MCW *AFNI* — User Manual

Medical College of Wisconsin
Analysis of Functional NeuroImages
Version 2.00, December 1996

Robert W. Cox, Ph.D.

Biophysics Research Institute
Medical College of Wisconsin
8701 Watertown Plank Road
Milwaukee, WI 53226

Summary: MCW *AFNI* displays three dimensional functional neuroimages overlaid onto anatomical reference scans. Data may be transformed to the Talairach-Tournoux (stereotaxic) proportional grid system. Images may be generated in each of the three cardinal orthogonal planes, and switched among multiple functional/anatomical data volumes. Auxiliary programs are provided to manipulate and combine 3D image sets. Time-dependent 3D image sets may also be stored, displayed, and manipulated. An application programmer interface is provided to allow users to extend the functionality of *AFNI* (via 'plugins').

Document: This manual describes the use of the *AFNI* program, version 2.00. Earlier versions (1.xx) are superseded by this version; bugs in the earlier programs will no longer be fixed. Separate manuals are provided for the auxiliary programs, and for the programming of plugins.

Distribution and Registration: MCW *AFNI* is available free for research purposes, but users must register with the Medical College of Wisconsin. For information, send an e-mail request to the author at `rwcox@mcw.edu`, write to the address above, or see the final page of this manual.

Acknowledgement: This work was developed with internal MCW funds and was also partly supported by the United States NIH through grants MH51358 and NS34798.

Contents

1 Introduction

AFNI is an interactive program for viewing the results of 3D functional neuroimaging. It can overlay the (usually) low resolution results of functional brain scans onto higher resolution structural volume data sets. By marking fiducial points, you may transform the data to the proportional grid (stereotaxic coordinates) of Talairach and Tournoux [1]. Time-dependent 3D volume data sets can also be created and viewed. Auxiliary programs are provided for combining and editing 3D and 3D+time functional data sets.

With this new version of MCW *AFNI*, my intention is to add many new analytical capabilities to the software system. Some of these are found in the auxiliary programs. Others are found in the *AFNI* program itself; in particular, the interactive ability to compute functional activation using the correlation method [2]. The new plugins capability offers C-literate users the ability to integrate their own analytical tools into *AFNI*. It is my hope that other sites will develop *AFNI* plugins and share them with the FMRI community.

⋆⋆⋆ ⟶ **Whats New?**

This document has been extensively rewritten from the version 1.0x manuals. Major new facilities in *AFNI* are outlined in the next section.

⋆⋆⋆ ⟶ **Computer requisites**

The program runs on Unix workstations, using the X11 windowing system and the Motif 1.2 toolkit for its graphical interface. Minimum system requirements to use *AFNI* are 32 MB RAM (64-128 MB will work much better), X11R5 with Motif 1.2, an ANSI C compiler, and enough disk space to hold the 3D data volumes required. *AFNI* is designed to work with 8- or 12-bit PseudoColor X11 visuals. A visual of this type must be the default visual on the system used to display *AFNI*. The program has been tested on the following systems:

- SGI Indigo workstations (R4400 and R10000 CPUs) running IRIX 5.3 and IRIX 6.2.
- HP 9000/735 workstations running HP-UX 9.05.
- Intel-based Linux 1.2.13 systems.
- Sun SPARCstations running Solaris 2.5.

It has not been tested on other platforms. It *will not* work with X11R4 or Motif 1.1 (in particular, it does not work under SGI IRIX 4.x). There are no plans to port it to radically different platforms such as Microsoft Windows or the Apple Macintosh.

AFNI is a powerful program for display and manipulation of functional neuroimages, with many options. Study of this manual and experience with the program are both required to make full use of *AFNI*. Some experience with using X11 is also very useful, and will be assumed in this manual.

Nomenclature

In general, I refer to "MCW *AFNI*" as the whole software package. The particular program "*AFNI*" is at the heart of the package, and is the subject of this manual. The package includes a number of other programs and the programming interface for creating plugins, both of which are documented elsewhere.

If you find MCW *AFNI* useful and wish to refer to it in a publication, the appropriate citation is [3].

2 What's New?

1. **Atomic Datum Types**
   Previously, the data stored in an *AFNI* dataset had to be 16 bit signed integers (`short` datum). The new dataset format allows for 8 bit unsigned integers (`byte` datum), 32 bit floating points numbers (`float` datum), and 64 bit complex numbers (`complex` datum). The purpose of the `byte` datum is to allow for compact storage of datasets where the 0–255 range has sufficient precision. The purpose of the `float` datum is to allow for more natural storage of statistical quantities, without the need for rescaling to the limited range of values allowed with `short`s. (But see item 2 below.) The purpose of the `complex` datum is to eventually allow for unprocessed reconstructed images to be directly imported into *AFNI*.

2. **Scale Factors**
   Each 3D sub-brick in a dataset can now have a floating point scale factor attached to it. The purpose of this is to allow data to be stored in the more compact `byte` or `short` formats, but always to be automatically scaled to the correct units when accessed by an MCW *AFNI* program. For example, correlation coefficients can still be stored as `short`s $\in[-10000, 10000]$, but will be scaled by 0.0001 to their true range $\in[-1.0, 1.0]$ before being displayed or processed. (The new interactive `FIM` utility in *AFNI* does this — see item 6 below.) The new to3d can take as input floating point volumes and scale them to produce `short` or `byte` datasets, with the appropriate scale factor(s) attached.

3. **Auxiliary Statistical Data**
   There are three new types of datasets: the `fico`, `fitt`, and `fift` functions. `fico` means "functional intensity with correlation"; `fitt` means "functional intensity with t-test"; `fift` means "functional intensity with F-test". These are similar in concept to the `fith` images, but the threshold data now has the statistical parameters (*e.g.*, degrees of freedom) attached that allow calculation of the significance (p) level. With such a dataset, *AFNI* will interactively display the p-value associated with the chosen threshold. At present, only *AFNI* and 3dfim generated `fico`, 3dttest generated `fitt`, and 3dANOVA generated `fift` datasets have the statistical parameters automatically attached. Using the new to3d, it is possible to create your own datasets of these types if you provide the needed auxiliary statistical parameters.

4. **Demise of Dataset 'Name' and 'Short Label'**
   These values are no longer used in to3d or *AFNI*. The filename of a dataset is now used for display in the *AFNI* window titlebars and selection choosers.

   The original purpose of the name and short label values was to assign descriptions to a dataset that did not depend on the filename. This is needed when one dataset refers to another, which happens during the warp-on-demand procedure. If you renamed a dataset, and if filenames were used as the internal method of keeping track of such references, *AFNI* would get confused. To solve this problem, I have instead put inside each dataset's `.HEAD` file an internally generated identifier code. This is designed to be unique, and independent of filenames. When you run an *AFNI* program on an old dataset, you will see a message that it is generating a 'new ID code'. These ID codes are pseudorandomly generated using the current time and machine name as seeds. For more information, see the plugins manual.

If you use the Unix command `cp` to copy a dataset, then the new dataset will have the same ID code as the original. *AFNI* will not run correctly if two such datasets are read into the program. To fix this, you can use the auxiliary program 3dnewid to attach a new ID code to a dataset.

5. **3D+time Datasets**
   AFNI now supports time-dependent 3D datasets, which I refer to as '3D+time datasets'. At present, only anatomical datasets may be usefully set up to be time-dependent. 3D+time datasets are created with to3d, using the new `-time:zt` or `-time:tz` command line switches.

   3D+time datasets can be viewed in the image viewing windows in the usual orthogonal slices. (Scrolling in time through these views is a good way to check for subject movement.) They can also be graphed vs. time (similar to the program FD2).

6. **Interactive Functional Activation Analysis**
   AFNI now has the ability to run the equivalent of the fim2 program on a 3D+time dataset, producing as output a new 3D `fico` dataset.

7. **Image Montage**
   One of the most visible changes to *AFNI* itself is the addition of the 'montage' (display of an array of slices) feature to the image viewing windows. This is accessed using the new `Mont` button.

8. **Big Talairach Box**
   The size of the default Talairach coordinates brick has been extended down (inferior) by 10 mm. This is to make sure that the brick includes the entire cerebellum, which is not the case using the old brick dimensions (taken from the Atlas).

9. **Threshold Resampling**
   The type of interpolation used for functional dataset resampling is controlled by the `Resam mode` button in the `Datamode` control panel. In *AFNI* 1.0x, both the functional intensity ('`fim`') and the threshold data were resampled using the chosen method. In this version of *AFNI*, the threshold data (*e.g.*, correlation coefficient) is always resampled using the `NN` method. This is because thresholding with an interpolated nonlinear statistic is a somewhat dubious procedure.

10. **Multiple *AFNI* Controllers**
    Using the `New` button, you can open up multiple controller windows in a single run of *AFNI*. This allows you to view more than one dataset at a time. Using the `Lock` menu, you can force the coordinates of the different viewing windows to be locked together. This feature allows you to scroll in unison through multiple datasets.

11. **Session Directories**
If you don't specify any directories on the command line, then *AFNI* acts as if you had typed '`./`' — that is, it will try to read datasets from the current directory. The new '`-R`' switch will tell *AFNI* to read from all subdirectories of the given session directories, recursively. Using the `Rescan` buttons, you can re-read sessions. This is useful if you use an auxiliary program (*e.g.*, 3dmerge) to create a dataset, and then want to import it into *AFNI*.

12. **Writing Many Datasets to Disk at Once**
A new button allows you to select many datasets at once for output to disk. This makes it possible to start a long Talairach output session, and then leave the computer unattended while it computes each output `.BRIK`.

3 Fundamentals

This section explains how data is organized in the MCW *AFNI* package. The two indispensable concepts are *datasets* and *sessions*.

3.1 Datasets

⋆⋆⋆ ⟶ **Crucial information**

The fundamental unit of data in *AFNI* is the *dataset*: one or more 3D bricks of imaging data, together with some auxiliary information (*e.g.*, axes orientation, coordinates of marked fiducial points, ...). There are two classes of datasets: *anatomical* and *functional*. An example of the former would be a Spoiled GRASS MRI scan. An example of the latter would be the results of cross-correlating a functional MRI (FMRI) time course of images [2]. When you create a dataset (using the to3d program), you must specify whether it is anatomical or functional in nature. When you are using *AFNI*, at any given time you will be displaying one anatomical dataset as the grayscale underlay, and (possibly) one functional dataset as the false color overlay.

Within the class of anatomical datasets, to3d provides you with a list of possible types (*e.g.*, Spoiled GRASS, Echo-Planar, MR Angiogram, ...). At present, all anatomical types of datasets are treated identically. The only reason for choosing a particular anatomical type is to remind yourself of the dataset's origin.

⋆⋆⋆ ⟶ **3D+time**

Anatomical datasets may be in the '3D' or the '3D+time' format. The 3D format stores one value per voxel. The 3D+time format stores a series of values per voxel. This can be used to stored all the data from a 3D FMRI imaging run. The auxiliary program 3dfim or the internal *AFNI* `FIM` capability can be used to produce functional activation datasets from 3D+time datasets.

Within the class of functional datasets, there are presently five types.

`fim` Functional Intensity
- one value is stored per voxel

`fith`	Functional Intensity + Threshold • two values are stored per voxel: • the first is 'intensity' (defined arbitrarily); • the second is 'threshold', which is a either a floating point number between -1.0 and 1.0 or a 2 byte integer between -10000 and 10000, which can be used to select which voxels are considered 'active'.
`fico`	Functional Intensity + Correlation • two values are stored per voxel: • the first is 'intensity'; • the second is a correlation coefficient (between -1.0 and 1.0), which can be used to select 'active' voxels at a given significance (p) value.
`fitt`	Functional Intensity + t-test • two values are stored per voxel: • the first is 'intensity'; • the second is a t-statistic, which can be used to select 'active' voxels at a given significance.
`fift`	Functional Intensity + F-test • two values are stored per voxel: • the first is 'intensity'; • the second is an F-statistic, which can be used to select 'active' voxels at a given significance.

