MCW AFNI 2.22 — Auxiliary Programs

B. Douglas Ward and Robert W. Cox

August 27, 1999

Abstract

A number of auxiliary programs are included with the *AFNI* 2.22 distribution. Those programs are documented here. The auxiliary programs may be roughly divided into two categories: The older "im" programs provide the capability to manipulate and study 2-dimensional (planar) images, whereas the newer "3D" programs are for manipulation and analysis of 3-dimensional (volume) images.

Among the many auxiliary programs are programs for converting 2D datasets into AFNI 3D datasets, and vice versa (to3d, from3d, 3drefit), programs for image processing (3dvolreg, 3drotate, 3dnoise, 3dIntracranial), programs to combine, edit, and analyze the 3D datasets (3dmerge, 3dclust, 3dcalc), programs to perform time series analysis on individual 3D+time datasets (3dfim, 3dNLfim, 3dDeconvolve), and programs to perform statistical analysis on collections of 3D datasets (3dttest, 3dANOVA, 3dMannWhitney, 3dRegAna, etc.).

People who use *AFNI*, particularly newcomers, are often confused about which program to use, or which programs are available, to accomplish a specific objective. For this reason, a number of changes have been made to the current documentation. *Every* stand-alone program that is included with the *AFNI* distribution is referenced here. For most programs, this file contains the complete documentation. For others, there is a reference to the external file containing documentation for that program. Also, the table of contents has been replaced with a table which gives a very brief description of every program. For those who are new to AFNI, and even for experieced users, the table on the following pages may be a good place to start if you have a specific objective, but do not know how to proceed.

This manual describes the purpose and features of the auxiliary programs, gives the command line syntax for execution of the programs, describes the options, and (in most cases) provides one or two examples illustrating possible applications of the programs.

Table of Programs

Program	Description	Page
1dplot	Graph columns of *.1D type time series to screen	004
24swap	Swaps byte pairs and/or quadruples on listed files	005
2dImReg	Slice-by-slice image registration of FMRI 3D datasets	006
2swap	Swap byte pairs for inter-operating system compatibility	007
3T_toafni	Send information about imaging sequence to AFNI	008
3dANOVA	Single factor Analysis of Variance for FMRI 3D datasets	008
3dANOVA2	Two factor Analysis of Variance for FMRI 3D datasets	009
3dANOVA3	Three factor Analysis of Variance for FMRI 3D datasets	009
3dDeconvolve	Deconvolution analysis of FMRI 3D time series data	010
3dFWHM	Estimation of image Filter Width Half Maximum	011
3dFriedman	Nonparametric Friedman test for blocked multiple sample	011
3dIntracranial	Automatic segmentation of intracranial region	012
3dKruskalWallis	Nonparametric Kruskal-Wallis test for multiple samples	012
3dMannWhitney	Nonparametric Mann-Whitney rank-sum two-sample test	013
3dNLfim	Nonlinear Regression Analysis of FMRI time series data	013
3dRegAna	Linear Regression Analysis of FMRI 3D datasets	014
3dTSgen	Generates random signal+noise 3D datasets	015
3dTcat	Concatenate sub-bricks into one 3D+time dataset	015
3dTsmooth	Smooth each voxel time series in a 3D+time dataset	017
3dWilcoxon	Nonparametric Wilcoxon signed-rank paired two-sample	018
3daxialize	Write out data brick oriented as axial slices	018
3dbuc2fim	Convert bucket sub-bricks to fim (fico, etc.) datasets	019
3dbucket	Concatenate individual sub-bricks into bucket dataset	020
3dcalc	Do arithmetic on 3D datasets, voxel-by-voxel	022
3dclust	Cluster detection and statistical summary	026
3ddup	Make duplicate copy of a 3D dataset	029
3dfim	Cross correlation analysis of FMRI 3D time series data	030
3dfractionize	Resample a mask dataset from a fine grid to a coarse grid	034
3dhistog	Compute histogram from FMRI 3D dataset	035
3dinfo	Print out useful information from a 3D dataset header file	036
3dmaskave	Compute average of all voxels specified by a 3D mask	039
3dmerge	Edit, cluster, filter, and merge FMRI 3D datasets	040
3dnewid	Assign a new ID code to a dataset	049
3dnoise	Set voxels below noise threshold to zero	049
3dnvals	Print out the number of sub-bricks in a 3D dataset	050
3dpc	Principal Component Analysis of 3D datasets	050
3dproject	Projection along cardinal axes from a 3D dataset	051
3drefit	Change information in a 3D dataset's header	053
3drotate	Rotate and/or translate all bricks from a 3D dataset	056
3dttest	Perform t-test for sets of FMRI 3D datasets	057
3dvolreg	Register each input 3D sub-brick to a base brick	061

Table of Programs (continued)

Program	Description	Page
4swap	Swap byte quadruples on the files listed	063
AlphaSim	Estimate stat. significance via Monte Carlo simulation	064
FD2	Visualization of FMRI 2D datasets	064
RSFgen	Generate random stimulus functions	064
abut	Put noncontiguous FMRI slices together (for to3d)	065
adwarp	Resample dataset to grid defined by 'anat parent' dataset	066
afni	Visualization of FMRI 3D datasets	067
byteorder	Indicates host CPU byte order	067
ccalc	Perform interactive arithmetic calculations	067
cdf	Calculates various cumulative distribution probabilities	068
count	Generates strings of numbers	068
fim2	Cross correlation analysis of FMRI 2D time series data	069
float_scan	Scan input file of floating point numbers for illegal values	076
from3d	Extract 2D data files from 3D datasets	077
ftosh	Convert float 2D images to short 2D images	079
imand	Produce logical "and" of a sequence of input images	080
imaver	Compute mean and std. dev. of sequence of 2D images	080
imcalc	Do arithmetic on 2D images, voxel-by-voxel	081
imdump	Prints out non-zero pixels in a 2D image	082
immask	Apply mask to input 2D image	083
imreg	Register a sequence of 2D images	084
imrotate	Rotate and/or translate a sequence of 2D images	088
imstack	Stack up a set of 2D images into one big file	089
imstat	Calculate statistics for one or more images	089
imupsam	Upsample the input 2D image	090
mritopgm	Convert an image to raw pgm format	091
nsize	Zero pads 2D image to next larger power of 2	091
p2t	Calculate tail probabilities	092
sfim	Selective averaging of 2D image time series	093
sqwave	Creates an ideal square wave time series file	094
tfim	Perform t-test for sets of 2D images	096
to3d (batch)	Convert 2D images into 3D datasets for AFNI	097
to3d (interactive)	Convert 2D images into 3D datasets for AFNI	105
waver	Creates an ideal waveform time series file	111

1 Program 1dplot

1.1 Purpose

Graphs the columns of a *.1D type time series file to the screen.

1.2 Usage

1dplot [options] tsfile

1.3 Options

-sep = Plot each column in a separate sub-graph.

-one = Plot all columns together in one big graph. [default = -sep]

-dx xx = Spacing between points on the x-axis is 'xx' [default = 1]

-ignore nn = Skip first 'nn' rows in the input file [default = 0]

-xlabel aa = Put string 'aa' below the x-axis [default = no axis label]

-ylabel aa = Put string 'aa' to the left of the y-axis [default = no axis label]

-ynames aa bb ... = Use the strings 'aa', 'bb', etc., as labels to the right of the graphs, corresponding to each input column. These strings CANNOT start with the '-' character.

-volreg = Makes the 'ynames' be the same as the 6 labels used in plug_volreg for Roll, Pitch, Yaw, I-S, R-L, and A-P movements, in that order.

You may also select a subset of columns to display using a tsfile specification like 'fred.1D[0,3,5]', indicating that columns #0, #3, and #5 will be the only ones plotted. For more details on this selection scheme, see the output of '3dcalc -help'.

1.4 Example

To graph a 'dfile' output by 3dvolreg, when TR=5, use:

```
1dplot -volreg -dx 5 -xlabel Time 'dfile[1..6]'
```

2 Program 24swap

2.1 Purpose

Swap byte pairs and/or quadruples on the files listed.

2.2 Usage

The command line format for program 24swap is as follows:

24swap [options] file ...

2.3 Options

-q Operate quietly (i.e., suppress output to the screen).

-pattern pat 'pat' determines the pattern of 2 and 4 byte swaps. Each element is of the form 2xN or 4xN, where N is the number of bytes to swap as pairs (for 2x) or as quadruples (for 4x). For 2x, N must be divisible by 2; for 4x, N must be divisible by 4. The whole pattern is made up of elements separated by colons, as in '-pattern 4x39984:2x0'. If bytes are left over after the pattern is used up, the pattern starts over. However, if a byte count N is zero, as in the example, then it means to continue until the end of the file.

2.4 Notes

- A default pattern can be stored in the Unix environment variable

  AFNI_24SWAP_PATTERN.

  If no -pattern option is given, the default will be used. If there is no default, then nothing will be done.

- If there are bytes 'left over' at the end of the file, they are written out unswapped. This will happen if the file is an odd number of bytes long.

- If you just want to swap pairs, see program 2swap. For quadruples only, see program 4swap.

- This program will overwrite the input file! You might want to test it first.

2.5 Example

24swap -pat 4x8:2x0 fred

If fred contains	'abcdabcdabcdabcdabcd'	on input,
then fred has	'dcbadcbabadcbadcbadc'	on output.

3 Program 2dImReg

3.1 Purpose

This program performs 2d image registration. Image alignment is performed on a slice-by-slice basis for the input 3d+time dataset, relative to a user specified base image.

3.2 Usage

The command line format for program 2dImReg is as follows:

2dImReg -input fname [**-basefile fname**] [**-base num**] [**-nofine**] [**-fine blur dxy dphi**] **-prefix pname** [**-dprefix dname**] [**-dmm**] [**-rprefix rname**] [**-debug**]

3.3 Options

-input fname Filename of input 3d+time dataset to process

-basefile fname Filename of 3d+time dataset for base image (default = current input dataset)

-base num Time index for base image (0 <= num) (default: num = 3)

-nofine Deactivate fine fit phase of image registration (default: fine fit is active)

-fine blur dxy dphi Set fine fit parameters

where:

blur	= FWHM of blurring prior to registration (in pixels)	(default: blur = 1.0)
dxy	= Convergence tolerance for translations (in pixels)	(default: dxy = 0.07)
dphi	= Convergence tolerance for rotations (in degrees)	(default: dphi = 0.21)

-prefix pname Prefix name for output 3d+time dataset

-dprefix dname Write files 'dname'.dx, 'dname'.dy, 'dname'.psi containing the registration parameters for each slice in chronological order.

File formats:

'dname'.dx:	time(sec)	dx(pixels)
'dname'.dy:	time(sec)	dy(pixels)
'dname'.psi:	time(sec)	psi(degrees)

-dmm Change dx and dy output format from pixels to mm

-rprefix rname Write files 'rname'.oldrms and 'rname'.newrms containing the volume RMS error for the original and the registered datasets, respectively.

File formats:

'rname'.oldrms:	volume(number)	rms_error
'rname'.newrms:	volume(number)	rms_error

-debug Lots of additional output to screen

4 Program 2swap

4.1 Purpose

Different computers use different formats for storage of two byte (short) integers. Some computers store the most significant (high) byte first, followed by the least significant (low) byte. Other computers store the low byte first, followed by the high byte. Program 2swap is provided to facilitate exchange of data between computers using different formats for short integers. The program simply swaps byte pairs on the files specified by the user. This program should be run on the *.BRIK data files only; it should *never* be run on *.HEAD files (which store data in ASCII format).

4.2 Usage

The command line format for program 2swap is as follows:

2swap [-q] file ...

The -q option means to work quietly (i.e., suppress output to the screen).

4.3 Examples

Example 1. A user has transferred the AFNI dataset fred.anat+orig (.HEAD and .BRIK) from a computer with an Intel CPU to a RISC workstation. Since these computers use different formats for storage of short integers, it will be necessary to swap bytes in the .BRIK file prior to viewing the dataset with program afni. The appropriate command line is:

2swap fred.anat+orig.BRIK

The computer responds with:

- opened fred.anat+orig.BRIK.........

File fred.anat+orig.BRIK now contains the byte-swapped data, suitable for viewing with afni.

Example 2. A group of 20 AFNI datasets is to be transferred from a RISC workstation to an Intel CPU computer. The data files have the names fred.func01+orig (.HEAD and .BRIK), ..., fred.func20+orig(.HEAD and .BRIK). Instead of entering the 2swap command 20 times (once for each .BRIK file), the wildcard character "*" can be used (assuming that the transferred files are the only files beginning with "fred.func" in the current directory):

2swap fred.func*.BRIK

The computer responds with:

- opened fred.func01+orig.BRIK.........
- opened fred.func02+orig.BRIK.........

etc.

- opened fred.func20+orig.BRIK.........

As a result, all .BRIK files which were transferred to the current directory have their bytes swapped.

5 Program 3T_toafni

5.1 Purpose

Get information about current imaging sequence from the MCW Bruker 3T/60 scanner, format it for AFNI, and send it to AFNI. The output is written to stdout (AFNI will pipe it into itself).

5.2 Usage

3T_toafni [**-dummy**]

5.3 Options

-dummy If the -dummy option is used, then the data from the scanner will be read from stdin instead of from the scanner console computer.

6 Program 3dANOVA

6.1 Purpose

Program 3dANOVA was developed to perform single-factor analysis of variance on collections of *AFNI* 3-dimensional data sets, voxel by voxel. Through the command line inputs, the user specifies which *AFNI* data sets are to be used in the analysis, and to which factor level they belong. Various output options are available, including the F-test for equality of factor level means, estimation of individual factor level means, estimation of the difference between two factor level means, and estimation of contrasts. The resulting output may be stored either as multiple *AFNI* 2 sub-brick datasets, or as a single *AFNI* "bucket" type dataset.

6.2 Further Details

For further details, see Section 1 of *Analysis of Variance for FMRI Data*, contained in file 3dANOVA.ps.

7 Program 3dANOVA2

7.1 Purpose

Program 3dANOVA2 was developed to perform two-factor (or two-way) analysis of variance on *AFNI* 3-dimensional data sets. Through the command line inputs, the user specifies which *AFNI* data sets are to be used in the analysis, and to which factor levels they belong. Various output options are available, including the F-test for factor interaction, F-test for equality of factor level means, estimation of individual factor level means, estimation of the difference between two factor level means, and estimation of contrasts. For the "fixed effects model" (described below), additional output includes estimation of individual cell (treatment) means, differences in cell means, contrasts in cell means, and their corresponding statistics. The resulting output may be stored either as multiple *AFNI* two sub-brick datasets, or as a single *AFNI* "bucket" type dataset.

Program 3dANOVA2 requires **equal sample sizes** for all combinations of factor levels.

7.2 Further Details

For further details, See Section 2 of *Analysis of Variance for FMRI Data*, contained in file 3dANOVA.ps.

8 Program 3dANOVA3

8.1 Purpose

Program 3dANOVA3 was developed to perform crossed and crossed-nested three factor analysis of variance (ANOVA) on *AFNI* 3-dimensional data sets. Five different ANOVA models are available, which are described below. Although these models do not cover every possible situation for three factor ANOVA, they should suffice for most practical purposes.

The command line inputs allow the user to specify which *AFNI* data sets are to be used in the analysis, and to which factor levels they belong. Various output options are available, including F-tests for main factor effects, F-tests for factor interactions, estimation of individual factor level means, estimation of the difference between two factor level means, and estimation of contrasts. For the "fixed effects model" (Model 1, described below), additional output includes estimation of individual cell (treatment) means, and differences in cell means, along with their corresponding statistics. The resulting output may be stored either as multiple *AFNI* 2 sub-brick datasets, or as a single *AFNI* "bucket" type dataset.

Program 3dANOVA3 requires **equal sample sizes** for all combinations of factor levels.

8.2 Further Details

For further details, see Section 3 of *Analysis of Variance for FMRI Data*, contained in file 3dANOVA.ps.

9 Program 3dDeconvolve

9.1 Purpose

Program 3dDeconvolve was developed to provide deconvolution analysis of FMRI time series data. This has two primary applications: (1) estimation of the system impulse response function, and (2) multiple linear regression analysis of time series data. Given the input stimulus function(s), and the measured FMRI signal data, program 3dDeconvolve first estimates the impulse response function(s); the impulse response function(s) is then convolved with the stimulus time series to yield the estimated response. Various statistics are calculated to indicate the "goodness" of the fit.

The capability of fitting *multiple* stimulus (or reference) waveforms differentiates program 3dDeconvolve from the cross-correlation analysis programs (such as AFNI fim and 3dfim). Another way that program 3dDeconvolve differs from the cross-correlation analysis programs is in the model for the output waveform. The cross-correlation programs model the system response as a scaled version of a fixed waveform (such as a square wave or a sine wave). Program 3dDeconvolve uses a sum of scaled and time-delayed versions of the stimulus time series. The data itself determines (within limits) the functional form of the estimated response. In fact, the shape of the fitted waveform can vary from voxel to voxel.

The input to program 3dDeconvolve consists of an *AFNI* 3d+time data set, along with one or more input stimulus time series. Output consists of the estimated system impulse response function, along with the statistical significance of the fit of this impulse response function to the original FMRI data, for each voxel in the dataset. The program calculates the F-statistic for significance of the multiple regression, as well as t-statistics for each of the impulse response function parameters. In the case of multiple input stimuli, the program calculates the partial F-statistics for significance of each individual stimulus.

Two new features have been added to the current version of program 3dDeconvolve. First, the -nodata option, which allows the user to evaluate an experimental design prior to collecting data. Specifically, the experimental design matrix can be tested for multicollinearity. Also, the relative accuracies of different hypothetical experimental designs can be evaluated. The second addition is the -glt option. This option allows the user to perform a general linear test on the model parameters; e.g., a within-run test for differences in response to different stimuli. The general linear test is defined by the user in the form of a matrix, whose rows represent the multiple linear constraints on the model parameters. The program calculates the specified linear combinations, as well as the F-statistic for significance of the general linear test.

This program, which was developed for use in a "batch" processing mode, should be used in conjunction with the interactive program plug_deconvolve. See the documentation for program plug_deconvolve for a description of the interactive version.

9.2 Further Details

For further details, see *Deconvolution of FMRI Time Series Data*, contained in file 3dDeconvolve.ps.

10 Program 3dFWHM

10.1 Purpose

Program 3dFWHM provides a means of estimating the spatial correlation of voxels in an *AFNI* 3d dataset. In order to use program AlphaSim, it is necessary to know the degree of voxel spatial correlation, so that this can be accounted for in the simulation. Spatial correlation is modeled by applying a Gaussian filter to the random image data. The extent of spatial correlation is specified by entering the width of the Gaussian filter corresponding to each axis. The numbers $FWHM_x$, $FWHM_y$, and $FWHM_z$ (FWHM, for "Full Width Half Maximum") are estimated by program 3dFWHM for a user specified input dataset.

10.2 Further Details

For further details, see Section 2 of *Simultaneous Inference for FMRI Data*, contained in file AlphaSim.ps.

11 Program 3dFriedman

11.1 Purpose

Program 3dFriedman compares blocked multiple treatments. This program performs the nonparametric Friedman test for randomized complete block design experiments, on a voxel-by-voxel basis. Output includes the index of the best (highest ranking) treatment, as well as the Friedman chi-square statistic, for each voxel.

The Friedman test is the nonparametric counterpart of the mixed effects two-way ANOVA. As such, program 3dFriedman roughly corresponds to program 3dANOVA2, which may be used to compare blocked multiple treatments, assuming that the underlying populations are normally distributed with equal variances.

11.2 Further Details

For further details, see Section 4 of *Nonparametric Statistical Analysis of FMRI Data*, contained in file Nonparametric.ps.

12 Program 3dIntracranial

12.1 Purpose

Program 3dIntracranial provides automatic segmentation of the intracranial region. Since this program does not require operator interaction, it can be used for batch processing of a number of images. The output segmented images can be in one of two formats: (1) a segmented anatomical dataset, where brain voxels retain their original gray-scale intensities, and non-brain voxels are set to zero, or (2) a functional dataset "mask", consisting of zero's (corresponding to brain voxels) and one's (corresponding to non-brain voxels).

The anatomical dataset output can be used directly as the underlay for display of functional activation maps. Also, the segmented anatomical dataset can be used as input to the volume rendering program which is included with the *AFNI* package.

The functional dataset mask output can be used as an overlay on top of the original anatomical dataset. This allows for convenient visual inspection of the automatic segmentation results, and manual editing of the output if necessary (see Draw Dataset plugin). Assuming that the original anatomical dataset will be retained, conversion of the functional mask into a segmented anatomical dataset is trivial. Since the functional mask dataset can be compressed to a tiny fraction of the disk space required by the anatomical dataset, this might be another advantage of the functional dataset output over the anatomical dataset output.

12.2 Further Details

For further details, see *Intracranial Segmentation*, contained in file 3dIntracranial.ps.

13 Program 3dKruskalWallis

13.1 Purpose

Program 3dKruskalWallis was developed for comparing multiple treatments. This program performs the nonparametric Kruskal-Wallis test for whether any of s treatments are different. Output includes the index of the best (highest ranking) treatment, as well as the Kruskal-Wallis chi-square statistic, for each voxel.

The Kruskal-Wallis test is the nonparametric counterpart of the one-way ANOVA. As such, program 3dKruskalWallis roughly corresponds to program 3dANOVA, which may be used to compare multiple treatments, assuming that the underlying populations are normally distributed with equal variances.

13.2 Further Details

For further details, see Section 3 of *Nonparametric Statistical Analysis of FMRI Data*, contained in file Nonparametric.ps.

14 Program 3dMannWhitney

14.1 Purpose

Program 3dMannWhitney was developed for nonparametric comparison of two treatments, or two samples. This program performs the Wilcoxon-Mann-Whitney rank-sum test on two groups of *AFNI* 3d datasets, voxel-by-voxel, to determine if the two samples are from the same population. Output includes an estimate for the treatment effect, as well as the normalized Wilcoxon rank-sum statistic, for each voxel.

The Wilcoxon-Mann-Whitney rank-sum test is the nonparametric counterpart of the (unpaired data) t-test. As such, program 3dMannWhitney roughly corresponds to program 3dttest, which may be used to compare two samples, assuming the underlying populations are normally distributed.

14.2 Further Details

For further details, see Section 1 of *Nonparametric Statistical Analysis of FMRI Data*, contained in file Nonparametric.ps.

15 Program 3dNLfim

15.1 Purpose

Program 3dNLfim was developed to provide nonlinear regression analysis of *AFNI* 3d+time data sets. The nonlinear regression is accomplished by calculating a least squares fit of the time series data to a user specified model of the data. The program comes with a selection of separately compiled signal and noise models, which may be chosen by the user. Alternatively, the user may define his own signal model, and add that model to those already accessible by the program (see the section on model definition). Program 3dNLfim makes a separate least squares estimate of the model parameters for each voxel in the input time series data set. Program output options include an *AFNI* 'fift' dataset containing the F-statistic for significance of the nonlinear model at each voxel location, and separate *AFNI* datasets for each parameter in the model. Also, the user has the option of storing the output as a single *AFNI* "bucket" type dataset.

This program is intended for use in a "batch" processing mode. See program plug_nlfit for description of an interactive version of this program. Programs 3dNLfim and plug_nlfit are complementary in nature; they may be used in combination for model building, data exploration, and data analysis. Also, the associated program 3dTSgen may be useful for experimental design and model validation.

15.2 Further Details

For further details, see Section 1 of *Nonlinear Regression Analysis of FMRI Time Series Data*, contained in file 3dNLfim.ps.

16 Program 3dRegAna

16.1 Purpose

Program 3dRegAna was developed to provide multiple linear regression analysis across *AFNI* 3d datasets. For each input dataset, the user must enter the quantitative level that applies for each of the independent (predictor) variables. The user also specifies which variables are to appear in the *full linear regression model*, as well as a simpler *reduced model*.

If repeat observations are available, program 3dRegAna first performs a "lack of fit" test for each voxel in the dataset. The F_{lof} statistic is used to determine if the full linear regression model is adequate for explaining variation in the data. Although the lack of fit test is not mandatory, it is strongly recommended for cases where repeat observations are available. Due to the large number of voxels in a typical FMRI dataset, visual inspection of each of the individual linear regression functions is not practical; therefore, automatic screening for adequacy of the regression model is important.

Program 3dRegAna continues with the regression analysis for those voxels where lack of fit is *not* indicated. The least squares fit of the regression parameters for the full model is calculated individually for each voxel. The regression statistic F_{reg} is calculated; this statistic indicates the significance of the full model relative to the reduced model. Therefore, the F_{reg} statistic can be used in tests of hypotheses about the model structure. The coefficient of multiple determination, R^2, represents the proportion of variation in the data that is explained by the full model. As such, it indicates how well the full model explains the data, and can be used in comparing alternative models. The t-statistics, which indicate the statistical significance of individual parameters within the full model, are also calculated for each voxel. Program 3dRegAna output includes separate *AFNI* 3d datasets containing the individual parameter estimates, along with the corresponding F_{reg} statistic, R^2, and t-statistics. Also, the user has the option of storing the output as a single *AFNI* "bucket" type dataset.

