on (from the **Draw Dataset** plugin GUI) and set the controls to its right to [Zero] [Show as Olay] and [As Is]
      - [Data] would make a copy of the underlay dataset with the actual voxel data values. [Zero] copies the geometry of underlay, but gives each voxel a data value of *zero* (this latter option is usually what you want when starting out)

- [As Is] keeps the voxel values in the copy as the same type as in the original underlay; you can also change the voxel values to be stored as:
  - [Byte] (integer values: 0..255) ⇒ 1 byte each
  - [Short] (integer values: -32767…32767) ⇒ 2 bytes each
  - [Float] (fractional values) ⇒ 4 bytes each
  - Bytes and Shorts make the most sense for ROI masks, where you are essentially attaching labels to voxels
- Click on [Choose Dataset], select the dataset you want a copy of (e.g., the anatomical underlay), and then press [Set]

- **Step 2: Drawing the ROI (or ROIs):**
  - *Choose the value to draw into the anatomical overlay dataset* (recall that all values in the copied/blank overlay dataset are zero at this point)
    - If you drawing only one ROI, then the default value of 1 is good
    - **Erase with value of zero!**
    - If you are drawing multiple ROIs, then you should choose a different numerical value for each so that they can be distinguished later
      - Pencil and paper are our friends -- write down which number corresponds with which ROI for later recall!
  - You could use ROI color maps (this is now the default):
    - Choose the '**' map. Set 'pos' on.
    - Right click on colormap --> Choose Colorscale --> ROI_i256 (or ROI_innn),
    - Set 'autoRange' off. Set range to 256
  - *Choose the drawing color*
    - This is the color that is shown *while you are drawing*

- After you finish a drawing motion, the voxels you drew will be filled with the drawing value, the image will be redisplayed, and the colors will be determined by the **[Define OverLay]** control panel
- *Choose the drawing mode*
  - **[Filled Curve]**
    Drawing action produces a continuous closed-ended curve (default setting)
  - **[Open Curve]**
    Drawing action produces a continuous open-ended curve
  - **[Closed Curve]**
    Drawing action produces a continuous closed-ended curce
  - **[Points]**
    Only points actually drawn over are filled (used to “touch up” and ROI)



- **[Flood→Value]**
  Flood fills space outward from the drawing point, stopping when the flood hits the current drawing value (used to fill a closed curve)
- **[Flood Nonzero]**
  Drawing action produces a continuous closed-ended curve (filled inside)
- **[Zero→Value]**
  Floods voxels with zero until the flood hits nonzero voxels (you can also do this more easily with Filled Curve, value=0)
- **[Flood→Nonzero]**
  Flood fills outwards from drawn point, stopping when the flood hits any nonzero voxel (used to fill between regions):

- Important Note:
  - An ROI is defined by the values stored in voxels of the 'mask' dataset
  - ROIs do not need to have contiguous voxels – they can be scattered
  - Two voxels are in the same ROI if they have the same value in the mask dataset (i.e., it doesn't matter where they are located in the volume)

- *Draw something*
  - Drawing is done with mouse Button 2 (“middle” button) in 2D slice image (shift-drag on Mac trackpad) or use “Pen” button
  - Hold the button down in the image window during a single drawing action
  - While the drawing action is happening, the chosen drawing color will trace the screen pixels you draw over
  - When you release the button, these pixels are converted to voxels, and the dataset is actually changed, using the drawing value and drawing mode you selected
  - At this point, the color of the drawn region will change to reflect the drawing value and the setup of the **[Define OverLay]** control panel
- **[Undo]** button will let you take back the last drawing action (you can go “undo” many levels back, i.e., multiple undo function)
- You can draw on one 2D slice image at a time (see sphere though)

- **Step 3: Save your results:**
  - **[Save]** will write the current dataset values to disk (overwriting any existing .BRIK file, i.e., if you had edited this ROI earlier, the new changes would overwrite the old file)
    - You could also then choose another dataset to edit
  - **[Save As]** will let you write the current dataset to disk under a new name, creating a new dataset, then continue editing the new dataset
  - **[Quit]** exits editing and closes the plugin window, without saving to disk any changes since the last [Save]
    - Exiting AFNI has the same effect
  - **[Done]** is equivalent to **[Save]** then **[Quit]**
- **Optional Drawing Steps:**
  - **[Linear Fillin]** lets you draw a 3D ROI not in every slice, but in every third slice (say), and then go back and fill in the gaps
    - For example, if you draw in coronal slices, then you want to fill in the **[A-P]** direction (the default)
    - If you draw every nth slice, then you want to set the Gap to n-1
    - Line segments of voxels in the fillin direction that have a current drawing value at each end, and have no more than [Gap] zero voxels in between, will get their gap voxels filled with the drawing value
      - After you try this, you will probably have to touch up the dataset manually

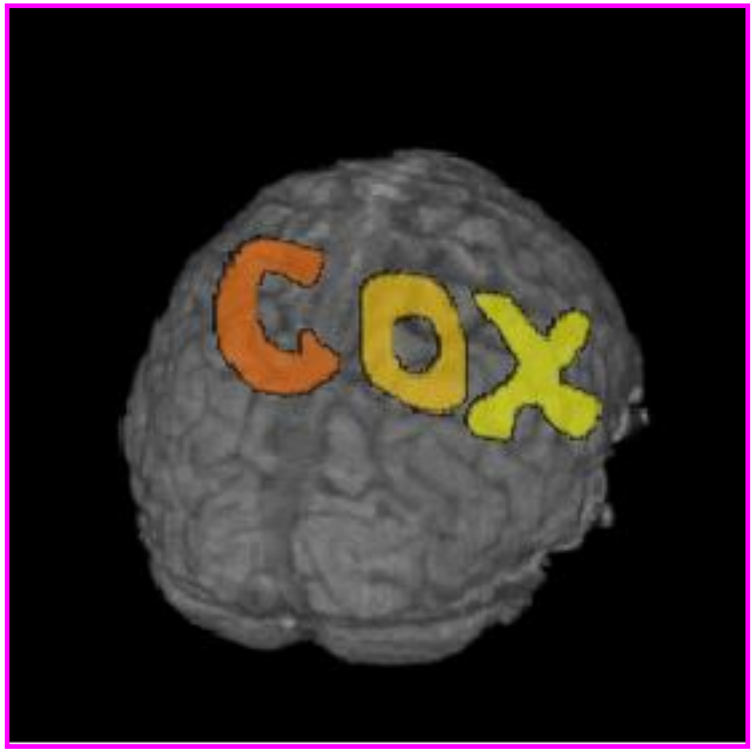
- This operation can also be done with program **3dRowFillin**, which creates a new dataset

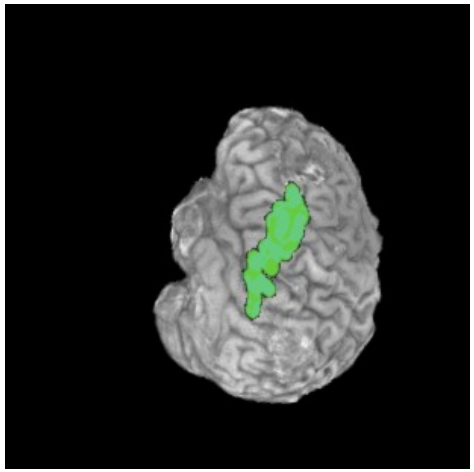
- **[Select Atlas and Region]** lets you load regions from any loaded atlas
  - Requires the dataset in standard space coordinates or with transformation from **+orig → +tlrc** computed in the current directory (only for affine alignment with @auto_tlrc)
  - Choose a region to draw into the dataset (e.g., Hippocampus)
  - **[Load: Overwrite]** will fill all voxels in the region with the drawing value
  - **[Load: Infill]** will fill only voxels in the region that are currently zero
  - Start with regions from atlas and combine or modify as needed

- **Drawing and Volume Rendering at the Same Time** (totally fun, and maybe useful)
  - You cannot draw into the rendering plugin, but you can use it to see in 3D what you are drawing in 2D
    - If you meet the criteria for rendering (usually in `+tlrc` coordinates)
  - How to set up the renderer:
    - Choose the underlay to be the current anatomical dataset (or a "scalped" version, from **3dSkullStrip** or **3dIntracranial**)
    - Choose the overlay dataset to be the dataset you are editing
    - Turn on **[See Overlay]**
    - Set **[Color Opacity]** to **[ShowThru]**

