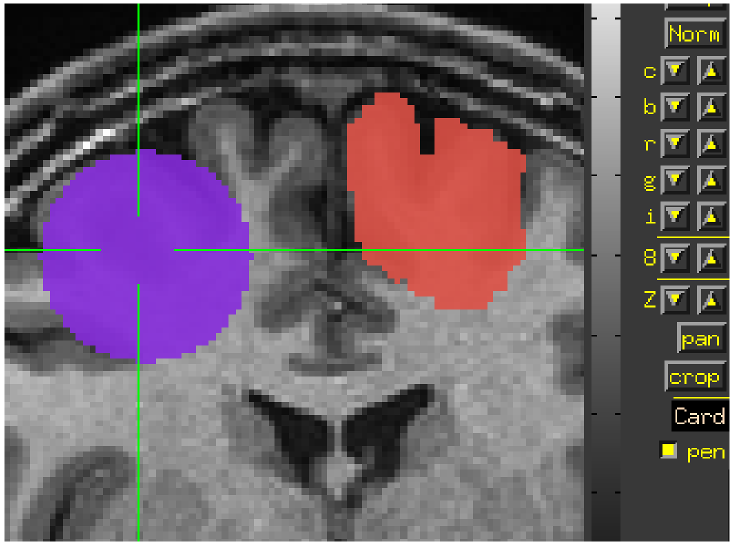
AFNI

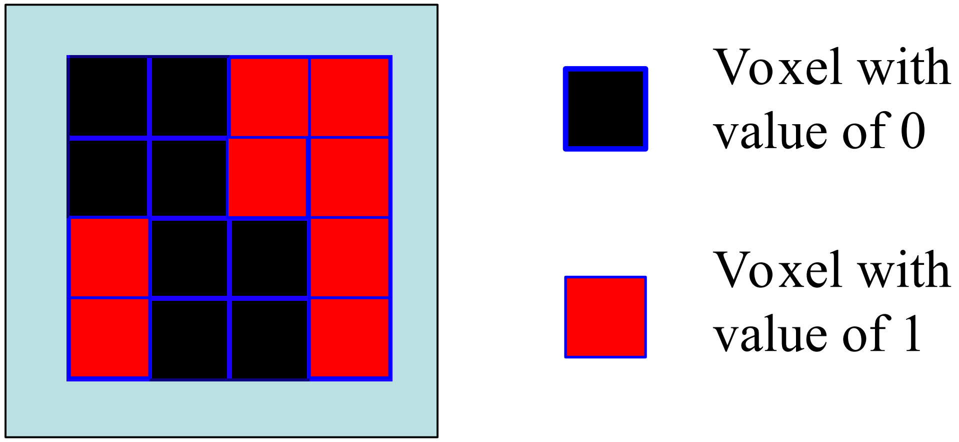
Didactics and Demonstrations

Regions of Interest (ROIs) Introduction



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

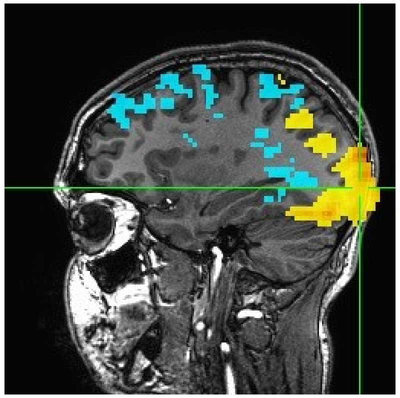


- Note any dataset can be considered as an ROI if it is non-zero in areas in which you are interested.
- Usually stored as byte integers to save disk space and memory

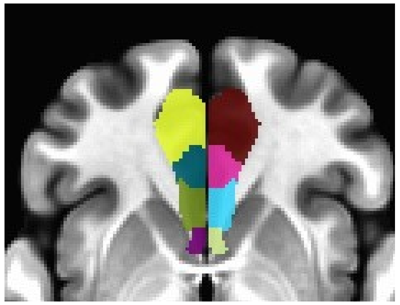
Creating Regions of Interest

- Method 1: **Draw.** Draw ROIs based on anatomical structures, then analyze functional datasets within these regions
  - Previous studies may identify particular regions
  - This method relies on 'a priori' assumptions about localization of brain function and neuroanatomy knowledge