Applications of program 3dRegAna include: simple linear regression, polynomial regression, multiple linear regression, regression analysis involving combinations of quantitative and qualitative predictor variables, and analysis of variance (ANOVA) for cases with unequal sample sizes (provided that the number of factor levels is not too large).

16.2 Further Details

For further details, see *Multiple Linear Regression Analysis across FMRI 3D Datasets*, contained in file 3dRegAna.ps.

17 Program 3dTSgen

17.1 Purpose

Program 3dTSgen provides a means of generating artificial time series data, and storing such data into an *AFNI* 3d+time dataset. The time series data is generated using the operator specified signal and noise models. Such artificial time series data is useful in several ways:

1) Testing of statistical analysis programs for significance of the results. Artificial time series can be generated corresponding to the null hypothesis under consideration. Then, when input to a statistical analysis program, one can determine how often a false positive occurs; i.e., the probability of rejecting the null hypothesis, when it is, in fact, true.

2) Calculation of the statistical power of a test. Artificial time series can be generated corresponding to the alternative hypothesis under consideration. Then, when input to a statistical analysis program, one can determine how often the signal is detected; i.e., the probability of rejecting the null hypothesis, when it is, in fact, false. This enables one to estimate the power of the test.

3) Design of experiments. Various parameters of an experiment are under the researcher's control. Program 3dTSgen can be used to determine the importance of the parameters. For example, the researcher may wish to change the length of the time series data. Using previous experience to specify the signal and noise models, the researcher could use program 3dTSgen to determine the effect of time series length upon the statistical power of the test.

17.2 Further Details

For further details, see Section 3 of *Nonlinear Regression Analysis of FMRI Time Series Data*, contained in file 3dNLfim.ps.

18 Program 3dTcat

18.1 Purpose

Concatenate sub-bricks from input datasets into one big 3D+time dataset.

18.2 Usage

3dTcat options

18.3 Options

-prefix pname
OR
-output pname = Use 'pname' for the output dataset prefix name. [default='tcat']

-session dir = Use 'dir' for the output dataset session directory. [default='./'=current working directory]

-glueto fname = Append bricks to the end of the 'fname' dataset. This command is an alternative to the -prefix and -session commands.

-dry = Execute a 'dry run'; that is, only print out what would be done. This is useful when combining sub-bricks from multiple inputs.

-verb = Print out some verbose output as the program proceeds (-dry implies -verb). Using -verb twice results in quite lengthy output.

-rlt = Remove linear trends in each voxel time series loaded from each input dataset, SEPARATELY. That is, the data from each dataset is detrended separately. At least 3 sub-bricks from a dataset must be input for this option to apply.

Command line arguments after the above are taken as input datasets. A dataset is specified using one of these forms:

'prefix+view', 'prefix+view.HEAD', or 'prefix+view.BRIK'.

18.4 Sub-brick selection

You can also add a sub-brick selection list after the end of the dataset name. This allows only a subset of the sub-bricks to be included into the output (by default, all of the input dataset is copied into the output). A sub-brick selection list looks like one of the following forms:

fred+orig[5]	==>	use only sub-brick #5
fred+orig[5,9,12]	==>	use #5, #9, and #12
fred+orig[5..8] or [5-8]	==>	use #5, #6, #7, and #8
fred+orig[5..13(2)] or [5-13(2)]	==>	use #5, #7, #9, #11, and #13

Sub-brick indexes start at 0. You can use the character '$' to indicate the last sub-brick in a dataset; for example, you can select every third sub-brick by using the selection list:
fred+orig[0..$(3)]

18.5 Notes

- The TR and other time-axis properties are taken from the first input dataset that is itself 3D+time. If no input datasets contain such information, then TR is set to 1.0. This can be altered using the 3drefit program.

- The sub-bricks are output in the order specified, which may not be the order in the original datasets. For example, using

fred+orig[0..$(2),1..$(2)]

will cause the sub-bricks in fred+orig to be output into the new dataset in an interleaved fashion. Using

fred+orig[$..0]

will reverse the order of the sub-bricks in the output. If the -rlt option is used, the sub-bricks selected from each input dataset will be re-ordered into the output dataset, and then this sequence will be detrended.

- You can use the '3dinfo' program to see how many sub-bricks a 3D+time or a bucket dataset contains.
- The '$', '(', ')', '[', and ']' characters are special to the shell, so you will have to escape them. This is most easily done by putting the entire dataset plus selection list inside single quotes, as in 'fred+orig[5..7,9]'.
- You may wish to use the 3drefit program on the output dataset to modify some of the .HEAD file parameters.

19 Program 3dTsmooth

19.1 Purpose

Smooths each voxel time series in a 3D+time dataset and produces as output a new 3D+time dataset.

19.2 Usage

3dTsmooth [options] dataset

19.3 Options

General Options:

-prefix ppp = Sets the prefix of the output dataset to be 'ppp'.
[default = 'smooth']

-datum type = Coerce the output data to be stored as the given type.
[default = input data type]

The following options define the smoothing filter to be used. All filters in this program use 3 input points to compute one output point:
a = input value before the current point
b = input value at the current point
c = input value after the current point
[at the left end, a=b; at the right end, c=b]

-lin = 3 point linear filter: 0.15*a + 0.70*b + 0.15*c
[This is the default smoother]

-med = 3 point median filter: median(a,b,c)

-osf = 3 point order statistics filter:
0.15*min(a,b,c) + 0.70*median(a,b,c) + 0.15*max(a,b,c)

-3lin m = 3 point linear filter: 0.5*(1-m)*a + m*b + 0.5*(1-m)*c
Here, 'm' is a number strictly between 0 and 1.

20 Program 3dWilcoxon

20.1 Purpose

Program 3dWilcoxon was developed for nonparametric paired comparison of two treatments. This program performs the Wilcoxon signed-rank test for paired *AFNI* 3d datasets. Output includes the estimate for the treatment effect, and the normalized Wilcoxon signed-rank statistic, for each voxel.

Unlike the Wilcoxon-Mann-Whitney rank-sum test, which makes no assumption about the distribution of the populations, the Wilcoxon signed-rank test assumes that the population (differences between pairs of observations) has a symmetric distribution.

The Wilcoxon signed-rank test is the nonparametric counterpart of the paired data t-test. As such, program 3dWilcoxon roughly corresponds to program 3dttest, which may be used to compare paired samples, assuming that the underlying populations are normally distributed.

20.2 Further Details

For further details, see Section 2 of *Nonparametric Statistical Analysis of FMRI Data*, contained in file Nonparametric.ps.

21 Program 3daxialize

21.1 Purpose

Read in a dataset and write it out as a new dataset with the data brick oriented as axial slices. The input dataset must have a .BRIK file. One application is to create a dataset that can be used with the AFNI volume rendering plugin.

21.2 Usage

3daxialize [options] dataset

21.3 Options

-prefix ppp = Use 'ppp' as the prefix for the new dataset. [default = 'axialize']

-verbose = Print out a progress pacifier.

22 Program 3dbuc2fim

22.1 Purpose

This program converts bucket sub-bricks to fim (fico, fitt, fift, ...) type dataset.

22.2 Usage

3dbuc2fim -prefix pname d1+orig[index]

> This produces a fim dataset.
>
> -or-

3dbuc2fim -prefix pname d1+orig[index1] d2+orig[index2]

> This produces a fico (fitt, fift, ...) dataset, depending on the statistic type of the 2nd subbrick, with
>
> d1+orig[index1] -> intensity sub-brick of pname
>
> d2+orig[index2] -> threshold sub-brick of pname
>
> -or-

3dbuc2fim -prefix pname d1+orig[index1,index2]

> This produces a fico (fitt, fift, ...) dataset, depending on the statistic type of the 2nd subbrick, with
>
> d1+orig[index1] -> intensity sub-brick of pname
>
> d1+orig[index2] -> threshold sub-brick of pname

22.3 Options

-prefix pname OR

-output pname = Use 'pname' for the output dataset prefix name. [default='buc2fim']

-session dir = Use 'dir' for the output dataset session directory. [default='./'=current working directory]

-verb = Print out some verbose output as the program proceeds.

Command line arguments after the above are taken as input datasets. A dataset is specified using one of these forms:

'prefix+view', 'prefix+view.HEAD', or 'prefix+view.BRIK'.

Sub-brick indexes start at 0.

22.4 Notes

- The sub-bricks are output in the order specified, which may not be the order in the original datasets. For example, using fred+orig[5,3] will cause the sub-brick #5 in fred+orig to be output as the intensity sub-brick, and sub-brick #3 to be output as the threshold sub-brick in the new dataset.
- The '$', '(', ')', '[', and ']' characters are special to the shell, so you will have to escape them. This is most easily done by putting the entire dataset plus selection list inside single quotes, as in 'fred+orig[5,9]'.

23 Program 3dbucket

23.1 Purpose

Concatenate sub-bricks from input datasets into one big 'bucket' dataset.

23.2 Usage

3dbucket options

23.3 Options

-prefix pname OR

-output pname = Use 'pname' for the output dataset prefix name. [default='buck']

-session dir = Use 'dir' for the output dataset session directory. [default='./'=current working directory]

-glueto fname = Append bricks to the end of the 'fname' dataset. This command is an alternative to the -prefix and -session commands.

-dry = Execute a 'dry run'; that is, only print out what would be done. This is useful when combining sub-bricks from multiple inputs.

-verb = Print out some verbose output as the program proceeds (-dry implies -verb).

-fbuc = Create a functional bucket.

-abuc = Create an anatomical bucket. If neither of these options is given, the output type is determined from the first input type.

Command line arguments after the above are taken as input datasets. A dataset is specified using one of these forms:

'prefix+view', 'prefix+view.HEAD', or 'prefix+view.BRIK'.

You can also add a sub-brick selection list after the end of the dataset name. This allows only a subset of the sub-bricks to be included into the output (by default, all of the input dataset is copied into the output). A sub-brick selection list looks like one of the following forms:

fred+orig[5]	==>	use only sub-brick #5
fred+orig[5,9,12]	==>	use #5, #9, and #12
fred+orig[5..8] or [5-8]	==>	use #5, #6, #7, and #8
fred+orig[5..13(2)] or [5-13(2)]	==>	use #5, #7, #9, #11, and #13

Sub-brick indexes start at 0. You can use the character '$' to indicate the last sub-brick in a dataset; for example, you can select every third sub-brick by using the selection list
fred+orig[0..$(3)]

23.4 Notes

- The sub-bricks are output in the order specified, which may not be the order in the original datasets. For example, using fred+orig[0..$(2),1..$(2)] will cause the sub-bricks in fred+orig to be output into the new dataset in an interleaved fashion. Using fred+orig[$..0] will reverse the order of the sub-bricks in the output.
- Bucket datasets have multiple sub-bricks, but do NOT have a time dimension. You can input sub-bricks from a 3D+time dataset into a bucket dataset. You can use the '3dinfo' program to see how many sub-bricks a 3D+time or a bucket dataset contains.
- The '$', '(', ')', '[', and ']' characters are special to the shell, so you will have to escape them. This is most easily done by putting the entire dataset plus selection list inside single quotes, as in 'fred+orig[5..7,9]'.

23.5 Example

In non-bucket functional datasets (like the 'fico' datasets output by FIM, or the 'fitt' datasets output by 3dttest), sub-brick [0] is the 'intensity' and sub-brick [1] is the statistical parameter used as a threshold. Thus, to create a bucket dataset using the intensity from dataset A and the threshold from dataset B, and calling the output dataset C, you would type

3dbucket -prefix C -fbuc 'A+orig[0]' 'B+orig[1]'

24 Program 3dcalc

24.1 Purpose

Do arithmetic on 3D datasets, voxel-by-voxel [no inter-voxel computation].

24.2 Usage

3dcalc [options]

24.3 Options

-verbose = Makes the program print out various information as it progresses.

-datum type = Coerce the output data to be stored as the given type, which may be byte, short, or float. [default = datum of first input dataset]

-fscale = Force scaling of the output to the maximum integer range. This only has effect if the output datum is byte or short (either forced or defaulted). This option is often necessary to eliminate unpleasant truncation artifacts. [The default is to scale only if the computed values seem to need it – are all less than 1 or there is at least one value beyond the integer upper limit.]

- In earlier versions of 3dcalc, scaling (if used) was applied to all sub-bricks equally – a common scale factor was used. This would cause trouble if the values in different sub-bricks were in vastly different scales. In this version, each sub-brick gets its own scale factor. To override this behavior, use the '-gscale option'.

-gscale = Same as '-fscale', but also forces each output sub-brick to get the same scaling factor. This may be desirable for 3D+time datasets, for example.

-a dname = Read dataset 'dname' and call the voxel values 'a' in the expression that is input below. 'a' may be any single letter from 'a' to 'z'.

- If some letter name is used in the expression, but not present in one of the dataset options here, then that variable is set to 0.

- If the letter is followed by a number, then that number is used to select the sub-brick of the dataset which will be used in the calculations. For example, '-b3 dname' specifies that the variable 'b' refers to sub-brick 3 of the dataset (indexes start at 0). N.B.: Another way to achieve the effect of '-b3' is described below in the 'INPUT DATASET SPECIFICATION' section.

-expr "expression" Apply the expression within quotes to the input datasets, one voxel at a time, to produce the output dataset. ("sqrt(a*b)" to compute the geometric mean, for example).

-prefix pname = Use 'pname' for the output dataset prefix name. [default='calc']

-session dir = Use 'dir' for the output dataset session directory. [default='./'=current working directory]

24.4 3D+TIME DATASETS

This version of 3dcalc can operate on 3D+time datasets. Each input dataset will be in one of these conditions:
(A) Is a regular 3D (no time) dataset; or
(B) Is a 3D+time dataset with a sub-brick index specified ('-b3'); or
(C) Is a 3D+time dataset with no sub-brick index specified ('-b').
If there is at least one case (C) dataset, then the output dataset will also be 3D+time; otherwise it will be a 3D dataset with one sub-brick. When producing a 3D+time dataset, datasets in case (A) or (B) will be treated as if the particular brick being used has the same value at each point in time.

Multi-brick 'bucket' datasets may also be used. Note that if multi-brick (bucket or 3D+time) datasets are used, the lowest letter dataset will serve as the template for the output; that is, '-b fred+tlrc' takes precedence over '-c wilma+tlrc'. (The program 3drefit can be used to alter the .HEAD parameters of the output dataset, if desired.)

24.5 INPUT DATASET SPECIFICATION

An input dataset is specified using one of these forms:

'prefix+view', 'prefix+view.HEAD', or 'prefix+view.BRIK'.

You can also add a sub-brick selection list after the end of the dataset name. This allows only a subset of the sub-bricks to be read in (by default, all of a dataset's sub-bricks are input). A sub-brick selection list looks like one of the following forms:

fred+orig[5]	==>	use only sub-brick #5
fred+orig[5,9,12]	==>	use #5, #9, and #12
fred+orig[5..8] or [5-8]	==>	use #5, #6, #7, and #8
fred+orig[5..13(2)] or [5-13(2)]	==>	use #5, #7, #9, #11, and #13

Sub-brick indexes start at 0. You can use the character '$' to indicate the last sub-brick in a dataset; for example, you can select every third sub-brick by using the selection list
fred+orig[0..$(3)]

- The sub-bricks are read in the order specified, which may not be the order in the original dataset. For example, using fred+orig[0..$(2),1..$(2)] will cause the sub-bricks in fred+orig to be input into memory in an interleaved fashion. Using fred+orig[$..0] will reverse the order of the sub-bricks.

- The '$', '(', ')', '[', and ']' characters are special to the shell, so you will have to escape them. This is most easily done by putting the entire dataset plus selection list inside forward single quotes, as in 'fred+orig[5..7,9]'.

24.6 1D TIME SERIES

You can also input a '*.1D' time series file in place of a dataset. In this case, the value at each spatial voxel at time index n will be the same, and will be the n-th value from the time series file. At least one true dataset must be input. If all the input datasets are 3D (single sub-brick) or are single sub-bricks from multi-brick datasets, then the output will be a 'manufactured' 3D+time dataset For example, suppose that 'a3D+orig' is a 3D dataset:

3dcalc -a a3D+orig -b b.1D -expr "a*b"

The output dataset will 3D+time with the value at (x,y,z,t) being computed by a3D(x,y,z)*b(t). The TR for this dataset will be set to 1 second – this could be altered later with program 3drefit. Another method to set up the correct timing would be to input an unused 3D+time dataset – 3dcalc will then copy that dataset's time information, but simply do not use that dataset's letter in -expr.

24.7 COORDINATES

If you don't use '-x', '-y', or '-z' for a dataset, then the voxel spatial coordinates will be loaded into those variables. For example, the expression

'a*step(x*x+y*y+z*z-100)'

will zero out all the voxels inside a 10 mm radius of the origin.

24.8 PROBLEMS

- Complex-valued datasets cannot be processed.
- This program is not very efficient (but is faster than before).

24.9 EXPRESSIONS

Arithmetic expressions are allowed, using + - * / ** and parentheses. As noted above, datasets are referred to by single letter variable names. At this time, C relational, boolean, and conditional expressions are NOT implemented. Built in functions include:

sin ,	cos ,	tan ,	asin ,	acos ,	atan ,	atan2,	
sinh ,	cosh ,	tanh ,	asinh ,	acosh ,	atanh ,	exp ,	
log ,	log10,	abs ,	int ,	sqrt ,	max ,	min ,	
J0 ,	J1 ,	Y0 ,	Y1 ,	erf ,	erfc ,	qginv,	qg ,
rect ,	step ,	astep,	bool ,	and ,	or ,	mofn ,	
sind ,	cosd ,	tand ,	median,				

where

qg(x) = reversed cdf of a standard normal distribution
qginv(x) = inverse function to qg,
min, max, atan2 each take 2 arguments,

J0, J1, Y0, Y1 are Bessel functions (see Watson),
erf, erfc are the error and complementary error functions,
sind, cosd, tand take arguments in degrees (vs. radians),
median(a,b,c,...) computes the median of its arguments [median takes a variable number of arguments].

The following functions are designed to help implement logical functions, such as masking of 3D volumes against some criterion:

step(x)	=	{1 if x>0 ,	0 if x<=0},
astep(x,y)	=	{1 if abs(x) > y ,	0 otherwise} = step(abs(x)-y)
rect(x)	=	{1 if abs(x)<=0.5,	0 if abs(x)>0.5},
bool(x)	=	{1 if x != 0.0 ,	0 if x == 0.0},
and(a,b,...,c)	=	{1 if all arguments are nonzero,	0 if any are zero}
or(a,b,...,c)	=	{1 if any arguments are nonzero,	0 if all are zero}
mofn(m,a,...,c)	=	{1 if at least 'm' arguments are nonzero,	0 otherwise}

These last 3 functions take a variable number of arguments.

The following 27 new [Mar 1999] functions are used for statistical conversions, as in the program 'cdf':

fico_t2p(t,a,b,c),	fico_p2t(p,a,b,c),	fico_t2z(t,a,b,c),
fitt_t2p(t,a) ,	fitt_p2t(p,a) ,	fitt_t2z(t,a) ,
fift_t2p(t,a,b) ,	fift_p2t(p,a,b) ,	fift_t2z(t,a,b) ,
fizt_t2p(t) ,	fizt_p2t(p) ,	fizt_t2z(t) ,
fict_t2p(t,a) ,	fict_p2t(p,a) ,	fict_t2z(t,a) ,
fibt_t2p(t,a,b) ,	fibt_p2t(p,a,b) ,	fibt_t2z(t,a,b) ,
fibn_t2p(t,a,b) ,	fibn_p2t(p,a,b) ,	fibn_t2z(t,a,b) ,
figt_t2p(t,a,b) ,	figt_p2t(p,a,b) ,	figt_t2z(t,a,b) ,
fipt_t2p(t,a) ,	fipt_p2t(p,a) ,	fipt_t2z(t,a) .

See the output of 'cdf -help' for documentation on the meanings of and arguments to these functions. (After using one of these, you may wish to use program '3drefit' to modify the dataset statistical auxiliary parameters.)

24.10 Notes

- Computations are carried out in double precision before being truncated to the final output 'datum'.
- Note that the quotes around the expression are needed so the shell doesn't try to expand * characters, or interpret parentheses.
- Try the ccalc program to see how the expression evaluator works. The arithmetic parser and evaluator is written in Fortran-77 and is derived from a program written long ago by RW Cox to facilitate compiling on an array processor hooked up to a VAX. It's a mess, but it works.

25 Program 3dclust

25.1 Purpose

Program 3dclust is used to detect clusters of activation (i.e., nonzero voxels) in 3D datasets, and to report various statistics about the location and intensity of cluster activation. A cluster is defined as a set of points, each of which has a neighbor in the cluster that is no more than some given distance away (the 'connectivity distance'). If you select this connectivity distance as being just slightly over one voxel width, then clusters must be connected in a nearest-neighbor fashion. If the connectivity distance is a little over $\sqrt{2}$ times the voxel dimension, then next-nearest-neighbor (diagonal, in-plane) connections are possible. And so forth. For example:

3dclust 1.9 200.0 friday/func01+tlrc.HEAD

This says to examine the given dataset for clusters, using a 1.9 mm connectivity distance, and discarding all clusters that aren't at least 200 μl (mm^3) in volume. If the voxels are $1 \times 1 \times 1$ mm^3, then the 1.9 mm connectivity distance means that clusters will be connected by rectangular neighbors (1 mm away), in-plane diagonal neighbors ($\sqrt{2}$ mm away), or by out-of-plane diagonal neighbors ($\sqrt{3}$ mm away).

The above definition of cluster is used in program 3dmerge (see user documentation for program 3dmerge). Its application there is to edit out functional results that are not in a cluster.

The information about the clusters will be sent to stdout and will report their volumes, their centroid (center of mass, or CM), spatial extent of the cluster, the standard error of the mean (SEM), and their 'hot spot' (voxel with maximum value) coordinates. These coordinates are in the DICOM standard order, and it is easy to copy-and-paste them (using X11) for use as input to the Jump to chooser in *AFNI*.

At present, only one hot spot per cluster is reported. For large clusters, this may not be adequate for database (e.g., BrainMap) purposes.

25.2 Usage

The command line format for program 3dclust is as follows:

3dclust [editing options] [-summarize] [-noabs] rmm vmul dset ...

25.3 Options

rmm = cluster connection radius (mm). It is required that rmm be positive.

vmul = minimum cluster volume (micro-liters). It is required that vmul be positive. Note that (1 mm)^3 is 1 micro-liter.

dset = input dataset (more than one allowed).

-noabs The -noabs option uses the signed voxel intensities (not the absolute values) for calculation of the mean and SEM.

-summarize The -summarize option will write out only the total nonzero voxel count and volume for each dataset.

The report is sent to stdout.

25.4 Notes

- The editing options are as in 3dmerge.
- The program does not work on complex-valued datasets!
- The center of mass calculations make use of the intensity at each point.
- Cluster calculations are all done on the absolute value of the intensity; hence, positive and negative voxels can be grouped together into the same cluster, which may skew results. As an alternative, one could use the -1noneg option or the -noabs option.
- The standard error of the mean (SEM) values are not realistic for interpolated data sets (because comparisons are not independent). A *rough* correction is to multiply the interpolated SEM of the interpolated data set by the square root of the number of interpolated voxels per original voxel.

25.5 Example

Example 1.

An investigator wishes to examine *AFNI* dataset file fred.test07.run35+tlrc (both .HEAD and .BRIK files must be present) for clusters of activation. A possible input command line sequence is as follows:

3dclust -1noneg -1thresh 0.5 2 500 fred.test07.run35+tlrc

This command line tells program 3dclust to identify clusters of active voxels, considering only those voxels whose intensity is non-negative, and whose threshold value is above 0.50. A voxel is "connected" to another voxel if the voxels are $\leq$ 2 mm apart. Further, only clusters having a volume of 500 mm^3 or more are considered.