- Turn on [DynaDraw]
- Drawing in a 2D image window immediately triggers a redraw in the rendering window

  (if the 2D and 3D overlay datasets are the same)
- This is only useful if your computer is fast enough to render quickly (<1 sec per frame)

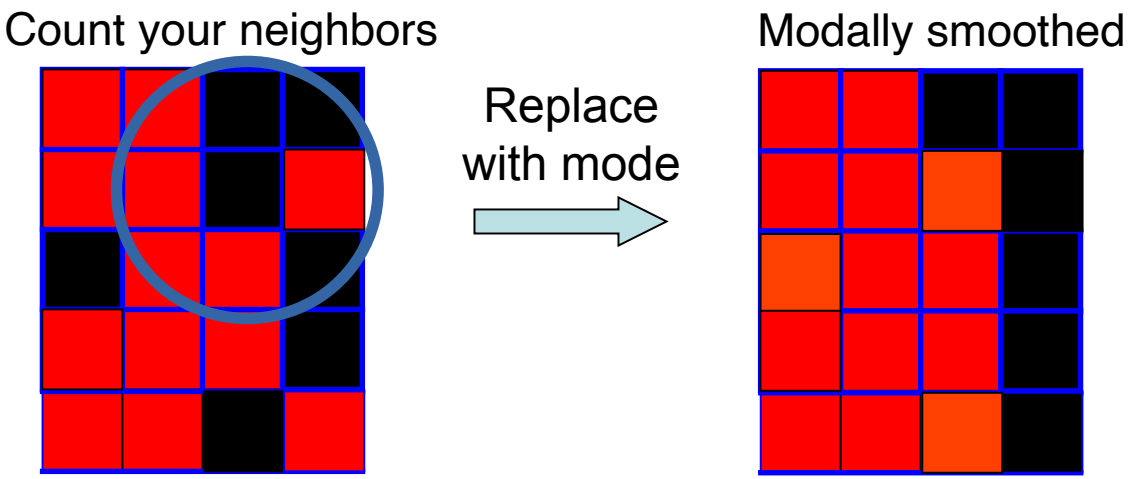




Modal Smoothing -fixing ROIs

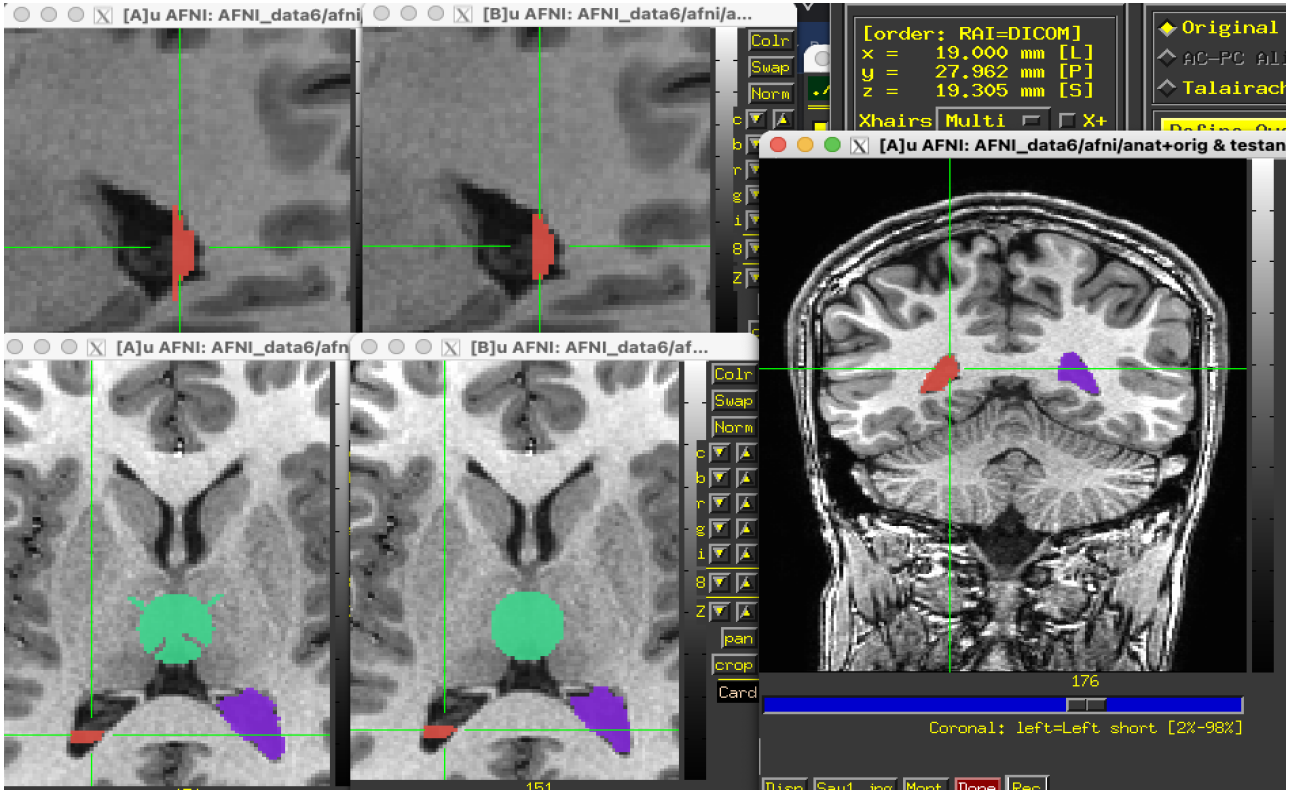
Drawing is not perferct!

- Manual errors - drawing in 2D slices, random “paintbrush” drips
- uneven tissue contrast
- arbitrary boundaries
- Nonlinear warping - distort regions
- Maximum probability maps - combinations of regions across subjects



Modal Smoothing (continued)

```
3dLocalstat -stat mode -nbhd 'SPHERE(-1.8)' \
   -prefix myrois_mode1.8 myrois+orig.
3drefit -cmap INT_CMAP -copytables myrois+orig myrois_mode1.8+orig
```



Modal Smoothing (continued)

Modally smooth on surface with SurfLocalstat

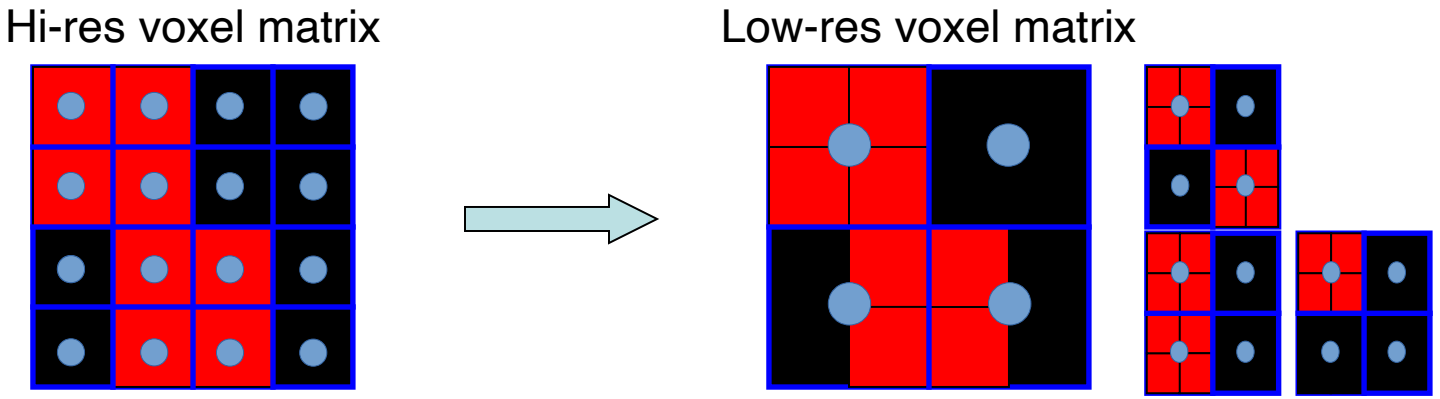
Schaefer-Yeo 400 region atlas (before and after)

Also see
@ROI_modal_grow - to grow regions iteratively to fill mask

@ROI_decluster -
to find lost clusters made during drawing or other processing like Maximum Probability Map generation