By *intensity*, I mean a signed number indicating the level of functional activity in each voxel.

I consider the `fith` dataset type to be obsolete. It is retained for compatibility with *AFNI* version 1.0x. The latter three functional types differ from the `fith` type in that *AFNI* knows how to statistically interpret the second value attached to each voxel.

⋆⋆⋆ ⟶ **Atomic datum**

The values stored at each voxel can be any of the following:

`byte`	= a `typedef` for `unsigned char`
`short`	= 2 byte signed `int`
`float`	= single precision; 4 bytes
`complex`	= a `struct` containing two `floats`; 8 bytes

I refer to these types as the 'atomic' datum types of a dataset. `Float` and `complex` datasets may not be portable between CPU architectures. Also, `short` datasets may need to be byte-swapped if the files are moved to a different computer. (Specifically, Intel CPUs are reversed from most other Unix systems, so that `short` brick files created on an SGI system would have to be byte-reversed before they could be used on an Intel based system.) The auxiliary program `2swap` can perform this function.

For most purposes, the `short` atomic datum is the most useful. Each 3D brick within a `short` (or `byte`) dataset can have a floating point scaling factor attached, so that the *AFNI* programs will interpret the value stored as *factor ∗ voxel value*.

Datasets are stored in two files: the *header* and *brick* files. The header file contains all the auxiliary information about a dataset, stored in an ASCII format. The brick file contains only the actual 3D volume data. For details on the storage, see the *AFNI* plugins manual.

⋆⋆⋆ ⟶ **File names**

The files in a dataset have highly structured names, and these names should not be casually altered, or *AFNI* will not be able to read them. The general form of the dataset filenames is '`prefix+view.NAME`', where '`prefix`' is to be supplied by the user (you), and presumably would be used to indicate the type of data stored in the file (*e.g.*, `spgr` for Spoiled GRASS, `func` for functional intensity, etc.) The '`+view`' code indicates the origin of the data and is assigned by *AFNI*; the possibilities are:

`+orig` for original data (untransformed by *AFNI*)

`+acpc` for datasets which have been aligned to the AC–PC line

`+tlrc` for datasets which have been transformed to the Talairach-Tournoux grid

Datasets in the `+acpc` and `+tlrc` views would normally be created by *AFNI*; `+orig` files would be created by `to3d`. The '`.NAME`' suffix is `.HEAD` for the header file, and is `.BRIK` for the brick file.

One reason for splitting the auxiliary information from the volume data in each dataset is for efficiency of access to the brick file. Another reason is that in an emergency, the auxiliary information is stored in ASCII form and so can be edited manually (this requires extreme care!). A third reason is that when transformed datasets (`+acpc` and `+tlrc`) are created, their brick files may be deleted later to save disk space — as long as the transformed header files and the `+orig` brick files exist, *AFNI* can recreate the transformed data.

Once raw images have been put into the *AFNI* dataset format, they are no longer needed for any of the programs described herein. It is always possible (using *AFNI* or `from3d`) to extract images out of the 3D data brick, although you may then have to convert them into whatever format you desire.

If you choose to rename the pair of files that make up a dataset, the only part you should touch is the prefix. If you alter the `+view` or `.NAME` parts, *AFNI* will probably refuse to read the files at all.

⋆⋆⋆ ⟶ **Warp-on-demand**

Not all datasets will have a `.BRIK` file. *AFNI* is capable of transforming data from a 'parent' dataset as needed for image display. If the necessary transformation (*e.g.*, from `+orig` to `+tlrc`) is available, then the 'child' dataset (*e.g.* the `+tlrc` dataset) need not have a `.BRIK` file—it can be "warped-on-demand" for display. Normally, `+orig` datasets do not have a warp parent dataset, so they must have a `.BRIK` file. (An exception to this rule can be created with the auxiliary program `3ddup`.)

At this time, no program but *AFNI* itself can deal with warp-on-demand datasets. That is, all the auxiliary programs (and plugins) must deal with actual dataset `.BRIK`s.

3.2 Sessions

⋆⋆⋆ ⟶ **Crucial information**

All of the dataset files that 'go together' should be gathered into a single directory. By 'go together', I specifically mean those datasets gathered during the same scanning session on a single subject. After their positions and orientations are set up (in to3d), all these datasets are presumed to be aligned to one another. *If this is not the case, then the images making up the datasets should be registered before entry into* to3d. The auxiliary program imreg may be useful for this purpose. Alternatively, the program AIR from UCLA might be needed. AIR is available at `http://bishopw.loni.ucla.edu/AIR/index.html`.

A directory containing datasets is called a `session`. The hierarchy of files that make up a session is pictured below:

```
                 session           <-- top-level directory
                /   |   \
               /    |    \
              /     |     \
    ... header   header  header ... <-- actual data files
        brick    brick   brick
        header   header  header
        brick    brick
```

It is permissible to save other files (*e.g.*, the original image files) in the session directory — these files will simply be ignored by *AFNI*.

AFNI takes as input a collection of sessions (specified by their directory names), and allows you to switch between them and between their constituent anatomical and functional datasets. It is important to understand the dataset concepts, file structure, and directory hierarchy described and depicted above.

⋆⋆⋆ ⟶ **Command line names for sessions, datasets**

Sessions are referred to by the top-level directory name under which all their datasets reside. An individual dataset is referred to (on auxiliary program command lines) by the name of its header file, the name of its brick file, or just by the `prefix+view` part of the filenames; for example,

`12dec94/func03+acpc.HEAD`   `12dec94/func03+acpc.BRIK`   `12dec94/func03+acpc`

would all refer to the same dataset residing in a given session directory.

By moving to the directory *above* the session directory, you can save and compress all the files in a session using the command

```
tar cvf - session_directory | gzip -9v > session.tgz
```

This presumes that you have the `GNU gzip` compression utility installed on your system. The command to uncompress and restore from the compressed archive would be

```
gzip -dc session.tgz | tar xvf -
```

4 A Tour of *AFNI*

The best way to learn the program is to read this tour through, and then sit down with the program and try it out.

4.1 Starting Up

The command line to run *AFNI* is quite simple:

```
afni session1 session2 ...
```

Here, `session*` is the name of a session directory to read in. All the 3D datasets under each named session directory will be read in. If no sessions are specified on the command line, the current working directory will be used. (Command line options for *AFNI* will be discussed in a later section.)

For *AFNI* to be able to use a session, it must contain at least one anatomical dataset (3D or 3D+time). If none are available, the auxiliary program `3ddup` can be used to create a warp-on-demand copy of a functional dataset. Alternatively, you could use the first image from the FMRI time course in each slice to form an anatomical dataset, but a separate higher-resolution scan will be more useful and look better. At any given moment in *AFNI*, you are viewing one given anatomical dataset and (possibly) one given functional dataset. Controls are supplied to let you switch among sessions and among datasets within sessions.

A useful model is to think of each session as being organized in a two dimensional layout:

```
                     --------- View Type --------
                     +orig        +acpc      +tlrc

       | anat  :       X            X
prefix | angio :       X            O
       | func1 :       X            O
       | func2 :       X            O
```

Across the top is the view type (Original, AC–PC aligned, or Talairach). Down is the dataset prefix. `X`'s mark datasets that actually exist on disk. In the sample above, the `anat` original data has also been transformed to the AC–PC aligned view (using the marker driven transformation described later).

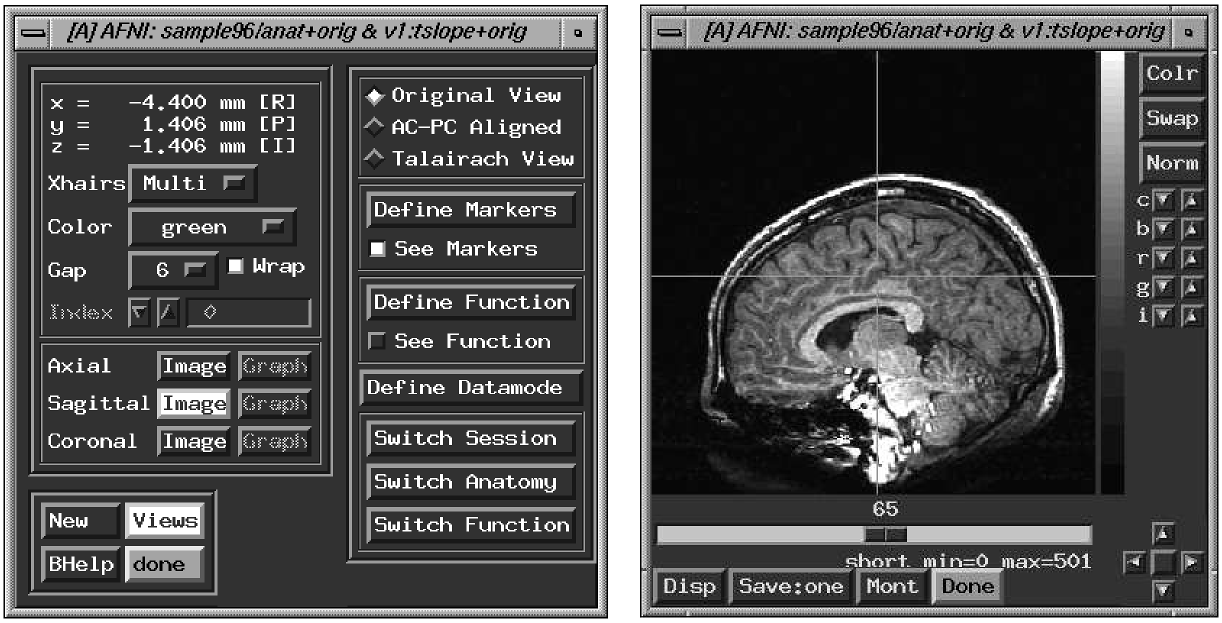
When *AFNI* starts, all the other datasets in this session will also have AC–PC aligned view versions made internally in the program — these are indicated by `O`'s in the table above. (No `.HEAD` files will be written for these datasets at this time. These datasets will be warp-on-demand until and unless you write out the `.BRIK` files using one of the `Write` buttons described later.) The transformation from `+orig` to `+acpc` in the `anat` dataset (stored in the `.HEAD` file) will be applied to the other datasets in the `+orig` view. These 'follower' datasets are what makes *AFNI* work and easy to use. When the `anat` transformation from `+acpc` to `+tlrc` is defined — when the `X` is placed at the top of the third column — then all the other datasets in this session will again follow along — `O`'s will fill in the rest of the third column.

For this to be possible, it is necessary that the correct geometrical relationship between the datasets comprising a session be established when `to3d` is run — that is the import of getting the axes orientations and origins correct in `to3d`.

★★★ ⟶ `afni -im`

AFNI is also capable of directly reading in and displaying a set of image files. Use the command '`afni -help`' for detailed instructions on how to do this. In this mode, none of the controls for dataset transformation, functional overlay, etc., are available.

After you start *AFNI*, a control window opens on the X11 screen:



AFNI controller and image viewing windows

4.2 Program Control

At the lower left of the control panel are four buttons which control various 'global' program functions:

`New` This button will open up a new *AFNI* controller window. In this way, it is possible to open up image viewers on more than one dataset (or session) at a time. (A maximum of 5 controller windows can be open at once.) The first controller window and its children are marked with `[A]` in their titlebars; the second is marked with `[B]`, and so forth.

`Views` This button will open and close the control panels to the right of the first column of the controller window (the first such control panel starts with `Original View` in the figure above). This function allows you to save some screen space without iconifying the controller window.

`BHelp` This button allows you to popup a help window for most controls within *AFNI*. Pressing `BHelp` will cause the mouse cursor to change to a small hand shape. Pressing mouse Button 1 while the hand is over an *AFNI* control will popup a help window for that control. Clicking Button 1 inside the help window will dismiss it.

