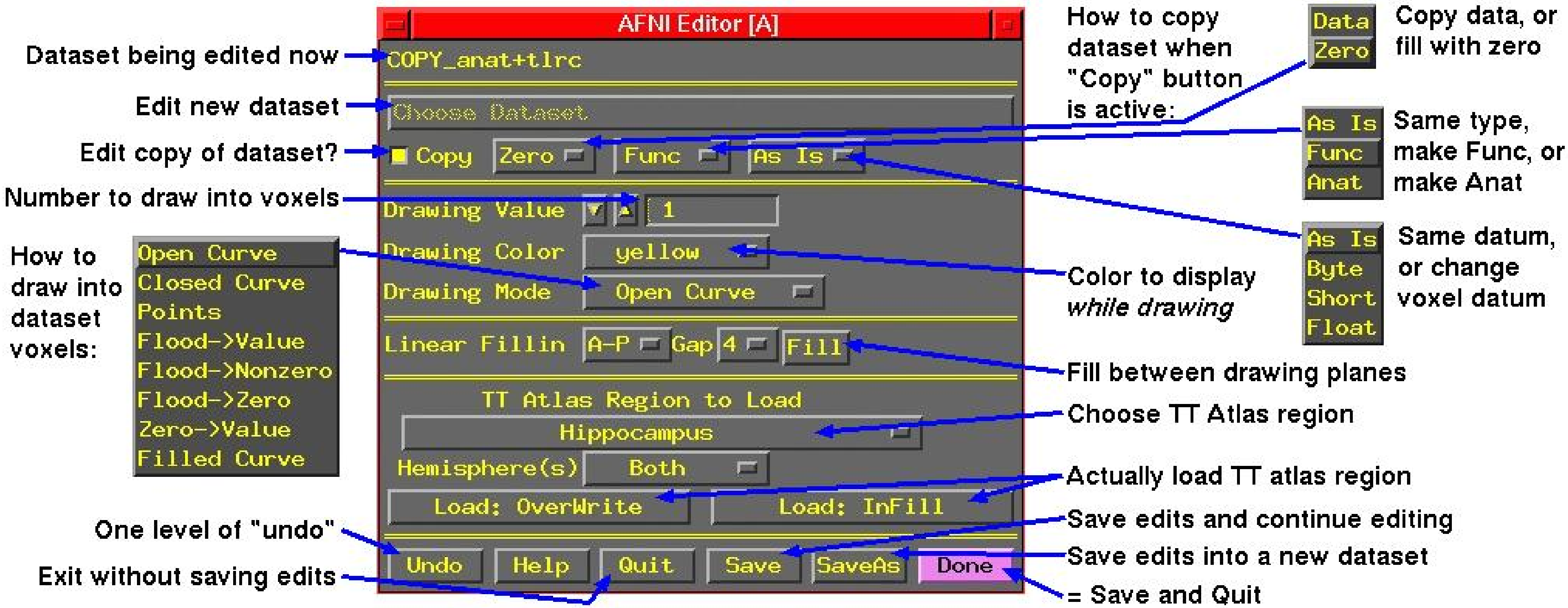
Region of Interest Drawing and Usage in AFNI

- Goal: manually select Regions of Interest (ROIs) based on anatomical structures, then analyze functional datasets within these regions
  - ◇ Alternative procedure for selection of ROIs: geometrically connected clusters of 'activity' (supra-threshold voxels in some functional statistical map)
  - ◇ Even with this method, you might want to manually adjust the ROIs
- ROIs are stored as regular AFNI datasets; it is only the user (you) that decides whether a particular dataset is a ROI mask
  - ◇ Nonzero voxels are "in" the mask; zero voxels are "outside"
  - ◇ It is best to store a ROI mask as a functional dataset, so you can overlay it in color on an anatomical dataset
- Outline of procedure:
  - ◇ On the main AFNI control panel, set the anatomical underlay dataset (with `Switch Anatomy`) to be what you want to draw on—usually a SPGR or MP-RAGE type of dataset
  - ◇ Start the `Draw Dataset` plugin (also called the Editor)