Resampling ROIs - Making a resolution

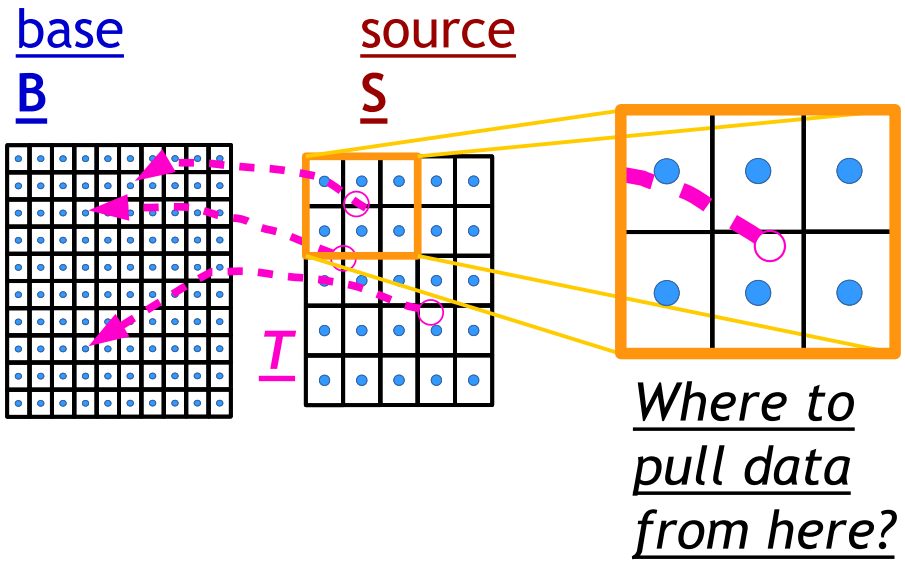
- ROIs created at the anatomical resolution applied to FMRI resolution
  - Regions can disappear - multiple regions compete for the same voxel, and narrow regions can be missed. Be sure they still are represented with enough voxels
  - If only 2 voxels thick, partial voluming (mixed tissue)
  - afni_proc.py QC report now includes information some ROI information
    - -regress_compute_tsnr_stats ROI_DSET_LABEL ROI_1 ROI_2 ...
    - : compute TSNR statistics per ROI



Nearest neighbor interpolation - tricky here, even for this simple case, and 3D

Think about the real resolution of your data - Can the ROI be reliably observed and analyzed?

Alignment interpolation review



- Let's do a class example of **3dresample**:

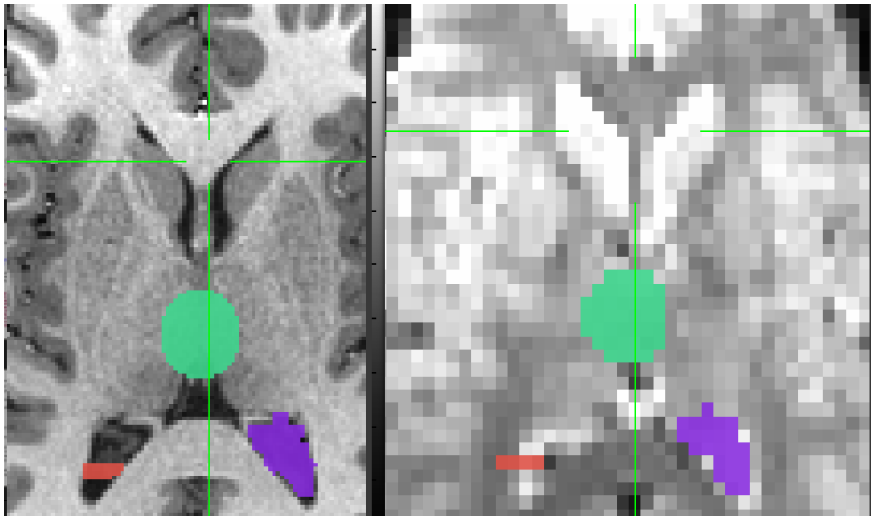
```
cd AFNI_data6/afni  (or roi_demo)
3dresample -master rall_vr+orig          \
           -prefix anat_roi_resam        \
           -inset anat_roi+orig          \
           -rmode NN
```

In this class example, we want to take our ROI mask, which has a high voxel resolution of 0.9 x 0.9 x 1.2 mm, and resample to it the lower resolution of the time-series dataset, **rall_vr+orig** (2.75 x 2.75 x 3.0mm).

Anatomical resolution

anat_roi+orig

0.9x0.9x1.2 mm voxel grid



Functional resolution

anat_roi_resam+orig

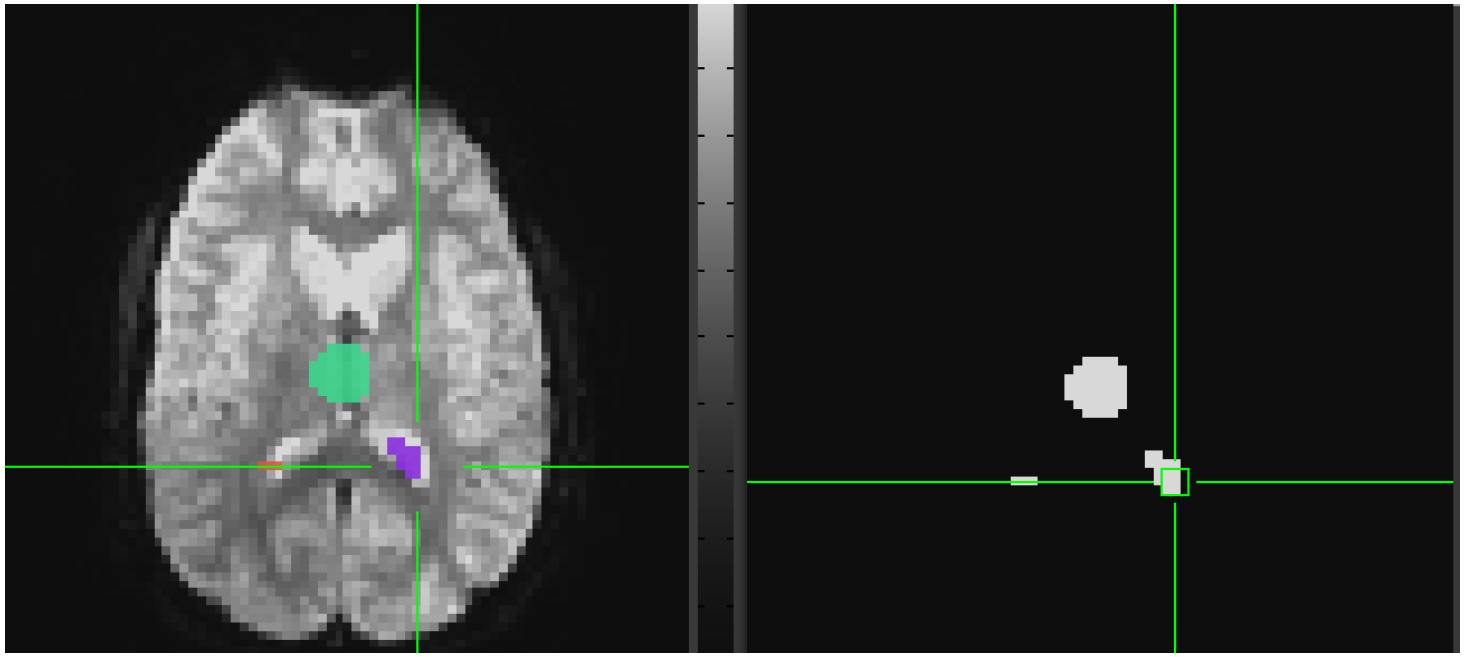
0.9x0.9x1.2 mm voxel grid

Underlay is functional dataset - rall_vr+orig

Masking Data with 3dcalc

Apply a mask to the data:

```
3dcalc -a rall_vr+orig. -b anat_roi_resam+orig \
          -expr 'a*step(b)' -prefix rall_masked
```



- **3dmaskave**
  - Compute the average of voxels within an ROI mask
    Class Example:
    - ```
      3dmaskave -mask anat_roi_resam+orig'<2>' -q \
                rall_vr+orig > epi_avg.1D
      ```

      Compute mean of all voxels within ROI 2 at every time point/volume.

      In this example, there are 450 time-points in this dataset, so the output will be a column of 450 means.