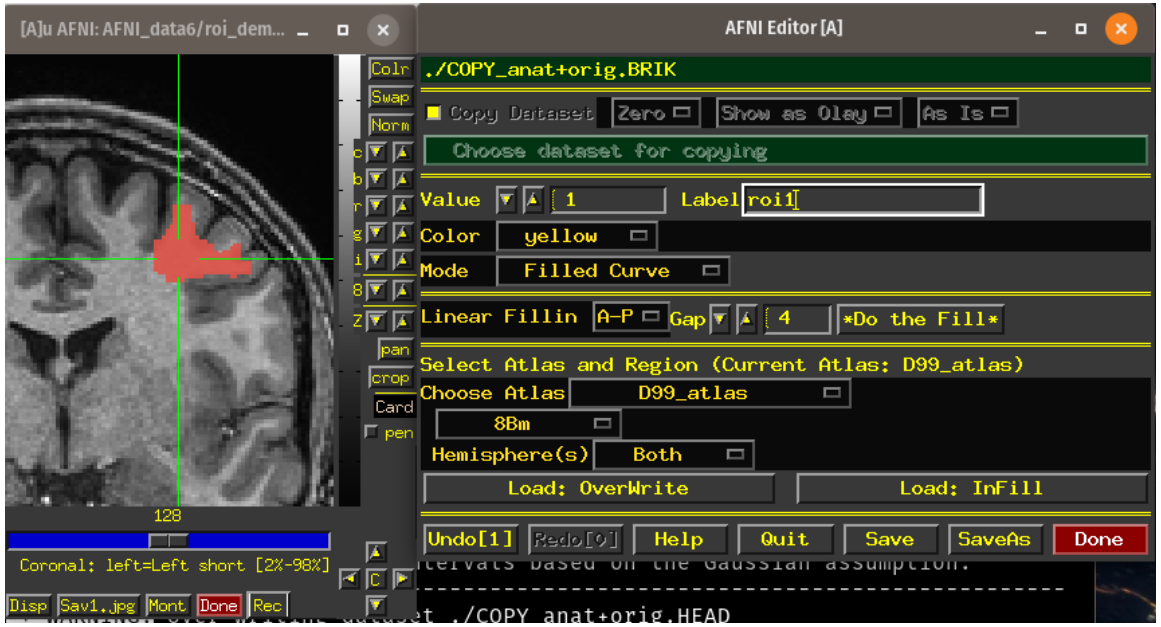
- Method 2: **Cluster.** Analyze functional dataset for *entire* brain first, then use clusters of 'activity' (neighboring voxels with values above a threshold in some statistical map)
  - Analyze the entire brain first and *then* find interesting areas of activity and do further analyses on those areas
  - Use **3dClusterize** or the AFNI graphical interface button **[Clusterize]** to find larger "blobs" or clusters of activity
  - Apply the clusters from a localizer task to apply to a separate experiment



- Method 3: **Atlas**. Use atlases to select anatomical regions
  - Use **whereami** program or symbolic notation to create masks on the command line (3dcalc, 3dresample, ...)
  - Parcellations and classifications from FreeSurfer, 3dSeg, 3dkmeans, etc.
  - Spheres around published coordinates