Button help is implemented as a `help callback` in Motif. If your terminal keyboard is appropriately set up, then pressing the Motif '`Help`' key (often `F1`) while the mouse cursor is over a button or other widget will also cause the help window for that widget to popup.

`done` This button will close the controller window when pressed twice within 5 seconds. If this is the only controller window running, *AFNI* will also exit. (For more details, try using `BHelp` on `done`.)

The MCW logo will appear in the empty space just to the right of these four buttons when the program is doing some operation that is potentially time consuming. At the same time, the mouse cursor will change to a watch shape. When the time consuming operation is over, the logo will be removed and the cursor will change back to its usual arrow shape.

A trick I sometime use to visually grab attention when a lengthy task is underway is to click the `Swap` button in an image window. This will turn the images to reverse video when the program catches up with you. Then `Swap` again, and proceed.

4.3 Image Display

The first column of the controller window contains the controls that enable you to open the image viewing windows: the three `Image` buttons. The windows open separately on the X11 display screen, and may be positioned and resized independently. A little practice is needed to decide upon a good layout scheme for these windows. The reason *AFNI* does not define their locations and sizes is that it is often desirable expand one window to look at some details, and temporarily cover up the other windows.

When an `Image` button is highlighted in inverted colors, this means that its window is already open. Pressing the button again will bring that window to the top of the display — this is useful if the image viewing window is hidden beneath some other window, or is iconified.

An image viewing window can be closed by using the `Done` button along its bottom edge. Alternatively, the window manager `Close` or `Delete` function may be used.

4.3.1 Crosshairs and the Viewpoint

⋆⋆⋆ ⟶ **Left is Right!**

You can have up to three image viewing windows open at any given time: one axial, one sagittal, and one coronal. (In the axial and coronal windows, images are displayed with the subject's left on the screen right — the usual radiological convention.) These windows are orthogonal slices through the 3D dataset. The colored crosshairs that overlay each image mark the slices that are visible in the other image windows. The point at which the crosshairs intersect is called the `viewpoint`.

In the upper left hand corner of the *AFNI* controller window are displayed the coordinates of the current viewpoint. These coordinates are presented in the DICOM 3.0 standard order:

x**-axis** is Right (negative) to Left (positive)

y-**axis** is Anterior (negative) to Posterior (positive)

z-**axis** is Inferior (negative) to Superior (positive)

Internally, *AFNI* uses DICOM coordinates to keep track of everything. (However, *AFNI* cannot read DICOM files!)

The button `Xhairs` just under the coordinate display allows you to switch between 3 modes for crosshair display:

`Off` Crosshairs are not displayed.

`Single` A single crosshair is displayed at the (x, y, z) coordinates of the 'viewpoint' (*i.e.*, the point whose coordinates are given in the upper left corner).

`Multi` If a montage of slices is displayed in an image viewing window, then the orthogonal slices will have crosshairs indicated for all the montaged slices. This is useful for indicating the anatomical location of the montage layout.

You may change the crosshair color and central gap with the selectors just underneath the `Xhairs` button. The colors available for the various overlays are built into *AFNI*, but can be altered by appropriate changes to your `.Xdefaults` file.

If you click the left mouse button while the cursor is in an image window, then the crosshairs will immediately jump to that location. This will usually mean that the other two windows will display new slices. The image windows are always 'linked' in this fashion.

The `Index` control on the *AFNI* controller window is used to control the time index of the viewpoint. This control is only active if the current anatomical dataset is in the 3D+time format. (The time index can also be controlled in a graph viewer.)

4.3.2 Colormap Controls

At the right of each image window is a set of buttons that are used to control the X11 colormap assigned to the windows. From top to bottom, these controls

`Colr` Change from grayscale to a colorscale, and back.

`Swap` Invert the grayscale or colorscale (swap it end-for-end).

`Norm` Return the colormap to its initial state (after you mess it up).

`c` Change the contrast of the grayscale (a multiplicative change to the intensity of each pixel).

`b` Change the brightness of the grayscale (an additive change to the intensity of each pixel).

`r` Rotate the grayscale or colorscale.

`g` Change the γ correction factor for the grayscale.

`i` Change the fraction of the viewing window taken up by the image.

You may have to drag the viewing windows to be larger than their initial sizes so that these and the other controls don't obscure each other. Changing the colormap in one viewing window affects all the other windows from the same *AFNI* process.

4.3.3 Position Controls

Below each image is a slider that indicates the image number in the current sequence. By dragging this slider, you can move through the slices to any given slice number. In doing so, you will also move the crosshairs in the other two image windows.

At the lower right of the image window is an 'arrowpad' of four arrows arranged in a N–E–W–S pattern, plus a central button. Clicking on one of the arrows will cause the crosshair viewpoint to move one voxel in the direction pointed by the arrow *in that window*. Clicking on the central (unlabeled) button causes the crosshair gap to close; clicking this button again causes the gap to open up again. This is very useful when positioning the crosshairs prior to setting an anatomical marker.

Image viewer `Disp` and `Mont` control panels

4.3.4 `Disp` Control

This button opens up a control panel (pictured above) that lets you change how the images will be displayed in this window. The `Rotation` and `Mirror` items control the orientation of the image in the window. The `No Overlay` item allows you to turn off all color overlay items (*i.e.*, crosshairs, anatomical markers, and function). The `Min-to-Max` and `2%-to-98%`

items choose how the values in the image array are mapped to grayscale levels on the screen. The former choice maps the minimum image value to black and the maximum to white; the latter choice computes the cumulative histogram of the image and maps the 2% point to black and the 98% point to white.

The `Free Aspect` control lets you resize the image window to any bizarre aspect ratio. Normally, the program restricts the image window resizing so as to keep the data-voxel to display-pixel geometric relationship correct (assuming that display pixels are square!).

The `Save` controls actually have nothing to do with image display. They are just here because it was a convenient place to put them. They control the operation of the `Save:` button on the image viewing window (next to the `Disp` button) — this is discussed below.

The `Tran` menus allow you to pick from a list of image transformations. The `Tran 0D` transformations are all 'pointwise'; that is, the image intensity output at a given pixel is a function of the image intensity input at that given pixel only. The built-in `Tran 0D` functions are `Log10` and `SSqrt`, which take the common logarithm ($\log_{10}$) of each pixel, and take the 'signed square root' ($\operatorname{sgn}(x)|x|^{1/2}$) of each pixel, respectively.

`Tran 2D` functions are more global image transformations, where the image intensity output at a given pixel can be a function of other pixels. The only built-in `Tran 2D` function is `Median9`, which replaces each pixel by the median overs its 3×3 neighborhood.

AFNI plugin authors can add functions to these `Tran` menus — hopefully, they will be documented. At the bottom of the `Disp` panel are 3 other built-in image processing functions:

`Flatten` Histogram 'flattening' (or equalization) is performed on the image prior to display.

`Sharpen` High-emphasis 'sharpening' is performed on the image prior to display. When printing images on printers with relatively few colors available per pixel, the combination of `Median9` and `Sharpen` gives nice results.

`Edge Detect` Sobel edge detection is performed on the image prior to display.

If more than one of these are selected, they are performed in the top-to-bottom order, as displayed; also, these operations are performed after any `Tran` functions.

4.3.5 `Save:` Control

This button takes one of three forms, depending on the choices made in the `Disp` panel:

`Save:one` This form of the `Save` function is selected by toggling on the `Save One` option on the `Disp` control panel. In this form, the action is to save the current image to disk. *This is the only way provided by AFNI to save a montage layout.*

This button pops up a little 'chooser' window which asks you to input the filename prefix for the output image. The image will be saved in the 'raw PNM' format, with the name '`prefix.pnm`'. Images in this format can be converted to other formats (such as `TIFF`) with command line utilities in the netpbm package, or the xv shareware program. (Images which contain no color will be saved in the PGM format; images with colored pixels will be saved in the PPM format.)

⋆⋆⋆ ⟶ netpbm and xv

`Save:pnm` This form of the `Save` function is selected by toggling on the `PNM Save` option on the `Disp` control panel. In this form, the action is to save a collection of slice images (underlay and color overlay) to disk in the raw PNM format. Even if a montage is being displayed, only single slices will be saved with this function.

This button also pops up a chooser window which asks you to input the filename prefix for the slice data. If you enter '`fred`' for the prefix, then the 238^{th} slice would be named `fred.0238.pnm`. After you type in the desired prefix, you click the `Set` button, and then must choose the first and last slice indexes for the save operation from the new choosers that will popup. When you `Set` the last slice index, the write-to-disk operation starts.

`Save:bkg` This form of the `Save` function is selected by toggling off the `PNM Save` and `Save One` options on the `Disp` control panel. In this form, the action is to save the background image pixel values. This does *not* mean the grayscale intensities displayed in the window. It means that the actual values stored in the dataset `.BRIK` file will be written to disk.

This button operates similarly to the `Save:pnm` function: you must choose a filename prefix, and the first and last slice indexes for the save operation. If `Nsize Save` is selected on the `Disp` control panel, then the saved images will be expanded to the next largest power of two.

Using the `Function` controls, you can switch to have the function displayed as the background. In that way, the functional dataset voxel values may be written to disk in slice format. Alternatively, the auxiliary program from3d can be used to write slice image files out of a dataset.

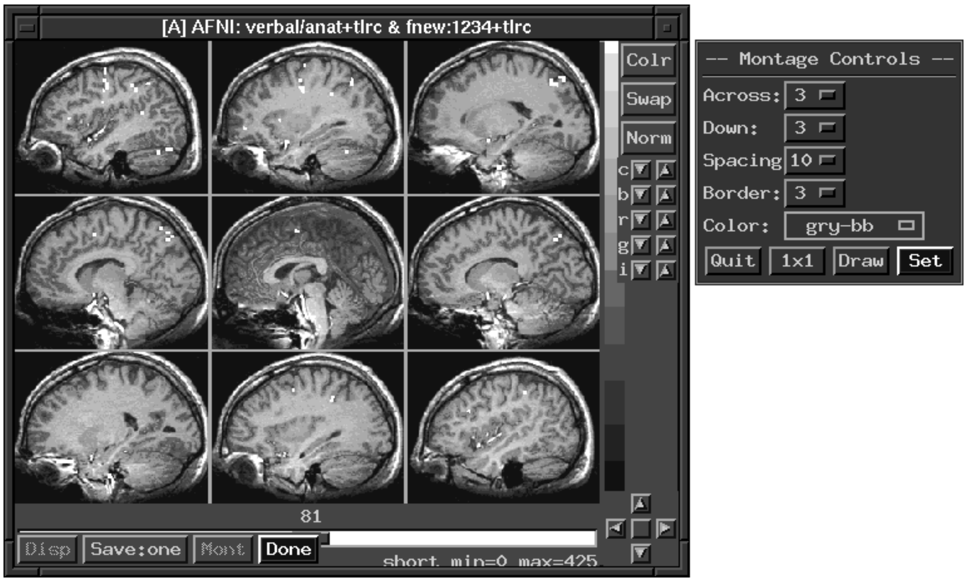
For all `Save` options, the actual size of the displayed window doesn't matter — the images saved to disk will reflect the voxel dimensions of the dataset. In particular, if the dataset voxels are not square in the plane of view, then the saved image aspect ratio will be distorted. The only way to rectify this in *AFNI* is to switch the dataset to be warp-on-demand (using the `Datamode` controls), which always interpolates to square pixels.

To actually save the pixels as displayed on the screen, some sort of 'snapshot' or 'window grab' utility is needed. If none other is available, the shareware program xv has a window grab function. Or the xwd command line program can be used, with appropriate conversion using the netpbm utilities. For example, the images displayed in this manual were captured with variants of the following command line:

```
xwd -frame|xwdtopnm|pnmdepth 255|ppmtopgm|pgmnorm|pnmtops -noturn>name.eps
```

4.3.6 `Mont` control

This button allows you to make a montage of more than one slice in the image window. It pops up a control panel (pictured earlier and below). For convenience in programming, only one of the `Disp` and `Mont` control panels can be open at a time (per image viewer). This restriction may be lifted in some later version of *AFNI* (but no guarantees).