      -q : Suppresses the voxel-count output (e.g., "[9 voxels] make up the ROI mask") from appearing next to each mean.

      Alternatively, instead of having the results of **3dmaskave** shown into the shell, you can redirect (**>**) the output into a text file (**epi_avg.1D**) for later use.
      ```

Output will look like this (450 means in the column):

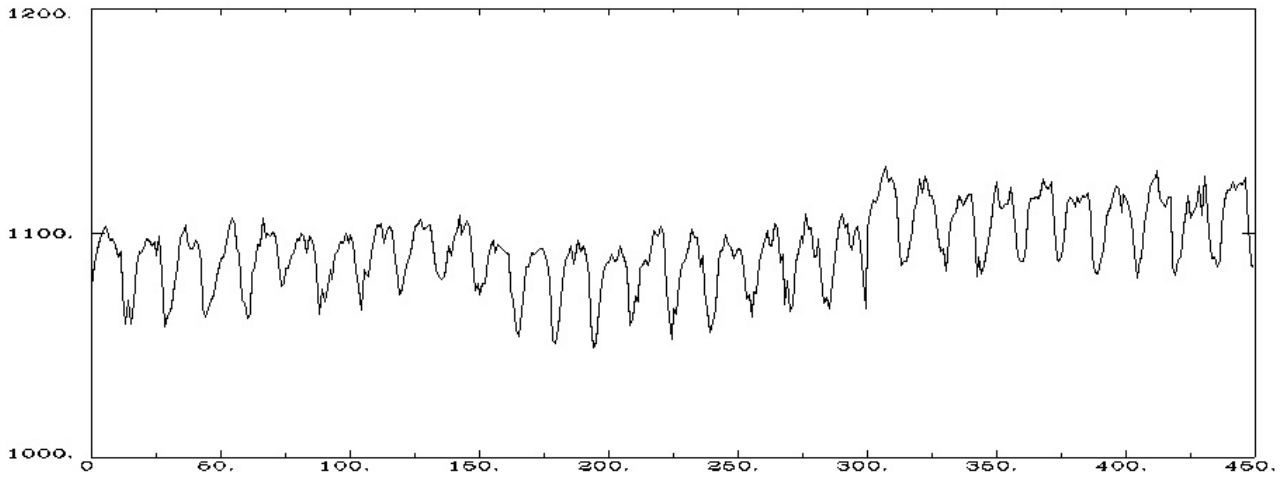
```
more epi_avg.1D
```

```
1076.11
1086.11
1092.33
1097.33
  ...
1084.76
```

Data can also be plotted out using **1dplot**:

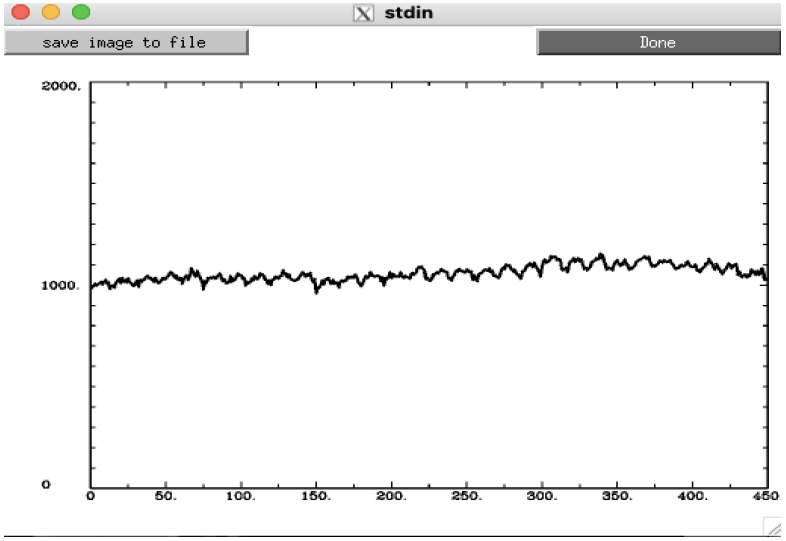
```
1dplot epi_avg.1D  or…
1dplot -yaxis 1000:1200:2:1 epi_avg.1D
       (adjust min and max for ROI data)
```

Mean voxel intensity for voxels falling within the ROI mask (at each timepoint).



- **3dmaskdump**
  - Get the individual voxel values for a dataset within an ROI
  - Class examples:

```
3dmaskdump -noijk -dbox 33 78 -11 rall_vr+orig.|1dplot -stdin
```




```
3dmaskdump -noijk -mask anat_roi_resam+orig     \
           'func_slim+orig[2]' > Vrel-tstats.txt
```

The output appears in the shell (unless you redirect it (**>**) into a text file). This example shows one column of numbers for all the voxel values for functional sub-brick #2 ('Visual-reliable' t-values) that fall within the ROI mask

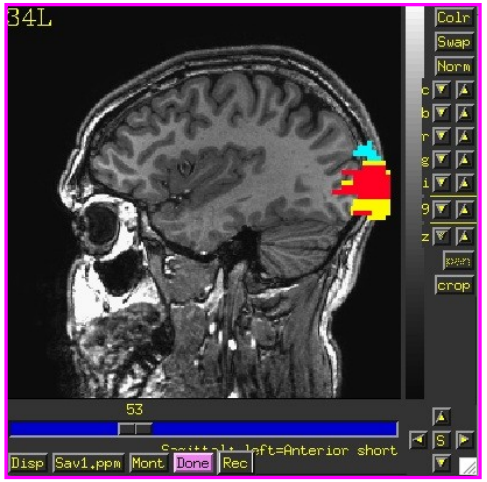
N.B. - **3dUndump** can be used to create a dataset from a text file with ijk-indexes and dataset values

Simple ROI Statistics - 3dROIstats

Program to compute separate statistics for each ROI in a dataset

- E.g. Mean can be computed for multiple ROIs
- This differs from **3dmaskave** because the ROIs within the mask are not combined

- Averaging is done over each region defined by a distinct numerical value in the ROI dataset




```
3dROIstats -mask anat_roi_resam+orig. func_slim+orig'[0]'

File                 Sub-brick    Mean_roi1    Mean_roi2    Mean_roi3
func_slim+orig[0] 0[Full_Fsta] 8.861037     5.241786     2.422280
```

Other stats can be added - median, mode, std. deviation, volume, ...

Even simpler ROI statistics!

```
3dBrickStat -mask anat_roi_resam3+orig.'<1>' -mean func_slim+orig'[0]'
8.86104
```

Method 2: Creating ROI datasets from Activation Maps

- ROI masks derived from functional data can be made by finding contiguous supra (above) threshold voxel clusters in an activation (functional) map and then converting each cluster into a ROI with a separate data value
  - These ROIs can then be used as starting points for some analysis

- Example:

  `cd AFNI_data6/roi_demo` and launch `afni &`

- Let's pick select some criteria that will determine how big our voxel clusters will be. Any voxels that survive these criteria will be part of our ROI mask(s)

  Select UnderLay: **`anat+orig`**

  Select OverLay: **`func_slim+orig`**

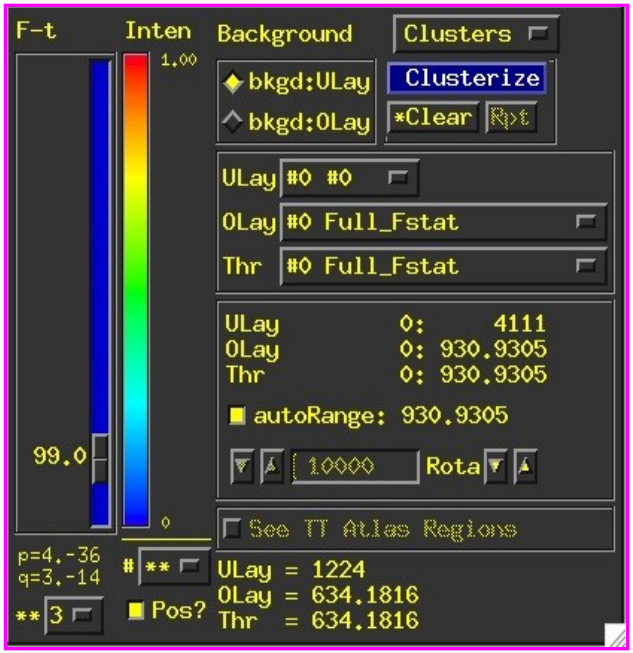
  --> Define OLay & Threshold: **Sub-brick 0 (Full-F)**