Drawing ROIs



./COPY_anat+orig.BRIK

Value 1 Label roi1

Color yellow

Mode Filled Curve

Linear Fillin A-P Gap 4

Choose Atlas D99_atlas

8Bm

Hemisphere(s) Both

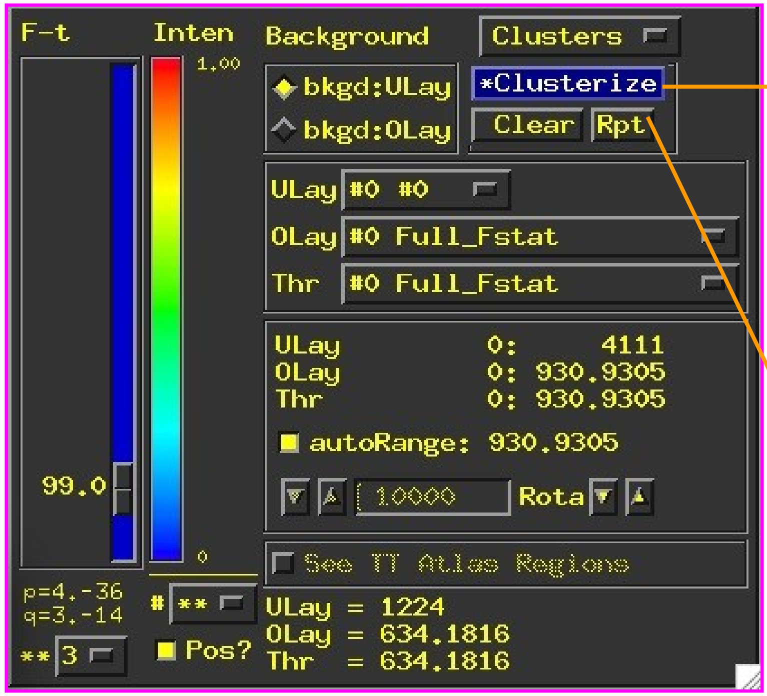
Use AFNI's ROI Draw Dataset plugin (see video)
For large projects, consider drawing tools like a stylus with a touchscreen or Wacom Cintiq.

Also other drawing tools that support NIFTI output can be used - MIPAV, Amira, ...

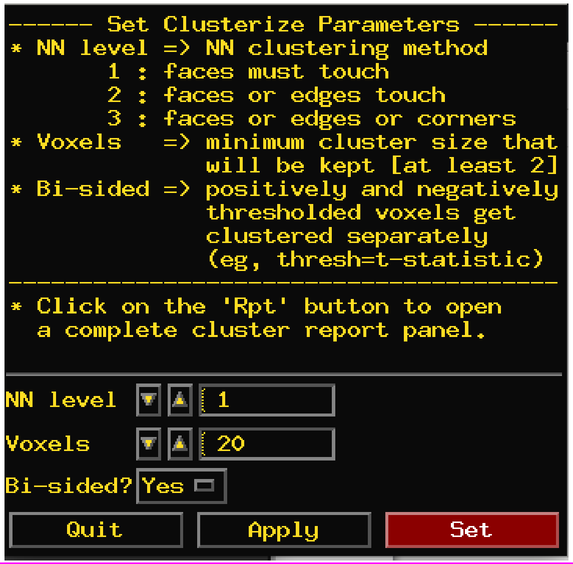
Clustering

Clusterize

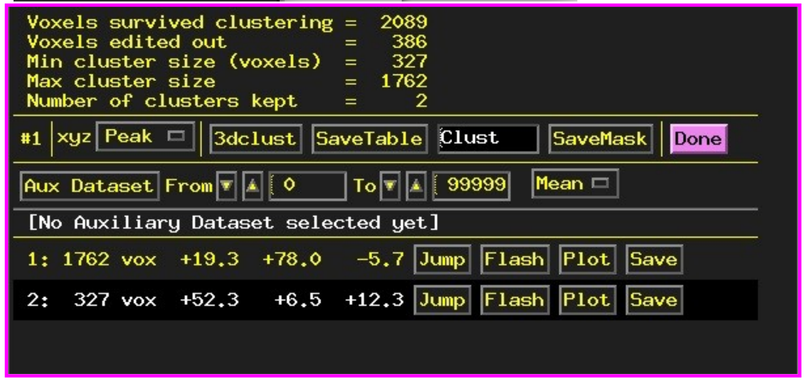
- The **Clusterize** button on the main AFNI graphical interface gives users a quick and easy way to locate clusters of activity in a functional dataset. Once the user sets the clusterize parameters, a complete cluster "report" is given, which details the number of clusters found, based on these parameters.



```
F-t      Inten   Background        Clusters
         1.00    bkgd:ULay         *Clusterize
                 bkgd:OLay         Clear  Rpt
                 ULay  #0 #0
                 OLay  #0 Full_Fstat
                 Thr   #0 Full_Fstat
                 ULay           0:      4111
                 OLay           0:  930.9305
                 Thr            0:  930.9305
                 autoRange:  930.9305
 99.0            1.0000   Rota
         0       See TT Atlas Regions
p=4,-36   # **   ULay = 1224
q=3,-14          OLay = 634.1816
** 3      Pos?   Thr  = 634.1816
```

