# mc-afni

```
Comments or questions: analysis-bugs@nmr.mgh.harvard.edu\\
$Id: mc-afni.tex,v 1.1 2005/05/04 17:00:48 greve Exp $
```

## 1 Introduction

**mc-afni** is a front-end for running the AFNI motion correction (see http://varda.biophysics.mcw.edu/ cox/ for more information about AFNI). You must have the AFNI software package installed on your computer and the binaries in your path. It converts from bfile format to AFNI format then uses AFNI's 3dvolreg to align the functional volume to a target volume. Algorithm used by 3dvolreg: 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.

## 2 Usage

Typing mc-afni at the command-line without any options will give the following message:

```
USAGE: mc-afni
Options:
   -i stem : input  volume
   -o stem : output volume
   -t stem : target (input volume)
   -toff offset  : target volume offset   (0)
   -session dir  : afni session directory (/tmp)
   -prefix  name : afni prefix            (procid)
   -scriptonly scriptname   : don't run, just generate a script
   -version : print version and exit
```

## 3 Command-line Arguments

**-i stem**: (Required) stem of the input functional volume for a single run. It is assumed that the data are stored in *bfile format*.

**-o stem**: (Required) stem of the output motion-corrected functional volume. This script will also produce a file called *stem.mcdat* in which the AFNI motion parameters are stored. There are 9 columns: *n roll pitch yaw dS dL dP rmsold rmsnew*, where n is the functional index, roll is the rotation about the I-S axis, pitch is the rotation about the R-L axis, yaw is the rotation about the A-P axis, dS is displacement in the Superior direction, dL is the displacement in the Left direction, dP is the displacement in the Posterior direction, rmsold is the RMS difference between input brick and base brick, rmsnew is RMS difference between output brick and base brick. All rotations are in degrees CCW, and all displacements are in mm. See the AFNI documentation for 3dvolreg for more information.

**-t stem**: (Optional) stem of the target volume. The target is the image to which AFNI attempts to register the input volume. The default is the first image of the input volume.

**-toff offset**: (Optional) register to the $n^{th}$ image in the target volume ($n = offset$), where the first image has offset 0.

**-session dir**: (Optional) location in which to place the temporary AFNI files. The default is to put it in /tmp which should be local to the computer executing the program. This can speed up the run time significantly as compared to when the session directory is someplace else across the network. All temprorary files are deleted by **mc-afni**.

**-prefix name**: (Optional) name to give the temporary AFNI files.

**-scriptonly scriptname**: (Optional) this does not do anything yet.

**-version**: print **mc-afni** version and exit.

## 4 Examples

### 4.1 Single Run Motion Correction

Consider the case where there is a functional volume with stem $func$ with 16 slices, each with 120 time points. To motion correct this volume, run:

```
% mc-afni -i func -o func_mc
```

This will create a new volume with stem *func_mc*. It will also create a file called *func_mc.mcdat* with the motion parameters. Note that this implicity aligned all of the images with the first image. Note that the above invocation is equivalent to

```
% mc-afni -i func -o func_mc -t func -toff 0
```

To align with the $20^{th}$ image, run:

```
% mc-afni -i func -o func_mc -toff 19
```

### 4.2 Mulitple Run Motion Correction

When there are multiple runs, you will want to align the images in both runs to a single (target) image. Consider two functional volumes, $func1$ and $func2$, and we want to align both volumes to the third image in $func1$. Then run:

```
% mc-afni -i func1 -o func1_mc -toff 2
% mc-afni -i func2 -o func2_mc -t func1 -toff 2
```

Again, this will create motion parameter files *func1_mc.mcdat* and *func2_mc.mcdat*.