  --> Set Threshold to **F 99.0**

  --> To be part of a cluster, voxels must be right next to each other, **rmm = 2.75**

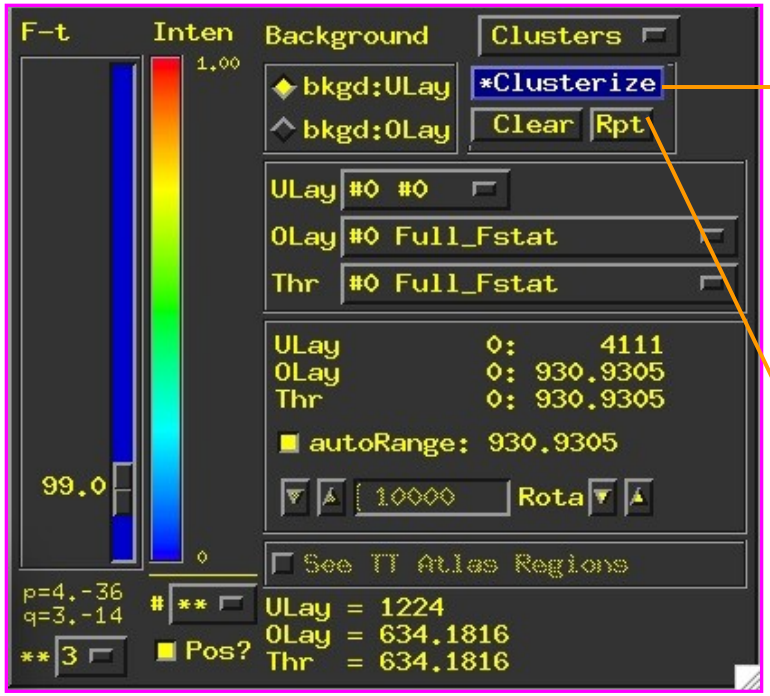
  --> Clusters must be 200+ voxels in size

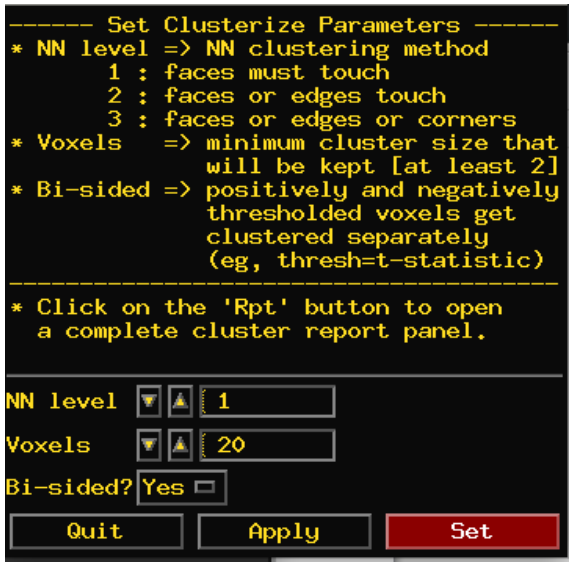
  2.75 x 2.75 x 3.0 = 22.69 x 200 = **vmul = 4538**

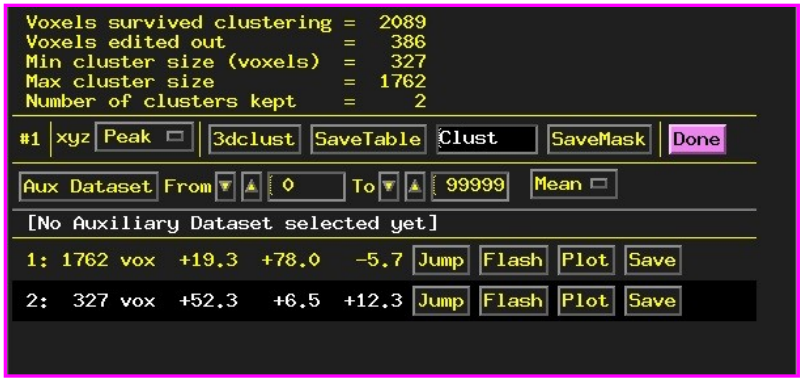


Clusterize

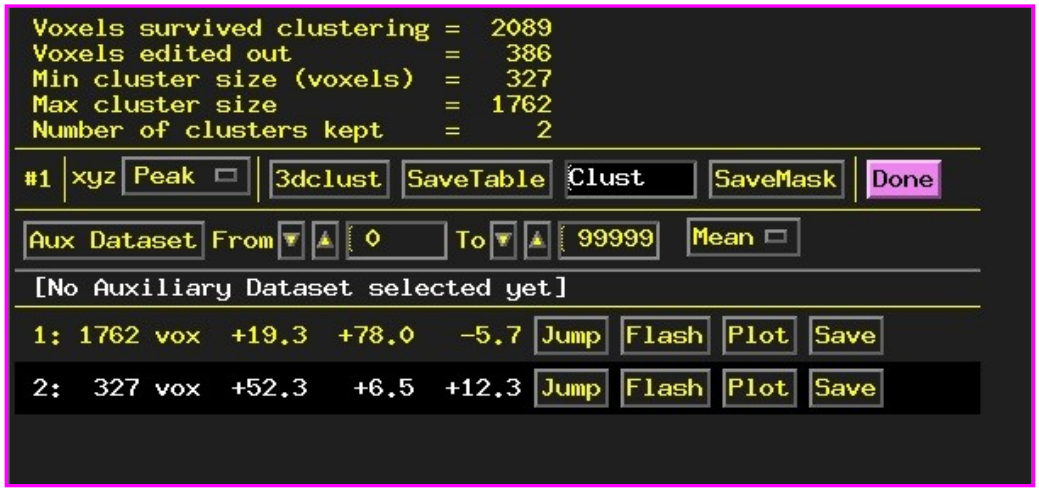
- The **Clusterize** button on the main AFNI graphical interface gives users a quick and easy way to locate clusters of activity in a functional dataset. Once the user sets the clusterize parameters, a complete cluster "report" is given, which details the number of clusters found, based on these parameters.







Cool Clusterize Features



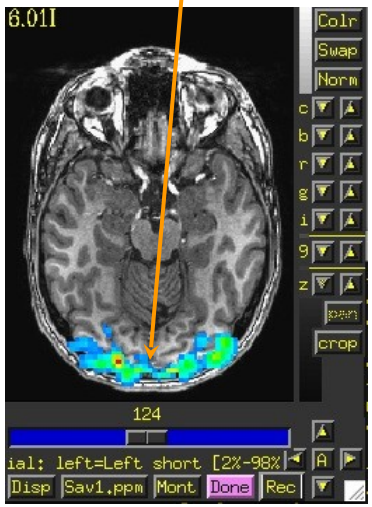
Jump: sets the crosshairs to the designated xyz coordinates (default is the *peak* of the ROI cluster)

Flash: flashes the cluster voxels in the image viewer

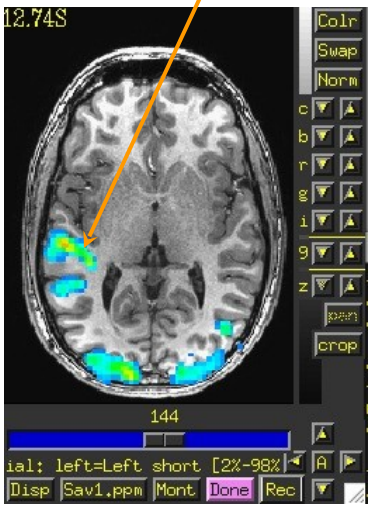
SaveMask: Click on this button to write clusters to a mask dataset called **Clust_mask+orig**

Plot/Save: Allows user to load a 3D+time dataset (Aux Dset button) and plot the avg time series over a cluster. Plot can be saved in .jpg or .png format.