Example of slice montage, with $\boxed{\texttt{Mont}}$ control panel

The five montage menu controls from top to bottom are:

$\boxed{\texttt{Across:}}$ This controls the number of slices to be displayed horizontally across the window. In the example above, `Across` and `Down` are both set to 3.

$\boxed{\texttt{Down:}}$ This controls the number of slices to be displayed vertically down the window.

$\boxed{\texttt{Spacing:}}$ This controls the frequency with which slices are displayed. `Spacing` of 1 means that adjacent slices will be displayed; 2 means that every other slice will be displayed, etc. The units of this selector are slices, not millimeters; thus, changing the resolution of a warp-on-demand dataset (using `Datamode` controls) will change the inter-slice distance as displayed here. In the example above, the images are 10 slices apart. Since this is in the Talairach view (which can be seen from the dataset names in the image viewer titlebar), and the default voxel size of 1 mm was used, these sagittal slices are 10 mm apart (center-to-center).

$\boxed{\texttt{Border:}}$ This controls the thickness (in dataset pixels, not screen pixels) of the border to draw between slice images. In the above example, this is set to 3.

$\boxed{\texttt{Color:}}$ This controls the color of the border drawn between slice images. In the above example, this is set to a gray color (under the presumption that you do not have a color PostScript printer with which to output this manual).

Across the bottom of the $\boxed{\texttt{Mont}}$ control panel are four action buttons. Their functions are:

$\boxed{\texttt{Quit}}$ This will close the $\boxed{\texttt{Mont}}$ control panel, and leave the current montage layout unchanged.

$\boxed{\texttt{1x1}}$ This will reset the $\boxed{\texttt{Across:}}$ and $\boxed{\texttt{Down:}}$ controls to each be 1. This is simply a convenience, for when you wish to go back to displaying a single slice.

$\boxed{\texttt{Draw}}$ This will instruct the image viewing window to redraw itself as currently commanded. (This button will display in inverted colors until the redraw operation is complete. If the dataset is warp-on-demand, and many slices are requested, this operation may take several seconds.)

$\boxed{\texttt{Set}}$ This combines the functions of $\boxed{\texttt{Draw}}$ and $\boxed{\texttt{Quit}}$: it will redraw the window as commanded, and also close the $\boxed{\texttt{Mont}}$ control panel.

Slices are displayed starting in the upper left corner, then from left-to-right, then top-to-bottom. If the $\boxed{\texttt{Disp}}$ controls are used to rotate or mirror the images, these operations apply to each slice individually, not to the montage as a whole.

If the number of slices displayed (`Across` $\times$ `Down`) is large, and `Spacing` is large, then the extreme slices requested may be outside the dataset. In such a case, the toggle $\boxed{\texttt{Wrap}}$ on the *AFNI* controller window (next to the $\boxed{\texttt{Gap}}$ menu) controls what happens. If $\boxed{\texttt{Wrap}}$ is turned on, then slices requested past the edge of the dataset will be wrapped back to the opposite edge. If $\boxed{\texttt{Wrap}}$ is turned off, then slices requested past the edge of the dataset will be filled with zeros.

The only slice image which will have crosshairs displayed is the one containing the current viewpoint. Clicking mouse Button 1 in any slice image will cause the current viewpoint to jump to that location in that slice. This will also cause that slice to jump to the center position in the montage layout.

If the $\boxed{\texttt{Xhairs}}$ button is set to $\boxed{\texttt{Multi}}$, then orthogonal image viewing windows will show crosshairs for each slice drawn in the montaged window.

The only way that *AFNI* provides to save a montage display to disk is the $\boxed{\texttt{Save:one}}$ function, described earlier. This function (as all `Save` functions) saves the image at its 'natural' size — one output pixel per dataset pixel, regardless of any window resizing you may have imposed.

4.3.7 Popup Menu

If you click-and-hold mouse Button 3 (usually the rightmost button) while the cursor is in an image window, a menu will popup at that point. The first item on the menu is $\boxed{\texttt{Jumpback}}$. This will reset the crosshair viewpoint to the last location clicked upon. The main use of this feature is to recover from accidentally clicking Button 1 in an inconvenient location.

The second item on the popup menu is $\boxed{\texttt{Jump to}}$. This will popup a window in which you may enter the (x, y, z) coordinates (DICOM order) of the point to which you wish to set the crosshair viewpoint. (One application: the cluster locations reported by 3dclust may be pasted into the window.)

The third item on the popup menu is $\boxed{\texttt{Image display}}$. This will collapse the viewing window to just include the image. Selecting this item again will bring the viewing window controls back. (One application: making snapshots of nice functional displays.) **N.B.**: This function does not work well on all X11 displays, for unknown reasons. It may collapse the image window to zero size, which is slightly inconvenient.

$\star\star\star \longrightarrow$ **Seeing individual voxel values**

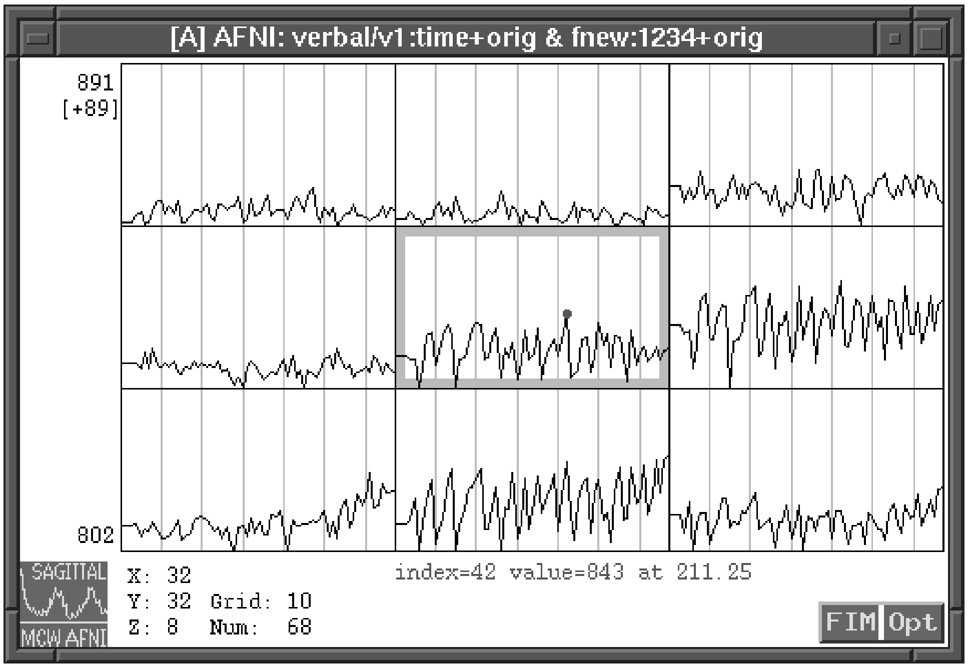
When all three viewing windows (`Axial`, `Sagittal`, and `Coronal`) are open, the bottom item on the popup menu will contain a display of the background pixel value at the center of the crosshairs. (The background is normally an anatomical image, but can also be switched to be a functional image using the `Function` controls.)

4.3.8 Resizing Image Windows

When a viewing window is first opened, it is at the 'natural' dimension for the resolution set in the `Datamode` controls (or in the dataset header file, if viewing from the data brick). When jumping from one coordinate system to another, or from one dataset to another, the program attempts to keep the on-screen pixels/mm the same, so that no sudden scale changes occur.

When the viewing windows are resized, the images are stretched by nearest-neighbor resampling. This has nothing to do with the resampling modes set in the `Datamode` and `Function` controls. For example, the crosshairs are exactly one pixel thick in the 'natural' dimension of each window. If a window is stretched, then the crosshairs will become thicker. If a window is shrunk, the crosshairs may disappear when they are missed during the display resampling.

When viewing from the data brick, and when the data voxels are not cubical, the viewing windows will similarly stretch the coarser direction to maintain the correct physical aspect ratio on the screen (assuming that the X11 pixels are square). This can produce a blocky looking image. To force a smoothing, the `Warp on Demand` mode should be used.



Sample Graphing Window

4.4 Graph Display

When the current anatomical dataset is 3D+time and is not set to warp-on-demand (using the `Datamode` controls), then the three `Graph` buttons (next to the `Image` buttons) will be activated. These allow the display of graphs of voxel intensity vs. time. (Although *AFNI*

graph windows look very like the graph window in the auxiliary program FD2, there is at
⋆⋆⋆ ⟶ least one major difference: *AFNI* graph windows can be resized.)

The above figure shows a sample, with 3×3 data time series sub-graphs displayed. The central sub-graph is bordered in a light color (here, a shade of gray). This sub-graph is a plot of the data time series at the viewpoint voxel — the one displayed at the crosshairs in the image viewing windows. The other sub-graphs are of neighboring pixels, in this case in the sagittal plane. (In the corresponding sagittal viewing window, the crosshairs will show a box outlining the pixels being graphed.) If you shift the viewpoint in an image viewer, then the graph viewpoint will shift accordingly.

The voxel indexes of the viewpoint are shown at the lower left edge of the graph display; for example, `Z: 8` means that this is slice number 8 (counting from 0). Then the spacing between vertical grid lines (in time steps) is shown; in this case, there is a grid line every 10^{th} time step. The `Num:` label shows the number of points in the time series. At the bottom of the graphs is a label starting with `index=`. This shows the time index of the current point (which can be altered with the `Index` control on the *AFNI* controller window), the value of the time series at that point, and the time coordinate of that time index (in this case, `211.25` seconds). The current time point is displayed with a little red ball overlaid n the central sub-graph (in the figure above, red is rendered in gray).

⋆⋆⋆ ⟶ **Graph scaling**

At the left of the graphs are two numeric labels; in this case, 802 and 891 (= 802 + 89). This shows the vertical range of the central sub-graph. In this display, each sub-graph has its minimum point at the bottom of its sub-window. *Every sub-graph has the same vertical scale factor, but has a different vertical offset.* This is so that they will all fit in the same display easily. This can be confusing when comparing levels of adjacent pixels: it is important to realize that there can be an arbitrary constant offset between adjacent sub-graphs. The `Baseline` button on the `Opt` menu can be used to ensure that all sub-graphs are plotted with the same vertical offset.

4.4.1 Button Clicks in a Graph

At the lower left of the graph window is a logo, whose main function is to remind you from which plane these graphs are drawn. This can be suppressed (for window snapshot purposes) by clicking Button 1 on the logo. Clicking Button 1 in that space again will restore the logo. (The `FIM` and `Opt` buttons will also be hidden and restored by these operations.)

Clicking Button 1 on the central sub-graph will cause the time index to jump to that point. If the Shift or Ctrl key is pressed while doing this, the time index instead will move up or down by 1, in whichever direction corresponds to the mouse cursor relative to the time index indicator ball.

Clicking Button 1 on any other sub-graph will cause the spatial viewpoint to jump to that location, without changing the time index. Clicking Button 3 on a sub-graph will popup a small window with some statistics about that time series.

Some keystrokes, if pressed while the graph window has ‘focus’, will carry out certain functions. These functions are also available from the `Opt` menu, and are described below.

Graphing `FIM` (left) and `Opt` (right) menus

4.4.2 `Opt` menu

The items on this menu control the appearance of the graphs:

`Scale` This is a 'pullright' menu, used to control the vertical scale of the graphs. The figure above shows the sub-menu that results from pulling-right on this item. The `Down` and `Up` buttons will cause the graphs to scale down (shrink) and up (grow). Pressing the keys '`-`' and '`+`' in the graph window will have the same effects. The `Choose` button will cause a little chooser to popup, which allows you set the vertical scale factor manually. If the scale factor is positive, then it is the number of screen pixels to use per unit of data. If the scale factor is negative, its absolute value is the number of units of data per screen pixel. Thus, increasing the scale factor will cause the graph to grow vertically. There is no provision in *AFNI* for automatic scaling to fit the dataset range; each graph window starts with its scale factor set to 1. Judicious use of the '`-`' or '`+`' keys may be needed to make a graph visible.