  ↪ `Define Datamode→Plugins→Draw Dataset`

- ◇ Create an all zero functional dataset the size of the anatomical dataset on top of which you are going to draw
  - ↪ Creating an all zero dataset can be done with the Editor plugin, or with the `Copy Dataset` plugin
  - ↪ You could instead edit a ROI dataset that you had already created before
  - ↪ It is required that the dataset being edited and the dataset that is the anatomical underlay have the same geometry (voxel size, etc.), since drawing/editing is done on a voxel-by-voxel basis
- ◇ Set the functional overlay dataset (with `Switch Function`) to be the dataset you are editing, and turn the functional overlay on (with `See Function`)
- ◇ Draw the ROIs into the functional dataset, eventually `Save`-ing the results
- ◇ Convert the anatomical-resolution ROI dataset into a dataset at the resolution of the functional datasets you want to analyze with the ROI
  - ↪ The ROI and functional datasets may already be at the same resolution, if you are operating in `+tlrc` coordinates
  - ↪ Resolution conversion of masks is done with program `3dfractionize`
- ◇ Use programs `3dmaskave`, `3dmaskdump`, and `3dROIstats` to extract ROI-based information about functional datasets
  - ↪ Also can use `ROI Average` plugin to extract interactively the average of a dataset over a ROI

Using the Drawing Plugin



- Critical things to remember:
  - ◇ You should have `See Function` turned on, and be viewing the same function 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

- Step 1: Load a dataset to be edited
  - ◇ `Choose Dataset` button gives you a list of datasets that
    - (a) Actually have brick data with only one sub-brick; and
    - (b) Are at the same voxel dimension, grid size, etc., as the current anatomical underlay dataset
  - ◇ When you are starting, you probably don't want to edit an existing dataset
  - ◇ To get an all zero copy of the anatomical underlay, click the `Copy` button on, and set the controls to its right to `Zero`, `Func`, `As Is`
    1. `Data` would make a copy of the dataset with the actual data values
    2. `Func` makes the copy be a functional dataset, no matter what the original is; `Anat` would make the copy be an anatomical dataset; `As Is` would leave the copy in whatever mode the original was
    3. `As Is` means to keep the voxel values in the copy as the same type as in the original; you can also change the voxels values to be stored as
       - ▷ `Byte`s (integer values: 0..255) — 1 byte each
       - ▷ `Short`s (integer values: $-32767..32767$) — 2 bytes each
       - ▷ `Float`s (fractional values) — 4 bytes each
       - ▷ Bytes and Shorts make the most sense for ROI masks, where you are essentially attaching labels to voxels
  - ◇ Then `Choose Dataset`, pick the dataset you want a copy of (e.g., the anatomical underlay), and then `Set`

- Step 2: Drawing the ROI (or ROIs)
  - ◇ Choose the value to draw into the dataset
    - ↪ If you are drawing only 1 ROI, then the default value of 1 is good
    - ↪ If you are drawing multiple ROIs, then you should choose a different numerical value for each so that they can be distinguished later
      - ▷ And write down which number corresponds to which ROI!
  - ◇ Choose the drawing color
    - ↪ This is the color that is shown while you are drawing
    - ↪ Color choice is only important to give a good contrast with underlay
    - ↪ 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 Function` control panel
  - ◇ Choose the drawing mode:
    - ↪ `Open Curve` ⇒
      Drawing action produces a continuous open-ended curve
    - ↪ `Closed Curve` ⇒
      Drawing action produces a continuous closed-ended curve (not filled inside)
    - ↪ `Points` ⇒
      Only points actually drawn over are filled (used to "touch up" an ROI)

↪ `Flood->Value` ⇒
Flood fill outwards from drawn point, stopping when the flood hits the current drawing value (used to fill a closed curve)

↪ `Flood->Nonzero` ⇒
Flood fill outwards from drawn point, stopping when the flood hits any nonzero voxel (used to fill between regions)

↪ `Zero->Value` ⇒
Flood fill with zero—instead of the drawing value—stopping when the flood hits the current drawing value (I forget why this is useful)

↪ `Filled Curve` ⇒
Drawing action produces a continuous closed-ended curve (filled inside)

◇ Actually draw something

↪ Drawing is done with mouse Button 2 ("middle" button) in a 2D slice image

↪ 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 Function` control panel

◇ Undo button will let you take back the last drawing action

↪ Only one level of undo is available: Undo then Undo will take you back to where you just finished drawing

◇ You can only draw on one 2D slice image at a time

↪ If you draw on a montage display, only screen pixels overlaying the first image you Button 2 click in will count

↪ While drawing, if you cross over between sub-images in the montage, unexpected effects will result

▷ But there is always Undo to the rescue!