The program would then send to the standard output (such as the user's terminal) the following report:

Cluster report for file fred.test07.run35+tlrc
[Connectivity radius = 2.00 mm Volume threshold = 500.00]

[Single voxel volume = 1.0 (microliters)]
[Voxel datum type = short]
[Voxel dimensions = 1.000 mm X 1.000 mm X 1.000 mm]

Volume	CM RL	CM AP	CM IS	minRL	maxRL	minAP	maxAP	minIS	maxIS
17059	-5.3	70.7	47.4	-49.0	35.0	29.0	99.0	11.0	72.0
3384	36.7	75.7	17.8	22.0	50.0	56.0	89.0	-1.0	37.0
1680	-17.2	67.2	-2.5	-31.0	-4.0	58.0	76.0	-12.0	13.0
1086	-28.6	4.4	51.3	-38.0	-18.0	-1.0	12.0	40.0	68.0
920	35.9	39.7	43.9	26.0	50.0	33.0	50.0	36.0	54.0
795	12.0	67.0	0.7	4.0	21.0	60.0	80.0	-4.0	7.0
725	-47.2	73.0	7.5	-52.0	-35.0	65.0	81.0	0.0	15.0
646	26.2	3.6	48.6	15.0	39.0	-1.0	8.0	44.0	56.0
535	-40.5	-0.3	37.0	-49.0	-32.0	-9.0	4.0	28.0	45.0
26830									

Mean|*|	SEM	Max Int	MI RL	MI AP	MI IS
641.21	2.5049	2717	-12.0	73.0	58.0
448.92	2.7841	1205	41.0	80.0	21.0
517.87	3.8714	971	-8.0	59.0	11.0
460.21	3.7005	784	-30.0	5.0	63.0
393.92	3.7619	732	27.0	39.0	41.0
422.13	4.353	722	9.0	66.0	-2.0
515.05	7.3938	1183	-51.0	75.0	9.0
411.84	4.725	686	31.0	3.0	46.0
405.86	5.3017	709	-37.0	2.0	34.0
573.31	1.7778				

From the output, we see that 9 clusters (each having a volume of at least 500 mm^3) have been identified. The first column lists the volume of each cluster (which, in this case, is numerically equal to the number of voxels in the cluster, since each voxel has volume $1mm^3$). The 2nd, 3rd, and 4th columns give the center of mass of the cluster, in the R-L, A-P, I-S coordinate system. The 5th through 10th columns give the spatial extent of the cluster, in terms of the minimum and maximum voxel coordinates in the R-L, A-P, and I-S directions. The 11th column lists the average absolute value of the intensity of the voxels within the cluster. The 12th column gives the cluster intensity standard error of the mean (SEM). The 13th column lists the (signed) maximum intensity of a voxel within the cluster. Finally, the 14th, 15th, and 16th columns give the coordinates of the voxel having the peak intensity within the cluster, in the R-L, A-P, I-S coordinate system.

The last row of the table lists the total volume of all clusters combined, the average absolute value of the intensity of all voxels in all clusters, and the standard error of the mean (SEM) for all voxels in all clusters.

26 Program 3ddup

26.1 Purpose

Program 3ddup will make a 'Warp-on-demand' duplicate copy of a given 3D dataset. The new dataset will consist only of a .HEAD file. There are two applications for this capability:

- It is possible to change the dataset type with 3ddup. This means that the original dataset can be anatomical and the new dataset functional (-fim type). The only disk space penalty will be a few Kbytes for the new .HEAD file—there is no need to duplicate the .BRIK file. In this way, for example, it is possible to gather Magnetic Resonance Angiograms (MRA data) and view the 3D dataset either as anatomy (gray background) or as function (overlaid in color).

- This is the only way to get *AFNI* to write out a resampled version of a dataset in the +orig view. *AFNI* will not overwrite a .BRIK file unless the dataset is warped from another dataset. (This is to prevent destruction of data that cannot easily be recreated by 'Warp-on-demand'.) With 3ddup, the new dataset can be overwritten at whatever resolution desired (creating a new .BRIK file), using the 'Define Datamode' controls. (Of course, after it is resampled and overwritten, it is no longer a perfect duplicate of the original.)

Note that when resampling a functional dataset (using the button 'Write Function Brick'), it will be written out onto the grid corresponding to the current anatomical dataset, at the currently chosen grid size. This grid will have cubical voxels if Warp-on-demand is chosen; however, if View-data-brick is chosen in *AFNI*, then the output grid will conform to the anatomy data brick grid, which is not necessarily cubical.

If the function is not defined on parts of the anatomy, then zeros are written in those locations. If the original function brick extends outside the domain of the anatomy brick, then the newly resampled function brick will be clipped to fit the anatomy brick.

26.2 Usage

3ddup [options] dataset

26.3 Options

-type Convert to the given 'type', which must be chosen from the same list as in to3d.

-session dirname Write output into given directory (default=./).

-prefix pname Use 'pname' for the output directory prefix (default=dup).

N.B.: Even if the new dataset is anatomical, it will not contain any markers, duplicated from the original, or otherwise.

27 Program 3dfim

27.1 Purpose

Program 3dfim calculates a functional image from a 3d+time data file by cross correlation of each voxel time series with a user specified reference time series. The functional image may then be used as input to program afni to provide a visual display of the locations of those voxels showing a statistically significant correlation of intensity with the reference time series.

27.2 Usage

The command line format for program 3dfim is as follows:

3dfim [-im1 num] -input fname -prefix name -ideal fname [-percent p] [-ort fname]

27.3 Options

-im1 num 'num' is the index of the first image to be used in time series correlation; prior images will be ignored. The default is 1.

-input fname 'fname' is the filename of a 3d + time data file for input. This is the source of the **x** data time series mentioned above (for each voxel).

-prefix name 'name' is the prefix of the filename for saving the output functional data.

The output file is an *AFNI* "fico" dataset; i.e., a 2 sub-brick data file, where the first sub-brick consists of the unnormalized image intensity (which is used to choose overlay colors), and the second sub-brick contains the partial correlation coefficient for each voxel. This may be represented schematically by:

$$\textit{AFNI}\text{ ``fico'' dataset} \quad \left\{ \begin{array}{l} \boxed{\alpha = \dfrac{\langle \mathbf{Pr}, \mathbf{Px} \rangle}{|\mathbf{Pr}|^2}} \\[2em] \boxed{\rho = \dfrac{\langle \mathbf{Pr}, \mathbf{Px} \rangle}{|\mathbf{Pr}|\,|\mathbf{Px}|}} \end{array} \right.$$

where

$\mathbf{x}$ = data time series (as a vector),
$\mathbf{r}$ = reference (ideal) time series,
$\mathbf{S}$ = matrix formed from 'ort' time series (including polynomials),
$\mathbf{P}$ = projection matrix that removes undesired components of signal
$= \mathbf{I} - \mathbf{S}[\mathbf{S}^T\mathbf{S}]^{-1}\mathbf{S}^T$,
α = unnormalized image intensity, used to choose overlay colors,
ρ = partial correlation coefficient.

-ideal fname 'fname' is the filename of a time series to which the image data is to be correlated. This is the source of the **r** reference (ideal) time series mentioned above. More than one ideal time series may be used (see below).

-percent p The -percent command provides an alternative output value for the intensity sub-brick. If this command is used, then the program calculates the percentage change of the input time series, relative to the baseline, accounted for by the ideal time series. If more than one ideal time series is specified, then this calculation uses the ideal which is most highly correlated with the input time series. The number p specifies the *maximum* value allowed for the calculated percentage (i.e., if the calculated percentage is greater than p, then it is set equal to p). Obviously, it is required that $p > 0$.

The output file is an *AFNI* "fico" dataset; i.e., a 2 sub-brick data file, where the first sub-brick consists of the percentage change, and the second sub-brick contains the partial correlation coefficient for each voxel.

$$\text{\textit{AFNI} ``fico'' dataset} \quad \begin{cases} \boxed{\% = 100 \cdot \dfrac{|\alpha|(r_{\max} - r_{\min})}{baseline}} \\ \\ \boxed{\rho = \dfrac{\langle \mathbf{Pr}, \mathbf{Px} \rangle}{|\mathbf{Pr}|\,|\mathbf{Px}|}} \end{cases}$$

where

$r_{\max}$ = maximum value of the reference (ideal) time series,
$r_{\min}$ = minimum value of the reference (ideal) time series,
baseline = absolute value of the average of the input data time series, i.e.,

$$baseline = |\bar{x}| = \left| \frac{1}{n} \sum_{i=1}^{n} x_i \right|$$

-ort fname 'fname' is the filename of a time series to which the image data is to be orthogonalized. The 'ort' time series is used to create the **S** matrix mentioned above. More than one 'ort' time series may be used (see below). Note: whether or not an ort time series is specified by the user, the program will always remove a constant plus linear trend from the image time series data.

Note that it is possible to specify more than one ideal time series file. Each ideal time series is separately correlated with the image time series and the one most highly correlated is selected for each voxel. Multiple ideals are specified using more than one '-ideal fname' option, or by using the form '-ideal [fname1 fname2 ...]' – this latter method allows the use of wildcarded ideal filenames. The '[' character that indicates the start of a group of ideals can actually be any ONE of these: [{/% and the ']' that ends the group can be:]}/%

The file containing the ideal time series must be in the following format:

- ASCII; one number per line;
- Same number of lines as images in the time series;
- Any value over 33333 means "don't use this image in the analysis".

It is also possible to specify more than one 'ort' time series file. The image time series is orthogonalized to each ort time series. Multiple orts are specified using more than one '-ort fname' option, or by using the form '-ort [fname1 fname2 ...]' – this latter method allows the use of wildcarded ort filenames. The '[' character that indicates the start of a group of ideals can actually be any ONE of these: [{/% and the ']' that ends the group can be:]}/%

The file containing the ort time series must be in the following format:

- ASCII; one number per line;
- At least same number of lines as images in the time series.

27.4 Examples

Example 1. Suppose that file fred.ts+orig.BRIK contains 3D+time data, and that the user wishes to determine which voxels are correlated with the reference time series in file cos7.25.1D (a cosine waveform). File cos7.25.1D might have the following appearance:

100000
100000
100000
100000
−195
555
980
831
195
−555
−980
−831
−195
etc.

Since the first 4 numbers in the file are greater than 33333, this tells program 3dfim to ignore the first 4 images.

The command line sequence to correlate this reference time series with the image data in file fred.ts+orig.BRIK is given by

```
3dfim -ideal cos7.25.1D -prefix fred.tscorr -input fred.ts+orig
```

The resulting output will be stored in file fred.tscorr+orig.BRIK (and .HEAD). Now, this output will show which voxels are correlated with the given reference time series, which is a cosine waveform of a specific frequency and phase. The frequency of the cosine wave is, of course, that of the task which was used during the experiment.

Example 2. (Continuation of previous example). Since the time delay will vary from voxel to voxel, it would be more meaningful to correlate the image time series with cosine waves of the same frequency, but having different phases (corresponding to different time delays). This is accomplished by using more than one ideal time series. Suppose that files cos7.00.1D, cos7.25.1D, cos7.50.1D, cos7.75.1D, cos8.00.1D, cos8.25.1D, cos8.50.1D, and cos8.75.1D contain copies of the cosine waveform, all at the same frequency, but with different phase shifts. Then, to find the best correlation at each voxel of the image time series with each of these ideal times series, one could use the command line:

```
3dfim -ideal [ cos7.00.1D cos7.25.1D cos7.50.1D \
cos7.75.1D cos8.00.1D cos8.25.1D cos8.50.1D cos8.75.1D ] \
-prefix fred.tscorr -input fred.ts+orig
```

where the list of ideal time series files is contained between the '[' and ']' symbols. Assuming that no other files in this directory begin with 'cos', this command line may be abbreviated by using the 'wildcard' symbol '*' as follows:

```
3dfim -ideal \[cos*\] -prefix fred.tscorr -input fred.ts+orig
```

In both cases, the output file fred.tscorr+orig.BRIK (and .HEAD) contains the *AFNI* 'fico' dataset, where, for each voxel, the image intensity α and the partial correlation coefficient ρ correspond to the ideal time series to which the voxel time series is most highly correlated.

(Note that '[' and ']' are special characters to the C-shell and so must be escaped (with '\' characters) on the command line.)

Example 3. The user suspects that external physiological activity is interfering with the signal analysis; specifically, that breathing and heart rate may be "contaminating" the true neurological response. In order to remove these unwanted effects from the data, suppose that measurements of breathing and heart rate, time synchronized with the fMRI data, are stored in files lung.1D and heart.1D. Then the '-ort' command is used to remove the influence of these factors:

```
3dfim -ideal [ cos7.00.1D cos7.25.1D cos7.50.1D \
cos7.75.1D cos8.00.1D cos8.25.1D cos8.50.1D cos8.75.1D ] \
-ort [ heart.1D lung.1D ] \
-prefix fred.tscorr -input fred.ts+orig
```

The output file fred.tscorr+orig.BRIK (and .HEAD) contains the *AFNI* 'fico' dataset, where, for each voxel, the image intensity α and the partial correlation coefficient ρ correspond to the ideal time series to which the voxel time series is most highly correlated, *after* the effect of the heart and lung data has been removed from the image data.

28 Program 3dfractionize

28.1 Purpose

The purpose of this program is to allow the resampling of a mask dataset (the input) from a fine grid to a coarse grid (defined by the template). When you are using the output, you will probably want to threshold the mask so that voxels with a tiny occupancy fraction aren't used. This can be done in 3dmaskave, by using 3calc, or with the '-clip' option below.

This program will also work in going from a coarse grid to a fine grid, but it isn't clear that this capability has any purpose.

28.2 Usage

3dfractionize [options]

28.3 Options

-template tset Use dataset 'tset' as a template for the output.

-input iset Use dataset 'iset' for the input. Only the sub-brick #0 of the input is used. You can use the sub-brick selection technique described in '3dcalc -help' to choose the desired sub-brick from a multi-brick dataset.

-prefix ppp Use 'ppp' for the prefix of the output. [default = 'fractionize']

-clip fff Clip off voxels that are less than 'fff' occupied. 'fff' can be a number between 0.0 and 1.0, meaning the fraction occupied, can be a number between 1.0 and 100.0, meaning the percent occupied, or can be a number between 100.0 and 10000.0, meaning the direct output value to use as a clip level. [default = 0.0]

28.4 Notes

- For each voxel in the output dataset, this program computes the fraction of it that is occupied by nonzero voxels from the input.
- The fraction is stored as a short in the range 0..10000, indicating fractions running from 0..1.
- The template dataset is used only to define the output grid; its brick(s) will not be read into memory.

- The actual values stored in the input dataset are irrelevant, except in that they are zero or nonzero.

29 Program 3dhistog

29.1 Purpose

Program 3dhistog computes a histogram of the voxel intensities in an *AFNI* 3D dataset.

29.2 Usage

The command line format for program 3dhistog is as follows:

3dhistog [**editing options**] [**histogram options**] **dataset**

29.3 Options

The editing options are the same as in program 3dmerge.

The histogram options are:

-nbin # Means to use '#' bins (default = 100).
Special Case: For short or byte dataset bricks, set '#' to zero to have the number of bins set by the brick range.

-thr r Means to count only voxels with the statistics threshold above 'r'.

-dind i Means to take data from sub-brick 'i'.

-tind j Means to take threshold values from sub-brick 'j'.

-omit x Means to omit the value 'x' from the count.

-notit Means to leave the title line off the output.

The report is sent to the standard output device (user's terminal, unless redirected).

29.4 Examples

Example 1. A researcher wishes to compute the histogram of voxel intensities for an *AFNI* 'fift' dataset. Only those voxels whose corresponding F-statistic is above 9.0 are of interest (this provides an overall significance level of $\alpha = 0.05$ for this particular case). The data is stored in file fred.smax+orig.BRIK (and .HEAD). The command line to calculate the histogram using 100 bins is given by:

3dhistog -thr 9.0 fred.smax+orig

The output is sent to the standard output device (usually, the user's terminal):

Magnitude	Freq	Cum_Freq	Thr_Freq	Cum_Thr_Freq
-458.567261	5	5	5	5
-449.316010	3	8	3	8
-440.064728	14	22	14	22
⋮	⋮	⋮	⋮	⋮
-23.757933	245	3060	6	2428
-14.506693	112	3172	0	2428
-5.255426	22434	25606	0	2428
3.995841	141	25747	0	2428
13.247108	356	26103	12	2440
⋮	⋮	⋮	⋮	⋮
429.553894	10	32749	10	8512
438.805145	9	32758	9	8521
448.056427	9	32767	9	8530
457.307678	1	32768	1	8531

The first column is the lower edge of the bin of voxel intensity. The second column is a count of the number of voxels that fall within the bin. The third column is the cumulative count; i.e., the number of voxels whose intensity falls within the bin corresponding to that row or to previous rows. The fourth column is a count of the number of voxels whose intensity falls within the bin for that row, and whose threshold (statistic) exceeds the specified value (9.0, in this particular case). The fifth column is the cumulative count of the voxels exceeding the threshold value. In the above example, 8531 out of a total of 32768 voxels (or 26%) of the voxels showed statistically significant activity.

30 Program 3dinfo

30.1 Purpose

Program 3dinfo prints out some useful information from an *AFNI* 3D dataset's header file.

30.2 Usage

The command line format for program 3dinfo is as follows:

3dinfo [-v] **dataset** [**dataset ...**]

The -v option means print out verbose information. At present, it just causes the printing of all the statistics for each time in a time-dependent dataset.

For *AFNI* 'bucket' type datasets, 3dinfo prints the label and statistical information for each sub-brick in the dataset.

30.3 Examples

Example 1.

Suppose that the user is uncertain about the contents of *AFNI* data file fred.test02.run35+tlrc.BRIK. Then, assuming that the .HEAD file is also present, the 3dinfo command line is:

```
3dinfo fred.test02.run35+tlrc
```

The program prints the following information to the screen:

```
Dataset File: fred.test02.run35+tlrc
Identifier Code: MCW_AGKZLWWPCSE
Creation Date: Tue Aug 13 20:17:10 1996
Dataset Type: Inten+Thr (-fith)
Anatomy Parent: [MCW_KDZRUNFMRVF]
Warp Parent: [MCW_YKCFQKMXFVJ]
Data Axes Orientation:
first = Right-to-Left
second = Anterior-to-Posterior
third = Inferior-to-Superior [-orient RAI]

   R-to-L extent:  -80.000 [R]  -to-  80.000 [L]  -step- 1.000 mm [161 voxels]
   A-to-P extent:  -80.000 [A]  -to- 110.000 [P]  -step- 1.000 mm [191 voxels]
   I-to-S extent:  -55.000 [I]  -to-  85.000 [S]  -step- 1.000 mm [141 voxels]

Number of values stored at each pixel = 2
– At sub-brick #0 datum type is short: -2186 to 2717
– At sub-brick #1 datum type is short: -7023 to 8226
```

■

The program lists the Dataset File name, the Identifier Code (this identifier was created by *AFNI* for internal use), the Dataset Type (in this case, an *AFNI* 'fith' dataset, consisting of an intensity sub-brick and a threshold sub-brick), the Anatomy Parent identifier, the Warp Parent identifier, the orientation and size of the Data Axes (in mm. and in voxels), the number of values stored at each voxel (2 in this case, since for each voxel there is one intensity value and one threshold value), and the range of data values stored in each sub-brick.

Example 2.

Program 3dNLfim was used to perform nonlinear regression analysis for the 3d+time dataset freg+orig. The results of the analysis were stored in an *AFNI* 'bucket' type dataset fred.analysis+orig. The command line

3dinfo fred.analysis+orig

produces the following screen output:

```
Dataset File:     fred.analysis+orig
Identifier Code:  MCW_ZQEUXZAVNGB
Creation Date:    Fri Dec 26 11:51:57 1997
Dataset Type:     Func-Bucket (-fbuc)
Data Axes Orientation:
   first   (x) =  Anterior-to-Posterior
   second  (y) =  Left-to-Right
   third   (z) =  Superior-to-Inferior   [-orient ALS]
R-to-L extent:  -118.125 [R] -to-  118.125 [L] -step-  3.750 mm  [ 64 voxels]
A-to-P extent:  -118.125 [A] -to-  118.125 [P] -step-  3.750 mm  [ 64 voxels]
I-to-S extent:   -28.000 [I] -to-   28.000 [S] -step-  8.000 mm  [  8 voxels]
Number of values stored at each pixel = 12

-- At sub-brick #0 'constant' datum type is short:          0 to     32767 [internal]
                                  [* 0.210506]             0 to   6897.63 [scaled]
-- At sub-brick #1 'linear' datum type is short:       -28594 to     32767 [internal]
                                  [* 0.000192914]    -5.5162 to   6.32123 [scaled]
-- At sub-brick #2 't0' datum type is short:                0 to     32767 [internal]
                                  [* 0.00228889]           0 to        75 [scaled]

-- At sub-brick #3 'k' datum type is short:            -32766 to     32767 [internal]
                                  [* 0.0152592]     -499.984 to       500 [scaled]
-- At sub-brick #4 'alpha1' datum type is short:            0 to     32767 [internal]
                                  [* 4.57775e-06]          0 to  0.149999 [scaled]
-- At sub-brick #5 'alpha2' datum type is short:            0 to     32767 [internal]
                                  [* 1.52592e-05]          0 to  0.499997 [scaled]

-- At sub-brick #6 'Signal TMax' datum type is short:       0 to     32767 [internal]
                                  [* 0.00424207]           0 to       139 [scaled]
-- At sub-brick #7 'Signal SMax' datum type is short:  -32767 to     32635 [internal]
                                  [* 0.013963]      -457.526 to   455.683 [scaled]
-- At sub-brick #8 'Signal % SMax' datum type is short: -13951 to    32767 [internal]
                                  [* 0.00295232]    -41.1879 to   96.7388 [scaled]
```

– At sub-brick #9 'Signal Area' datum type is short:	0 to	32767	[internal]
[∗ 1.28015]	0 to	41946.8	[scaled]
– At sub-brick #10 'Signal % Area' datum type is short:	−17719 to	32767	[internal]
[∗ 0.00128149]	−22.7067 to	41.9905	[scaled]
– At sub-brick #11 'F-stat Regress' datum type is short:	0 to	32766	[internal]
[∗ 0.0616718]	0 to	2020.74	[scaled]

statcode = fift; statpar = 4 191

■

For each sub-brick in the 'bucket' dataset, program 3dinfo prints the label and the range of values. Since sub-brick #11 is a statistical sub-brick, additional information is displayed. The "statcode = fift" indicates that this sub-brick contains F-statistics; "statpar = 4 191" means that for the voxels in this sub-brick, the numerator dof = 4 and the denominator dof = 191.

31 Program 3dmaskave

31.1 Purpose

Computes average of all voxels in the input dataset which satisfy the criterion in the options list. If no options are given, then all voxels are included.

31.2 Usage

3dmaskave [options] dataset

31.3 Options

-mask mset Means to use the dataset 'mset' as a mask. Only voxels with nonzero values in 'mset' will be averaged from 'dataset'. Note that the mask dataset and the input dataset must have the same number of voxels.

-mindex miv Means to use sub-brick #'miv' from the mask dataset. If not given, then miv=0.

-mrange a b Means to further restrict the voxels from 'mset' so that only those mask values between 'a' and 'b' (inclusive) will be used. If this option is not given, all nonzero values from 'mset' are used. Note that if a voxel is zero in 'mset', then it won't be included, even if $a < 0 < b$.

-dindex div Means to use sub-brick #'div' from the dataset. If not given, all sub-bricks will be processed.

-drange a b Means to only include voxels from the dataset whose values fall in the range 'a' to 'b' (inclusive). Otherwise, all voxel values are included.

-sigma Means to compute the standard deviation as well as the mean.

-dump Means to print out all the voxel values that go into the average. This option cannot be used unless the -mask option is also used.

-udump Means to print out all the voxel values that go into the average, UNSCALED by any internal factors. Also requires -mask. N.B.: the scale factors for a sub-brick can be found using program 3dinfo.

-indump Means to print out the voxel indexes (i,j,k) for each dumped voxel. Has no effect if -dump or -udump is not also used. N.B.: if nx,ny,nz are the number of voxels in each direction, then the array offset in the brick corresponding to (i,j,k) is i+j*nx+k*nx*ny.

The output is printed to stdout (the terminal), and can be saved to a file using the usual redirection operation '>'.

32 Program 3dmerge

32.1 Purpose

Program 3dmerge is for editing and/or merging of 3D datasets. The editing options, which are applied to each individual 3D dataset, include:

- Discard negative intensities, or take their absolute values.
- Discard intensities below a given 'clipping' level.
- Use the threshold in a fith (or fico, or fitt, or fift) file to censor the dataset.
- Blur voxels with a Gaussian tapering function.
- Discard voxels not in a cluster of at least a given volume.
- Apply a spatial filter to voxel intensities.
- Linearly scale voxel intensities so that the largest present is set to 10000.

These functions are applied in the order given.

Various options are provided for forming a single 3D dataset from a group of 3D datasets (the merging options). The merging functions include:

- Discard voxels that aren't nonzero in a given number of datasets (principally used to edit out one-time-only results).

- Combine using voxel means or voxel max.
- After combination, again discard voxels not in a cluster of at least a given volume.

These functions are applied in the order given.

For some applications, you may wish to edit functional datasets using 3dmerge before using 3dproject or 3dclust on them.