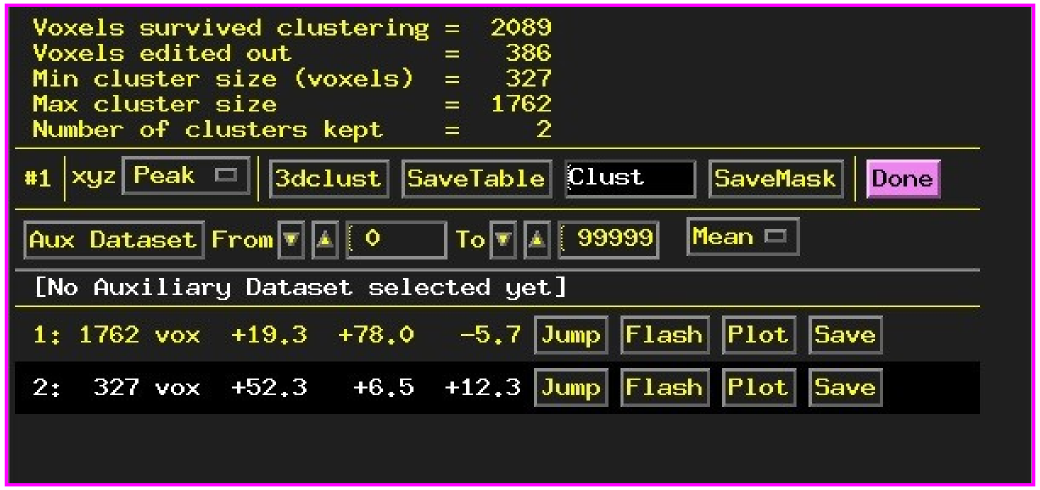


```
------ Set Clusterize Parameters ------
* NN level => NN clustering method
       1 : faces must touch
       2 : faces or edges touch
       3 : faces or edges or corners
* Voxels   => minimum cluster size that
              will be kept [at least 2]
* Bi-sided => positively and negatively
              thresholded voxels get
              clustered separately
              (eg, thresh=t-statistic)
----------------------------------------
* Click on the 'Rpt' button to open
  a complete cluster report panel.

NN level    1
Voxels      20
Bi-sided?   Yes
Quit   Apply   Set
```



```
Voxels survived clustering =  2089
Voxels edited out          =   386
Min cluster size (voxels)  =   327
Max cluster size           =  1762
Number of clusters kept    =     2

#1 xyz Peak  3dclust  SaveTable  Clust  SaveMask  Done
Aux Dataset  From 0  To 99999  Mean
[No Auxiliary Dataset selected yet]
1: 1762 vox  +19.3  +78.0   -5.7  Jump Flash Plot Save
2:  327 vox  +52.3   +6.5  +12.3  Jump Flash Plot Save
```

Clusterize Features



```
Voxels survived clustering =  2089
Voxels edited out          =   386
Min cluster size (voxels)  =   327
Max cluster size           =  1762
Number of clusters kept    =     2
```

```
[No Auxiliary Dataset selected yet]
1: 1762 vox  +19.3  +78.0   -5.7
2:  327 vox  +52.3   +6.5  +12.3
```

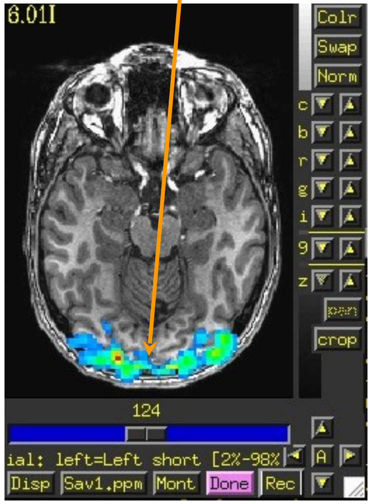
Jump: sets the crosshairs to the designated xyz coordinates (default is the *peak* of the ROI cluster)