- Step 3: Save your results

◇ Save will write the current dataset values to disk (overwriting any `.BRIK` file), and let you continue editing the same dataset

↪ You could also then choose another dataset to edit

◇ SaveAs will let you write the current dataset to disk under a new name, creating a new dataset, then continue editing the new dataset

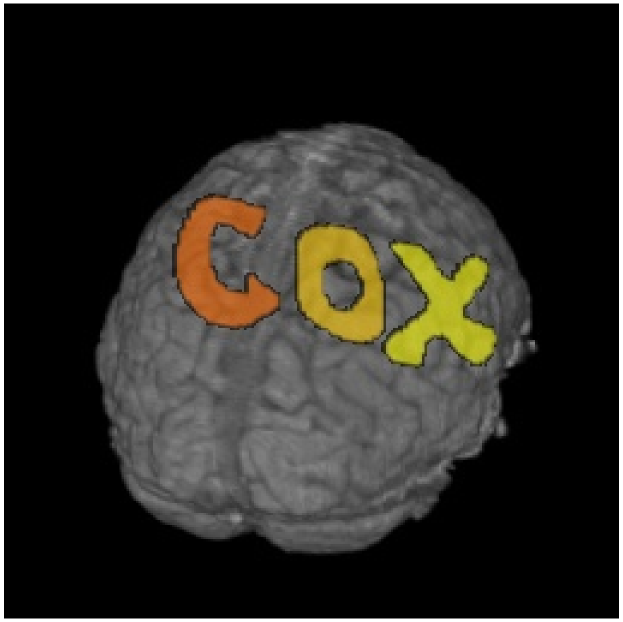
◇ Quit exits editing and closes the plugin window, without saving to disk any changes since the last Save

↪ 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 n^{th} slice, then you want to set `Gap` to $n-1$
    - ↪ Line segments of voxels in the fillin direction that have the 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
    - ↪ Fillin cannot be undone — be careful!
    - ↪ This operation can also be done with program `3dRowFillin`, which creates a new dataset (and thus provides a way to undo: the `rm` command)
  - ◇ `TT Atlas Region to Load` lets you load regions from the Talaraich Daemon database into the dataset voxels
    - ↪ Requires that you be drawing in `+tlrc` coordinates, or at least have a transformation from `+orig`→`+tlrc` computed in the current directory
    - ↪ 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
    - ↪ You probably want to edit the results manually to fit the subject

- Drawing and 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 dataset 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 `3dIntracranial`)
    - ↪ Choose the overlay dataset to be the dataset you are editing
    - ↪ Turn on `See Overlay`
    - ↪ Set `Color Opacity` to `ShowThru` (or `ST+Dcue`)
    - ↪ 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 s per frame)



Things to Do with ROI Datasets

- ROIs are used on a voxel-by-voxel basis to select parts of datasets
- If you draw at the anatomical resolution and want to use the ROI dataset at the functional resolution, you probably have to convert the high-resolution ROI dataset to a low-resolution dataset (unless you are working in `+tlrc` coordinates)
- Program `3dfractionize` does this resolution conversion:

```
3dfractionize -template func+orig
              -input ROI_hires+orig
              -clip 0.5 -preserve -prefix ROI_lores
```

  - ◇ `-template func+orig` ⇒ New dataset is written at resolution of `func+orig`
  - ◇ `-input ROI_hires+orig` ⇒ Defines the input high-resolution dataset
  - ◇ `-clip 0.5` ⇒ Output voxels will only get a nonzero value if they are at least 50% filled by nonzero input voxels
  - ◇ `-preserve` ⇒ Output voxels will preserve the input dataset's values (without this option, the output value of a voxel is the fraction of the voxel that is filled with nonzero input voxels)
- One large output voxel may be partially or fully covered by several nonzero smaller input voxels; `3dfractionize` computes the overlap fraction for each output voxel and uses that fraction to decide what value to put in the voxel