32.2 Usage

The command line format for program 3dmerge is as follows:

3dmerge [options] datasets . . .
where the options are:

32.3 Editing Options Applied to Each Input Dataset

-1thtoin Copy threshold data over intensity data. This is only valid for datasets with some thresholding statistic attached. All subsequent operations apply to this substituted data.

-2thtoin The same as -1thtoin, but do NOT scale the threshold values from shorts to floats when processing. This option is only provided for compatibility with the earlier versions of the AFNI package '3d*' programs.

-1noneg Zero out voxels with negative intensities.

-1abs Take absolute values of intensities.

-1clip val Clip intensities in range [-val,val] to zero.

-2clip v1 v2 Clip intensities in range [v1,v2] to zero.

-1uclip val , -2uclip v1 v2
These options are like the above, but do not apply any automatic scaling factor that may be attached to the data. These are for use only in special circumstances. (The 'u' means 'unscaled'. Program '3dinfo' can be used to find the scaling factors.) N.B.: Only one of these 'clip' options can be used; you cannot combine them to have multiple clipping executed.

-1thresh thr Use the threshold data to censor the intensities (only valid for 'fith', 'fico', or 'fitt' datasets). N.B.: The value 'thr' is floating point, in the range 0.0<thr<1.0 for 'fith' and 'fico' datasets, and 0.0<thr<32.7 for 'fitt' datasets.

Gaussian blur options:

-1blur_sigma bmm Isotropic Gaussian blur with sigma = bmm (in mm).

-1blur_rms bmm Isotropic Gaussian blur with rms deviation = bmm.

-1blur_fwhm bmm Isotropic Gaussian blur with FWHM = bmm.

-t1blur_sigma bmm Isotropic Gaussian blur of the threshold values with sigma = bmm (in mm).

-t1blur_rms bmm Isotropic Gaussian blur of the threshold values with rms deviation = bmm.

-t1blur_fwhm bmm Isotropic Gaussian blur of the threshold values with FWHM = bmm.

N.B.: The 3 '-1blur' and 3 '-t1blur' options just provide different ways of specifying the radius used for the blurring function. The relationships among these specifications are

sigma = 0.57735027 * rms = 0.42466090 * fwhm

The requisite convolutions are done using FFTs; this is one of the slowest operations among the editing options.

-1zvol x1 x2 y1 y2 z1 z2 Zero out entries inside the 3D volume defined by x1≤x≤x2, y1≤y≤y2, z1≤z≤z2. N.B.: The ranges of x, y, z in a dataset can be found using the '3dinfo' program. Dimensions are in mm.

N.B.: This option may not work correctly at this time, but I've not figured out why!

The following cluster options are mutually exclusive:

-1clust rmm vmul Form clusters with connection distance rmm and clip off data not in clusters of volume at least vmul microliters.

-1clust_mean rmm vmul Same as -1clust, but all voxel intensities within a cluster are replaced by the average intensity of the cluster.

-1clust_max rmm vmul Same as -1clust, but all voxel intensities within a cluster are replaced by the maximum intensity of the cluster.

-1clust_amax rmm vmul Same as -1clust, but all voxel intensities within a cluster are replaced by the maximum absolute value of the intensity of the cluster.

-1clust_smax rmm vmul Same as -1clust, but all voxel intensities within a cluster are replaced by the maximum signed absolute value of the intensity of the cluster, i.e., the sign is preserved.

-1clust_size rmm vmul Same as -1clust, but all voxel intensities within a cluster are replaced by the size of the cluster (in multiples of vmul).

-1clust_order rmm vmul Same as -1clust, but all voxel intensities within a cluster are replaced by the cluster size index (largest cluster = 1, next largest = 2, etc.).

The following commands produce erosion and dilation of 3D clusters. These commands assume that one of the -1clust commands has been used. The purpose is to avoid forming strange clusters with 2 (or more) main bodies connected by thin 'necks'. Erosion can cut off the neck. Dilation will minimize erosion of the main bodies.

Note: Manipulation of values inside a cluster (-1clust commands) occurs AFTER the following two commands have been executed.

-1erode pv For each voxel, set the intensity to zero unless *pv* % of the voxels within radius rmm are nonzero.

-1dilate Restore voxels that were removed by the previous command if there remains a nonzero voxel within rmm.

The following spatial filtering options are mutually exclusive:

-1filter_mean rmm Set each voxel to the average intensity of the voxels within a radius of rmm.

-1filter_nzmean rmm Set each voxel to the average intensity of the non-zero voxels within a radius of rmm.

-1filter_max rmm Set each voxel to the maximum intensity of the voxels within a radius of rmm.

-1filter_amax rmm Set each voxel to the maximum absolute value of the intensity of the voxels within a radius of rmm.

-1filter_smax rmm Set each voxel to the maximum absolute value (preserving the sign) of the intensity of the voxels within a radius of rmm.

-1filter_aver rmm Same idea as '_mean', but implemented using a new code that should be faster.

The following threshold spatial filtering options are mutually exclusive:

-t1filter_mean rmm Set each correlation or threshold voxel to the average of the voxels within a radius of rmm.

-t1filter_nzmean rmm Set each correlation or threshold voxel to the average of the non-zero voxels within a radius of rmm.

-t1filter_max rmm Set each correlation or threshold voxel to the maximum of the voxels within a radius of rmm.

-t1filter_amax rmm Set each correlation or threshold voxel to the maximum absolute value of the intensity of the voxels within a radius of rmm.

-t1filter_smax rmm Set each correlation or threshold voxel to the maximum absolute value (preserving the sign) of the intensity of the voxels within a radius of rmm.

-t1filter_aver rmm Same idea as '_mean', but implemented using a new code that should be faster.

-1mult factor Multiply intensities by the given factor.

-1zscore If the sub-brick is labeled as a statistic from a known distribution, it will be converted to an equivalent $N(0,1)$ deviate (or a 'z score'). If the sub-brick is not so labeled, nothing will be done.

The above '-1' options are carried out in the order given above, regardless of the order in which they are entered on the command line.

-datum type Coerce the output data to be stored as the given type, which may be byte, short, or float.

N.B.: Byte data cannot be negative. If this datum type is chosen, any negative values in the edited and/or merged dataset will be set to zero.

-keepthr When using 3dmerge to edit exactly one dataset of a functional type with a threshold statistic attached, normally the resulting dataset is of the 'fim' (intensity only) type. This option tells 3dmerge to copy the threshold data (unedited in any way) into the output dataset.

N.B.: This option is ignored if 3dmerge is being used to combine 2 or more datasets.

N.B.: The -datum option has no effect on the storage of the threshold data. Instead use '-thdatum type'.

-doall Apply editing and merging options to ALL sub-bricks uniformly in a dataset.

N.B.: All datasets must have the same number of sub-bricks when using the -doall option.

N.B.: The threshold specific options (such as -1thresh, -keepthr, -trfisher, etc.) are not compatible with the -doall command. Neither are the -1dindex or the -1tindex options, described below.

N.B.: All labels and statistical parameters for individual sub-bricks are copied from the first dataset. It is the responsibility of the user to verify that these are appropriate. Note that sub-brick auxiliary data can be modified using program 3drefit.

-1dindex j Use sub-brick #j as the data source.

-1tindex k Use sub-brick #k as the threshold source.

With these commands, you can operate on any given sub-brick of the input dataset(s) to produce as output a 1 brick dataset. If desired, a collection of 1 brick datasets can later be assembled into a multi-brick bucket dataset using program 3dbucket or into a 3D+time dataset using program 3dTcat.

N.B.: If these options aren't used, j=0 and k=1 are the defaults.

32.4 Merging Options Applied to Form the Output Dataset

Here we describe different ways to combine several datasets. The following '-g' options are mutually exclusive!

-gmean Combine datasets by averaging intensities (including zeros) – this is the default.

-gnzmean Combine datasets by averaging intensities (not counting zeros).

-gmax Combine datasets by taking max intensity (e.g., -7 and 2 combine to 2)

-gamax Combine datasets by taking max absolute intensity (e.g., -7 and 2 combine to 7)

-gsmax Combine datasets by taking max signed intensity (e.g., -7 and 2 combine to -7)

-gcount Combine datasets by counting number of 'hits' in each voxel (a 'hit' occurs if a voxel is nonzero).

-gorder Combine datasets in order of input:

- If a voxel is nonzero in dataset #1, then that value goes into the voxel.
- If a voxel is zero in dataset #1 but nonzero in dataset #2, then the value from #2 is used.
- And so forth: the first dataset with a nonzero entry in a given voxel 'wins'

-gfisher Take the arctanh of each input, average these, and output the tanh of the average. If the input datum is 'short', then input values are scaled by 0.0001 and output values by 10000. This option is for merging bricks of correlation coefficients.

32.5 Merging Operations Applied to the Threshold Data

Here we describe different ways to combine the thresholds. If none of these are given, the thresholds will not be merged and the output dataset will not have threshold data attached. Note that the following '-tg' command line options are mutually exclusive, but are independent of the '-g' options given above for merging the intensity data values.

-tgfisher This option is only applicable if each input dataset is of the 'fico' or 'fith' types — functional intensity plus correlation or plus threshold. (In the latter case, the threshold values are interpreted as correlation coefficients.) The correlation coefficients are averaged as described by -gfisher above, and the output dataset will be of the fico type if all inputs are fico type; otherwise, the output datasets will be of the fith type. N.B.: The difference between the -tgfisher and -gfisher methods is that -tgfisher applies to the threshold data stored with a dataset, while -gfisher applies to the intensity data. Thus, -gfisher would normally be applied to a dataset created from correlation coefficients directly, or from the application of the -1thtoin option to a fico or fith dataset.

32.6 Optional Ways to Postprocess the Combined Results

These options may be combined with the above methods. Any combination of these options may be used.

-ghits count Delete voxels that aren't $\neq 0$ in at least count datasets ($\neq 0$ is a 'hit')

-gclust rmm vmul Form clusters with connection distance rmm and clip off data not in clusters of volume at least vmul microliters

The '-g' and '-tg' options apply to the entire group of input datasets.

32.7 Options that Control the Names of the Output Dataset

-session dirname Write output into given directory (default=./)

-prefix pname Use 'pname' for the output directory prefix (default=mrg)

32.8 Notes

- If only one dataset is read into this program, then the '-g' options do not apply, and the output dataset is simply the '-1' options applied to the input dataset (i.e., edited).
- A merged output dataset is ALWAYS of the intensity only variety.

- Complex-valued datasets cannot be merged.
- This program cannot handle time-dependent datasets.
- Note that the input datasets are specified by their .HEAD files, but that their .BRIK files must exist also!

32.9 Examples

Example 1. An investigator wishes to combine functional images fred.01+tlrc, fred.02+tlrc, fred.03+tlrc, and fred.04+tlrc (.BRIK and .HEAD) by taking the average of the intensity, at each voxel, over the 4 datasets. A batch command file to accomplish this is as follows:

```
3dmerge -gmean -prefix fred.gmean \
fred.01+tlrc fred.02+tlrc fred.03+tlrc fred.04+tlrc
```

After execution, file fred.gmean+tlrc (.BRIK and .HEAD) will contain the average intensities of the input data sets.

Example 2. The investigator decides, after examining the previous output using program afni, that it would be better to combine, by averaging, only those voxels which show a significant functional activation. This is accomplished through use of the -1thresh command. Assuming that the functional images fred.01+tlrc through fred.04+tlrc are of the afni 'fico' type, i.e., functional intensity + correlation coefficient, the correlation coefficient sub-brick can be used to set a threshold to censor the intensities. A batch command file to do this might be:

```
3dmerge -gmean –1thresh 0.5 -prefix fred.thresh.gmean \
fred.01+tlrc fred.02+tlrc fred.03+tlrc fred.04+tlrc
```

Only voxels whose correlation coefficient is greater than 0.5 (in absolute value) will be averaged, and the output is sent to file fred.thresh.gmean+tlrc (.BRIK and .HEAD).

Example 3. After examining the previous output, the investigator decides that slight mis-alignment of the functional images has occurred. Therefore, voxels that should overlap in the different images do not. One way to cope with this is to artificially enlarge the regions of functional activation. This can be accomplished in several different ways. One possibility is to use a Gaussian blur, as in:

```
3dmerge -gmean –1thresh 0.5 -1blur_sigma 3 \
-prefix fred.thresh.blur.gmean \
fred.01+tlrc fred.02+tlrc fred.03+tlrc fred.04+tlrc
```

Each input functional image is "blurred" by convolution with a Gaussian function with $\sigma = 3$ mm. Another possibility is to use one of the '-1filter' commands, such as:

```
3dmerge -gmean –1thresh 0.5 -1filter_nzmean 2 \
-prefix fred.thresh.filtnzmean.gmean \
fred.01+tlrc fred.02+tlrc fred.03+tlrc fred.04+tlrc
```

The result of the '-1filter_nzmean 2' command is as follows: the intensity of each voxel in each input data set is replaced by the average of the voxel intensities of all voxels (not counting voxels having zero intensity) within a radius of 2 mm of the original voxel. This has the effect of "spreading" the regions of non-zero activation.

Example 4. The above example showed how to "grow" regions of functional activation. However, this also "grows" isolated activated voxels, which may be due to noise. Since clusters of activated voxels are far less likely to be the result of random noise than isolated voxels, the researcher decides to keep only clusters of activated voxels from the input data sets. This is accomplished using one of the '-1clust' commands. For example, consider the following batch command file:

```
3dmerge -gmean –1thresh 0.5 –1clust_mean 2 300 -1filter_nzmean 2 \
-prefix fred.thresh.clustmean.filtnzmean.gmean \
fred.01+tlrc fred.02+tlrc fred.03+tlrc fred.04+tlrc
```

The '-1clust_mean 2 300' does the following: For each input data set, the program identifies clusters of active (nonzero) voxels. Two activated voxels are in the same cluster if they are separated by not more than 2 mm. Therefore, every voxel is a member of exactly 1 cluster (even if the cluster contains only 1 voxel). After the clusters have been identified, any cluster with volume less than 300 mm^3 is discarded. Therefore, only active voxels belonging to a cluster of volume $\geq$ 300 mm^3 remain. Now, since the '-1clust_*mean*' option is used, the intensity of each voxel within a cluster is replaced by the average intensity of the voxels within that cluster. Since this is followed by the -1filter_nzmean, the effect is to "grow" those clusters having the specified minimum size. The output is stored in file fred.thresh.clustmean.filtnzmean.gmean+tlrc (.HEAD and .BRIK).

Example 5. An experimenter has created a set of 3D functional images. These images were generated for a group of 10 subjects. Each subject was subjected to 4 different tests on 5 consecutive days, for a total of 200 functional volumes. Due to the amount of computer time required to process so many images, the experimenter decided that it would be best to "pre-process" the images, producing data files that contain the clustered and filtered images.

Suppose that the data file names have the format: fred.ii.jj.kk+orig (.HEAD and .BRIK), where ii = subject id number, jj = test number, and kk = day number. A C-shell batch command file to threshold, cluster, and filter each of the data sets is given by:

```
foreach subject ( 1 2 3 4 5 6 7 8 9 10 )
  foreach test ( 1 2 3 4 )
    foreach day ( 1 2 3 4 5 )
      3dmerge -1thresh 0.5 -1clust_mean 2 300 -1filter_nzmean 2 \
     -prefix fred.proc.${subject}.${test}.${day} \
      fred.${subject}.${test}.${day}+orig
    end
  end
end
```

Each input file fred.ii.jj.kk+orig (.HEAD and .BRIK) produces a corresponding output file fred.proc.ii.jj.kk+orig (.HEAD and .BRIK). The user may now experiment with different ways of combining these output files, without having to repeat the lengthy process of thresholding, clustering, and filtering for each individual data set.

33 Program 3dnewid

33.1 Purpose

Assigns a new ID code to a dataset; this is useful when making a copy of a dataset, so that the internal ID codes remain unique.

33.2 Usage

3dnewid dataset [dataset ...]

34 Program 3dnoise

34.1 Purpose

Estimates noise level in 3D datasets, and optionally set voxels below the noise threshold to zero. This only works on datasets that are stored as shorts, and whose elements are all nonnegative.

34.2 Usage

3dnoise [-blast] [-snr fac] [-nl x] datasets ...

34.3 Options

-blast Set values at or below the cutoff to zero. In 3D+time datasets, a spatial location is set to zero only if a majority of time points fall below the cutoff; in that case all the values at that location are zeroed.

-snr fac Set cutoff to 'fac' times the estimated noise level. Default fac = 2.5. What to use for this depends strongly on your MRI system – I often use 5, but our true SNR is about 100 for EPI.

-nl x Set the noise level to 'x', skipping the estimation procedure.

35 Program 3dnvals

35.1 Purpose

Prints out the number of sub-bricks in a 3D dataset

35.2 Usage

3dnvals dataset

36 Program 3dpc

Principal Component Analysis of 3D Datasets

36.1 Usage

3dpc [options] dataset dataset ...

Each dataset may have a sub-brick selector list. Otherwise, all sub-bricks from a dataset will be used.

36.2 Options

-dmean = remove the mean from each input brick (across space)

-vmean = remove the mean from each input voxel (across bricks)
[N.B.: -dmean and -vmean are mutually exclusive]
[default: don't remove either mean]

-normalize = L2 normalize each input brick (after mean subtraction)
[default: don't normalize]

-pcsave sss = 'sss' is the number of components to save in the output; it can't be more than the number of input bricks
[default = all of them = number of input bricks]

-prefix pname = Name for output dataset (will be a bucket type); also, the eigen-timeseries will be in 'pname'.1D (all of them) and in 'pnameNN.1D' for eigenvalue #NN individually (NN=00 .. 'sss'-1, corresponding to the brick index in the output dataset)
[default value of pname = 'pc']

-1ddum ddd = Add 'ddd' dummy lines to the top of each *.1D file. These lines will have the value 999999, and can be used to align the files appropriately.
[default value of ddd = 0]

-verbose = Print progress reports during the computations

-float = Save eigen-bricks as floats
[default = shorts, scaled so that |max|=10000]

37 Program 3dproject

37.1 Purpose

Program 3dproject projects datasets along the three orthogonal directions to produce (up to) three 2D image files. Its options are to

- Project using the voxel sum, or using the voxel max along each ray.
- Edit out negative values.
- Project over the entire thickness of the dataset, or over any subrange (e.g., it is possible to get left hemisphere and right hemisphere projections separately).

Program 3dproject does not produce a brain outline or Talairach grid to provide any reference for projected images.

The output image files are by default in the internal *AFNI* format. This may not be useful for you, so you can either write them as 'normal sized' files with the -nsize option, or use program nsize (supplied with *AFNI*) to convert them later.

37.2 Usage

The command line format for program 3dproject is as follows:
3dproject [editing options][-sum|-max|-amax|-smax] [-output root] [-nsize] [-mirror] [-RL {all | x1 x2}] [-AP {all | y1 y2}] [-IS {all | z1 z2}] [-ALL] dataset

The options are explained below.

37.3 Options

-sum Add the dataset voxels along the projection direction.

-max Take the maximum of the voxels along the projection direction.

-amax Take the absolute maximum of the voxels along the projection direction.

-smax Take the signed maximum of the voxels along the projection direction.

For example:

Option	Voxel values	Projected value
-sum	-7, 2, 4, 5	4
-max	-7, 2, 4, 5	5
-amax	-7, 2, 4, 5	7
-smax	-7, 2, 4, 5	-7

Note: The default is -sum.

-nsize Scale the output images up to 'normal' sizes (e.g., 64x64, 128x128, or 256x256). This option only applies to byte or short datasets.

-mirror The radiologists' and AFNI convention is to display axial and coronal images with the subject's left on the right of the image; the use of this option will mirror the axial and coronal projections so that left is left and right is right.

-output root Output projections will be named root.sag, root.cor, and root.axi. The default root is 'proj'.

-RL all Project in the Right-to-Left direction along all the data (produces root.sag)

-RL x1 x2 Project in the Right-to-Left direction from x-coordinate x1 to x2 (mm) (negative x is Right, positive x is Left). Or, you may use something like -RL 10R 20L to project from x=-10 mm to x=+20 mm.

-AP all Project in the Anterior-to-Posterior direction along all the data (produces root.cor)

-AP y1 y2 Project in the Anterior-to-Posterior direction from y-coordinate y1 to y2 (mm) (negative y is Anterior, positive y is Posterior). Or, you may use something like -AP 10A 20P to project from y=-10 mm to y=+20 mm.

-IS all Project in the Inferior-to-Superior direction along all the data (produces root.axi)

-IS y1 y2 Project in the Inferior-to-Superior direction from z-coordinate z1 to z2 (mm) (negative z is Inferior, positive z is Superior). Or, you may use something like -IS 10I 20S to project from z=-10 mm to z=+20 mm.

-ALL ==> Equivalent to '-RL all -AP all -IS all'.

37.4 Notes

- A projection direction will not be used if the bounds aren't given for that direction; thus, at least one of -RL, -AP, or -IS must be used, or nothing will be computed!
- In the directions transverse to the projection direction, all the data is used; that is, '-RL -5 5' will produce a full sagittal image summed over a 10 mm slice, irrespective of the -IS or -AP extents.
- The [editing options] are the same as in 3dmerge. In particular, the '-1thtoin' option can be used to project the threshold data (if available).

37.5 Example

Example 1. The user wishes to project, over the entire thickness of the dataset, the *AFNI* 3D dataset anat+orig along the three orthogonal directions to produce three 2D image files. The command line is:

3dproject -ALL -output fred anat+orig

The resulting *AFNI* 2D datasets are: fred.axi, fred.cor, and fred.sag.

38 Program 3drefit

38.1 Purpose

Changes some of the information inside a 3D dataset's header. Note that this program does NOT change the .BRIK file at all; the main purpose of 3drefit is to fix up errors made when using to3d. To see the current values stored in a .HEAD file, use the command '3dinfo dataset'. Using 3dinfo both before and after 3drefit is a good idea to make sure the changes have been made correctly!

38.2 Usage

3drefit [options] dataset ...

38.3 Options

-orient code

Sets the orientation of the 3D volume(s) in the .BRIK. The code must be 3 letters, one each from the pairs {R,L} {A,P} {I,S}. The first letter gives the orientation of the x-axis, the second the orientation of the y-axis, the third the z-axis:

R	= right-to-left	L	= left-to-right
A	= anterior-to-posterior	P	= posterior-to-anterior
I	= inferior-to-superior	S	= superior-to-inferior

** WARNING: when changing the orientation, you must be sure to check the origins as well, to make sure that the volume is positioned correctly in space.

-xorigin distx
-yorigin disty
-zorigin distz

Puts the center of the edge voxel off at the given distance, for the given axis (x,y,z); distances in mm. (x=first axis, y=second axis, z=third axis). Usually, only -zorigin makes sense. Note that this distance is in the direction given by the corresponding letter in the -orient code. For example, '-orient RAI' would mean that '-zorigin 30' sets the center of the first slice at 30 mm Inferior. See the to3d manual for more explanations of axes origins.

** SPECIAL CASE: you can use the string 'cen' in place of a distance to force that axis to be re-centered.

-xdel dimx
-ydel dimy
-zdel dimz

Makes the size of the voxel the given dimension, for the given axis (x,y,z); dimensions in mm.

** WARNING: if you change a voxel dimension, you will probably have to change the origin as well.

-TR time

Changes the TR time to a new value (see 'to3d -help').

** WARNING: this only applies to 3D+time datasets.

-newid

Changes the ID code of this dataset as well.

-nowarp

Removes all warping information from dataset.

-statpar v ...

Changes the statistical parameters stored in this dataset. See 'to3d -help' for more details.

-markers

Adds an empty set of AC-PC markers to the dataset, if it can handle them (is anatomical, is in the +orig view, and isn't 3D+time).
** WARNING: this will erase any markers that already exist!

-view code
Changes the 'view' to be 'code', where the string 'code' is one of 'orig', 'acpc', or 'tlrc'.
** WARNING: The program will also change the .HEAD and .BRIK filenames to match. If the dataset filenames already exist in the '+code' view, then this option will fail. You will have to rename the dataset files before trying to use '-view'. If you copy the files and then use '-view', don't forget to use '-newid' as well!

-byteorder bbb
Sets the byte order string in the header. Allowable values for 'bbb' are:
LSB_FIRST MSB_FIRST NATIVE_ORDER
Note that this does not change the .BRIK file! This is done by programs 2swap and 4swap.

-appkey ll
Appends the string 'll' to the keyword list for the whole dataset.

-repkey ll
Replaces the keyword list for the dataset with the string 'll'.

-empkey
Destroys the keyword list for the dataset.