`Matrix` This pullright menu is used to control the number of sub-graphs displayed. It shows a sub-menu similar to the `Scale` sub-menu. In this case, `Down` (keystroke '`m`') will cause the number of sub-graphs displayed across and down to decrease by 1; `Up` (keystroke '`M`') will increase the array size by 1; `Choose` will let you directly set the number of sub-graphs.

`Grid` This pullright menu lets you control the spacing (in time steps) between vertical grid lines.

`Slice` This pullright menu lets you move the spatial viewpoint between slices in the dataset (moving within a slice is controlled by Button 1 clicks, as described above).

`Grid Color` This item just rotates the vertical grid color between the available choices.

`Baseline` This item switches the way that graph baselines are computed. By default, each sub-graph has the minimum value of its time series mapped to the bottom of its sub-window. By pressing this item (or the '`b`' key), the sub-graphs will switch to having a common baseline. That is, the minimum value in `all` the displayed time series will mapped to the bottom of each sub-window. (In many cases, doing this will require scaling down the graphs with the '`-`' key.) Choosing this item again will restore the original graph baseline mode.

`Write Center` Selecting this item (or pressing the '`w`' key) will write the dataset time series in the central sub-graph to disk, in a filename like `033_034_008.suffix.1D`. The '`suffix`' is selected by the next menu item, `Set 'w' Suffix`, and the prefix numbers are from the voxel spatial index in the dataset. The output file is ASCII, one number per line. (**N.B.**: the output is the dataset time series, and is unaffected by the `Tran` transformations below.)

`Tran 0D` This item is the same as the `Tran 0D` item on the image viewer `Disp` panel. It allows the application of a pointwise transformation function to the time series before graphing occurs.

`Tran 1D` This item allows the application of general transformations to the time series before graphing. The two built-in `Tran 1D` functions are `Median3` and `OSfilt3`, which are order-statistics smoothing filters using a 3-wide neighborhood. (In addition, the `plug_lsqfit` plugin defines two more functions, which can be used to do linear least squares fits of various functions to dataset time series.)

`Double Plot` Normally, if a `Tran 1D` transformation is applied, the graph of the transformed data replaces the graph of the original data. If this item is selected, then both graphs will be displayed. (This was added to be able to see the least squares fits from `plug_lsqfit` plotted over the input data.)

`Done` This closes the graphing window (the keystroke '`q`' has the same effect).

4.4.3 `FIM` menu

This menu is used to control the computation of functional datasets (of the `fico` type) from 3D+time datasets. Some of the functions of this menu are duplicated on the `Function` control panel `FIM` menu.

Theory of `FIM`

If x_n is a data time series, and r_n is a known reference (or ideal) vector corresponding to the expected activation time course, then the statistical model for x_n is

$$\begin{aligned} x_n &= \alpha r_n + a + bn + \eta_n \qquad (n = 0, 1, \ldots) \\ \eta_n &\sim N(0, \sigma^2) \qquad \text{i.i.d.} \end{aligned}$$

where α is the unknown amplitude of the activation, a is the unknown mean signal level, b is the unknown linear drift in time, and σ^2 is the unknown noise variance. The correlation method [2, 4] computes the sample correlation coefficient between x_n and r_n, and uses that to determine the significance of the hypothesis that $\alpha \neq 0$.

Time Series Files

MCW *AFNI* stores single time series (such as r_n) in the format described earlier: ASCII format, one number per line. A number greater than 33333.0 means that particular point in time should be ignored. This convention goes back to the original fim program in 1990, and is due to Andrzej Jesmanowicz of MCW. (The `Write Center` button described above is one way to create such a time series file. Another is with the auxiliary program sqwave. Yet another is simply to use a text editor, such as `vi`.)

When *AFNI* starts up, it will read in time series files from the session directories that it opens. The program will attempt to read a time series from any file whose name ends in '`.1D`'. These time series will be pooled into a 'library', whose entries can be selected from a menu, as described below.

You may wish to keep some time series files stored separately from any particular session directory. *AFNI* can be made to read such files by defining the shell environment variable `AFNI_TSPATH`. Like the executables variable `PATH`, this is a colon-separated list of directories in which to search for particular files: in this case, '`*.1D`' files. For example:

```
setenv AFNI_TSPATH ${HOME}/timeseries:./
```

will cause *AFNI* to search for time series in the `timeseries` directory under your home directory, and in the current working directory. Normally, you would put this type of command in your `.cshrc` file so that it would always be executed when you login (I'm assuming that you use the C-shell `csh` or its superset, `tcsh`.)

I said above that time series files are stored as one number per line. This is true of the older programs such as fim2, FD2, etc., but *AFNI* itself allows more than one number per line. In this way, more than one time series can be stored per file:

```
99999   99999   99999   99999
99999   99999   99999   99999
    0       0       0       0
   10       0       0       0
   10      10       0       0
    0      10      10       0
    0       0      10      10
   10       0       0      10
   10      10       0       0
  ...     ...     ...     ...
```

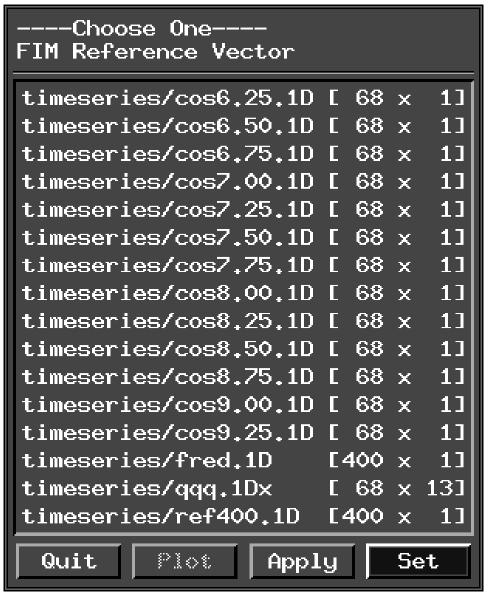
The above example defines four time series in one file. Each one starts with two `99999`s, which indicates that these two points should not be used in any analysis with these time series. The subsequent values could be used as an ideal waveform in `FIM` analysis. In this example, each timeseries (after the first) is a time-delayed copy of the one to the left. A Unix command of the form

```
pr -m -t -s" " a.1D b.1D c.1D d.1D > abcd.1Dx
```

can be used to 'glue' multiple files together horizontally into a single file with multiple columns. You should be sure that all the files have the same number of lines before using the `pr` command, which is really a program designed for formatting files for printout. (On SGI workstations, the `pr` command has a small upper limit to the number of columns it will output. On such systems, it may be necessary to 'paste up' a large number of columns in two stages.)

To allow such time series files to be distinguished from 'ordinary' files (containing only one time series), *AFNI* allows the use of the filename suffix '`.1Dx`', as in the `pr` command example above. However, there is no difference between the '`.1Dx`' files and the '`.1D`' files as far as *AFNI* is concerned: it is perfectly acceptable to have a multi-column time series filename end in '`.1D`', or to have a single column time series filename end in '`.1Dx`'.

At this release, only the actual *AFNI* program will recognize these multi-column time series files correctly. Other programs, such as fim2, still require each file to contain only one time series.



Sample time series chooser menu

`FIM` Menu Items

`Pick Ideal` This button pops up a chooser window which allows you to select a time series file to be used as the ideal waveform r_n (an example of this chooser is shown above). With each time series filename is shown the dimensions of the time series. In the example, the file `qqq.1Dx` has 68 points in time (68 lines of data) and has 13 columns.

If the ideal time series has more than one column, the `FIM` computations compute the correlation coefficient of each voxel with each column separately. Then the column that is most highly correlated with the voxel time series is selected to compute the α for that voxel; α becomes the 'intensity' and the correlation coefficient becomes the second sub-brick stored in the resulting `fico` dataset.

When the ideal time series is selected, it will be plotted at the top of the center sub-graph window, in red. Sections of it that are marked to be ignored will be plotted in blue. If there is more than one column in the ideal time series, by default they will all be plotted.

(By the way, the `Plot` button on the time series chooser is permanently deactivated. I intended to provide the ability to preview a time series in a little window, but never got around to implementing it. The button remains to remind me of this, and to remind me of all my other goals for *AFNI* that have yet to come to fruition.)

`Pick Ort` In the mathematical model for x_n given earlier, note that we included $a+bn$ in the model to represent the unknown baseline and drift of the data time series. These are 'orts': time series to which the data time series is orthogonalized prior to the correlation coefficient calculation [4]. *AFNI* always orthogonalizes the data to these two time series. In addition, you may specify another time series to which the data time series in each voxel should be orthogonalized. If the ort time series has more than one column, each data time series will be orthogonalized to all columns prior to the correlation coefficient calculation.

When the ort time series is selected, it will be plotted in the middle of the center sub-graph window, in green. Sections of it that are marked to be ignored will be plotted in blue.

`Edit Ideal` This pullright sub-menu (shown in a figure far above) is used to control the ideal waveform. The menu sub-items are:

- `Ideal = Center` will take the central sub-graph data time series and make it the ideal time series.
- `Ideal+= Center` will take the central sub-graph data time series and average it into the ideal time series. In this way, it is possible to select a group of voxels (one at a time) and average them together to make a single waveform.
- `Smooth Ideal` will apply a 3-neighborhood order-statistics filter to smooth the current ideal waveform. If the 3 points input are a, b, and c, then the output is

$$y_n = 0.70 \cdot \text{median}\,(a,b,c) + 0.15 \cdot \max\,(a,b,c) + 0.15 \cdot \min\,(a,b,c)\ .$$

- `Shift Ideal` will popup a control panel that allows you to generate shifted copies of the current ideal waveform. All the shifted copies form a multi-column time series. In this way, it is possible to construct a single column ideal waveform, use *AFNI* to create time shifted copies of it, and then correlate them all with the data, picking out the 'best' time shift at each voxel.
- `Clear Ideal` will clear the current ideal waveform; that is, the ideal waveform will be undefined after this is pressed. The graph of the ideal waveform will also be cleared.
- `Clear Ort` will clear the current ort waveform.
- `Read Ideal` allows you to read in the ideal waveform from an external file.

- `Write Ideal` allows you to write the current ideal waveform to disk. In this way, an ideal created from the current dataset can be saved for later analysis.
- `Store Ideal` allows you to store the current ideal waveform into the *AFNI* menu for possible later selection. (The only way to create an ort from a dataset is to first create it as an ideal, then to store it with this button, and then to use `Pick Ort` to get it off the time series menu.) The `Write` and `Store` operations are independent: writing to disk does not imply storing in the menu system, or *vice versa*.

`Ignore` This pullright menu allows you set the number of points to be ignored at the beginning of each time series. (This is often desirable, since in rapid scan MRI the longitudinal magnetization may take several images to reach steady state.) Time points that are ignored — either through the use of this control or large values (33333+) in the ideal waveform — will be skipped in the correlation analysis. In addition, points that are ignored with this control will not be graphed. This allows large initial transients to be suppressed, and makes it easier to read the graphs.

`FIM Plots` This pullright menu allows you to specify whether all the columns of the ideal and ort waveforms should be plotted, or just the first columns.

`Compute FIM` This will start the correlation calculations. At the least, an ideal waveform must be specified before this button will work. You may also wish to set the `Ignore` value and the ort waveform prior to these computations.