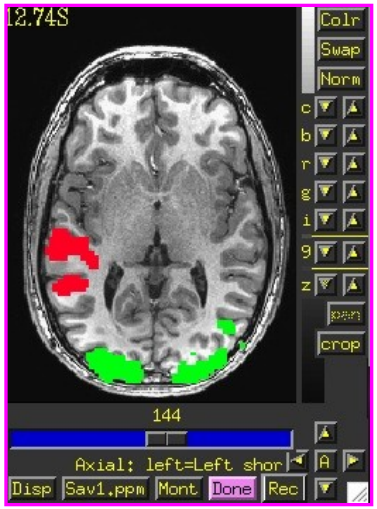
Cluster #1



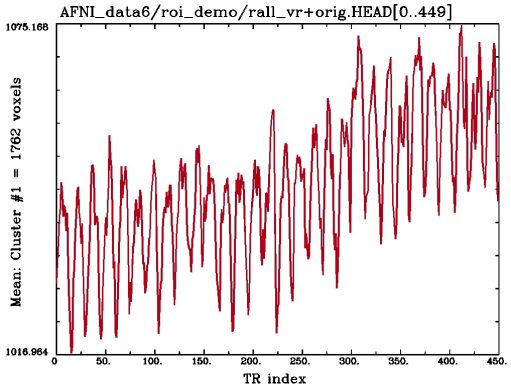
Cluster #2



Clust_mask+orig



E.g., 3D+time dataset **rall_vr+orig** loaded and avg time series plotted for voxels within Cluster #1

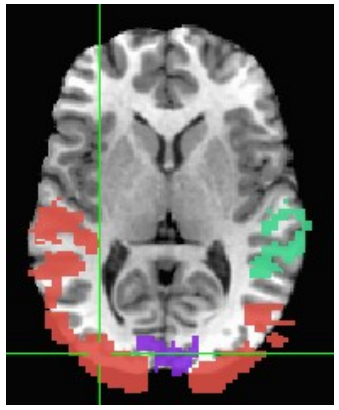


3dClusterize

- The program **3dClusterize** looks for clusters of activity that fit the criteria set on the command line to get report and and cluster datasest -- similar to 3dclust and 3dmerge
- Example:

3dClusterize -clust_nvox 200 -bisided -2 2 -ithr 2 -idat 1 -NN 1 -inset func_slim+orig. -pref_map myclusters

The above command tells `3dClusterize` to find potential cluster volumes for dataset `func_slim+orig`, sub-brick #2, where the threshold has been set to 2.0 (i.e., ignore voxels with an activation threshold absolute value <2.0). Voxels must be facing each other in the cluster, and cluster volume must be at least 200 voxels (these are not guidelines, just an example!).

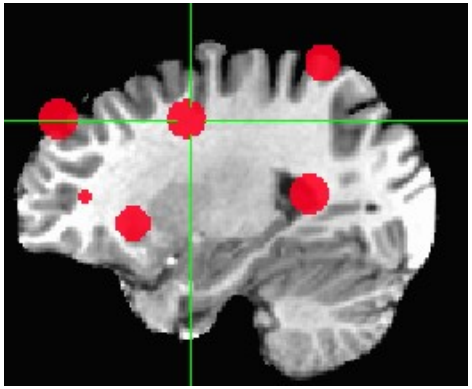


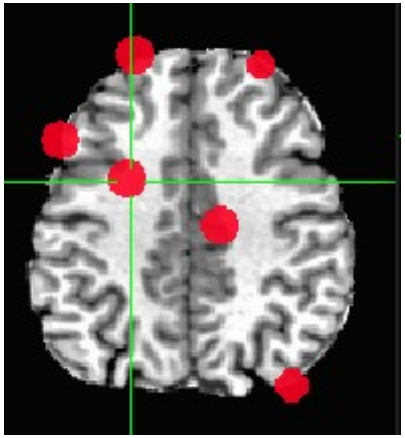
```
#  Cluster report
#[ Option summary      = bisided,-2,2,clust_nvox,200,NN1 ]
#[ Threshold value(s)  = left-tail stat=-2.000000;right-tail stat=2.000000 ]
#[ Nvoxel threshold    = 200;  Volume threshold = 4537.500 ]
#[ Single voxel volume = 22.688 (microliters) ]
#[ Neighbor type, NN   = 1 ]
#[ Voxel datum type    = float ]
#[ Voxel dimensions    = 2.750 mm X 2.750 mm X 3.000 mm ]
#[ Coordinates Order   = RAI ]
# Mean and SEM based on absolute value of voxel intensities ]
#
#Volume  CM RL  CM AP  CM IS  minRL  maxRL  minAP  maxAP  minIS  maxIS    Mean     SEM    Max Int  MI RL  MI AP  MI IS
#------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -------  -------  -------  -----  -----  -----
  16791  -11.0   13.5    9.6  -96.2   82.6  -120.0   94.5  -17.7   78.3   1.0198   0.0087   11.135   41.3  -70.5  -14.7
  15563  -14.6   20.2   34.4  -93.4   66.1  -103.5   94.5  -17.7   78.3   0.4392   0.0037   -8.114  -90.7   -7.3   30.3
    991   50.9   -5.4   43.7   16.6   68.8   -26.5   23.0   12.3   78.3   0.4297   0.0139   -4.655   55.1    3.7   78.3
    421   48.2   -1.7   -9.9   24.8   63.3   -26.5   36.7  -17.7    6.3   0.4582   0.0126  -1.6187   57.8   -4.5   -2.7
    418  -52.4   -4.0   -9.5  -74.2  -21.9   -29.3   25.7  -17.7    6.3   0.4287   0.0132  -2.2152  -24.7  -12.8  -14.7
    326   -2.6  -55.6   61.6  -24.7   11.1   -89.8  -23.8   45.3   78.3   0.2991    0.011   1.4813    2.8  -87.0   60.3
    206  -23.5  -30.9   -5.5  -49.4  -13.7   -45.8   -1.8  -17.7   12.3   0.6163   0.0445  -4.0194  -16.4  -40.3  -14.7
#------  -----  -----  -----  -----  -----  -----  -----  -----  -----  -------  -------  -------  -----  -----  -----
# 34716  -10.8   14.3   16.8                                             0.7196   0.0048
```

Getting around with spheres

Make spheres from the cluster peaks, centers of mass or from previous coordinates from literature or from locator tasks

```
3dUndump -srad 7.5 -master func_slim+tlrc -orient RAI   \
    -prefix clust_spheres -xyz Clust_PeakXYZ.1D
```





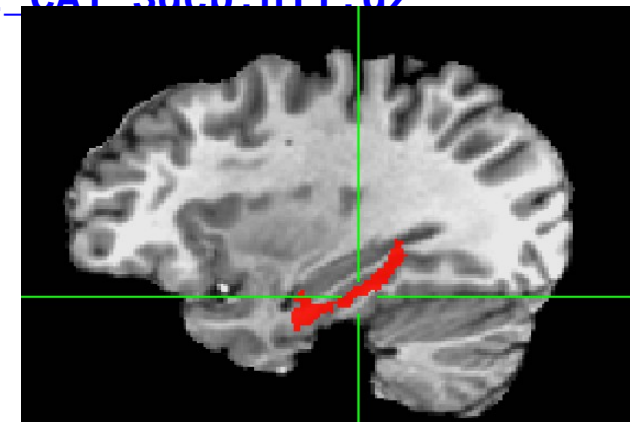
Method 3: ROIs from Atlases

- **whereami_afni** and all afni programs can extract ROIs from atlases

Example: Use the Julich_Brain atlas to create an ROI from right_CA1 region

```
whereami_afni -mask_atlas_region 'Julich_MNI2009c::right_CA1' \
      -prefix right_CA1.nii.gz
or
3dcalc -a ~/abin/Julich_MNI2009c.nii.gz'<27>' \
      -expr 'step(a)' -prefix right_CA1_3dc.nii.gz
or
3dcalc -a ~/abin/Julich_MNI2009c.nii.gz'<right_CA1_(Hippocampus)>' \
      -expr 'step(a)' -prefix right_CA1_3dcb.nii.gz
```

ULay: **anat+tlrc**
OLay: **right_CA1.nii.gz**



- **whereami** can provide the user with more detailed information regarding the output of **3dClusterize**
  - For instance, say you want more information regarding the center of mass voxels from each cluster (from the 3dclust output). I.e., where do they fall approximately within the atlases?

```
3dClusterize -clust_nvox 200 -bisided -8.0 8.0 -ithr 2 -idat 1 -
  NN 1 -inset func_slim+orig. -quiet > visual_clusters.1D
whereami -coord_file clusts.1D'[1,2,3]' -tab | less
```