-type
Changes the type of data that is declared for this dataset, where 'type' is chosen from the following:

ANATOMICAL TYPES

spgr	==	Spoiled GRASS	fse	==	Fast Spin Echo
epan	==	Echo Planar	anat	==	MRI Anatomy
ct	==	CT Scan	spct	==	SPECT Anatomy
pet	==	PET Anatomy	mra	==	MR Angiography
bmap	==	B-field Map	diff	==	Diffusion Map
omri	==	Other MRI	abuc	==	Anat Bucket

FUNCTIONAL TYPES

fim	==	Intensity	fith	==	Inten+Thr
fico	==	Inten+Cor	fitt	==	Inten+Ttest
fift	==	Inten+Ftest	fizt	==	Inten+Ztest
fict	==	Inten+ChiSq	fibt	==	Inten+Beta
fibn	==	Inten+Binom	figt	==	Inten+Gamma
fipt	==	Inten+Poisson	fbuc	==	Func-Bucket

The options below allow you to attach auxiliary data to sub-bricks in the dataset. Each option may be used more than once so that multiple sub-bricks can be modified in a single run of 3drefit.

-sublabel	**n**	**ll**	Attach to sub-brick #n the label string 'll'.
-subappkey	**n**	**ll**	Add to sub-brick #n the keyword string 'll'.
-subrepkey	**n**	**ll**	Replace sub-brick #n's keyword string with 'll'.
-subempkey	**n**		Empty out sub-brick #n' keyword string

-substatpar n type v ...
Attach to sub-brick #n the statistical type and the auxiliary parameters given by values 'v ...', where 'type' is one of the following:

type	Description	PARAMETERS
fico	Cor	Samples, Fit-Parameters, Ort-Parameters
fitt	Ttest	Degrees-of-Freedom
fift	Ftest	Numerator and Denominator Degrees-of-Freedom
fizt	Ztest	N/A
fict	ChiSq	Degrees-of-Freedom
fibt	Beta	Alpha and Beta (exponenents)
fibn	Binom	Number-of-Trials and Probability-per-Trial
figt	Gamma	Shape and Scale
fipt	Poisson	Mean

39 Program 3drotate

39.1 Purpose

Rotates and/or translates all bricks from an AFNI dataset.

39.2 Usage

3drotate [options] dataset
'dataset' may contain a sub-brick selector list.

39.3 Options

-prefix fname
Sets the output dataset prefix name to be 'fname'

-verbose
Prints out progress reports

At most one of these shift options can be used:
-ashift dx dy dz = Shifts the dataset 'dx' mm in the x-direction, etc., AFTER rotation.
-bshift dx dy dz = Shifts the dataset 'dx' mm in the x-direction, etc., BEFORE rotation.

The shift distances by default are along the (x,y,z) axes of the dataset storage directions (see the output of '3dinfo dataset'). To specify them anatomically, you can suffix a distance with one of the symbols 'R', 'L', 'A', 'P', 'I', and 'S', meaning 'Right', 'Left', 'Anterior', 'Posterior', 'Inferior', and 'Superior', respectively.

-rotate th1 th2 th3
Specifies the 3D rotation to be composed of 3 planar rotations:
1) 'th1' degrees about the 1st axis, followed by
2) 'th2' degrees about the (rotated) 2nd axis, followed by
3) 'th3' degrees about the (doubly rotated) 3rd axis.

Which axes are used for these rotations is specified by placing one of the symbols 'R', 'L', 'A', 'P', 'I', and 'S' at the end of each angle (e.g., '10.7A'). These symbols denote rotation about the 'Right-to-Left', 'Left-to-Right', 'Anterior-to-Posterior', 'Posterior-to-Anterior', 'Inferior-to-Superior', and 'Superior-to-Inferior' axes, respectively. A positive rotation is defined by the right-hand rule.

Algorithm: The rotation+shift is decomposed into 4 1D shearing operations (the 3D generalization of Paeth's algorithm). The interpolation (i.e., resampling) method used for these shears can be controlled by the following options:

-Fourier	=	Use a Fourier method (the default: most accurate; slowest).
-linear	=	Use linear (1st order polynomial) interpolation (least accurate).
-cubic	=	Use the cubic (3rd order) Lagrange polynomial method.
-quintic	=	Use the quintic (5th order) Lagrange polynomial method.
-heptic	=	Use the heptic (7th order) Lagrange polynomial method.

39.4 Example

3drotate -prefix Elvis -bshift 10S 0 0 -rotate 30R 0 0 Sinatra+orig

This will shift the input 10 mm in the superior direction, followed by a 30 degree rotation about the Right-to-Left axis (i.e., nod the head forward).

40 Program 3dttest

Program 3dttest performs a t-test on sets of 3D datasets. This test is used to determine, on a voxel by voxel basis, if the two sets of data are significantly different from each other (or, if a single set of data is significantly different from a fixed constant). When two sets of data are to be compared, the user has the option of using a paired t-test.

40.1 Usage

The command line format for program 3dttest depends on whether the program is to be used for a 1-sample or a 2-sample t-test:

Usage 1: 3dttest [options] -set1 datasets . . . -set2 datasets . . . This usage is for comparing the means of 2 sets of datasets (voxel by voxel).

Usage 2: 3dttest [options] -base1 bval -set2 datasets . . . This usage is for comparing the mean of 1 set of datasets against a constant.

40.2 Outputs

A single dataset is created that is the voxel-by-voxel difference of the mean of set2 minus the mean of set1 (or minus 'bval'). The output dataset will be of the intensity+Ttest ('fitt') type. The t-statistic at each voxel can be used as an interactive thresholding tool in *AFNI*.

The *AFNI* 'fitt' type dataset consists of two sub-bricks. In the case of a 2 sample unpaired t-test, the first sub-brick contains the estimated differences $\hat{D} = \bar{Y}_{2.} - \bar{Y}_{1.}$, where $\bar{Y}_{1.}$ and $\bar{Y}_{2.}$ are the average, at each voxel, over the 1st and 2nd datasets, respectively:

$$\begin{aligned} \bar{Y}_{1.} &= \frac{\sum_i Y_{1i}}{n_1} \\ \bar{Y}_{2.} &= \frac{\sum_i Y_{2i}}{n_2} \end{aligned}$$

The second sub-brick contains the corresponding t-statistics.

$$\textit{AFNI}\ \text{“fitt” dataset} \quad \begin{cases} \boxed{\hat{D} = \bar{Y}_{2.} - \bar{Y}_{1.}} \\ \\ \boxed{t^* = \dfrac{\hat{D}}{\sqrt{s^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}} \end{cases}$$

Here, s^2 is the estimator of the common variance:

$$s^2 = \frac{\sum_i \left(Y_{1i} - \bar{Y}_1\right)^2 + \sum_i \left(Y_{2i} - \bar{Y}_2\right)^2}{n_1 + n_2 - 2} \ .$$

Under the hypothesis that the mean of set 1 is equal to the mean of set 2, t^* has the t-distribution with $n_1 + n_2 - 2$ degrees of freedom.

40.3 t-Testing Options

-set1 datasets . . .

Specifies the collection of datasets to put into the first set. The mean of set1 will be tested with a 2-sample t-test against the mean of set2. Note that options -set1 and -base1 are mutually exclusive!

-base1 bval

'bval' is a numerical value that the mean of set2 will be tested against with a 1-sample t-test.

When the -base1 option is used, the output dataset looks like:

$$AFNI\ \text{“fitt” dataset} \left\{ \begin{array}{l} \boxed{\hat{D} = \frac{1}{n}\sum_i (Y_{2i} - bval)} \\ \\ \boxed{t^* = \dfrac{\hat{D}}{\sqrt{s^2\left(\frac{1}{n}\right)}}} \end{array} \right.$$

where

$$s^2 = \frac{\sum_i \left(Y_{2i} - bval - \hat{D}\right)^2}{n-1}$$

.

-set2 datasets . . .
Specifies the collection of datasets to put into the second set. There must be at least 2 datasets in each of set1 (if used) and set 2.

-paired
Specifies the use of a paired-sample t-test to compare set1 and set2. If this option is used, set1 and set2 must have the same cardinality. Note that a paired t-test is intended for use when the set1 and set2 dataset function values may be pairwise correlated. If they are in fact uncorrelated, this test has less statistical 'power' than the unpaired (default) t-test. This loss of power is the price that is paid for insurance against pairwise correlations.

When the -paired option is used, the output dataset looks like:

$$AFNI\ \text{“fitt” dataset} \left\{ \begin{array}{l} \boxed{\hat{D} = \frac{1}{n}\sum_i (Y_{2i} - Y_{1i})} \\ \\ \boxed{t^* = \dfrac{\hat{D}}{\sqrt{s^2\left(\frac{1}{n}\right)}}} \end{array} \right.$$

where

$$s^2 = \frac{\sum_i \left(Y_{2i} - Y_{1i} - \hat{D}\right)^2}{n-1}$$

-unpooled
Specifies that the variance estimates for set1 and set2 be computed separately (not pooled together). This only makes sense if -paired is NOT given. N.B.: If this option is used, the number of degrees of freedom per voxel is a variable, rather than a constant. NOT RECOMMENDED.

-workmen mega
'mega' specifies the number of megabytes of RAM to use for statistical workspace. It defaults to 12. The program will run faster if this is larger (see the NOTES section below).

The -base1 or -set1 command line switches must follow all other options (including those described below) except for the -set2 switch.

40.4 Input Editing Options

The same as are available in 3dmerge.

40.5 Output Options

These options control the output files.

-session dirname
Write output into given directory (default = ./).

-prefix pname
Use 'pname' for the output directory prefix (default = tdif).

-datum type
Use 'type' to store the output difference in the means; 'type' may be short or float. How the default is determined is described in the notes below.

40.6 Notes

- To economize on memory, 3dttest makes multiple passes through the input datasets. On each pass, the entire editing process will be carried out again. For efficiency's sake, it is better to carry out the editing using 3dmerge to produce temporary datasets, and then run 3dttest on them. This applies with particular force if a 'blurring' or 'filtering' option is used. Note also that editing a dataset requires that it be read into memory in its entirety (so that the disk file is not altered). This will increase the memory needs of the program far beyond the level set by the -workmem option.
- The input datasets are specified by their .HEAD files, but their .BRIK files must exist also! This program cannot 'warp-on-demand' from other datasets.
- This program cannot deal with time-dependent or complex-valued datasets! By default, the output dataset function values will be shorts if the first input dataset is byte- or short-valued; otherwise they will be floats. This behavior may be overridden using the -datum option. However, the t-statistic at each voxel will always be stored as a short integer that is 1000 times the actual t-value.

40.7 Examples

Example 1: A researcher wishes to test whether left-handed and right-handed people have significantly different neural activation when performing a particular task. The collection of 3D data for left-handed people, selected at random, is stored in files L001+tlrc through L005+tlrc (both .HEAD and .BRIK files). The collection of 3D data for right-handed people is stored in files R001+tlrc through R007+tlrc (both .HEAD and .BRIK files). A command sequence for performing a t-test on this data is:

```
3dttest -prefix LRttest
-set1 L001+tlrc L002+tlrc L003+tlrc L004+tlrc L005+tlrc \
-set2 R001+tlrc R002+tlrc R003+tlrc R004+tlrc R005+tlrc \
R006+tlrc R007+tlrc
```

Example 1 illustrates the 2-sample, unpaired t-test. The output is stored in the *AFNI* 3D dataset LRttest+tlrc.BRIK (and .HEAD). For Example 1, $n_1 = 5$ and $n_2 = 7$.

Example 2: An investigator is trying to determine if there is a significant difference between two tasks. Each of the 10 randomly selected subjects is asked to perform both tasks, and the resulting set of 20 functional images is used as input to program 3dttest. To "remove" person-to-person variation, a paired t-test is used:

```
3dttest -prefix ttestabc -paired \
-set1 subj01.task1+tlrc subj02.task1+tlrc ... subj10.task1+tlrc \
-set2 subj01.task2+tlrc subj02.task2+tlrc ... subj10.task2+tlrc
```

Example 2 illustrates the paired t-test. In this example, $n = 10$ (the number of pairs of datasets). The output *AFNI* 'fitt' dataset is stored in file ttestabc+tlrc.HEAD (and .BRIK).

Example 3: To determine for which voxels in a dataset the mean is significantly different from the predicted value of 34.5, the following command line is used:

```
3dttest -prefix ttestout -base1 34.5 \
-set2 fim001+tlrc fim002+tlrc fim003+tlrc fim004+tlrc fim005+tlrc \
fim006+tlrc fim007+tlrc fim008+tlrc fim009+tlrc fim010+trlc \
fim011+tlrc fim012+tlrc
```

Example 3 illustrates the -base1 option. The mean of set2 is to be tested against the value $bval = 34.5$. In this example, $n = 12$ (the number of datasets). The output is stored in file ttestout+tlrc.HEAD (and .BRIK).

41 Program 3dvolreg

41.1 Purpose

Registers each 3D sub-brick from the input dataset to the base brick.

41.2 Usage

3dvolreg [options] dataset
'dataset' may contain a sub-brick selector list.

41.3 Options

-verbose Print progress reports. Use twice for LOTS of output.

Interpolation options:

-Fourier	Perform the alignments using Fourier interpolation.
-heptic	Use heptic polynomial interpolation.
-quintic	Use quintic polynomical interpolation.
-cubic	Use cubic polynomial interpolation.

Default = Fourier [slowest and most accurate interpolator]

-clipit Clips the values in each output sub-brick to be in the same range as the corresponding input volume. The interpolation schemes can produce values outside the input range, which is sometimes annoying.

-prefix fname Use 'fname' for the output dataset prefix. The program tries not to overwrite an existing dataset.
Default = 'volreg'.

-base n Sets the base brick to be the 'n'th sub-brick from the input dataset (indexing starts at 0).
Default = 0 (first sub-brick).

-base 'bset[n]' Sets the base brick to be the 'n'th sub-brick from the dataset specified by 'bset', as in

-base 'elvis+orig[4]'

The quotes are needed because the '[]' characters are special to the shell.

-dfile dname Save the motion parameters in file 'dname'. The output is in 9 ASCII formatted columns:

n roll pitch yaw dS dL dP rmsold rmsnew

where:

n = sub-brick index

$$\left.\begin{array}{lcl} \text{roll} & = & \text{rotation about the I-S axis} \\ \text{pitch} & = & \text{rotation about the R-L axis} \\ \text{yaw} & = & \text{rotation about the A-P axis} \end{array}\right\}\text{degrees CCW}$$

$$\left.\begin{array}{lcl} \text{dS} & = & \text{displacement in the Superior direction} \\ \text{dL} & = & \text{displacement in the Left direction} \\ \text{dP} & = & \text{displacement in the Posterior direction} \end{array}\right\}\text{mm}$$

rmsold = RMS difference between input brick and base brick
rmsnew = RMS difference between output brick and base brick

N.B.: If the '-dfile' option is not given, the parameters aren't saved.

N.B.: The motion parameters are those needed to bring the sub-brick back into alignment with the base. In 3drotate, it is as if the following options were applied to each input sub-brick:

-rotate <roll>I <pitch>R <yaw>A -ashift <dS>S <dL>L <dP>P

Algorithm: Iterated linearized weighted least squares to make each sub-brick as like as possible to the base brick. This method is useful for finding SMALL MOTIONS ONLY. See program 3drotate for the volume shift/rotate algorithm. The following options can be used to control the iterations:

-maxite m = allow up to 'm' iterations for convergence [default = 9].

-x_thresh x = iterations converge when maximum movement is less than 'x' voxels [default=0.050000],

-rot_thresh r = and when maximum rotation is less than 'r' degrees [default=0.070000].

-delta d = distance, in voxel size, used to compute image derivatives using finite differences [default=0.700000].

-final mode = do the final interpolation using the method defined by 'mode', which is one of the strings 'cubic', 'quintic', 'heptic', or 'Fourier' [default=mode used to estimate parameters].

-weight 'wset[n]' = set the weighting applyed to each voxel proportional to the brick specified here [default=smoothed base brick].

Warning: This program can consume very large quantities of memory.

Use of '-verbose -verbose' will show the amount of workspace, and the steps used in each iteration.

42 Program 4swap

42.1 Purpose

Swaps byte quadruples on the files listed. This may be useful for transferring .BRIK files between some CPU architectures (see also program 2swap).

42.2 Usage

4swap [-q] file ...
The -q option means to work quietly.

43 Program AlphaSim

43.1 Purpose

Program AlphaSim provides a means of estimating the overall significance level (the probability of a false detection) for an entire *AFNI* 3D functional image. This is accomplished by Monte Carlo simulation of the process of image generation, spatial correlation of voxels, voxel intensity thresholding, and cluster identification. Based upon the combination of individual voxel probability thresholding and minimum cluster size thresholding, the probability of a false positive detection per image is determined from the frequency count of cluster sizes.

Program AlphaSim can also estimate the statistical power (the probability of a true detection) for a user specified region of true activation. The probability of a true positive detection per image is similarly determined from Monte Carlo simulation.

A word of caution: The applicability of simulation results is limited by the validity of the assumptions involved. In particular, the modelling of spatial correlation of voxels assumes that the underlying population of voxel intensities has a normal distribution. The sensitivity of the results to departures from normality is not well known at this time.

43.2 Further Details

For further details, see *Simultaneous Inference for FMRI Data*, contained in file AlphaSim.ps.

44 Program FD2

44.1 Purpose

Visualization of FMRI 2D datasets.

44.2 Further Details

For further details, see the documentation in file FD2.ps.

45 Program RSFgen

45.1 Purpose

Sample program to generate random stimulus functions.

45.2 Further Details

For further information, see Section 3 of *Deconvolution of FMRI Time Series Data*, contained in file 3dDeconvolve.ps.

46 Program abut

46.1 Purpose

This program can be used to put a set of noncontiguous functional slices together to produce a contiguous set, as required by to3d. It works by supplying slices of all zeros in the 'gaps'. If the gaps are not integer multiples of the data slices thickness (by the way, all the data slices must be the same thickness), then the data slices will be subdivided to produce a higher resolution (in the z-direction) new set of slices. The subdivided slices may be simple replicas of the parent slices, or they may be interpolated from the parent slices.

46.2 Usage

abut [-dzin thickness] [-dzout thickness] [-root name]
[-linear | -blocky] [-verbose] [-skip n+gap] ... images ...

46.3 Options

-dzin The thickness value in mm. If not given, it is taken to be 1.0 (in which case, the output thickness and gap sizes are simply relative to the slice thickness, rather than absolute)

-dzout The output slice thickness, usually smaller than the input thickness. If not given, the program will compute a value (the smaller the ratio dzout/dzin is, the more slices will be output)

-root name 'name' is the root (or prefix) for the output slice filename. For example, '-root fred.' will result in files fred.0001, fred.0002, ...

-linear If present, this flag indicates that subdivided slices will be linearly interpolated rather than simply replicated – this will make the results smoother in the through-slice direction (if dzout < dzin)

-blocky Similar to -linear, but uses *AFNI*'s 'blocky' interpolation when possible to put out intermediate slices. Both interpolation options only apply when dzout < dzin and when an output slice has a non-gappy neighbor.

-skip n+gap Indicates that a gap is to be inserted between input slices #n and #n+1, where n=1,2,...; for example, -skip 6+5.5 means put a gap of 5.5 mm between slices 6 and 7. More than one -skip option is allowed. They must all occur before the list of input image filenames.

47 Program adwarp

47.1 Purpose

Resamples a 'data parent' dataset to the grid defined by an 'anat parent' dataset. The anat parent dataset must contain in its .HEAD file the coordinate transformation (warp) needed to bring the data parent dataset to the output grid. This program provides a batch implementation of the interactive afni 'Write' buttons, one dataset at a time.

47.2 Usage

adwarp [**options**]

47.3 Options

-apar aset
= Set the anat parent dataset to 'aset'. This is a nonoptional option (must be present).

-dpar dset
= Set the data parent dataset to 'dset'. This is a nonoptional option (must be present).

-prefix ppp
= Set the prefix for the output dataset to 'ppp'. The default is the prefix of 'dset'.

-dxyz ddd
= Set the grid spacing in the output datset to 'ddd' mm. The default is 1 mm.

-verbose
= Print out progress reports.

-resam rrr
= Set the resampling mode to 'rrr', which must be one of the following [default is Li]:

NN	=	Nearest Neighbor
Li	=	Linear Interpolation
Cu	=	Cubic Interpolation
Bk	=	Blocky Interpolation

47.4 Example

adwarp -apar anat+tlrc -dpar func+orig

Note: If func+tlrc already exists, it will be overwritten!

48 Program afni

48.1 Purpose

Visualization of FMRI 3D datasets.

48.2 Further Details

For further information, see the documentation in file afni200.ps.

49 Program byteorder

49.1 Purpose

Indicates host CPU byte order (LSB_FIRST or MSB_FIRST)

49.2 Usage

byteorder

50 Program ccalc

50.1 Purpose

Perform interactive arithmetic calculations.

50.2 Usage

ccalc

The computer will respond with the prompt:
`calc>`

50.3 Notes

- The user may enter any valid arithmetic expression, using +, -, *, /, **, and parentheses.
- Intermediate results may be stored in variables 'a' through 'z'.
- Refer to the documentation for program 3dcalc for a list of built-in functions.
- To exit the program, type 'quit'.

51 Program cdf

51.1 Purpose

This program does various conversions using the cumulative distribution function (cdf) of certain canonical probability functions.

51.2 Usage

Usage 1: **cdf [-v] -t2p statname t params**
Usage 2: **cdf [-v] -p2t statname p params**
Usage 3: **cdf [-v] -t2z statname t params**

51.3 Options

The optional '**-v**' indicates to be verbose – this is for debugging purposes, mostly.

Usage 1: Converts a statistic 't' to a tail probability.
Usage 2: Converts a tail probability 'p' to a statistic.
Usage 3: Converts a statistic 't' to a N(0,1) value (or z-score)
that has the same tail probability.

The parameter 'statname' refers to the type of distribution to be used. The numbers in the params list are the auxiliary parameters for the particular distribution. The following table shows the available distribution functions and their parameters:

statname	Description	PARAMETERS
fico	Cor	Samples, Fit-Parameters, Ort-Parameters
fitt	Ttest	Degrees-of-Freedom
fift	Ftest	Numerator and Denominator Degrees-of-Freedom
fizt	Ztest	N/A
fict	ChiSq	Degrees-of-Freedom
fibt	Beta	Alpha and Beta (exponenents)
fibn	Binom	Number-of-Trials and Probability-per-Trial
figt	Gamma	Shape and Scale
fipt	Poisson	Mean

52 Program count

52.1 Purpose

This program produces a list of numbers, optionally with a root at the beginning of each number and a suffix at the end.

52.2 Usage

count [**-digits** #] [**-root name**] [**-suffix name**] **bot top** [**step**]

52.3 Options

-digits Number of digits to represent each number. This defaults to 4.

-root name The string 'name' will precede each number. This defaults to the empty string.

-suffix name The string 'name' will follow each number. This defaults to the empty string

bot top The first and last numbers in the list.

step The interval between numbers in the list. This defaults to 1. If step is of the form 'R#', then # random counts are produced in the range bot..top, inclusive. If bot > top, then count backwards.

52.4 Examples

Example 1. The command line

```
count -root bob. -suffix .cox -digits 3 9 12
```

produces on stdout the following:

```
bob.009.cox   bob.010.cox   bob.011.cox   box.012.cox
```

Example 2. The count command is mostly useful with the C-shell's backquote operator and foreach command. For example,

```
foreach fred ( `count 1 100` )
    mv bob.$fred cox.$fred
end
```

will rename bob.0001 to cox.0001, etc.

53 Program fim2

53.1 Purpose

This program applies the correlation method to match an 'ideal' waveform to the MR intensity time series in each pixel [1]. The actual algorithm used is the recursive projection technique described in [2]. Features of fim2 include:

1. The ability to run several 'canonical' or 'ideal' waveforms, and select the results from the one most highly correlated in each pixel.

2. An 'ort' is a function of time that is orthogonally projected out of each data time series before the correlation is computed. The collection of orts is the matrix **S** in [2]. fim2 has the ability to use several 'orts' at once, and to generate internally polynomial functions of time to use as orts. This means that it is not necessary to provide a ramp time series file in order to remove linear trends in time from the data pixels; the -polort 1 option has the desired effect. (The name 'ort' was invented by Andrzej Jesmanowicz.)

3. The ability to output the contrast-to-noise-ratio (CNR) in each pixel. CNR is defined as α/σ, where α is the amplitude of the normalized ideal waveform, and σ is the standard deviation of the noise left after all ort and the ideal waveforms are projected out. In this context, 'normalized ideal waveform' means the waveform scaled so that the trough-to-peak height is 1. If the ideal waveform is a square wave, for example, then CNR will be the mean in the 'up' times minus the mean in the 'down' times, divided by σ.

4. The ability to output the standard deviation σ in each pixel, and the least squares fit coefficients of the ort and ideal waveforms.