Flash: flashes the cluster voxels in the image viewer

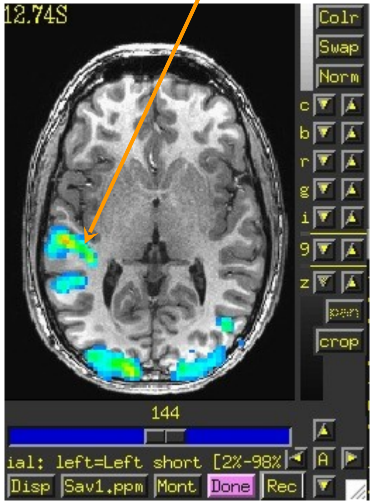
SaveMask: Click on this button to write clusters to a mask dataset called **`Clust_mask+orig`**

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.

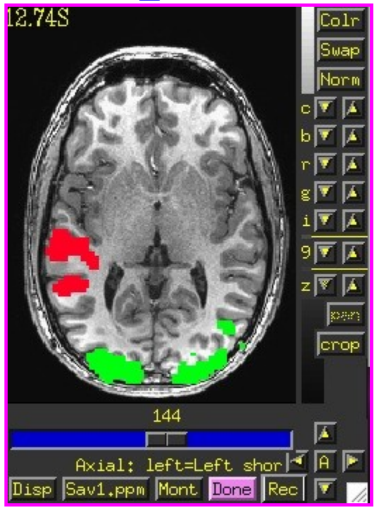
Cluster #1



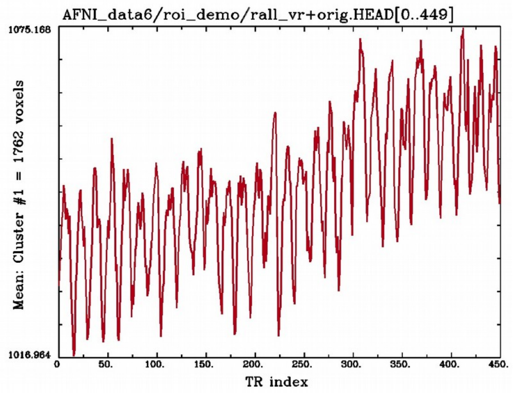
Cluster #2



Clust_mask+orig



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

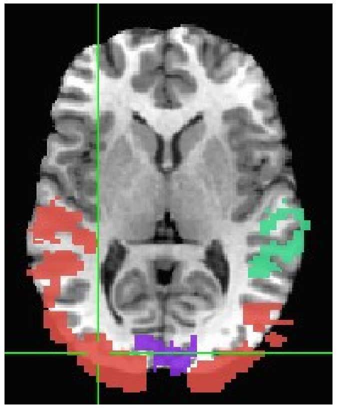


3dClusterize

The program **3dClusterize** looks for clusters – groups of voxels together that meet some threshold
Example:

```
3dClusterize -clust_nvox 200 -bisided -2.0 2.0 -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
```

(similar alternatives in AFNI 3dclust and 3dmerge)

Creating Regions of Interest from Atlases - whereami

`whereami` can extract ROIs from atlases using symbolic notation

```
whereami -mask_atlas_region TT_Daemon:left:amy
```

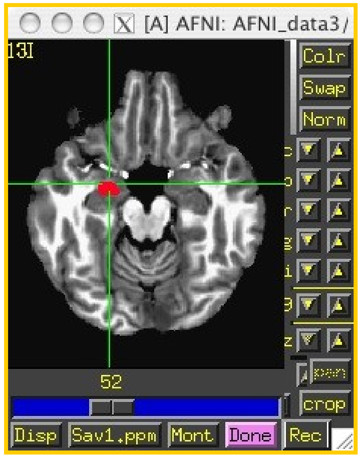
Use the Talairach-Tourneaux atlas (`TT_Daemon`) to create an ROI of the left amygdala.