The calculations use the recursive method of Cox *et al.* [4]. While they are proceeding, a 'progress meter' will display over the titlebar of the *AFNI* controller window. When the results are finished, the new `fico` dataset will become the current functional dataset, and can be examined immediately in the image viewers. If the input 3D+time dataset were named `elvis+orig`, then the `fico` dataset will be named `elvis@#+orig`, where `#` will be one of `1`, `2`, `3`, ... If desired, the `Dataset Rename` (`plug_rename.c`) plugin can be used to change the prefix to something more convenient.

4.5 Viewing Controls

The second column of the *AFNI* control window contains many controls that affect how the datasets are viewed, transformed, and output.

4.5.1 View Modes

At the top of this second column are the view mode controls. These let you switch between the available views: `Original`, `AC-PC aligned`, or `Talairach`. If a particular view is not available for the currently active anatomical dataset, then the corresponding button will be inactive (grayed-out).

When you switch view modes, you will see that the windows change size. That is because in the `+acpc` and `+tlrc` views, the images are clipped to the known dimensions of human heads. This is to save space when such datasets are written to disk. The window resizing is set up to preserve the pixels-per-brain-millimeter ratio. The program also attempts to keep the crosshair viewpoint in the same anatomical location as they were in the previous view mode, but they may shift slightly. Since the `+orig` view will generally be rotated from the `+acpc` and `+tlrc` views, keeping the viewpoint at the same location does *not* mean keeping the slices in the same location.

4.5.2 `Define` Controls

Each `Define` button opens up another control panel to the right of this second column. They are discussed separately later. Note that to close a `Define` control panel, you press its button again. To help with this, the `Define` buttons that are currently open are displayed in inverted colors. You may open more than one `Define` control panel at a time, but they will all lie on top of each other (most recently opened panel on top).

4.5.3 `Switch` Controls

These controls let you choose which session, and which datasets from that session, you are viewing at any given moment. They popup a chooser that lets you cycle between the session directories, the anatomical dataset prefixes, and the functional dataset prefixes, respectively.

When you switch datasets, *AFNI* may be forced to switch views as well. This can occur if the new dataset doesn't exist in the view mode you were formerly in. For example, if you save several functional datasets in the Talairach view mode (using the `Datamode` controls), then combine them with `3dmerge`, the result will only exist in the `+tlrc` view. You can only view this functional dataset when the view mode corresponds, so the program will jump to that mode. If such a view mode switch occurs, the program will beep when it makes the transition.

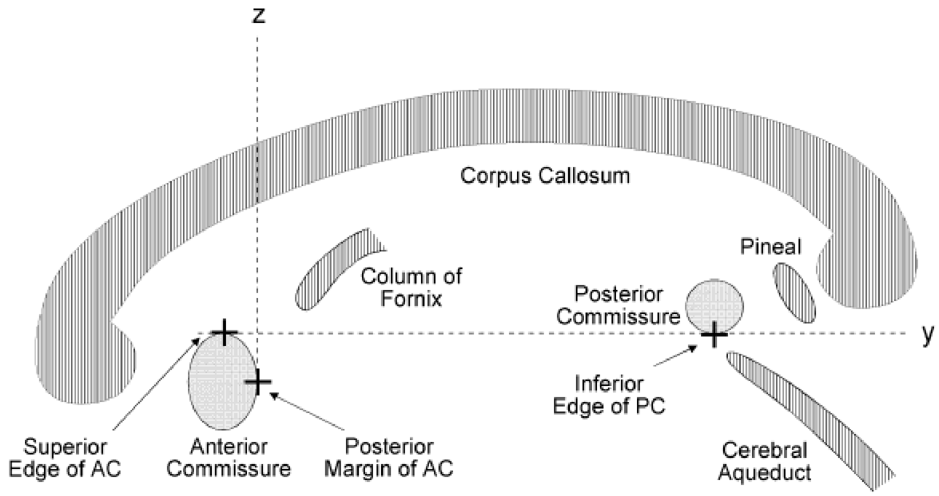
4.6 `Define Markers`

The transformation to the AC–PC aligned view, and from there to the Talairach view, is accomplished by means of 'markers'. These are anatomical landmarks that you manually select using the control column opened up by this control.

Only 3D anatomical datasets can have markers set; 3D+time datasets have the markers disabled. It is only possible to transform a 3D+time dataset to Talairach coordinates using a 'parent' 3D dataset, as described earlier (the '`X`'s and '`O`'s diagram).

4.6.1 Markers for the AC–PC Aligned Transformation

The AC–PC aligned view mode is defined by a rigid body transformation that makes the AC–PC line the new y-axis, makes the longitudinal fissure the new xz-plane, and makes the line perpendicular to that the new x-axis (right-to-left). The new origin is put at the intersection of the AC–PC line and the vertical line passing through the posterior margin of the AC. See [1] for details.



Cartoon of central brain, showing key anatomical locations

To select the landmarks, you must first choose `Allow edits`, so that *AFNI* will let you change the markers. After that, you choose the five landmarks:

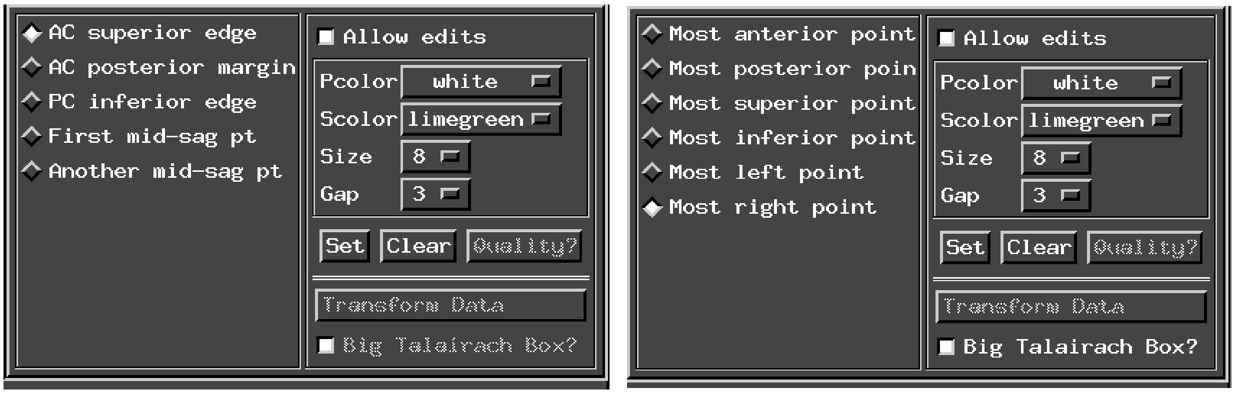
AC superior edge The highest point on the anterior commissure, in the mid-sagittal plane.

AC posterior margin The rearmost point on the anterior commissure, in the mid-sagittal plane.

PC inferior edge The lowest point on the posterior commissure, in the mid-sagittal plane.

First mid-sag pt Two points are required in the longitudinal fissure — they are used to define the new z-axis. (Two points are required to make sure that the new vertical plane is defined adequately; the mismatch between the AC–PC line and these two points must be less than 2°.)

Another mid-sag pt Should be at least 20 mm away from the first one.



Original (left) and AC-PC Aligned (right) `Define Markers` Control Panels

You select a landmark by depressing its button, then moving the crosshairs to the desired location, then clicking the `Set` button. A visible marker will appear. You can reset this point, or `Clear` it. When a marker is set, its toggle button will appear in inverted colors.

The atlas definitions of the marker locations are always in terms of a boundary. That is slightly ambiguous when it comes to discrete images: do you mark the last voxel visible in the structure, or the first voxel just outside the structure? My arbitrary solution to this problem is always to mark the last voxel visible in the structure. For example, in marking the 'top' of the AC, I move up until it is just no longer visible in the axial image. Then I drop back down one axial slice. It is possible to use 'warp-on-demand' to place markers at subvoxel locations — see the `Datamode` controls.

⋆⋆⋆ ⟶ **Sub-voxel marker locations**

On MR images with 1 mm^3 voxels and with good gray-white matter contrast, it is usually quite easy to set the AC markers. (At MCW, we use a GE Signa SPGR sequence for this purpose.) The PC is harder to spot, but it is always near the top of the cerebral aqueduct (in subjects with normal anatomy — it might be displaced by some pathological conditions; if you suspect this has happened, consult a neuroradiologist immediately!). Examination of the sample *AFNI* dataset, and some consultation with a neuroanatomy textbook will probably be helpful in learning to recognize the required anatomical landmarks.

4.6.2 Making the Transformation

When all markers are set, the `Quality?` button will become active. This button will check the marker set for elementary consistency. If the markers *don't* pass the test, an error message pops up to explain what happened. If they do pass the test, then the `Transform Data` button becomes active. When you press this, the new dataset will be created (in this case, the `+acpc` view of the anatomical dataset you were just marker-ing).

This new dataset will *not* have a `.BRIK` file on disk, just a `.HEAD` file. When *AFNI* comes to display this dataset, it will simply transform the needed data from the `+orig` dataset. If you wish, you may save the data voxels in the new view to disk using the `Datamode` controls. (If you wish to manipulate a transformed dataset using the auxiliary programs, you will have to save the data voxels to disk.)

4.6.3 Transformation to Stereotaxic (*i.e.*, Talairach) Coordinates

After you create the AC–PC aligned view, you can switch to it with the `AC-PC Aligned` button. At this point, you can set markers for the scaling to the Talairach view. Six markers are required to define the bounding box of the cerebral cortex. These should be set quite carefully, since it is often easy to mistake the sagittal sinus for cortex, which would give an erroneously large box in the $+z$-direction. Generally, I find that the most inferior point (in one of the temporal lobes) is the hardest to pick out.

In the `+acpc` → `+tlrc` transformation marker control panel, a toggle switch labeled `Big Talairach Box` appears just below the `Transform Data` button. In the earliest versions of *AFNI*, the brain data was clipped at 55 mm inferior to the AC-PC line, this being the location of the bottom of the Talairach-Tournoux atlas figures. In subsequent work, we found this was inadequate for cerebellar imaging. The new default clipping level is 65 mm inferior to the AC-PC line, and this is referred to as the `Big Talairach Box`. Datasets `.BRIK`s created with the old (small) box size are smaller than the new box size. There is no way to mix old and new box size datasets together when using the auxiliary analysis programs such as 3dANOVA. For this reason, and to maintain compatibility with old analyses, it is possible to write out old box size dataset `.BRIK`s by toggling this switch off. For all future work, I strongly recommend using the new Talairach box size.

The transformation carried out is precisely the one described in [1], and I recommend that anyone using *AFNI* read this atlas. The dataset is divided into 12 subvolumes:

- In x: Right-to-Midsagittal, Midsagittal-to-Left
- In y: Anterior-to-AC, AC-to-PC, PC-to-Posterior
- In z: Inferior-to-AC, AC-to-Superior

Each region is scaled separately to match the millimetric coordinates in the atlas:

x axis: AC–PC line to most left point of cerebrum	=	68 mm
x axis: AC–PC line to most right point of cerebrum	=	68 mm
y axis: Most anterior point of cerebrum to AC	=	70 mm
y axis: AC to PC	=	23 mm
y axis: PC to most posterior point of cerebrum	=	79 mm
z axis: Most inferior point of cerebrum to AC–PC line	=	42 mm
z axis: AC–PC line to most superior point of cerebrum	=	74 mm

The atlas brain is from an adult female, and so a typical adult male brain will be slightly compressed by these standard measurements. (Any jokes at this point will be sternly dealt with.) This transformation is then combined with the `+orig` → `+acpc` transformation, so that the data viewed in the Talairach mode is only interpolated once, not twice.

The whole procedure to produce the `+tlrc` view from the `+orig` view only takes a few moments, once you become adept at recognizing the requisite anatomical landmarks.

You will find that the marker toggle buttons and set/clear buttons are duplicated on the popup menu from the imaging windows. This is for convenience, since it is often necessary to peer closely at the screen to decide on a marker point, and I personally find it annoying to have to switch my attention to another window in order to set the marker. Also, note that depressing the toggle button (in the marker control column or on the popup menu) for an already set marker will cause *AFNI* to jump the crosshair viewpoint to that marker location.