5. The ability to clip to zero output correlations and functional intensities in pixels where the underlying images have low intensity. This is intended to help eliminate false positives that sometimes occur far outside the brain.

6. The ability to register and compensate for **in plane** movement between images by trying to match each image in the time series to a base image. Each image is resampled to the shifted and rotated grid estimated by the 'dfspace' filter. The usual correlation calculations then proceed using these registered images.

53.2 Usage

fim2 [options] imagefiles

where 'imagefiles … ' is a sequence of MRI filenames. The options are:

53.3 Options

-pcnt # Correlation coefficient threshold will be 1 - 0.01 × '#' , where '#' should be between 1 and 100.

-pcthresh # Correlation coefficient threshold will be '#' (0.0 to 0.99).

-im1 # Index of image file to use as first in time series; default is 1; previous images are filled with this image to synchronize with the reference time series.

-num # Number of images to actually use, if more than this many are specified on the command line; default is to use all images.

-non This option turns off the default normalization of the output activation image; the user should provide a scaling factor via '-coef #', or '1' will be used.

-coef # The scaling factor used to convert the activation output from floats to short ints (if '-non' is also present).

-ort fname fname = filename of a time series to which the image data will be orthogonalized before correlations are computed; any number of '-ort' options (from 0 on up) may be used.

-ideal fname fname = filename of a time series to which the image data is to be correlated; at least one such time series is required; if the '-ideal' option is not used, then the first filename after all the options will be used (this is for compatibility with the older fim program by Andrzej Jesmanowicz of MCW).

This version of fim2 allows the specification of more than one ideal time series file. Each one is *separately* correlated with the image time series and the one most highly correlated is selected for each pixel. Multiple ideals are specified using more than one -ideal fname option, or by using the form

-ideal [fname1 fname2 ...]

This latter method allows the use of wildcarded ideal filenames. The '[' character that indicates the start of a group of ideals can actually be any *one* of these: $\boxed{[\ \{\ /\ \%}$ and the ']' that ends the group can be *one* of: $\boxed{]\ \}\ /\ \%}$. (The C-shell doesn't like the [or { characters, which is why the more innocuous / and % are available.)

The format of ort and ideal time series files:

ASCII; one number per line;
Same number of lines as images in the time series;
Value over 33333 means "don't use this image in the analysis".

Yet more options for fim2 are:

-polref # **or** **-polort #** Use polynomials of order 0..# as extra 'orts'; default is 0 (yielding a constant vector). Use # = -1 to suppress this feature.

-fimfile fname fname = filename to save activation magnitudes in; if not given, the last name on the command line will be used.

-corr If present, indicates to write correlation output to image file 'fimfile.CORR' (next option is better).

-corfile fname fname = filename to save correlation image in; if not present, and '-corr' is not present, correlation image is not saved.

-cnrfile fname fname = filename to save contrast-to-noise image in; if not present, will not be computed or saved; CNR is scaled by 100 if images are output as shorts and is written 'as-is' if output as floats (see '-flim').

-sigfile fname fname = filename to save standard deviation image in; the standard deviation is of what is left after the least squares removal of the -orts, -polrefs, and -ideal. **N.B.:** This is always output in the -flim format!

-fitfile fname Flag to save image files of the least squares fit coefficients of all the -orts and -polrefs series that are projected out of the data time series before the -ideal is fit. The actual filenames will be fname.01 fname.02 .... Their order is -orts, then -polrefs, and last -ideal. **N.B.:** These are always output in the -flim format!

-subort fname A new timeseries of images is written to disk, with names of the form 'fname.0001', etc. These images have the orts and polrefs (but not ideals) subtracted out. **N.B.:** These are always output in the -flim format.

-flim If present, write outputs in the libmri.a 'floating point' format, rather than scale and convert to integers. [The ftosh program can later convert to short integers.]

-clean If present, then output correlation and functional images won't have the +/- 10000 values forced into their corners for scaling purposes.

-clip If present, output correlations, etc., will be set to zero in regions of low intensity. 'Low intensity' is defined as follows: the maximum absolute intensity in the base image (see -regbase) is found; all pixels above 10% of this intensity are averaged; 10% of this average defines the -clip threshold. Pixels with intensity below this value in the base image will be set to zero in the -fimfile and -corfile.

-q If present, indicates 'quiet' operation.

-dfspace[:0] Indicates to use the 'dfspace' filter (*à la* program imreg) to register the images spatially before filtering. This uses the 'iterated differential spatial' method to align the images. The optional :0 indicates to skip the iteration of the method, and to use the simpler linear differential spatial alignment method.

-regbase fname Indicates to read the image in file 'fname' and use it for the base image to which other images are registered. If not present, then the first image in the time series that will be used in the correlation computations will be selected (e.g., skipping the -im1 images). This image is also used to define 'low intensity' if the -clip option is used.

53.4 Notes

1. Although I have spent a great deal of effort to make the alignment routines efficient, registration can easily double or triple the CPU time used by fim2. This is one reason for allowing multiple -ideal options, since the overhead of registration only need be performed once.

2. The MR imaging process is not as simple as just taking a picture with a camera. When the subject moves his/her head, not everything in the imaging process moves with it. Here are some things for which 'mere' spatial registration (-dfspace) will not compensate, especially in echo-planar imaging:

   Nonuniform B_z from magnet $\Longrightarrow$ Distortions caused by shimming errors don't move with head.

   Susceptibility changes to B_z are angle dependent $\Longrightarrow$ Rotation of head causes image distortions to change.

   B_1 inhomogeneities $\Longrightarrow$ Signal intensity at a given anatomical location will fluctuate with motion.

   Gradient field inhomogeneities $\Longrightarrow$ Yet more position dependent image distortions.

   Slice selection $\Longrightarrow$ $TR \approx T_1$ means that some tissue may have different relaxation histories as it moves in and out of the image slices.

   It is better to think of the MR image as a picture taken with a distorting pane of glass between the camera and the subject. If the subject moves, the pane of glass (the imager) doesn't, and so the distortions that were present in (say) the hippocampus are suddenly now at the parahippocampal gyrus. Simply moving the image back to the location it should have come from won't make the changed distortions go away.

3. There are at least two aspects to subject head motion in FMRI. First, if the motion is random, then the signal changes induced by movement will be a source of noise, and will tend to mask truly activated pixels (increase the false negative rate). Second, if the motion is coherent with the task/stimulus protocol, then the signal changes induced by movement will tend to look like activated pixels (increase the false positive rate).

4. The file used in -regbase should be of the same character as the images which will be registered to it. By 'same character' I mean acquired with the same MR parameters under the same circumstances. It is crucial to note that the first image in an EPI sequence often does *not* meet this requirement, since the longitudinal magnetization

has not previously been excited within a few T_1. It is also not likely to be possible to use the program imreg to register functional images with higher resolution anatomical images.

In some cases, it may be practical to use -regbase to register a series of fim2 runs (from separate image sequences taken in the same scanning session) to a common image. This will be possible if there is no significant out-of-plane motion between scanning runs. For example, if the subject is in the scanner in the normal flat-on-his-back position, and he tends to slide along the tube during the scanning session, and the images are acquired in the sagittal plane, then the motion may mostly be in-plane and this application of -regbase may be appropriate. *I have not tried to do this*, as of today.

53.5 Example

Example 1. Below is a C-shell script to run fim2 and then to3d to create a collection of AFNI datasets.

- The time series image files (64×64) are in four directories (`v1`, `v2`, `v3`, and `v4`), 16 slices each, 68 images in each time series, with filenames such as `v1/v107.0033`, which is the 33^{rd} image in time, in the 7^{th} slice, in run `v1`.

- The C-shell 'foreach' command is used to loop over the runs and over the slices within the runs. The shell variable `$run` will successively be set to '`v1`', '`v2`', '`v3`', and '`v4`':

  foreach run (v1 v2 v3 v4)

  Similarly, the shell variable `$slice` will successively be set to '`01`', '`02`', . . . , '`16`':

  foreach slice(01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16)

- There are a number of ideal waveform files (cosine functions, with various phase shifts) in files with names like cos9.25. These are invoked by the option

  `-ideal % cos*.* %`

- The mean and linear trend in each pixel time series are projected out using:

  `-polort 1`

- The data is not thresholded on the correlation coefficient:

  `-pcthresh 0.0`

- The functional intensities are scaled to integers using the factor 100,000:

  `-non -coef 100000`

- Functional correlations and intensities will be set to zero in faint parts of the image time series:

  `-clip`

- The images are registered spatially:

  dfspace

- The output functional images are stored in files with names like v1/07.FIM (for slice 7 of run `v1`); the correponding correlation coefficient image will be v1/07.COR:

  -fimfile $run/$slice.FIM -corfile $run/$slice.COR

- When all slices are done for a run, the .FIM and .COR files are put into an *AFNI* dataset using batch-mode to3d. The geometry parent dataset zzzz+orig was created earlier using the interactive mode of to3d.

- This script is executed from the directory that contains the `v1`, `v2`, `v3`, and `v4` data directories. The .FIM and .COR files are left in those directories.

The Actual Script!

```
#!/bin/csh
# set part of the prefix for the output AFNI bricks
set prefix = dfst
foreach run ( v1 v2 v3 v4 )
 echo ''----- starting run $run -----''
 foreach slice( 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 )
 fim2 -ideal % cos*.* % \
 -polort 1 \
 -pcthresh 0.0 \
 -non -coef 100000 \
 -clip \
 -dfspace \
 -fimfile ${run}/${slice}.FIM -corfile ${run}/${slice}.COR \
 ${run}/${run}${slice}.0*
 end
 set pref = ${run}:${prefix}
 to3d -fith -geomparent zzzz+orig \
 -dname $pref -dlabel $pref -prefix $pref \
 ${run}/*.FIM ${run}/*.COR
end
```

53.6 References

[1] P.A. Bandettini, A. Jesmanowicz, E.C. Wong, and J.S. Hyde. Processing strategies for time-course data sets in functional MRI of the human brain. *Magn. Reson. Med.* **30**, 161–173 (1993).
[2] R.W. Cox, A. Jesmanowicz, and J.S. Hyde. Real-time functional magnetic resonance imaging. *Magn. Reson. Med.* **33**, 230–236 (1995).

54 Program float_scan

54.1 Purpose

Scans the input file of IEEE floating point numbers for illegal values: infinities and not-a-number (NaN) values.

54.2 Usage

float_scan [options] input_filename

54.3 Options

-fix = Writes a copy of the input file to stdout (which should be redirected using '>'), replacing illegal values with 0. If this option is not used, the program just prints out a report.

-v = Verbose mode: print out index of each illegal value.

-skip n = Skip the first n floating point locations (i.e., the first 4*n bytes) in the file

54.4 Notes

N.B.: This program does NOT work on compressed files, nor does it work on byte-swapped files (e.g., files transferred between Sun/SGI/HP and Intel platforms)!

The program 'exit status' is 1 if any illegal values were found in the input file. If no errors were found, then the exit status is 0. You can check the exit status by using the shell variable $status.

54.5 Example

A C-shell example:

```
float_scan fff
if ( $status == 1 ) then
        float_scan -fix fff > Elvis.Aaron.Presley
        rm -f fff
        mv Elvis.Aaron.Presley fff
endif
```

55 Program from3d

55.1 Purpose

Program from3d is used to extract 2D data files from a 3D or a 3D+time data set. As such, it may be considered as an inverse to program to3d, which combines 2D data files into a 3D data set. The slices are extracted only along the z-direction of the dataset, which can be determined by using program 3dinfo.

55.2 Usage

The command line format for program from3d is as follows:

from3d [-v] [-nsize][-zfirst num] [-zlast num] [-tfirst num] [-tlast num] -input fname -prefix rname

55.3 Options

-v Print out "verbose" information. At present, this just prints the names of the 2D data files that are being created.

-nsize It may be desirable, particularly for Fast Fourier Transform applications, that the dimensions of the 2D dataset be powers of 2. The -nsize option adjusts the size of the 2D data file to be NxN, by padding with zeros, where N is a power of 2. The value of N is chosen to be the smallest power of 2 greater than or equal to the dimensions of the original dataset.

-zfirst num This command sets 'num' = number of first z slice to be extracted. The default value is 1.

-zlast num Set 'num' = number of last z slice to be extracted. The default value is the largest z slice present in the 3D dataset.

-tfirst num This command sets 'num' = number of first time slice to be extracted. The default value is 1.

-tlast num Set 'num' = number of last time slice to be extracted. The default value is the last time slice present in the 3D+time dataset.

-input fname Read 3d data set from file 'fname'.

-prefix rname Write 2d data sets using prefix 'rname'.

55.4 Examples

Example 1: Suppose that file fred.anat+orig (.HEAD and .BRIK) contains a 256×256×124 voxel AFNI 3D dataset. The user wishes to extract the 61st through 67th z-slices. The command line to do this is:

```
from3d -v -zfirst 61 -zlast 67 -input fred.anat+orig -prefix fred.anat
```

The output on the screen (obtained by using the -v option) is:

```
3d Dataset File: fred.anat+orig
nx = 256 ny = 256 nz = 124 nv = 1
Data type: short
Writing 2d data file: fred.anat.061
Writing 2d data file: fred.anat.062
Writing 2d data file: fred.anat.063
    etc.
Writing 2d data file: fred.anat.067
Created 7 2d data files.
```

Program from3d has created 7 2D data files. Since the output option -prefix fred.anat was used, the output file names have the form "fred.anat.ijk", where "ijk" = number of the z-slice.

Example 2: Suppose that file fred.ts+orig (.HEAD and .BRIK) contains a 3D+time *AFNI* dataset (64×64×16 voxels for each of 68 time steps).

In order to extract the 8th through 10th z-slices, for time steps 40 through 45, the command line is:

```
from3d -v -zfirst 8 -zlast 10 -tfirst 40 -tlast 45 \
-input fred.ts+orig -prefix fred.ts.2d
```

The output on the screen (obtained by using the -v option) is:

3d Dataset File: fred.ts+orig
nx = 64 ny = 64 nz = 16 nv = 68
Data type: short
Writing 2d data file: fred.ts.2d08.0040
Writing 2d data file: fred.ts.2d09.0040
Writing 2d data file: fred.ts.2d10.0040
Writing 2d data file: fred.ts.2d08.0041
Writing 2d data file: fred.ts.2d09.0041
etc.
Writing 2d data file: fred.ts.2d10.0045
Created 18 2d data files.

As a result, program from3d has created 18 2D data files. Since the output option -prefix fred.ts.2d was used, the output file names have the form "fred.ts.2d*ij*.*klmn*", where "*ij*" = number of z-slice, and "*klmn*" = number of time step.

56 Program ftosh

56.1 Purpose

Convert float images to shorts.

56.2 Usage

ftosh [options] image_files ...
where the image_files are in the same format to3d accepts

56.3 Options

-prefix pname	pname = prefix for output files	Default pname = 'sh'
-suffix sname	sname = suffix for output files	Default sname = nothing at all
-start si	initial index number	Default si = 1
-step ss	index number increment	Default ss = 1

The output files will be named in the format:
'pname.index.sname'
where 'pname' and 'sname' are strings given by the first 2 options, and 'index' is a number, given by 'si+(i-1)*ss' for the i-th output file, for i=1,2,...

-nsize	Enforce the 'normal size' option, to make the output images 64x64, 128x128, or 256x256.
-scale sval	Scale factor for output (see below).
-base bval	Baseline value for output (see below).
-top tval	Used to set default value for sval (see below).

- 'sval' and 'bval' are numeric values; if sval is given, then the output images are formed by scaling the inputs by the formula:

  output=sval*(input-bval).

- Default sval is determined by finding:

  V = largest abs(input-bval) in all the input images, and then
  sval = tval / V.

- Default tval is 32000; note that tval is only used if sval is not given on the command line.

- Default bval is 0.

57 Program imand

57.1 Purpose

Produce logical "and" of a sequence of input images.

57.2 Usage

imand [-thresh #] input_images ... output_image

Only pixels nonzero in all input images (and above the threshold, if given) will be output.

58 Program imaver

58.1 Purpose

Compute the mean and standard deviation of a group of 2D images, pixel by pixel.

58.2 Usage

imaver out_ave out_sig input_images ...

58.3 Notes

- Use **-** to skip output of out_ave and/or out_sig.

- This program writes the output images in 'short int' format if inputs are short ints, otherwise output images are floating point.

58.4 Example

Example 1. Suppose that the group of files fred.0001, fred.0002, ..., fred.0068 is a collection of 2D images. To find the mean and standard deviation, pixel by pixel, for these images, use the command line:

imaver fred.ave fred.sig fred.*

Files fred.ave and fred.sig will be produced, containing the pixel by pixel means and standard deviations, respectively.

59 Program imcalc

59.1 Purpose

Do arithmetic on 2D images, pixel-by-pixel.

59.2 Usage

imcalc options

59.3 Options

-datum type = Coerce the output data to be stored as the given type, which may be byte, short, or float. [default = datum of first input image]

-a dname = Read image 'dname' and call the voxel values 'a' in the expression. 'a' may be any letter from 'a' to 'z'.

Note: If some letter name is used in the expression, but not present in one of the image options here, then that variable is set to 0.

-expr "expression" Apply the expression within quotes to the input images, one voxel at a time, to produce the output image.

("sqrt(a*b)" to compute the geometric mean, for example)

-output name = Use 'name' for the output image filename.

[default='imcalc.out']

59.4 Problems

- Complex-valued images cannot be processed (byte, short, and float are OK).
- Evaluation of expressions is not very efficient.

59.5 Expressions

Arithmetic expressions are allowed, using + - * / ** and parentheses. As noted above, datasets are referred to by single letter variable names. At this time, C relational, boolean, and conditional expressions are NOT implemented. Built in functions include:

sin ,	cos ,	tan ,	asin ,	acos ,	atan ,	atan2,	
sinh ,	cosh ,	tanh ,	asinh ,	acosh ,	atanh ,	exp ,	
log ,	log10,	abs ,	int ,	sqrt ,	max ,	min ,	
J0 ,	J1 ,	Y0 ,	Y1 ,	erf ,	erfc ,	qginv,	qg ,
rect ,	step ,	astep,	bool ,	and ,	or ,	mofn ,	

where
qg(x) = reversed cdf of a standard normal distribution
qginv(x) = inverse function to qg,
min, max, atan2 each take 2 arguments,
J0, J1, Y0, Y1 are Bessel functions (see Watson),
erf, erfc are the error and complementary error functions.

The following functions are designed to help implement logical functions, such as masking of 3D volumes against some criterion:

step(x)	=	{1 if x>0 ,	0 if x<=0},
astep(x,y)	=	{1 if abs(x) > y ,	0 otherwise} = step(abs(x)-y)
rect(x)	=	{1 if abs(x)<=0.5,	0 if abs(x)>0.5},
bool(x)	=	{1 if x != 0.0 ,	0 if x == 0.0},
and(a,b,...,c)	=	{1 if all arguments are nonzero,	0 if any are zero}
or(a,b,...,c)	=	{1 if any arguments are nonzero,	0 if all are zero}
mofn(m,a,...,c)	=	{1 if at least 'm' arguments are nonzero,	0 otherwise}

These last 3 functions take a variable number of arguments.

All computations are carried out in double precision before being truncated to the final output 'datum'.

Note that the quotes around the expression are needed so the shell doesn't try to expand * characters, or interpret parentheses.

(Try the 'ccalc' program to see how the expression evaluator works.)

60 Program imdump

60.1 Purpose

This program prints out the list of nonzero pixels in an image, with each one's x-index, y-index, and value. One application: importing data from functional images into spreadsheet programs for further analysis.

60.2 Usage

imdump input_image

60.3 Notes

- This prints out only the nonzero pixels in an image.
- The results are sent to stdout; this can be redirected (with >) to save to a file.
- Format: x-index y-index value, with one pixel per line.

60.4 Example

Example 1. Suppose that file fred08.0037 contains a 64×64 2D image.

imdump fred08.0037 > frednz.out

The output has been redirected to file frednz.out, which now contains:
1 0 8
2 0 8
3 0 13
4 0 4
⋮
etc.
⋮
60 63 25
61 63 13
62 63 9
63 63 9

61 Program immask

61.1 Purpose

Masks the input_image and produces the output_image.

61.2 Usage

immask [**-thresh** #] [**-mask mask_image**] [**-pos**] **input_image output_image**

61.3 Options

-thresh # means all pixels with absolute value below # in input_image will be set to zero in the output_image.

-mask mask_image means that only locations that are nonzero in the mask_image will be nonzero in the output_image.

-pos means only positive pixels from input_image will be used.

At least one of -thresh, -mask, -pos must be used; more than one is OK.

62 Program imreg

62.1 Purpose

This program takes as input a sequence of (2D) images and a base image, and produces as output a sequence of images registered to the base image. Note that the images must be acquired in the same modality — there is no allowance for having different types of tissue-to-grayscale mapping functions between the base image and the sequential images. Besides the output images, imreg also reports the Δx, Δy, and $\Delta\theta$ values needed for each registered image (in pixels, pixels, and degrees, respectively).

By the way, imreg (and imrotate) use bicubic interpolation when resampling the images to the new grid system. Imreg uses the same routines as fim2 for registration, which are designed only to deal with small displacements (at most 3-4 pixels).

62.2 Usage

imreg [options] base_image image_sequence ...

- Registers each 2D image in 'image_sequence' to 'base_image'.
- If 'base_image' = '+AVER', will compute the base image as the average of the images in 'image_sequence'.
- If 'base_image' = '+count', will use the count-th image in the sequence as the base image. Here, count is 1,2,3, ...

62.3 Output Options

-nowrite Don't write outputs, just print progress reports.

-prefix pname	Default pname	=	'reg.'
-suffix sname	Default sname	=	nothing at all
-start si	Default si	=	1
-step ss	Default ss	=	1

The output files will be named in the format

'pname.index.sname'

where 'pname' and 'sname' are strings given by the first 2 options. 'index' is a number, given by 'si+(i-1)*ss' for the i-th output file, for i=1,2,...

-flim Write output in mrilib floating point format (which can be converted to shorts using program ftosh).

- Default is to write images in format of first input file in the image_sequence.

-quiet Don't write progress report messages.

-debug Write lots of debugging output!

-dprefix dname Write files 'dname'.dx, 'dname'.dy, 'dname'.phi for use in time series analysis.

62.4 Alignment Algorithms

-bilinear Uses bilinear interpolation during the iterative adjustment procedure, rather than the default bicubic interpolation. NOT RECOMMENDED!

-modes c f r Uses interpolation modes 'c', 'f', and 'r' during the coarse, fine, and registration phases of the algorithm, respectively. The modes can be selected from 'bilinear', 'bicubic', and 'Fourier'. The default is '-modes bicubic bicubic bicubic'.

-mlcF Equivalent to '-modes bilinear bicubic Fourier'.

-wtim filename Uses the image in 'filename' as a weighting factor for each voxel (the larger the value the more importance is given to that voxel).

-dfspace[:0] Uses the 'iterated differential spatial' method to align the images. The optional :0 indicates to skip the iteration of the method, and to use the simpler linear differential spatial alignment method. ACCURACY: displacments of at most a few pixels.
Note: This is the default method (without the :0).

The next two options are used to play with the -dfspace algorithm, which has a 'coarse' fit phase and a 'fine' fit phase:

-fine blur dxy dphi Set the 3 'fine' fit parameters:

blur	=	FWHM of image blur prior to registration, in pixels [must be > 0];
dxy	=	convergence tolerance for translations, in pixels;
dphi	=	convergence tolerance for rotations, in degrees.

-nofine Turn off the 'fine' fit algorithm. By default, the algorithm is on,
with parameters 1.0, 0.07, 0.21.

62.5 How Program "imreg" Works

Registration is accomplished by minimizing the error functional

$$E(I,J) = \sum_{x,y} [I(x,y) - J(T(u,v,h)(x,y))]^2$$

where $J(x,y)$ is the "base" image, $I(x,y)$ is the image to be registered to it, $T(u,v,h)$ is the transformation with shift parameters (u,v) and rotation angle h. In practice, $J(T(x,y))$ is expanded in a Taylor series in (u,v,h) to be $J0(x,y)+u*Ju(x,y)+v*Jv(x,y)+h*Jh(x,y)$. The minimization is then a linear least squares problem. The (u,v,h) that minimizes E is found. At this point, $I(x,y)$ is transformed with $T(-u,-v,-h)$ to bring it closer to $J(x,y)$ (bicubic interpolation is used for resampling I). Then the minimization is repeated – that is, simple gradient descent is used to minimize E.

Actually, this procedure is performed with a $J(x,y)$ that has been smoothed with a Gaussian filter with FWHM=4 pixels. In this way, the effects of pixels a little farther away than nearest neighbors are included in the derivatives, and displacements of up to 2-3 pixels can be detected.