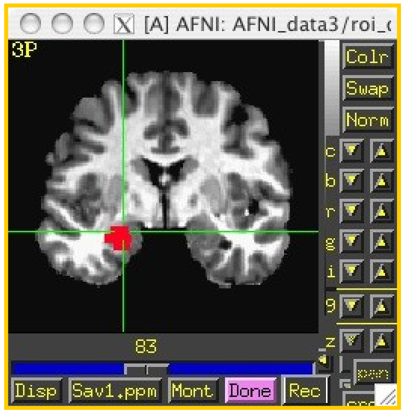
Find available atlases with whereami -show_atlases

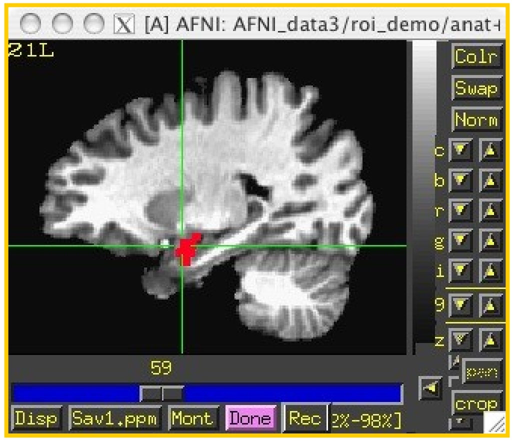
Find available regions with whereami -show_atlas_code

Another example with approximate name and prefix:

```
whereami -mask_atlas_region MNI_Glasser_HCP_v1.0::L_front_opercular \
          -prefix mniglass_lfront_oper
```







TT_Daemon.amy.l.tlrc

You can also specify atlas-based ROI masks directly like this:

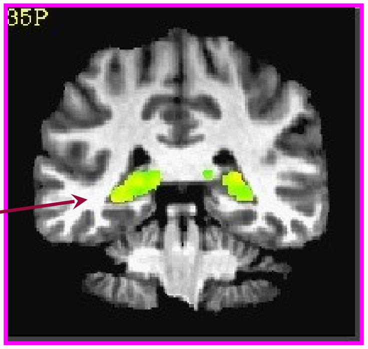
```
3dcalc -a ~/abin/MNI_Glasser_HCP_v1.0.nii.gz'<169>' -expr a \
 -prefix left_front_operc
```

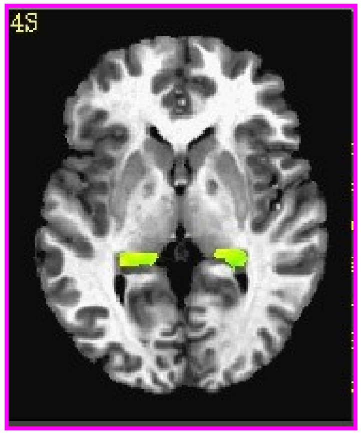
or this way is preferable:

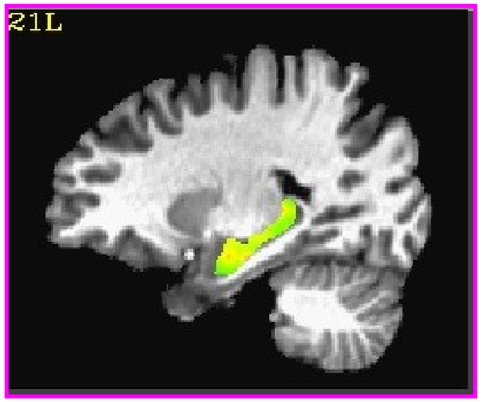
```
3dcalc -a   \
 ~/abin/MNI_Glasser_HCP_v1.0.nii.gz'<L_Area_Frontal_Opercular>' \
 -expr a -prefix left_front_operc2
```

Another example:

```
3dcalc  -prefix nice_roi                              \
        -a 'CA_N27_ML::hippo' -b 'func_FullF_1mm+tlrc'    \
        -expr '(step(a)*b)'
```