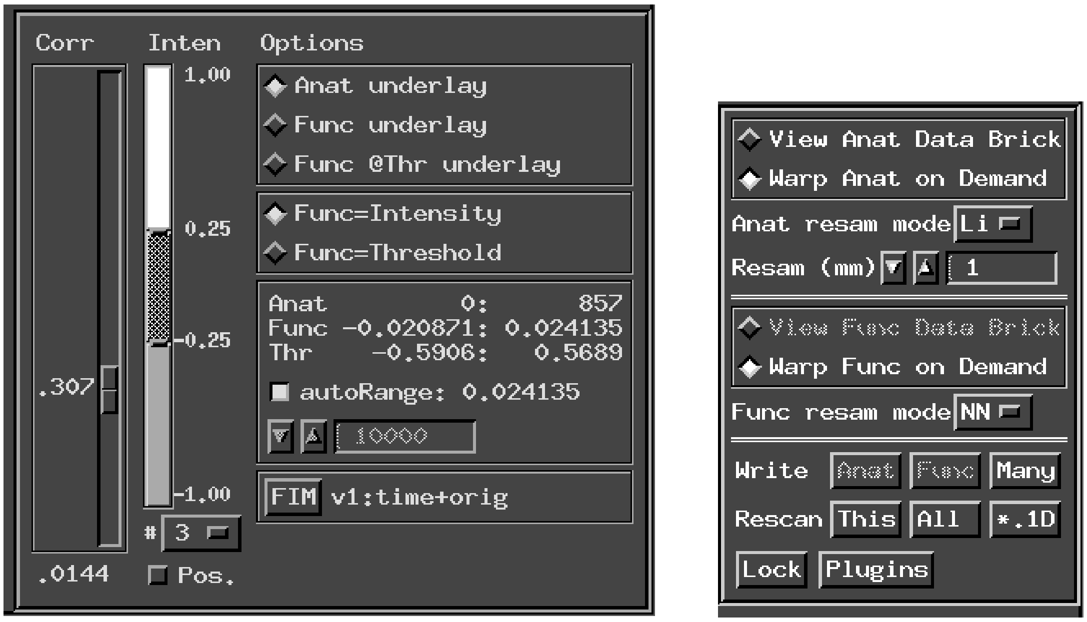
Finally, note that there are no markers available to be set in the Talairach view mode. At present, there are no transformations beyond this one — maybe someday.

4.6.4 Re-transformation and Re-creation of Datasets

If you decide to re-mark and re-transform an anatomical dataset, then the old transformed version will be deleted from disk before the new version is made. Not only that, but all the automatically manufactured transformed datasets that follow on this transformation will also be destroyed and remade. This can be disconcerting at first, but it is the only logical course of action for *AFNI* to take — otherwise, the transformed functional datasets would have been made with a different transformation than the new anatomical dataset on which they will be overlaid.

The only time this destruction is inappropriate is when the functional datasets in the transformed view are not in fact transformations from original datasets in the same session. This could happen if you copied a `+tlrc` dataset into a session from another directory, or if you created a `+tlrc` dataset using 3dmerge, say. *AFNI* will not delete a dataset if there is no 'parent' dataset from which it was warped.

AFNI will not let you mark and transform more than one anatomical dataset per session. Once you have transformed one dataset in a session, the others will be off-limits to direct marking and transformation. Their transformations will be derived from the 'master' dataset that you mark first.



`Define Function` and `Define Datamode` control panels

4.7 `Define Function`

The buttons and other 'widgets' on this control panel are used to manipulate how functional datasets are displayed in the viewer windows.

4.7.1 Threshold Slider

If the current functional dataset has a threshold data sub-brick (*i.e.*, is not of the `fim` type), then the first item in the Function control panel is a slider that lets you select the threshold to apply. Only voxels whose associated value is equal to or above this threshold will be overlaid in color. If the functional dataset type is `fico`, `fitt`, or `fift`, then *AFNI* knows how to interpret the threshold level in terms of the standard normal models for these statistics, and will show the corresponding 'p' value (per voxel) just below the slider. The dataset threshold type is shown at the top of the slider.

If the functional dataset is of the `fim` type, then this slider will not be visible. In this case, the leftmost item in the Function panel will be:

4.7.2 Color Pbar

The multi-colored vertical bar with numerical labels to the right, and a number selector labeled `#` below is called the 'pbar' (after its name in the C code). This device controls the colors for the functional overlay.

The number selector beneath the pbar control how many color panes are present: from 2 to 10 are available (the default in *AFNI* is set to 9 panes). Just below that is a toggle switch `Pos.`. This allows you to specify that only positive values will the shown in the color pbar, or that both positive and negative values will be shown. (Some functional datasets are naturally nonnegative; for these, allocating colors to the negative range is pointless.)

The color in any pane may be altered by clicking inside the pane itself. A chooser will then popup to let you select from the available palette (which is hard-coded into *AFNI*, and can only be changed via the `.Xdefaults` file). One color choice is 'none', which means that no color will display for that range of functional intensities, even if they appear in voxels that are over the selected threshold.

Each color pane applies to the indicated range of functional intensities. These intensities are relative to the `Range` control settings (to the lower right). The `Range` setting is is mapped to '`1.0`' on the pbar, and all other pbar settings correspond to the similarly scaled values in the functional dataset sub-brick being viewed. You may click-and-drag on the 'sashes' between the color panes to change the intensity thresholds.

4.7.3 Options

The options column is a grab bag of functional dataset stuff. The first boxed set of buttons (`Anat underlay`, etc.) lets you choose whether the anatomical dataset or the functional dataset appears as the background (grayscale) images in the viewing windows. The second boxed set (`Func=Intensity`, etc.) lets you choose between displaying the intensity or the threshold as the color-determining function.

The third boxed set comprised the range controls for the functional coloring. At the top are the minimum and maximum values found in the current datasets (*cf.* auxiliary program 3dinfo). The next control is a toggle labeled `autoRange:  xxxx`. The value represented by `xxxx` is the automatically assigned range for the functional range (which corresponds to `1.0` on the pbar). This `autoRange` is chosen by *AFNI* as the largest absolute value in the dataset sub-brick. If the toggle is off, then the control below is activated, and allows you to specify the functional value that maps to `1.0` on the pbar. Controlling this range may be desirable when comparing several datasets.

The lowest (so far) control in this column provides another `FIM` menu button (see the Graph section). The label to the right of this button shows the dataset that will be processed once the `Compute FIM` button is pressed.

4.8 Define Datamode

This control column determines how datasets are manipulated by *AFNI*. It also contains some miscellaneous controls that didn't 'fit in' elsewhere.

4.8.1 Resampling

⋆⋆⋆ ⟶ **Warp-on-demand; subvoxel markers**

The top boxed set of controls lets you choose how the anatomical data that is actually displayed will be generated. You can view the data directly from the `.BRIK` file, if it is available. In this case, you are limited to seeing the dataset at the voxel resolution at which the data was generated. Alternatively, you can choose `Warp Anat on Demand` viewing, which means that the data will be interpolated from its source to whatever resolution you order (using the controls just below). In this way, it is possible to place markers at subvoxel locations.

At present, *AFNI* cannot display graphs from a warp-on-demand dataset. If you take an action that causes a 3D+time dataset to be switched to warp-on-demand mode, then any open graph windows will be destroyed. This will happen, for example, if you switch from `+orig` to `+tlrc` view, and have not yet written the 3D+time dataset to disk.

⋆⋆⋆ ⟶ **Resampling modes**

The interpolation modes are nearest-neighbor ('`NN`'), linear ('`Li`'), cubic ('`Cu`'), and 'blocky' ('`Bk`') [this latter is intended mainly for functional datasets, and is intermediate between `NN` and `Li`; its mathematical definition is at the end of this manual]. `NN`, `Li`, and `Bk` are fairly rapid on a decent workstation, but `Cu` can be noticeably slow. Since it uses 64 neighboring grid points to interpolate, vs. 8 for `Li` and `Bk`, and it uses more complex formulas, this is understandable. For most purposes, `Li` interpolation for anatomical images and `Bk` interpolation for functional images will be the best. **N.B.**: threshold data in functional datasets is *always* resampled using the `NN` mode. This is because it is somewhat unreasonable to interpolate a nonlinear statistic (such as correlation coefficient) between voxels, and then to interpret this statistic using probabilistic models that assume independence.

The smallest resolution allowed by the `Resam (mm)` selector is 0.1 mm. This is very tiny, and images will display very slowly. You can do it if you wish; however, don't try to write out a whole human head dataset to disk at this resolution! The disk space required would be rather large.

Controls for determining whether the functional dataset `.BRIK` (if available) or warp-on-demand will be used for computing the functional slices are the next set down. The functional `.BRIK` can be used only if it actually exists in the current view (*i.e.*, coordinate system) and if it is at the same spatial resolution as the anatomical dataset being viewed.

4.8.2 Dataset Output and Input

The `Write` buttons will compute and write the `.BRIK` files to disk for the indicated datasets. The current resampling mode and resampling dimension will be used for this purpose. The `Anat` button will output the current anatomical dataset; the `Func` button outputs the current functional dataset. The `Many` button allows you to select more than one dataset (from all sessions) and write them all out. This new control is provided since the resampling process can be relatively slow — now you can select many datasets, start their output in Talairach coordinates, and then go get something good to eat while *AFNI* churns away.

⋆⋆⋆ ⟶ *AFNI* will not write over a dataset .BRIK which cannot be recreated by warping from a parent. (This precaution does not extend to plugins, which can be written so as to destroy unrecoverable .BRIKs.) It *is* possible to use *AFNI* to resample a dataset in the +orig view, but only if the dataset is warp-on-demand from another dataset as parent. This can be arranged using the 3ddup auxiliary program.

The Rescan buttons are designed to re-read data from disk. The first one, This, will re-read the current session. This is useful if you use an auxiliary program such as 3dANOVA to create a new dataset outside of *AFNI*, and then wish to view it. In earlier versions, it was necessary to exit *AFNI* and restart it to get new datasets into the program. The All button will re-read all the sessions that were initially loaded (there is no facility to read in an entire new session at this time). Finally, the *.1D button will re-read the time series directories, and load any new time series files that are found.

4.8.3 Controller Lock

With the New button (lower left of *AFNI* controller window), it is possible to view several datasets at once in several sets of viewing (and graphing) windows. Normally, the viewpoints of the separate controllers are independent; that is, clicking in the sagittal window of controller [A] will change the viewpoint of [A]'s coronal and axial windows, but will have no effect on [B]'s image and graph windows.

Under some circumstances, you may wish to lock some controllers together, so that their viewer windows move in unison. This can be done with the Lock menu. There is only one lock in *AFNI*. This menu determines which controller windows participate in the lock. If the spatial viewpoint is changed in any window that is affected by the lock, then *AFNI* will attempt to jump all other locked windows to the corresponding viewpoint.

In addition, the Lock menu has a Clear button, to detach all controllers from the lock, and an Enforce button, which can be used to make all viewer windows affected jump to the locked position. (Enforce is only needed just after choosing which controllers to be locked together.)

A couple of applications for the lock:

⋆⋆⋆ ⟶ **Try this!**

- Scrolling through several anatomies in Talairach coordinates, simultaneously viewing the similarities and differences.
- Viewing a dataset in Original and Talairach coordinates simultaneously.
- Comparing several different functional datasets overlaid on the same anatomy.