When the FWHM=4 pixel smoothing iteration converges, it is then repeated with $J(x,y)$ smoothed with FWHM=1.0 pixels. Typically, 2-4 iterations are required to converge in the first step, and 1-2 in the second, if the displacements are larger than 1/2 pixel.

The programs imreg and fim2 both use the routines in "mri_align.c", if you wish to see the details. The Gaussian smoothing and the differentiation of $J(x,y)$ are both done with FFTs.

One reference to a similar method is:

> Michael Irani and Shmuel Peleg, Improving Resolution by Image Registration. CVGIP (Computer Vision, Graphics, and Image Processing), Vol 53, pp. 231-239 (1991).

There are references in this paper to earlier works.

I strongly recommend that you use imreg to register image time series and look at the results to make sure that the program is doing a good job for you. Note that this method will only work for registration of similar images displaced by only a few pixels. It cannot be used to register PET and MRI, for example, or to register FSE and EPI.

Some further notes on motion and FMRI in general can be found at

http://www.biophysics.mcw.edu/BRI-people/rwcox/regnotes.ps

– this is a 780 K PostScript file, 17 pages long.

62.6 Examples

Example 1. Suppose that the user has a time series of 2D images, for slice z = 8, in files fred08.0001, fred08.0002, ..., fred08.0068. The command to register these images to the *average* of the 68 images is given by:

imreg -prefix fredreg -dprefix dname +aver fred*

to which the computer responds with:

– will set base image to AVER
– computing AVER image now
– beginning alignment
– mri_align_dfspace..............
– registering..............
– image fredreg.0001: dx = 0.057 dy = 0.104 phi = -0.171
– image fredreg.0002: dx = 0.104 dy = 0.010 phi = 0.079
– image fredreg.0003: dx = 0.054 dy = 0.010 phi = 0.131
– image fredreg.0004: dx = 0.017 dy = 0.028 phi = 0.068
⋮
etc.
⋮
– image fredreg.0065: dx = 0.051 dy = -0.013 phi = -0.072
– image fredreg.0066: dx = -0.000 dy = -0.007 phi = -0.087
– image fredreg.0067: dx = -0.001 dy = 0.029 phi = -0.141
– image fredreg.0068: dx = 0.031 dy = 0.023 phi = -0.183
– MAX: dx = 0.104 dy = 0.104 phi = -0.183

At this point, the registered 2D images are contained in files fredreg.0001 through fredreg.0068. Note that for each of the registered images, the computer printed out the "adjustment": dx and dy, in pixels, and the rotation angle phi, in degrees. If the user wishes to study the time series of adjustments, the -dprefix dname command can be used as shown above. For example, file dname.dx contains the dx values:

0.057
0.104
0.054
0.017
⋮
etc.
⋮
0.051
-0.000
-0.001
0.031

Similarly for files dname.dy and dname.phi.

63 Program imrotate

63.1 Purpose

This program rotates and shifts one image with given Δx, Δy, and $\Delta\theta$ values. *AFNI* requires that the functional and anatomical datasets be acquired on parallel grids. This means that the acquisition planes must be parallel or perpendicular — oblique (to the anatomy) functional data is not supported at this time. An application of imrotate might be to rotate a whole set of images before input to to3d, as a way around this limitation of *AFNI*.

63.2 Usage

imrotate [-linear|-Fourier] dx dy phi input_image output_image

63.3 Options

The above command line shifts and rotates an image:

dx	pixels rightwards (not necessarily an integer)
dy	pixels downwards
phi	degrees clockwise

-linear means to use bilinear interpolation (default is bicubic)

-Fourier means to use Fourier interpolation

Values outside the input_image are taken to be zero.

63.4 Example

Example 1. Suppose that the functional data is gathered in a set of oblique coronal slices, and the anatomical data is in a set of sagittal slices. By rotating the sagittal images to be aligned with the functional image axes, the whole to3d+*AFNI* process can be carried out.

I have not tried this, since at **MCW**, we do not gather oblique functional data. Here is a possible set of C-shell commands to do the desired work:

```
foreach slice ( `count 1 124` )
    imrotate 0 0 15.0 sag_in.${slice} sag_out.${slice}
end
to3d -spgr sag_out.*
```

This assumes that the functional slices are 15° oblique to the axes defined by the MR gradient coil set. The necessary steps would be considerably more complicated if the functional and anatomical data were "doubly oblique" to one another.

64 Program imstack

64.1 Purpose

Stack up a set of 2D images into one big file (a la MGH).

64.2 Usage

imstack [options] image_filenames ...

64.3 Options

-datum type Converts the output data file to be 'type', which is either 'short' or 'float'. The default type is the type of the first image.

-prefix name Names the output files to be 'name'.b'type' and 'name'.hdr.
The default name is 'obi-wan-kenobi'.

65 Program imstat

65.1 Purpose

Write out some statistics about (one or more) 2D image files.

65.2 Usage

imstat [-nolabel] [-pixstat prefix] [-quiet] image_file ...

65.3 Options

-nolabel Don't write labels on each file's summary line

-quiet Don't print statistics for each file

-pixstat prefix If more than one image file is given, then 'prefix.mean', 'prefix.sdev', and 'prefix.cvar' will be written as the pixel-wise statistics images (for the mean, standard deviation, and coefficient of variation, respectively) of the whole collection. These images will be in the 'flim' floating point format. [This option only works on 2D images!]

65.4 Examples

Example 1. Suppose that files fred08.0001, fred08.0002, fred08.0003,..., fred08.0068 contain 2D images. Assuming that these are the only files with prefix "fred08." in the current directory, the command line to calculate the pixel-wise statistics across the whole collection is given by:

imstat -pixstat fredstat fred08.*

And the computer responds with:

file = fred08.0001 nx = 64 ny = 64 data type = short
min = 0 next min= 1 max= 1808 next max= 1733
mean= 207.1 std.dev.= 331.2 number of zero pixels = 68

file = fred08.0002 nx = 64 ny = 64 data type = short
min = 0 next min= 1 max= 1489 next max= 1421
mean= 190.9 std.dev.= 298.5 number of zero pixels = 63

file = fred08.0003 nx = 64 ny = 64 data type = short
min = 0 next min= 1 max= 1346 next max= 1328
mean= 189.8 std.dev.= 296.9 number of zero pixels = 68

⋮
etc.
⋮

file = fred08.0068 nx = 64 ny = 64 data type = short
min = 0 next min= 1 max= 1416 next max= 1348
mean= 187.5 std.dev.= 294 number of zero pixels = 73

– Wrote mean image to fredstat.mean
– Wrote standard deviation image to fredstat.sdev
– Wrote coefficient of variation image to fredstat.cvar

So, files fredstat.mean, fredstat.sdev, and fredstat.cvar contain floating point images, where each individual pixel contains the mean, standard deviation, and coefficient of variation (respectively) for that pixel across the collection of fred08.* images.

66 Program imupsam

66.1 Purpose

Upsample the input 2D image.

66.2 Usage

imupsam n input_image output_image

66.3 Notes

- This upsamples the input 2D image by a factor of n and writes the result into output_image.
- n must be an integer in the range 2..30.
- 7th order polynomial interpolation is used in each direction.
- Inputs can be complex, float, short, PGM, or PPM.

67 Program mritopgm

67.1 Purpose

Converts an image to raw pgm format. Results go to stdout and should be redirected. This program is only useful if the netpbm package is installed. (The shareware program xv can also read pgm images).

67.2 Usage

The command line format for program mritopgm is as follows:

mritopgm input_image

67.3 Example

mritopgm fred.001 | ppmtogif > fred.001.gif

68 Program nsize

68.1 Purpose

Expand a 2D image file up to a 'normal' (i.e., power-of-2) size. This is useful when the 2D image is to be used as input to an FFT (Fast Fourier Transform) program.

68.2 Usage

nsize image_in image_out

68.3 Notes

- The program zero pads 'image_in' to size NxN, where N=64, 128, 256, 512, or 1024, whichever is the closest size larger than 'image_in'.
- The program works only for byte and short images.

69 Program p2t

69.1 Purpose

Calculation of tail probabilities for various probability distributions.

69.2 Note

This program has been superseded by program `cdf`.

69.3 Usage

Usage #1: p2t p dof

p	=	double sided tail probability for t-distribution
dof	=	number of degrees of freedom to use
OUTPUT	=	t value that matches the input p

Usage #2: p2t p N L M

p	=	double sided tail probability of beta distribution
N	=	number of measured data points
L	=	number of nuisance parameters (orts)
M	=	number of fit parameters
OUTPUT	=	threshold for correlation coefficient

Usage #3: p2t p

p	=	one sided tail probability of Gaussian distribution
OUTPUT	=	z value for which P(x>z) = p

Usage #4: p2t p dof N

p	=	double sided tail probability for distribution of the mean of N iid zero-mean t-variables
dof	=	number of degrees of freedom of each t-variable
N	=	number of t variables averaged
OUTPUT	=	threshold for the t average statistic

N.B.: The method used for this calculation is the Cornish-Fisher expansion in N, and is only an approximation. This also requires dof > 6, and the results will be less accurate as dof approaches 6 from above!

69.4 Examples

Example 1.

To calculate the t value which corresponds to a double sided tail probability of 0.05, when there are 10 degrees of freedom, the command is:

p2t .05 10

and the computer response is:

p=0.05 dof=10 t=2.22814

Thus, t=2.22814 yields a double sided tail probability of 0.05, when there are 10 df.

Example 2.

To find the z-value for which the one sided tail probability of the Gaussian distribution is 0.05, use the following command line:

p2t .05

to which the computer responds with:

p=0.05 z=1.64485

Therefore, under the Gaussian distribution, P(x > 1.64485) = 0.05.

70 Program sfim

70.1 Purpose

Calculate Stepwise Functional IMages.

70.2 Usage

sfim [options] image_files ...

where image_files are in the same format AFNI accepts.

70.3 Options

-sfint iname

'iname' is the name of a file which has the interval definitions; an example is:

3*# 5*rest 4*A 5*rest 4*B 5*rest 4*A 5*rest which says:

- ignore the 1st 3 images
- take the next 5 as being in task state 'rest'
- take the next 4 as being in task state 'A'

and so on; task names that start with a nonalphabetic character are like the '#' above and mean 'ignore'.

*** The default 'iname' is 'sfint'.

-base bname

'bname' is the task state name to use as the baseline; other task states will have the mean baseline state subtracted; if there are no task states from 'iname' that match 'bname', this subtraction will not occur.

*** The default 'bname' is 'rest'.

-localbase

if this option is present, then each non-base task state interval has the mean of the two nearest base intervals subtracted instead of the grand mean of all the base task intervals.

-prefix pname

'pname' is the prefix for output image filenames for all states: the i'th interval with task state name 'fred' will be writen to file 'pname.fred.i'.

*** The default 'pname' is 'sfim'.

Output files are the base-mean-removed averages for each non-base task interval, and simply the mean for each base task interval. Output images are in the 'flim' (floating pt. image) format, and may be converted to 16 bit shorts using the program 'ftosh'.

71 Program sqwave

71.1 Purpose

This is a simple program that generates a time series file consisting of 'on' periods (10's), 'off' periods (0's), and 'unused' periods. The time series is suitable for input as an ideal for 3dfim or afni.

71.2 Usage

sqwave [**-on** #] [**-off** #] [**-length** #] [**-cycles** #] [**-init** #] [**-onkill** #] [**-offkill** #] [**-initkill** #] [**-name name**]

71.3 Options

-on # Length of the 'on' period (# of 10's per cycle).

-off # Length of the 'off' period (# of 0's per cycle).

-length # Total length of the time series.

-cycles # An alternative way of specifying the length of the time series. If the -length command is *not* used, then

$$\text{length } \# = \text{cycles } \# \times (\text{on } \# + \text{off } \#) + \text{init } \#.$$

-init # Length of time before the first 'on' period.

-onkill # Ignore first # of points in the 'on' period.

-offkill # Ignore first # of points in the 'off' period.

-initkill # Ignore first # of points in the 'initialization' period.

-name name Name of output time series file.

71.4 Examples

Example 1. To create a square wave time series of length 100, with 'on' period of length 10, and 'off' period of length 10, the command line would be:

```
sqwave -on 10 -off 10 -length 100 -name sqwave1.1D
```

The output time series would then be stored in file sqwave1.1D (the actual output is stored one number per line):

10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0 10 10 10 10 10 10 10 10 10 10 0 0 0 0 0 0 0 0 0 0

Example 2. We again create a square wave time series of length 100, with 'on' period of length 10, and 'off' period of length 10. However, the 'on' cycle does not begin until after 17 time intervals. Also, for correlation calculations, we want to ignore the first 3 points of each 'on' period, and we want to ignore the first point of each 'off' period. The command line to accomplish this is:

```
sqwave -on 10 -off 10 -length 100 -init 17 -onkill 3 -offkill 1 \
-name sqwave2.1D
```

The output time series stored in file sqwave2.1D would be:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 99999 99999 99999 10 10 10 10 10 10 10 99999 0 0 0 0 0 0 0 0 0 99999 99999 99999 10 10 10 10 10 10 10 99999 0 0 0 0 0 0 0 0 0 99999 99999 99999 10 10 10 10 10 10 10 99999 0 0 0 0 0 0 0 0 0 99999 99999 99999 10 10 10 10 10 10 10 99999 0 0 0 0 0 0 0 0 0 99999 99999 99999

Note that the '99999' tells program 3dfim (or program afni) to ignore this point in the correlation calculations (see user documentation for these programs).

72 Program tfim

72.1 Purpose

To perform t-tests on sets of functional images.

72.2 Usage

Usage 1: **tfim [options] -set1 image_files ... -set2 image_files ...**
Usage 2: **tfim [options] -base1 bval -set2 image_files ...**

In usage 1, the collection of images files after '-set1' and the collection after '-set2' are averaged and differenced, and the difference is tested for significance with a 2 sample Student t-test.

In usage 2, the collection of image files after '-set2' is averaged and then has the constant numerical value 'bval' subtracted, and the difference is tested for significance with a 1 sample Student t-test.

> N.B.: The input images can be in the usual 'short' or 'byte' formats, or in the floating point 'flim' format.
>
> N.B.: If in either set of images, a given pixel has zero variance (i.e., is constant), then the t-test is not performed. In that pixel, the .tspm file will be zero.

72.3 Options

-prefix pname:
'pname' is used as the prefix for the output filenames. The output image files are:
pname.diff = average of set2 minus average of set1 (or minus 'bval')
pname.tspm = t-statistic of difference
Output images are in the 'flim' (floating pt. image) format, and may be converted to 16 bit shorts using the program ftosh.
*** The default 'pname' is 'tfim', if -prefix isn't used.

-pthresh pval:
'pval' is a numeric value between 0 and 1, giving the significance level (per voxel) to threshold the output with; voxels with (2-sided) t-statistic less significant than 'pval' will have their diff output zeroed.

*** The default is no threshold, if -pthresh isn't used.

-eqcorr dval:
If present, this option means to write out the file

$$\begin{aligned} \text{pname.corr} &= \text{equivalent correlation statistic} \\ &= t/\sqrt{(dof + t\hat{\ }2)} \end{aligned}$$

The number 'dval' is the value to use for 'dof' if dval is positive. This would typically be the total number of data images used in forming the image sets, if the image sets are from sfim or fim. If dval is zero, then dof is computed from the number of images in -set1 and -set2; if these are averages from program sfim, then dof will be smallish, which in turn means that significant corr values will be higher than you may be used to from using program fim.
*** The default is not to write, if -eqcorr isn't used.

-paired:
If present, this means that -set1 and -set2 should be compared using a paired sample t-test. This option is illegal with the -base1 option. The number of samples in the two sets of images must be equal. [This test is implemented by subtracting -set1 images from the -set2 images, then testing as in '-base1 0'.]
*** The default is to do an unpaired test, if -paired isn't used. In that case, -set1 and -set2 don't need to have the same number of images.

73 Program to3d (batch mode)

73.1 Purpose

Program to3d converts 2D image files into 3D datasets for use with AFNI.

73.2 Usage

The command line format for program to3d is as follows:

to3d [options] image_files ...

73.3 Options

The available options are

-help show this message

-type declare images to contain data of a given type, where 'type' is chosen from the following options:

ANATOMICAL TYPES		FUNCTIONAL TYPES	
spgr	Spoiled GRASS	fim	Intensity
fse	Fast Spin Echo	fith	Inten+Thr
epan	Echo Planar	fico	Inten+Cor
anat	MRI Anatomy	fitt	Inten+Ttest
ct	CT Scan	fift	Inten+Ftest
spct	SPECT Anatomy	fizt	Inten+Ztest
pet	PET Anatomy	fict	Inten+ChiSq
mra	MR Angiography	fibt	Inten+Beta
bmap	B-field Map	fibn	Inten+Binom
diff	Diffusion Map	figt	Inten+Gamma
omri	Other MRI	fipt	Inten+Poisson
abuc	Anat Bucket	fbuc	Func Bucket

Note: for paired (+) types above, images are fim first, then followed by the threshold (etc.) image files.

-statpar value value ... value This option is used to supply the auxiliary statistical parameters needed for certain dataset types (e.g., 'fico' and 'fitt'). For example, a correlation coefficient computed using program fim2 from 64 images, with 1 ideal, and with 2 orts could be specified with

-statpar 64 1 2

-prefix name will write 3D dataset using prefix 'name'

-session name will write 3D dataset into session directory 'name'

-geomparent fname will read geometry data from dataset file 'fname'. Note that geometry data does NOT include time-dependence

-anatparent fname will take anatomy parent from dataset file 'fname'

-nosave will suppress autosave of 3D dataset, which normally occurs when the command line options supply all needed data correctly

-view type Will set the dataset's viewing coordinates to 'type', which must be one of these strings: orig acpc tlrc

73.4 Time Dependent Datasets

-time:zt nz nt TR tpattern OR -time:tz nt nz TR tpattern
These options are used to specify a time dependent dataset.

-time:zt is used when the slices are input in the order z-axis first, then t-axis.

-time:tz is used when the slices are input in the order t-axis first, then z-axis.

nz = number of points in the z-direction

nt = number of points in the t-direction (thus exactly nt * nz slices must be read in)

TR = repetition interval between acquisitions of the same slice, in milliseconds (or other units, as given below)

tpattern = Code word that identifies how the slices (z-direction) were gathered in time. The values that can be used:

alt+z	=	altplus	=	alternating in the plus direction
alt-z	=	altminus	=	alternating in the minus direction
seq+z	=	seqplus	=	sequential in the plus direction
seq-z	=	seqminus	=	sequential in the minus direction
zero	=	simult	=	simultaneous acquisition
		@filename	=	read temporal offsets from 'filename'

For example, if nz = 5 and TR = 1000, then the inter-slice time is taken to be dt = TR/nz = 200. In this case, the slices are offset in time by the following amounts:

	SLICE NUMBER					
tpattern	0	1	2	3	4	Comment
altplus	0	600	200	800	400	Alternating in the +z direction
altminus	400	800	200	600	0	Alternating in the -z direction
seqplus	0	200	400	600	800	Sequential in the +z direction
seqminus	800	600	400	200	0	Sequential in the -z direction
simult	0	0	0	0	0	All slices acquired at once

If @filename is used for tpattern, then nz ASCII-formatted numbers are read from the file. These are used to indicate the time offsets (in ms) for each slice. For example, if 'filename' contains

0 600 200 800 400

then this is equivalent to 'altplus' in the above example.

73.5 Notes

- Time-dependent functional datasets are not yet supported by `to3d` or any other MCW *AFNI* package software. For many users, the proper dataset type for these datasets is '-epan'.
- Time-dependent datasets with more than one value per time point (e.g., 'fith', 'fico', 'fitt', 'fift') are also not allowed by `to3d`.
- If you use program `abut` to fill in gaps in the data and/or to subdivide the data slices, you will have to use the @filename form for tpattern, unless 'simult' or 'zero' is acceptable.
- At this time, the value of 'tpattern' is not actually used in any MCW *AFNI* program. The values are stored in the dataset .HEAD files, and will be used in the future.
- The values set on the command line can't be altered interactively.
- The units of TR can be specified by the command line options below:

-t=ms or -t=msec	–>	milliseconds (the default)
-t=s or -t=sec	–>	seconds
-t=Hz or -t=Hertz	–>	Hertz (for chemical shift images?)

  Alternatively, the units symbol ('ms', 'msec', 's', 'sec', 'Hz', or 'Hertz') may be attached to TR in the '-time:' option, as in '-time:zt 16 64 4.0sec alt+z'

73.6 Command Line Geometry Specification

-xFOV <dimen1><direc1>-<dimen2><direc2>
or

-xSLAB <dimen1><direc1>-<direc2>
(Similar -yFOV, -ySLAB, -zFOV and -zSLAB option are also present.)

These options specify the size and orientation of the x-axis extent of the dataset. <dimen#> means a dimension (in mm); <direc> is an anatomical direction code, chosen from

A (Anterior)	P (Posterior)	L (Left)
I (Inferior)	S (Superior)	R (Right)

Thus, 20A-30P means that the x-axis of the input images runs from 20 mm Anterior to 30 mm Posterior. For convenience, 20A-20P can be abbreviated as 20A-P.

-xFOV is used to mean that the distances are from edge-to-edge of the outermost voxels in the x-direction.

-xSLAB is used to mean that the distances are from center-to-center of the outermost voxels in the x-direction.

Under most circumstance, -xFOV , -yFOV , and -zSLAB would be the correct combination of geometry specifiers to use. For example, a common type of run at MCW would be entered as

-xFOV 120I-S -yFOV 120A-P -zSLAB 60S-50I

73.7 Input Image Formats

Image files may be single images of unsigned bytes or signed shorts (64x64, 128x128, 256x256, 512x512, or 1024x1024) or may be grouped images (that is, 3- or 4-dimensional blocks of data). In the grouped case, the string for the command line file spec is like

3D:hglobal:himage:nx:ny:nz:fname	16 bit input
3Ds:hglobal:himage:nx:ny:nz:fname	16 bit input, swapped bytes
3Db:hglobal:himage:nx:ny:nz:fname	8 bit input
3Di:hglobal:himage:nx:ny:nz:fname	32 bit input
3Df:hglobal:himage:nx:ny:nz:fname	floating point input
3Dc:hglobal:himage:nx:ny:nz:fname	complex input

where

'3D:' or '3Ds:'	signals this is a 3D input file of signed shorts
'3Db:'	signals this is a 3D input file of unsigned bytes
'3Di:'	signals this is a 3D input file of signed ints
'3Df:'	signals this is a 3D input file of floats
'3Dc:'	signals this is a 3D input file of complex numbers (real and imaginary pairs of floats)

hglobal	= number of bytes to skip at start of whole file
himage	= number of bytes to skip at start of each image
nx	= x dimension of each image
ny	= y dimension of each image
nz	= number of images
fname	= actual filename on disk to read

- The ':' separators are required. The k-th image starts at BYTE offset

  hglobal + (k+1)*himage + vs*k*nx*ny

  in file 'fname' for k=0,1,...,nz-1.

- Here, vs = voxel length = 1 for bytes, 2 for shorts, 4 for ints and floats, and 8 for complex numbers.

- As a special case, hglobal = -1 means read data starting at offset

  len - nz*(vs*nx*ny+himage),

  where len = file size in bytes. (That is, to read the needed data from the END of the file.)

- Note that there is no provision for skips between data rows inside a 2D slice, only for skips between 2D slice images.

- The int, float, and complex formats presume that the data in the image file are in the 'native' format for this CPU; that is, there is no provision for data conversion (unlike the 3Ds: format).

- Whether the 2D image data is interpreted as a 3D block or a 3D+time block depends on the rest of the command line parameters. The various 3D: input formats are just ways of inputting multiple 3D slices from a single file.

- The 'raw pgm' image format is also supported; it reads data into 'byte' images.

73.8 Notes (continued)

- Not all MCW AFNI programs support all datum types. Shorts and floats are safest. (See the '-datum' option below.)

- If '-datum short' is used or implied, then int, float, and complex data will be scaled to fit into a 16 bit integer. If the '-gsfac' option below is NOT used, then each slice will be SEPARATELY scaled according to the following choice:

  (a) If the slice values all fall in the range -32767 .. 32767, then no scaling is performed.

  (b) Otherwise, the image values are scaled to lie in the range 0 .. 10000 (original slice min -> 0, original max -> 10000.

  This latter option is almost surely not what you want! Therefore, if you use the 3Di:, 3Df:, or 3Dc: input methods and store the data as shorts, I suggest you supply a global scaling factor. Similar remarks apply to '-datum byte' scaling, with even more force.

- `to3d` now incoporates POSIX filename 'globbing', which means that you can input filenames using 'escaped wildcards', and then to3d will internally do the expansion to the list of files. This is only desirable because some systems limit the number of command-line arguments to a program. It is possible that you would wish to input more slice files than your computer supports. For example,

  to3d exp.?.*

might overflow the system command line limitations. The way to do this using internal globbing would be

to3d exp.\?.*

where the \ characters indicate to pass the wildcards ? and * through to the program, rather than expand them in the shell.