Center of mass output, columns 1,2,3, from 3dClusterize, Clusterize plugin or 3dclust reports.

```
++ Input coordinates orientation set by default rules to RAI
+++++++ nearby Atlas structures +++++++

Original input data coordinates in TLRC space

Focus point (LPI)                Coord.Space
  -3 mm [L],  -83 mm [P],    2 mm [S]   {TLRC}
  -3 mm [L],  -86 mm [P],   -3 mm [I]   {MNI}
  -3 mm [L],  -90 mm [P],    2 mm [S]   {MNI_ANAT}
Atlas          Within  Label                          Prob.   Code
TT_Daemon       0.0    Left  Lingual Gyrus             MPM    232
TT_Daemon       0.0    Left  Brodmann area 18          MPM    295
TT_Daemon       1.0    Left  Brodmann area 17          MPM    294
TT_Daemon       4.0    Left  Cuneus                    MPM    240
TT_Daemon       4.0    Right Lingual Gyrus             MPM    32
TT_Daemon       4.0    Right Brodmann area 18          MPM    95
TT_Daemon       5.0    Right Brodmann area 17          MPM    94
TT_Daemon       7.0    Right Cuneus                    MPM    40
CA_ML_18_MNIA   0.0    Left Calcarine Gyrus            ---    43
CA_ML_18_MNIA   3.0    Left Lingual Gyrus              ---    47
CA_ML_18_MNIA   7.0    Right Lingual Gyrus             ---    48
CA_MPM_18_MNIA  0.0    Area 17                         ---    181
CA_MPM_18_MNIA  1.0    Area 18                         ---    240
```

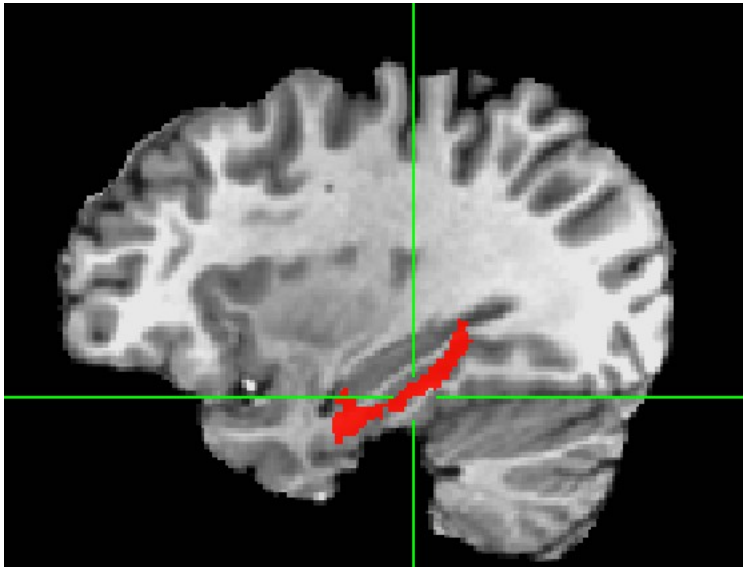
Shown: Cluster #1's coordinates according to various atlases (TT, MNI, etc), as well as the name of the anatomical structure that is located at or near these coordinates (varying by atlas)

- **whereami_afni** can extract ROIs for various atlas regions using symbolic notation

```
whereami_afni -mask_atlas_region 'Julich_MNI2009c::right_CA1' \
    -prefix right_CA1.nii.gz
```

  - The above command will use the Julich_Brain atlas to create an ROI from the right_CA1 region

ULay: **anat+tlrc**
OLay: **right_CA1.nii.gz**



- **whereami_afni** can report on the overlap of ROIs with atlas-defined regions

```
whereami_afni -omask anat_roi+tlrc
```

```
++ Input coordinates orientation set by default rules to RAI
++ Input coordinates space set by default rules to TLRC
++ In ordered mode ...
++ Have 2 unique values of:
   0   1
++ Skipping unique value of 0
++ Processing unique value of 1
++    195 voxels in ROI
++    195 voxels in atlas-resampled mask
Intersection of ROI (valued 1) with atlas TT_Daemon (sb0):
   89.2 % overlap with Middle Occipital Gyrus, code 33
   6.7  % overlap with Middle Temporal Gyrus, code 35
   -----
   95.9 % of cluster accounted for.

Intersection of ROI (valued 1) with atlas TT_Daemon (sb1):
   19.5 % overlap with Brodmann area 37, code 113
   1.5  % overlap with Brodmann area 19, code 96
   -----
   21.0 % of cluster accounted for.

++    195 voxels in atlas-resampled mask
Intersection of ROI (valued 1) with atlas CA_N27_MPM (sb0):
   1.5  % overlap with hOC5 (V5 / MT+), code 110
   -----
   1.5  % of cluster accounted for.

++    195 voxels in atlas-resampled mask
Intersection of ROI (valued 1) with atlas CA_N27_ML (sb0):
   61.0 % overlap with Right Middle Occipital Gyrus, code 52
   20.0 % overlap with Right Middle Temporal Gyrus, code 86
   -----
   81.0 % of cluster accounted for.
```

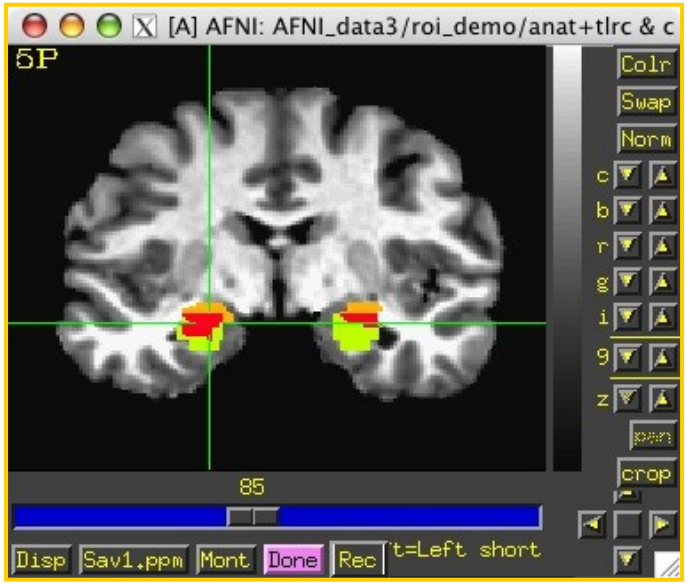
- Above examples use a feature of the AFNI software package to create datasets from atlas regions on the fly
  - Creation of a 0-or-1 mask dataset *directly* on the command line using a dataset name of the form **`Atlas_name:Hemisphere:Region_name`**
  - Using this feature, you don't have to create the mask using the **`whereami`** program and then use it later— you can create it and use it at the same time

- Example 9 from **`3dcalc -help`**:

  Compare the left and right amygdala between the Talairach atlas, and the CA_N27_ML atlas. The result will be **1** if a voxel is marked as amygdala in the **TT_Daemon** only, **2** if it is marked as amygdala in the **CA_N27_ML** only, and **3** where they overlap.

```
3dcalc  -a 'TT_Daemon::amygdala' \
  -b 'CA_N27_ML::amygdala'       \
       -expr 'step(a)+2*step(b)'
       -prefix compare.maps
```

Note: **`compare.maps+tlrc`** displays the TT amygdala (value=1) in green, the N27 amygdala (value=2) in orange, and the overlap between the two atlases (value=3) in red.