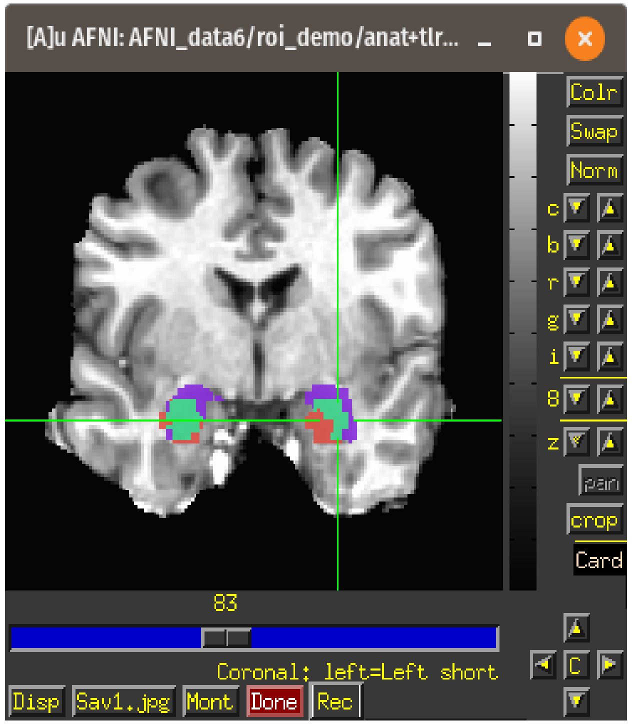
ULay: **anat+tlrc**
OLay: **nice_roi+tlrc**

F-stat voxels in func dataset that fall within the TT left hippocampus ROI

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

3drefit -cmap INT_CMAP compare.maps+tlrc
```

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



Getting around with spheres

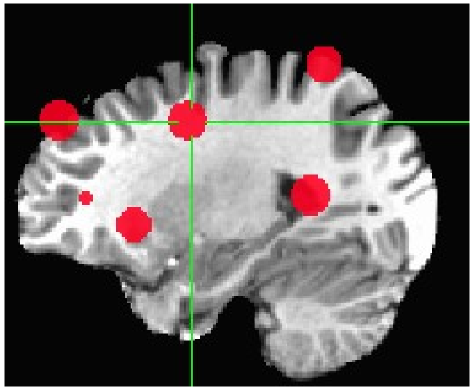
Another way to use cluster results –
make spheres from the cluster peaks or centers of mass

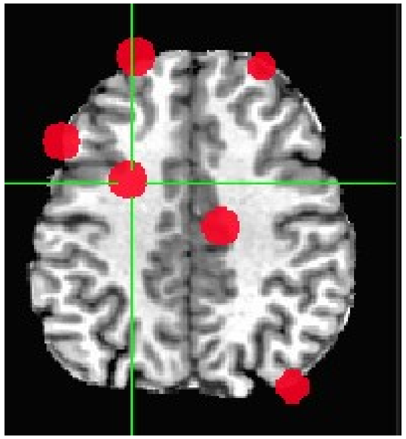
Try this:

```
adwarp -apar anat+tlrc -dpar func_slim+orig –dxyz 3
```

In afni GUI, select Underlay: anat, Overlay: func_slim
Switch view to Talairach
Overlay to vrel_coef, Threshold to vrel_tstat
Clusterize, and Save Tabl

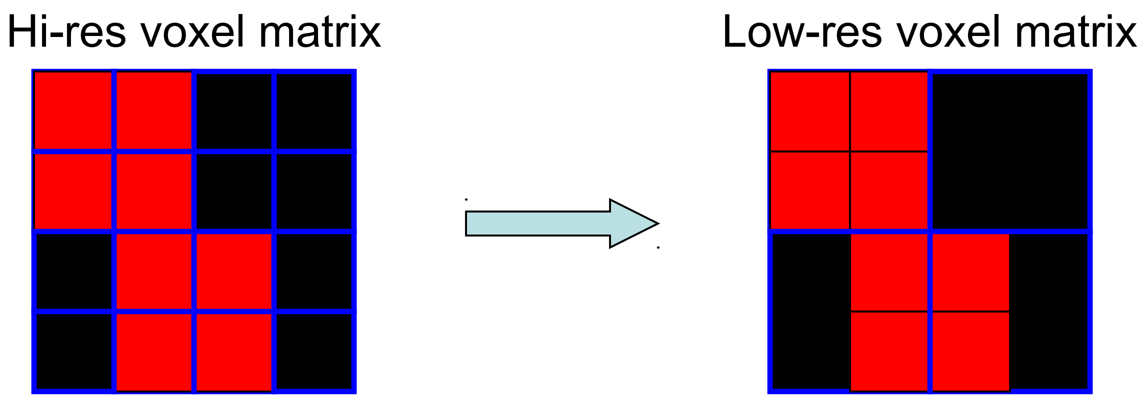
```
1dcat Clust_table.1D'[4..6]' > Clust_PeakXYZ.1D
3dresample -orient RAI -prefix func_slim_RAI  \
     -inset func_slim+tlrc
3dUndump -srad 7.5 -master func_slim_RAI+tlrc   \
     -prefix clust_spheres -xyz Clust_PeakXYZ.1D
```





Using ROIs - Resampling

- ROIs are typically applied to functional datasets – low resolution
- Draw on anatomy or use atlas regions - high resolution



Each voxel inside the original ROI has a nonzero value

When the resolution is changed, what do you do with voxels that are only partially filled?

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

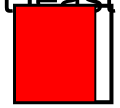
```
3dfractionize  -template low_res_dset+orig         \
               -input ROI_high_res+orig            \
               -clip 0.5 -preserve -prefix ROI_low_res
```

★ `-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`

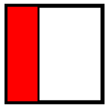
➥ **(Also useful for transforming std space back to orig space with the -warp dataset)**

 `-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:

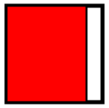


This voxel is 80% filled with the ROI value -- keep it



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:

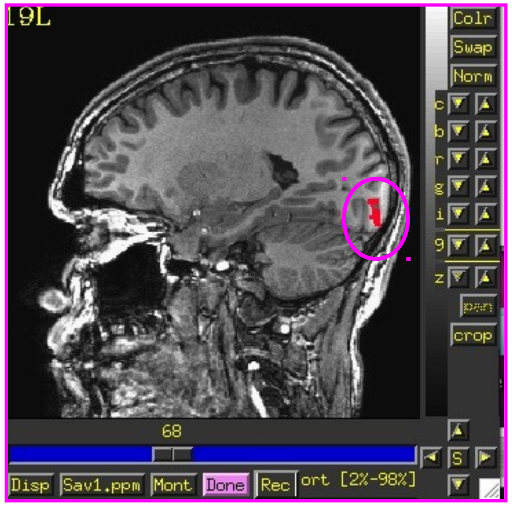
```
3dresample -master low_res_dset+orig      \
                -prefix ROI_low_res             \
                -inset ROI_high_res+orig        \
                -rmode NN
```

- **-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

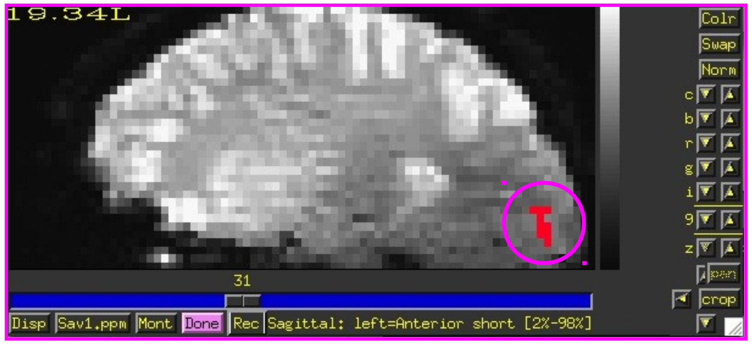
Let's do a class example of **3dresample**:

```
cd AFNI_data6/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).



Before, overlay ROI is:
anat_roi+orig
0.9x0.9x1.2 mm voxel grid



After, overlay ROI is:
anat_roi_resam+orig
2.75x2.75x3.0 voxel grid

Moving ROIs - 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
```

Moving ROIS - Back to the "Original" 2
Standard Space to Native Subject Space

Nonlinear and Affine transformation combinations – 3 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.   \
    -ainterp NN              <====== Nearest neighbor interpolation for ROIs
```

Moving ROIs - 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

# 3dNwarpCalc
3dNwarpCalc "&readwarp(anat_qw9_WARP+tlrc.)" "&invert" \
     "&read4x4(at_matrix.1D)" "&compose" "&write(combo_warp3)"
3dNwarpApply -prefix anat_backtoorig5 -nwarp combo_warp3+tlrc. \
    -source anat_qw9+tlrc -master anat+orig
```

That's All for Now