(a) Note that if you choose to use this feature, ALL wildcards in a filename must be escaped with \ or NONE must be escaped.

(b) Using the C shell, it is possible to turn off shell globbing by using the command 'set noglob' – if you do this, then you do not need to use the \ character to escape the wildcards.

(c) Internal globbing of 3D: file specifiers is supported in `to3d`. For example,

'3D:0:0:64:64:100:sl.*'

could be used to input a series of 64x64x100 files with names 'sl.01', 'sl.02' .... This type of expansion is specific to `to3d`; the shell will not properly expand such 3D: file specifications.

(d) In the C shell (csh or tcsh), you can use forward single 'quotes' to prevent shell expansion of the wildcards, as in the command

to3d '3D:0:0:64:64:100:sl.*'

The globbing code is adapted from software developed by the University of California, Berkeley, and is copyrighted by the Regents of the University of California (see file mcw_glob.c).

73.9 More Options

-2swap This option will force all input 2 byte images to be byte-swapped after they are read in.

-4swap This option will force all input 4 byte images to be byte-swapped after they are read in.

-gsfac value will scale each input slice by 'value'. For example, '-gsfac 0.31830989' will scale by 1/Pi (approximately). This option only has meaning if one of '-datum short' or '-datum byte' is used or implied. Otherwise, it is ignored.

-datum type will set the voxel data to be stored as 'type', which is currently allowed to be short, float, byte, or complex. If -datum is not used, then the datum type of the first input image will determine what is used. In that case, the first input image will determine the type as follows:

byte	–>	byte
short	–>	short
int, float	–>	float
complex	–>	complex

If -datum IS specified, then all input images will be converted to the desired type. Note that the list of allowed types may grow in the future, so you should not rely on the automatic conversion scheme. Also note that floating point datasets may not be portable between CPU architectures.

-in:1 Input of huge 3D: files (with all the data from a 3D+time run, say) can cause `to3d` to fail from lack of memory. The reason is that the images from a file are all read into RAM at once, and then are scaled, converted, etc., as needed, then put into the final dataset brick. This switch will cause the images from a 3D: file to be read and processed one slice at a time, which will lower the amount of memory needed. The penalty is somewhat more I/O overhead.

-orient code Tells the orientation of the 3D volumes. The code must be 3 letters, one each from the pairs (R,L) (A,P) (I,S). The first letter gives the orientation of the x-axis, the second the orientation of the y-axis, the third the z-axis:

R	=	right-to-left	L	=	left-to-right
A	=	anterior-to-posterior	P	=	posterior-to-anterior
I	=	inferior-to-superior	S	=	superior-to-inferior

Note that the -xFOV, -zSLAB constructions can convey this information.

73.10 Options that Affect the X11 Image Display

These options only affect `to3d` if it is run in "interactive" mode (see the next section).

-gamma gg The gamma correction factor for the monitor is 'gg' (default gg is 1.0; greater than 1.0 makes the image contrast larger – this may also be adjusted interactively).

-ncolors nn Use 'nn' gray levels for the image displays (default is 80).

-xtwarns Turn on display of Xt warning messages.

73.11 Notes (continued)

- Different computers use different formats for storage of two byte (short) integers. Some computers store the most significant (high) byte first, followed by the least significant (low) byte. Other computers store the low byte first, followed by the high byte. It is *very important* that the image data files be converted to the correct format *before* running program `to3d`, or else the -2swap option should be used *inside* program `to3d`. For example, suppose that the 2D image files have been transferred from a RISC workstation to an Intel CPU computer. Since these computers use different formats for storage of short integers, it is necessary to swap bytes in the image files prior to using program `to3d` (by using program `2swap`), or the -2swap option should be used when running `to3d`.

73.12 Examples

Example 1. Suppose that the user has a collection of 2D image files, consisting of 16 z-slices for 68 time steps, 5 seconds apart. The files fred01.0001, fred01.0002, ... , fred16.0068 contain the 2D image data, where the file format is: fred*ij*.*klmn*, where ij = number of z-slice, and $klmn$ = number of time slice. The command line to create an *AFNI* 3D+time dataset is:

```
to3d  -epan -time:tz 68 16 5000 alt+z \
      -xFOV 120S-I -yFOV 120A-P -zSLAB 52L-53R \
      -prefix fred -session ../ \
      fred*.*
```

The command line options indicate that this is an Echo Planar (epan) dataset; this is a time dependent data set, with files ordered by time first, then by z-slice; there are 68 time slices and 16 z-slices, with 5 seconds between time slices. The order of the z-slices is alternating in the plus z-direction. The output, an *AFNI* 3D+time dataset, is to be stored in file fred+orig.BRIK (and .HEAD) in the next higher directory (../). The wild card characters '*.*' indicate that the data files all begin with 'fred'.

74 Program to3d (interactive mode)

74.1 Purpose

This program reads in a collection of images and formats them into an *AFNI* dataset. It presents you with a control panel that lets you enter various information about the images that *AFNI* needs to know (e.g., the grid spacing, the axes orientation, etc.). `to3d` may also be run in a batch mode (see previous section), which is useful when creating a large number of similar datasets.

74.2 Usage

The command line to run to3d interactively takes the form:

to3d image_filename image_filename ...

For example, if the images are named anat.001, anat.002, ..., then "to3d anat.*" would start the program, read the files into memory, and load the control panel onto the X11 display.

The type of images that to3d can read are described first, followed by a description of each of the controls with which the program presents you. This discussion is fairly detailed, since it is important to get the conversion right.

You may get help on how to use the command line by typing the command "to3d -help". This is a general feature of the programs in the **MCW** *AFNI* package.

74.3 Input Image Formats

See the Input Image Formats subsection of program to3d (batch mode).

74.4 Time Dependent Datasets

See the Time Dependent Datasets subsection of program to3d (batch mode).

74.5 Orientation Controls

In the upper left corner of the to3d control panel are three selectors for you to enter the orientation of the input image axes. In to3d, the x-axis is across the screen, left-to_right; the y-axis is down the screen, top-to-bottom; and the z-axis is "through" the screen, in slice order, first-to-last. The six possible anatomical orientations are *Anterior-to-Posterior, Posterior-to-Anterior, Superior-to-Inferior, Inferior-to-Superior, Left-to-Right*, and *Right-to-Left*. By clicking the left mouse button when 'on' the arrows, you can select the anatomical orientations that correspond to the input image axes. (You should use the View Images button to help you in this task.)

to3d will not let you save the images into a 3D dataset until you have set up a consistent set of axes orientations. *Consistent* means, for example, that you cannot have the x-axis be Left-to-Right and have the y-axis be Right-to-Left. There are 48 $(6 \cdot 4 \cdot 2)$ consistent sets of orientations, out of the 216 $(6 \cdot 6 \cdot 6)$ possible combinations.

Note that there is no requirement that the different datasets in a given session be acquired in the same planes. It is quite possible to have the anatomical images be acquired in sagittal planes and have the functional images acquired coronally, for example. The important thing is to use to3d to set up the geometry of the datasets so that they will correspond. As discussed in subsection Axes Centering Controls, this is most simply done using the coordinates reported by the MR scanning system.

74.6 Voxel Dimension Controls

In the upper middle of the to3d control panel are selectors for entering the voxel dimensions (all **MCW** *AFNI* dimensions are given in mm). Just to the left of the Field of View control is a set of buttons which controls which voxel dimension controls are active. If cubical is selected, then only the Field of View control is active. If square is selected, then the z voxel size control is also active. If irregular is selected, then the x voxel size, y voxel size, and z voxel size controls are active and the Field of View control is not.

You may use the arrows to scroll through the range of numerical values allowed for voxel dimensions, or you may place the mouse cursor in the numerical display window to the right of the arrows, click (Button 1), and then type in the chosen value. Pressing Return, Tab, or moving the mouse cursor out of the window will cause the value to 'take effect' (e.g., changing the Field of View will cause the x voxel size and y voxel size values to change).

74.7 Time Dimension

When the command line specifies that the output is to be a 3D+time dataset, then the Time Dimension display appears just below the Field of View control. The Time Dimension display indicates the time parameters, and these parameters are *not* operator adjustable. For example, suppose the operator uses the command line

to3d -time:tz 68 16 5s alt+z fred*.*

Then the Time Dimension display shows: TR=5.000 (s), NR=68, Nz=16. This indicates that there are 68 time steps, 16 z-slices, and 5 seconds between images.

74.8 Axes Centering Controls

The default location for the dataset is centered on the (0,0,0) point of the coordinate system. For any given dataset, this is fine, but when multiple datasets are gathered in the same scanning session, they may not both be centered on the same point. The controls at the upper right of the to3d control panel are for offsetting the axes from the (0,0,0) point.

The set of buttons x axis centered, etc., controls whether each axis is to be centered on its 0 value. Be clicking on z axis centered, for example, you will 'unlock' the z origin control above. The value here is the absolute (unsigned) distance the *center* of the first voxel in the z-direction (i.e., slice 0) is offset from the location $z = 0$. The other directions may be off-centered in a similar fashion. For the x-direction, the control specifies how far the left edge of the images is offset from $x = 0$. For the y-direction, the control specifies how far the top edge of the images are offset from $y = 0$. Notice that when a control is 'locked' (that axis is specified as centered), then changing the voxel dimension in that direction will cause the origin value to alter correspondingly.

In MRI, data images are commonly displayed as being so many mm Left, Right, Anterior, Posterior, Inferior, or Superior (L, R, A, P, I, or S) from the (0,0,0) point of the gradient coil set. These values are directly translatable into the origin values needed here. For example, is a set of 151 sagittal anatomical slices is 1 mm thick, and runs from 72 L to

78 R (in that order), then the z origin value to use would be 72.0 (instead of the centered 75.0). If a corresponding set of 20 functional sices (6 mm thick) runs from 60 L to 54 R, then you should enter the z origin as 60.0 for that dataset. When the two datasets are conbined in *AFNI*, they will then be properly aligned.

One of the most confusing parts of using to3d is setting the origin correctly. This is necessary to get functional data to align with anatomical data if the two sets of images weren't acquired on exactly the same grids.

The "origin" controls (at the far right of the top of the to3d control panel) are used to offset the center of the first voxel from the $(0, 0, 0)$ of the coordinate system. By default (with the "centered" controls pressed in), each axis is offset exactly $\frac{1}{2}$ the span of the data in that direction.

If the z-axis is "Posterior–to–Anterior", the "z origin" control is used to set the distance the center of the first input slice is from $z = 0$ in the Posterior direction (i.e., in the direction before the word "–to–"). If "z axis centered" is pressed in, this offset will be computed from the "z voxel size" control and the number of input slices (so that when one alters the "z voxel size" control, the "z origin" control will change to match).

If one were to change the z-axis to be "Anterior–to–Posterior", then the "z origin" distance would now be interpreted to be the offset from $z = 0$ to the center of the first slice in the Anterior direction. For example, if the slices run from A=40 mm to P=60 mm, then if the images are input in the A-to-P order, then "z origin" should be set to 40, but if the images are input in the reverse (P-to-A) order, then "z origin" should be set to 60.

Only under unusual circumstances will the "x origin" and "y origin" controls need to be used. This would happen if a set of slices were not gathered centered on the origin of the scanner's coordinate system. I've never seen a case where these controls are useful, but no doubt someone somewhere has.

In some previous versions of MCW *AFNI* , the value entered in the "origin" controls was required to be non-negative. This meant that it was not easy to deal with data that was only on one side of the origin. For example, if a set of slices runs from L=5 mm to L=60 mm, from Right–to–Left (e.g., data from only one brain hemisphere), then what is the correct "z origin" value? Since the slices run from Right–to–Left, the value entered in "z origin" is interpreted as the distance to the Right. Since the actual offset is 5 mm to the Left, the correct entry in "z origin" should thus be -5.0. To3d now allows the "origin" control values to be negative just for this purpose.

74.9 Datum

Just below the Orientation Controls is the Datum display. This indicates the type of data that has been read by the program, such as: Datum: short.

74.10 View Control

Just below the Datum display is the View control. This control allows the operator to specify the view type extension for the output file name. When the output file is written to the disk, the filename will have the corresponding extension. There are 3 options: Original

View, AC-PC Aligned, and Talairach View, which may be selected by using the mouse cursor to scroll through the list. For example, if the prefix is specified to be "fred" (see below), then the output file will be fred+orig, fred+acpc, or fred+tlrc (.HEAD and .BRIK), respectively.

74.11 Geometry Parent

Above the double line were the 'geometry' controls. Below are the 'identity' controls. The uppermost of these is for specification of a 'geometry parent'.

It is common to set up many datasets with the same geometrical information. If you type in a dataset header file name in the Copy geometry text field, to3d will read the dataset's geometry information and load it into the geometry controls above the double line. (The geometry parent is also the mechanism for batch mode operation of to3d; see the previous section (program to3d (batch mode)).

74.12 Anatomy Parent

An 'anatomy parent', to *AFNI*, is an anatomical dataset that the current dataset is presumed to be aligned with. Normally, you do not need to enter an anatomy parent for any datasets, since *AFNI* will choose one for each functional dataset (from the same session). Under some circumstances, it may be necessary to specify an anatomy parent from a different session. That is the function of this control. Again: *under normal usage, this control is not required.* (It may be needed for some new functions planned for *AFNI*.)

74.13 Dataset Type

On the next-to-bottom row of the to3d control panel are two selectors for choosing the type of data. The one to the left lets you choose between 'Head anatomy', 'Head function', 'General anatomy', and 'General function'. (The difference between 'Head' and 'General' is that the former allows the Talairach transformation, and the latter does not.)

The selector to the right lets you specify the type of anatomical or functional data in the dataset. The only confusing issue is the choice between functional intensity and intensity+threshold (fith). In the former case, all the images input are taken to be intensity, and are stacked up in the z-direction. In the latter case, the images are divided into two sets: the first half is the collection of intensity images, and the second half is the collection of threshold images. Note well that the z origin control changes its value when it is locked (auto-centering) and when you toggle between the intensity and intensity+threshold choices. This is because the z-extent of the dataset depends on the slice thickness and the number of images through the volume.

74.14 Output Directory and Prefix

The bottom row of the to3d control panel contains two text fields for you to enter the session directory and the dataset prefix names under which you wish to save this dataset. These are required, and any attempt to save the dataset without proper names will fail.

The program attempts to check the names for characters (e.g., spaces), which make it difficult to deal with the files, but it is not foolproof. So don't be a fool.

N.B.: Some older Unix filesystems limit filenames to be 14 characters long. Since the +view.NAME part of the dataset filenames takes up 10 characters, on such a system, you can only use prefixes of 4 characters (or less). If you use a longer prefix, the filenames will probably be mangled by Unix and *AFNI* will not be able to read the files.

74.15 Action Controls

At the lower right of the to3d control panel is a set of five buttons. The first is Byte Swap[2], explained below. The second is button help. If you click on this button, the mouse cursor changes shape, into a little hand. Clicking the hand on any button in to3d will pop up a little help window for that button. To dismiss the popup help, click in the help window. (Only one popup help window will be open at a time.)

The third action control is View Images. This will open a window that lets you scroll through the input images. This ability is especially useful when deciding on the axes orientations — although for functional images, it is usually necessary to have the orientation information written down. Since humans are very left-right symmetric, you usually have to know which direction is left and which is right — seeing the images won't help much here.

The fourth action control is Save Dataset. It will write the 3D dataset to disk in the session directory chosen (creating it, if necessary). If any error occurs, one or more windows will pop up with messages explaining what the problem is: for example, inconsistent axes orientations will prevent the dataset from being written out.

The last action control is Quit. You must press this button twice *within five seconds* in order to exit to3d. (This is to prevent accidental exit.) You may also quit to3d by using the window manager's Close or Quit function. (This is not safeguarded by the double quit feature.)

74.16 Notes

- Different computers use different formats for storage of two byte (short) integers. Some computers store the most significant (high) byte first, followed by the least significant (low) byte. Other computers store the low byte first, followed by the high byte. It is *very important* that the image data files be converted to the correct format, either *before* running program to3d by using program 2swap, or *while* running program to3d, by pressing the Byte Swap[2] button. For example, suppose that the 2D image files have been transferred from a RISC workstation to an Intel CPU computer. Since these computers use different formats for storage of short integers, it is necessary to swap bytes in the image files prior to using program to3d. This can be done easily with program 2swap. The program simply swaps byte pairs on the files specified by the user. (See user documentation for program 2swap).

75 Program waver

75.1 Purpose

Creates an ideal waveform timeseries file. The output goes to stdout, and normally would be redirected to a file.

75.2 Usage

waver [**options**] > **output_filename**

75.3 Options

(# refers to a number; [xx] is the default value)

-WAV	=	Sets waveform to Cox special	[default]
-GAM	=	Sets waveform to form t^b * exp(-t/c)	

(waveforms will also be chosen if one of the options below is used)

These options set parameters for the -WAV waveform.

-delaytime #	=	Sets delay time to # seconds	[2]
-risetime #	=	Sets rise time to # seconds	[4]
-falltime #	=	Sets fall time to # seconds	[6]
-undershoot #	=	Sets undershoot to # times the peak (this should be a nonnegative factor)	[0.2]
-restoretime #	=	Sets time to restore from undershoot	[2]

These options set parameters for the -GAM waveform:

-gamb #	=	Sets the parameter 'b' to #	[8.6]
-gamc #	=	Sets the parameter 'c' to #	[0.547]

These options apply to any waveform type:

-peak #	=	Sets peak value to #	[100]
-dt #	=	Sets time step of output AND input	[0.1]

The default is just to output the waveform defined by the parameters above. If an input file is specified by one of the options below, then the timeseries defined by that file will be convolved with the ideal waveform defined above – that is, each nonzero point in the input timeseries will generate a copy of the waveform starting at that point in time, with the amplitude scaled by the input timeseries value.

-xyout	=	Output data in 2 columns: 1=time 2=waveform (useful for graphing) [default is 1 column=waveform]
-input infile	=	Read timeseries from 'infile'; convolve with waveform to produce output
-inline DATA	=	Read timeseries from command line DATA; convolve with waveform to produce output

DATA is in the form of numbers and count@value, as in
    -inline 20@0.0 5@1.0 30@0.0 1.0 20@0.0 2.0
which means a timeseries with 20 zeros, then 5 ones, then 30 zeros, a single 1, 20 more zeros, and a final 2. [The '@' character may actually be any of: '@', '*', 'x', 'X'. Note that * must be typed as * to prevent the shell from trying to interpret it as a filename wildcard.]

At least one option is required, or the program will just print this message to stdout. Only one of the timeseries input options above can be used.

If you have the 'xmgr' graphing program, then a useful way to preview the results of this program is through a command pipe like
    waver -dt 0.25 -xyout -inline 16@1 40@0 16@1 40@0 | xmgr -source stdin

If a square wave is desired, see the 'sqwave' program.

76 Using the 3D: Input File Specification

AFNI and its associated programs can read files in many different formats. Rather than encode all different image formats in the software (which would require one format for every programmer that ever lived), the "generic" image file format is an uncompressed contiguous 2D or 3D array of 8 bit unsigned bytes, 16 bit signed shorts, 32 bit signed ints, or 32 bit floats.

The burden is on the user to specify exactly WHERE in the file the image data is stored. This is done with the rather clumsy mechanism of prepending the necessary information to each input filename.

An example:

3D:80:0:64:64:1:fred.001

Piece	Meaning
3D:	Indicates that this is a file of shorts.
80:	Indicates to skip 80 bytes at the start of the file.
0:	Indicates to skip 0 bytes at the start of each 2D image within the file.
64:	Indicates that the "x" dimension of each 2D image is 64.
64:	Indicates that the "y" dimension of each 2D image is 64.
1:	Indicates that there is only 1 2D image in this file.
fred.001	Indicates that the actual filename is "fred.001".

If there were 10 such files, you could put them into a 3D block with the Unix command

cat fred.* > fred3d

This file could be read into to3d with the specification

3D:0:80:64:64:10:fred3d

Notice that the "80" has moved to the "per image header" slot, since each image has the 80 bytes of header information attached. If there were only one 80 byte header for all the data in this big file, then the appropriate specification would be

3D:80:0:64:64:10:fred3d

In general, the format for images of shorts is

3D:hglobal:himage:nx:ny:nz:fname	for 16 bit input
3Ds:hglobal:himage:nx:ny:nz:fname	for 16 bit input, swapped bytes
3Db:hglobal:himage:nx:ny:nz:fname	for 8 bit input
3Di:hglobal:himage:nx:ny:nz:fname	for 32 bit input
3Df:hglobal:himage:nx:ny:nz:fname	for floating point input

where

hglobal	= number of bytes to skip at start of whole file
himage	= number of bytes to skip at start of each image
nx	= x dimension of each image
ny	= y dimension of each image
nz	= number of images (at least 1)
fname	= actual filename on disk to read

The ':' separators are required. The k-th image starts at BYTE offset

hglobal + (k+1)*himage + vs*k*nx*ny

in file 'fname' for k=0,1,...,nz-1. Here, vs=voxel length=1 for bytes, 2 for shorts, 4 for ints and floats.

As a special case, hglobal = -1 means read data starting at offset len-nz*(vs*nx*ny+himage), where len=file size in bytes. In particular, this means that if you have a 64x64 image of shorts in a file, you can read it with

```
3D:-1:0:64:64:1:filename
```

76.1 Reading a Number of Files, Method 1

This is clearly a rather clumsy method to use if you have a bunch of files in a funny format. The count program (part of the *AFNI* package) and the C-shell backquote operator can be combined to remove a little of the pain.

The count program simply generates strings of numbers. For example, the output of "count -digits 3 5 17" is

005 006 007 008 009 010 011 012 013 014 015 016 017

"-digits 3" means to use 3 digits (at least); "5 17" means to count from 5 to 17. A more complex use of count can attach a root to the beginning of each number: "count -digits 3 -root fred. 5 17" produces

fred.005 fred.006 fred.007 fred.008 fred.009 fred.010 fred.011 fred.012 fred.013 fred.014 fred.015 fred.016 fred.017

So, to read in these files, we would use the count command

```
count -digits 3 -root 3D:-1:0:64:64:1:fred. 5 17
```

which produces the output

3D:-1:0:64:64:1:fred.005 3D:-1:0:64:64:1:fred.006 3D:-1:0:64:64:1:fred.007
3D:-1:0:64:64:1:fred.008 3D:-1:0:64:64:1:fred.009 3D:-1:0:64:64:1:fred.010
3D:-1:0:64:64:1:fred.011 3D:-1:0:64:64:1:fred.012 3D:-1:0:64:64:1:fred.013
3D:-1:0:64:64:1:fred.014 3D:-1:0:64:64:1:fred.015 3D:-1:0:64:64:1:fred.016
3D:-1:0:64:64:1:fred.017

Finally, we need to get this string onto a command line. This is done by putting the whole count command inside backquotes. In the Unix C-shell, this means to execute the command that is backquoted, then put its output into the whole command line about to be executed. Thus, we could do

```
to3d -fim `count -digits 3 -root 3D:-1:0:64:64:1:fred. 5 17`
```

This is clumsy, but not as bad as typing all the 3D: prefixes for each file.

Finally, note that the program FD2 cannot read in 3D blocks of data. Each input file must contain only 1 image. It can, however, use the 3D: prefix to properly skip arbitrary header information before the actual image data is read. The F64 and F128 scripts in the AFNI 1.04a distribution show how this can be done to trick the program into reading slices out of a 3D block.

76.2 Reading a Number of Files, Method 2

If you have AFNI version 1.04a (or higher), then there is an easier way to customize the input formats. It still uses the 3D: specifiers, so you need to understand those. The method is to define a Unix environment variable to associate a 3D: specifier with a given file size. For example,

setenv MCW_IMSIZE_1 8272=3D:-1:0:64:64:1:

means that a file whose length is 8272 bytes should be read with the string "3D:-1:0:64:64:1:" attached before its filename. This will allow you to tell the AFNI programs how to deal with your particular filesizes. You can use the environment variables MCW_IMSIZE_1 up to MCW_IMSIZE_99. If this is done, then the command "FD2 fred.*" would work to read the "fred.001" ... files described earlier.

The following shows the second form of an allowed MCW_IMSIZE environment variable:

setenv MCW_IMSIZE_2 %16096+80=3D:80:7904:64:64:

The "%" that starts the value signals that this is a variable length file type that can contain more than one image. In this case, if the file length is of the form 16096*N+80 bytes, where N=1,2,..., then the string "3D:80:7904:64:64:N:" will be put before the filename. This example allows the input of a 3D brick of images, each 2D image with a 7904 byte header, and the whole collection with an 80 byte header at the very beginning.