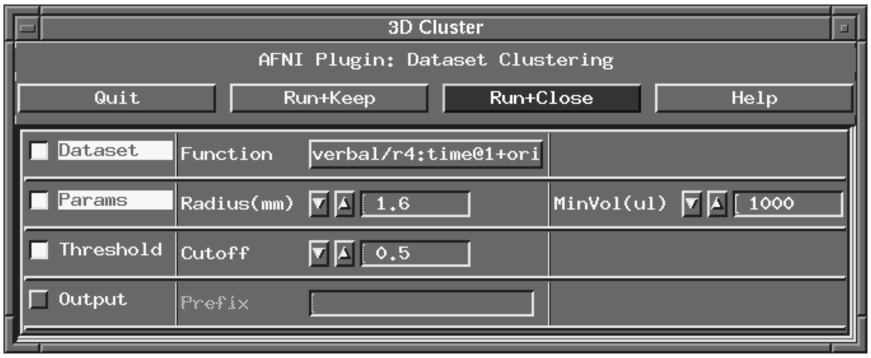
AFNI can have trouble when the lock is used between controllers in different coordinate systems. The lock subroutine will work properly if the dataset being viewed in one controller is just the realization of the other controller's dataset in a different coordinate system — it will carry out the Talairach transformation (or its inverse) as needed to keep the lock anatomically reasonable. It will fail if the two controllers have different coordinate systems and different datasets; for example, it doesn't know how to transform from Talairach coordinates in one dataset to Original coordinates in another dataset.

4.8.4 Plugins

The last item in the `Datamode` control panel is the `Plugins` menu button. This will only be present if *AFNI* is compiled with plugin support, and if the program finds at least one plugin when it starts up.

Plugins are external C functions, written in conformance with the plugins manual, that provide extra functionality to *AFNI*. They are compiled into 'shared objects' (or 'shared libraries') with the filename suffix `.so` (or `.sl` on HP-UX). *AFNI* searches for them in a set of directories specified in the shell environment variable `AFNI_PLUGINPATH`. If this is not defined, then `PATH` is used; that way, storing the compiled plugins in the same place as the MCW *AFNI* executables will work.

Each plugin will create one or more 'interface panels'. When you select a plugin from the `Plugins` menu, its interface panel will pop up. At that point, you fill in the desired parameters, and then execute the actual plugin code with one of the `Run` buttons.



Clustering plugin (`plug_clust.c`) interface panel

5 Command line switches

The general form for the *AFNI* command line is

`afni [options] [session_directory ...]`

where the options, listed below, all start with the character '`-`'. If no session directories are entered, then the program acts as if the user had typed '`./`' for the session, which means that the current working directory will be scanned for datasets.

What I consider to be the more useful options are listed below. A complete list can be found by entering the command '`afni -help`'.

- `-purge`
  Conserves memory by purging datasets to disk when not in use. Use this if you run out of memory when running AFNI. This will slow the code down, so use only if needed.

  Another application for the `-purge` switch arises when using Unix symbolic links to make a dataset appear as if it is in more than one session directory at once (without copying the brick file). For example, suppose one creates (using `3dmerge`) an averaged anatomical dataset: GRP+tlrc.HEAD and GRP+tlrc.BRIK, from sessions fred, ethel, and lucy. This could be done (from the directory above the sessions) by the commands

```
3dmerge -prefix GRP -gmean */anat+tlrc.HEAD
ln -s GRP+tlrc.HEAD fred/GRP+tlrc.HEAD
ln -s GRP+tlrc.BRIK fred/GRP+tlrc.BRIK
ln -s GRP+tlrc.HEAD ethel/GRP+tlrc.HEAD
ln -s GRP+tlrc.BRIK ethel/GRP+tlrc.BRIK
ln -s GRP+tlrc.HEAD lucy/GRP+tlrc.HEAD
ln -s GRP+tlrc.BRIK lucy/GRP+tlrc.BRIK
```

The ln -s commands put links to the GRP anatomy dataset into each of the session directories. In this way, if the user runs *AFNI* with the command

```
afni fred ethel lucy
```

then each of the sessions will have access to the GRP dataset, but there will only be one physical copy of it on disk.

A problem arises with this scheme because of the manner in which *AFNI* accesses large brick files. The program will not recognize that the links point to the same actual file, and it will attempt to mmap the GRP+tlrc.BRIK file more than once. This will likely fail, and the program will crash when the second access is attempted (when the user switches to the second session).

Using the -purge switch will avoid this problem. Before a new dataset is accessed (when switching sessions, anatomies, or functions), existing dataset bricks will be be munmap-ed. This will prevent the attempt to mmap the same brick file twice at the same instant.

Another way to approach this problem would be through the use of the 3ddup program to create warp-on-demand copies of the GRP+tlrc dataset in each of the session directories.

- -R
  Recursively searches each session_directory for more session subdirectories. This will descend the entire filesystem hierarchy from each session_directory given on the command line. On a large disk, this may take a long time. To limit the recursion to 5 levels (for example), use -R5.

- -ignore N
  Tells the program to ignore the first N points in time series for graphs and FIM calculations.

- -unique
  Tells the program to create a unique set of colors for each AFNI controller window. This allows different datasets to be viewed with different grayscales or colorscales. -unique will only work on 12-bit PseudoColor displays (for example, SGI workstations).

- `-ncolors nn`
  Tells *AFNI* to use `nn` gray levels for the image displays (default is 80). Since *AFNI* always uses the default colormap, on a 8-bit graphics system you may run out of colors if you run several graphics programs at once. WWW browsers are notorious for causing this problem, since they usually allocate many colormap entries and hold onto them (just like *AFNI* does).

6 Technical Notes

The MCW *AFNI* package is distributed as a compressed Unix `tar` file: `afni96.tgz`. It is unpacked with the command

```
gzip -dc afni96.tgz | tar xf -
```

The files will go into a directory named `AFNI96`.

AFNI requires an ANSI C compiler, X11R5, and Motif 1.2. It also requires that the default X11 Visual be an 8- or 12-bit PseudoColor visual; the program will not work with anything else. This is ordinarily not a problem except on very low end and very high end workstations and/or X-terminals.

Several `Makefile`s are included in the distribution. The machines they are intended for are indicated by their filename suffix. In general, you will have to modify one of these to fit your needs. You may also need to edit the file `machdep.h` to set up the flags for the `mmap` routine appropriately.

Copy the appropriate file to be `Makefile`. Examine it to make sure that it makes sense on your system! To build the executables, use the command `make all`. If you set the `INSTALLDIR` macro correctly in the `Makefile`, then `make install` will `mv` the executable images to their final resting spot. After that, a `make clean` is appropriate.

To make and install the plugins supplied with *AFNI*, the commands `make plugins` and `make install_plugins` will work. Not all system specific `Makefile`s have the commands needed to compile plugins. This is because I do not have access to such computer systems.

There are undoubtedly still bugs in this software. Suggestions for further improvements will be gladly received, but no action on such suggestions can be guaranteed! The e-mail address for *AFNI* comments is `rwcox@mcw.edu`.

6.1 mmap-ing

AFNI uses the Unix function `mmap` to access the `.BRIK` files. This is what makes it possible to read in many large datasets and not choke the memory or swap space of the computer. Data is only read from disk using `mmap` — all writes are done using `fwrite`.

If you need to disable the use of `mmap`, edit the file `3ddata.h` and `#define MMAP_THRESHOLD` to be `-1`. This will make *AFNI* use `malloc` and `fread` to access the `.BRIK` data. This will also strongly limit the number of datasets that can be used.

If you receive a message that *AFNI* cannot load a dataset into memory (or `mmap` it), then you should restart the program with the `-purge` option. This will force datasets not in immediate use to be purged from memory and to be `munmap`-ed, which might solve your problem. This will also make the program run slower when you switch between datasets.

6.2 `machdep.h`

This C header file contains machine specific settings. If you are porting *AFNI* to a system not available at MCW, you will have to create a `Makefile` appropriate for your computer, and will have to edit `machdep.h` to set various flags correctly. In particular, flags for `mmap` and dynamic loading of plugins must be set correctly. Comments in this file describe the options that are available.

6.3 X11 Resources for *AFNI*

Included in the MCW *AFNI* distribution is a file called `AFNI.Xdefaults`. This contains examples of how various features of *AFNI* can be controlled using X11 resources.

6.4 Formula for `Bk` Resampling

Define the cardinal basis function

$$\phi(x) = \begin{cases} 1 - 8x^4 & 0 \le |x| \le \frac{1}{2} \\ 8(1-|x|)^4 & \frac{1}{2} < |x| < 1 \\ 0 & 1 \le |x| \end{cases}$$

Then 'blocky' interpolation in 3D of a function defined on a grid with spacings $(\Delta x, \Delta y, \Delta z)$ to an arbitrary point in space is

$$f(x,y,z) = \sum_i \sum_j \sum_k f(i\Delta x, j\Delta y, k\Delta z)\,\phi\left(\frac{x - i\Delta x}{\Delta x}\right)\phi\left(\frac{y - j\Delta y}{\Delta y}\right)\phi\left(\frac{z - k\Delta z}{\Delta z}\right) .$$

The actual implementation of interpolation is done in a somewhat different fashion, for the sake of efficiency. See the source code in `afni_slice.c`.

7 Acknowledgements

Many thanks to Jim Hyde for much support and many discussions on the direction of FMRI analyses. Thanks also are due to Andrzej Jesmanowicz for forging the way with *AFNI*'s grandfather, FD. Doug Ward has contributed a lot with the 3dfim and 3dANOVA programs, new features in 3dmerge, plus the creation of the auxiliary programs manual. Mike Beauchamp has aided immeasurably by testing earlier versions of this software and by coming up with many useful ideas. Jay Kummer contributed the initial idea for plugins. Many other people at MCW have also helped, particularly with "quick questions" (you know who you are) and the occasional warm pumpernickel bagel. This work was partly supported by the United States NIH through grants MH51358 and NS34798, for which I am also grateful.

References

[1] Jean Talairach and Pierre Tournoux, *"Co-Planar Stereotaxic Atlas of the Human Brain"*, Thieme Medical Publishers, New York, 1988.

[2] Peter A. Bandettini, Andrzej Jesmanowicz, Eric C. Wong, and James S. Hyde, Processing strategies for time-course data sets in functional MRI of the human brain. *Magn. Reson. Med.* **30**, 161–173 (1993).

[3] Robert W. Cox, AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical Research* **29**, 162–173 (1996).

[4] Robert W. Cox, Andrzej Jesmanowicz, and James S. Hyde, Real-time functional magnetic resonance imaging. *Magn. Reson. Med.* **33**, 230–236 (1995).

Disclaimer:

MCW *AFNI*, its associated programs, and its documentation are provided as is, and no warranty for their correctness or usefulness for any purpose is made or implied by the Medical College of Wisconsin (MCW), or by the author of the software. Neither MCW nor the author accepts any liability for any defects in this software or its manuals, or for any damages caused by use of this software.

Ownership, Conditions of Use, and Restrictions:

- Permission is granted to make use of and to make copies of the MCW *AFNI* software and documentation for non-commercial research purposes only. Ownership of MCW *AFNI* and all copies is retained by the Medical College of Wisconsin.
- Patient-care applications are not recommended. MCW *AFNI* has not been evaluated by or approved by the United States Food and Drug Administration.
- Use for *any* purpose by for-profit organizations is prohibited without prior arrangement and written permission.
- Redistribution of MCW *AFNI*, or any derived work, outside the receiving institution is prohibited without prior permission.
- Copies may be made within the receiving institution without separate permission from the Medical College of Wisconsin.
- Technical support (*e.g.*, the fixing of bugs) for MCW *AFNI* is not guaranteed.

I agree to abide by the terms and restrictions above.

SIGNATURE:

NAME:
(print clearly, or type)

DATE:

E-MAIL ADDRESS:
(print clearly, or type)

To Register

Copy this page onto your departmental or institutional letterhead, sign and date, include your e-mail address, and return via mail or FAX to

Robert W. Cox, PhD
Biophysics Research Institute
Medical College of Wisconsin
8701 Watertown Plank Road
Milwaukee WI 53226 USA
FAX: 414–456–6512

Instructions on how to obtain *AFNI* by `anonymous ftp` will be sent by e-mail.