- N.B.: A ROI is defined by the values stored in the voxels of the 'mask' dataset
  - ◇ Contiguity of voxels has no meaning to the ROI software described below
  - ◇ Two voxels are in the same ROI if they have the same value in the mask dataset
- `3dmaskave`: program to compute the average of voxels from a dataset, with the voxels selected from a mask dataset (interactive version: `ROI Average` plugin)
  - ◇ Example:

    ```
    3dmaskave -mask ROI_fred+tlrc -mrange 1 3
                'func_fred+tlrc[2,5,7]'
    ```

    will print out 1 line for each of the 3 selected sub-bricks in the input dataset `func_fred+tlrc`:

    ```
    44.287 [137 voxels]
    27.021 [137 voxels]
    -33.22 [137 voxels]
    ```

    Each line is the average of the sub-brick values over the selected voxels: all voxels whose value in the mask dataset are between 1 and 3 (inclusive), as specified by `-mrange 1 3`
  - ◇ The `-q` option suppresses the voxel count printout; it is useful if you want to compute the average time series over a ROI and save it to a 1D file for later use in AFNI:

    ```
    3dmaskave -mask ROI_fred+tlrc -mrange 1 1
                epi01_fred+tlrc > epi01_fred_ROI_1.1D
    ```

- 3dmaskdump: program to dump out all voxels in a dataset (or datasets) that match some values given in mask dataset
  - ◇ Mask dataset is usually a ROI dataset
  - ◇ Example:

```
3dmaskdump -noijk -mask ROI_fred+tlrc
           -mrange 1 1 'func_fred+tlrc[2,5,7]'
```

    would produce output like so for each voxel whose value in ROI_fred+tlrc was 1:

```
12 37 42
27 0 321
4 4 444
```

    That is, each voxel that passes the -mask and -mrange tests (i.e., is nonzero in the mask and has its value in the range of numbers given by -mrange) gets one line printed with the values that are in sub-bricks #2, #5, and #7 from dataset func_fred+tlrc at that voxel.
  - ◇ More than one dataset can be given at the end of the command line
  - ◇ Main application is to dump out the data or functional activation values that match a ROI so they can be processed in some other program (e.g., Excel)
  - ◇ If -noijk option is omitted, each output line starts with ijk-indexes of the voxel
  - ↪ Program 3dUndump can be used to create a dataset from a text file with ijk-indexes and dataset values

- 3dROIstats: program to compute average of voxels from a dataset, using multiple regions selected by a single ROI dataset
  - ◇ Averaging is done over each region defined by a distinct value in the ROI dataset
  - ◇ Example:

    ```
    3dROIstats -mask ROI_fred+tlrc 'func_fred+tlrc'
    ```

    produces output that looks like this:

    ```
    File             Sub-brick   Mean_1  Mean_2  Mean_3
    func_fred+tlrc   0           33.3    44.4    55.5
    func_fred+tlrc   1           22.2    66.6    -21.2
    func_fred+tlrc   2           2.3     9.7     666.666
    ```

    The Mean_1 column is the average over the ROI whose mask value is 1, Mean_2 over ROI with mask value 2, etc.
  - ◇ Very useful if you create ROI masks for a number of subjects, using the same number codes for the same anatomical regions (e.g., 1=hippocampus, 2=amygdala, 3=superior temporal gyrus ...).
  - ◇ You can load the output of 3dROIstats into a spreadsheet for further analysis (e.g., statistics with other subjects' data).

Creating ROI Datasets from Activation Maps

- The program 3dmerge can find contiguous supra-threshold voxel clusters in an activation map and then convert each cluster into a ROI with a separate data value
  - ◇ These ROIs can then be used as starting points for some analysis
- Example:

```
3dmerge -1clust_order 1 500
         -1tindex 1 -1thresh 0.4
         -prefix ROI_func func+tlrc
```

  Does the following:
  - ◇ Thresholds the dataset on sub-brick #1 at value 0.4
  - ◇ Clusters together all the surviving nonzero voxels using a contiguity test of 1 mm and keeping only clusters at least 500 mm^3 in volume
  - ◇ All voxels in the largest cluster are assigned value 1, in the second largest assigned 2, etc., and the result is written to disk in dataset ROI_func+tlrc
- You can now use this dataset as a mask, edit it with the drawing plugin, . . .