- For more information about **`3dcalc`** , see the AFNI Utilities presentation

Back to the "Original" Standard Space to Native Subject Space

Useful for putting atlas regions into the native space

Affine transformations only (@auto_tlrc) – 3 ways to "inverse talairach":

3dAllineate

```
cat_matvec -ONELINE anat+tlrc::WARP_DATA > tlrc.aff12.1D
3dAllineate -1Dmatrix_apply tlrc.aff12.1D -prefix invtlrc3dAl+orig \
   -source anat+tlrc -master anat+orig
```

3dWarp

```
cat_matvec anat+tlrc::WARP_DATA > tlrc.1D
3dWarp -matvec_out2in tlrc.1D -prefix invtlrc_3dWarp+orig \
   -gridset anat+orig anat+tlrc
3drefit -view orig invtlrc_3dWarp+tlrc.
```

3dfractionize - slow but useful voting option for multiple ROIs
and manual Talairach transformations

```
3dfractionize -input anat+tlrc -warp anat+tlrc –preserve \
   -prefix invtlrc_3dfrac -template anat+orig
```

Back to the "Original" 2
Standard Space to Native Subject Space

Nonlinear and Affine transformation combinations – multi-ways to "inverse talairach":

```
# getting data to a standard space with @auto_tlrc and auto_warp.py
# affinely align to template  with @auto_tlrc
@auto_tlrc -base TT_N27+tlrc -input strip_shift+orig. -no_ss \
   -init_xform AUTO_CENTER
# nonlinearly align to template
auto_warp.py -skip_affine -base TT_N27+tlrc -input strip_shift+tlrc

# 3dNwarpApply
cat_matvec -ONELINE "strip_shift+tlrc::WARP_DATA" > at_shift.1D
# one step concatenate and apply
3dNwarpApply -prefix tw3    \
    -nwarp 'at_shift.1D INV(awpy/anat.un.aff.qw_WARP.nii)'   \
    -source awpy/strip_shift.aw.nii  \
    -master strip_shift+orig.
```

Back to the "Original" 2b
Standard Space to Native Subject Space

Nonlinear and Affine transformation combinations –

3dNwarpCat

```
3dNwarpCat -prefix anat_total_WARPINV2 \
    -warp2 'INV(anat_qw9_WARP+tlrc)' -warp1 'at.1D'
3dNwarpApply -prefix anat_backtoorig2    \
    -nwarp anat_total_WARPINV2+tlrc.     \
    -source anat_qw9+tlrc -master anat+orig
```

Back to the "Original" 2c
Standard Space to Native Subject Space

sswarper2 or @SSwarper warps

```
# move native to standard space
# for ROIs only, use NN interpolation
# (remove that line for continuous data
3dNwarpApply                                                    \
-nwarp 'anatQQ.sub007_WARP.nii anatQQ.sub007.aff12.1D' \
-prefix roi_standard_space.nii.gz                               \
-source native2.nii.gz                                          \
-interp NN                                                      \
-master  ~/abin/MNI152_2009_template_SSW.nii.gz

# move native to standard space - added -iwarp
3dNwarpApply -iwarp                                             \
-nwarp 'anatQQ.sub007_WARP.nii anatQQ.sub007.aff12.1D' \
-prefix atlas_native_space.nii.gz                               \
-source atlas.nii.gz                                            \
-interp NN                                                      \
-master myanat.sub007.nii.gz
```

Circularity – "double dipping"

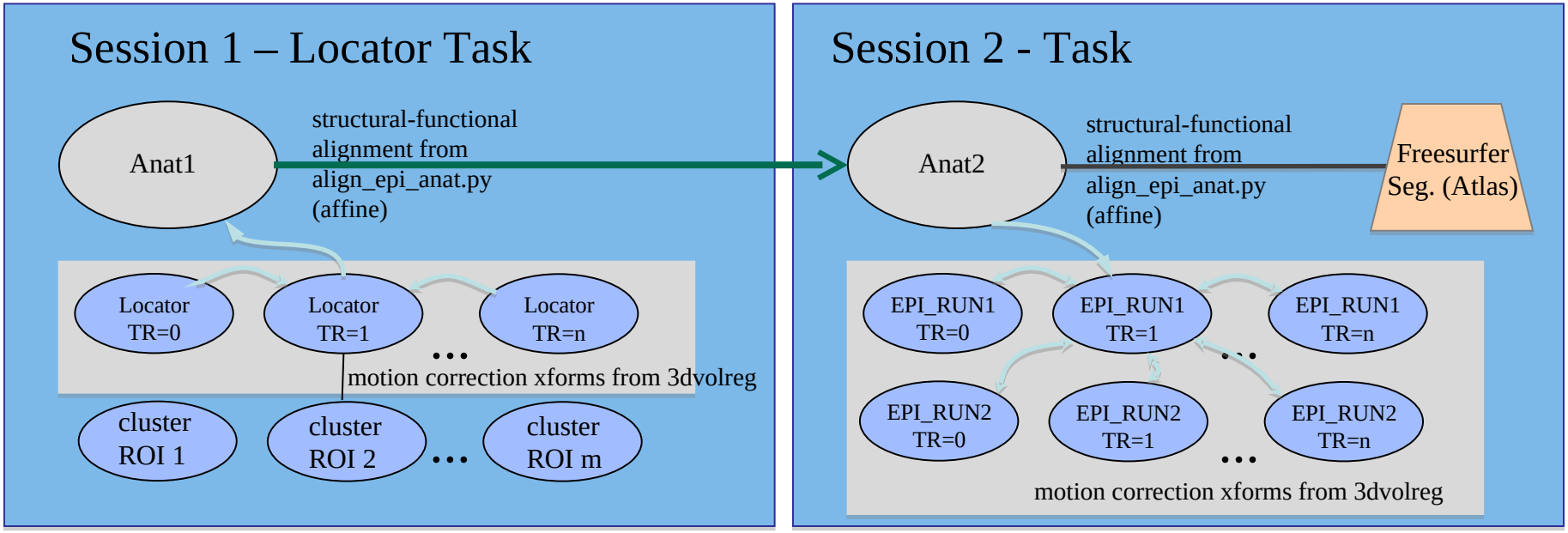
Using results from one set of data to limit the data in that same set.

FMRI example:

- Find largest differences with some threshold between groups A and B
- Create ROIs from those differences
- Show differences between ROIs for the same two groups showing that indeed A is very different from B with a p-value of 0.05. They're different because they're different.

- Solution : Use ROIs from independent data: Atlas regions, locator tasks, other subjects

ROIs from Locator session day 1 applied to session day 2 EPI



align_epi_anat a2 to e2 => a2e2.aff12.1D
align_epi_anat a1 to a2 => a1a2.aff12.1D
align_epi_anat L1 to a1 => L1a1.aff12.1D

cat_matvec a2e2.aff12.1D a1a2.aff12.1D L1a1.aff12.1D > L1e2.aff12.1D

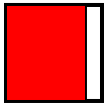
e2 -> a2 ⇨ a2 -> a1 ⇨ a1 -> L1 e2 -> L1

3dAllineate -1Dmatrix_apply L1e2.aff12.1D –final NN –prefix ROI1_e2 \
-base epi_run1+orig'[1]' –input ROI1+orig

- **3dfractionize** does this resolution conversion:

```
3dfractionize  -template low_res_dset+orig          \
               -input ROI_high_res+orig             \
               -clip 0.5 -preserve -prefix ROI_low_res
```

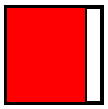
- **-template** → The destination grid you want your ROI grid to be resampled to (we're going from high to low resolution here). Our output dataset **ROI_low_res+orig** will be written at the resolution of **func+orig**
  - **(Also useful for transforming std space back to orig space with the -warp dataset)**
- **-input** → Defines the input high-resolution dataset (that needs to be converted from high resolution to low resolution)
- **-clip 0.5** → Output voxels will only get a nonzero value if they are at least 50% filled by nonzero input voxels (you decide the percentage here). E.g., when going from high to low res, keep a label a voxel as part of the ROI if it is filled with at least 50% (or more) of the voxel value. For example:



This voxel is 80% filled with the ROI value -- keep it

This voxel is 30% filled with the ROI value -- lose it

- **`-preserve`** → once it has been determined that the output voxel will be part of the ROI, preserve the original ROI value of that voxel (and not some fraction of that value). This option also allows for "voting" – determine the ROI that would most fill that voxel. For example, if our ROI mask has values of "4":



This voxel is 80% filled with the ROI value -- keep it.

Without the `-preserve` option, this voxel would be given a value of "3.2" (i.e., 80% of "4").

With `-preserve`, it is labeled as "4"

- **`3dresample`** does conversion too but you have less controls for handling partial overlaps:

```
3dresample -master low_res_dset+orig      \
           -prefix ROI_low_res                \
           -inset ROI_high_res+orig           \
           -rmode NN
```

- **`-master`**: the destination grid we want our ROI mask resampled to
- **`-prefix`**: The output from **`3dresample`** -- in this example, a low resolution ROI mask that corresponds with the voxel resolution of our master dataset
- **`-inset`**: The ROI mask dataset that is being resampled from high to low resolution
- **`-rmode NN`**: If a voxel's "neighbor" is included in the ROI mask, include the voxel in question as well