spm12Batch System
for systematically processing BOLD fMRI data.

This is an ecosystem for processing fMRI data in a coherent, flexible, streamlined, and self-documenting way. This ecosystem is based on bash and in general provides wrapping of **MATLAB** (along with **FSL** utilities, **ANTs**, **AFNI**, and **ANIMA**) to provide a full pipeline of data processing.

This system has successfully run on MAC OS X and Linux. Because the system is bash based, it is highly transportable, and as long as the underlying third party executables are accessible, the code should work out of the box after a short configuration.

While the system is called **spm12Batch**, it does leverage functionality from other processing systems to result in a more robust processing pipeline. This includes have the option to use **ANTs** for normalization, and to use **AFNI** for despiking of data. The steps of slice-time correction, motion correction, and co-registration steps done with **SPM**. There is also an option to use **SPM** to do tissue segmentation to provide for WM and CSF masks that can be used further down for physiological denoising using a process such as COMPCOR. **ANIMA** is used for EPI distortion correction.

The code is written such that you call the code with a list of parameters, if other than default values, and a list of fMRI sessions (subjects/participants). The code will then build the necessary (unique) scripts and launch into background, providing a log file. The result in a fully logged process of how your data have moved through pre-processing.

The code is such that while session names are unique, file names below that level are not unique, which provides for ease of specification.

A general philosophy of this ecosystem is to keep different functional tasks apart from each other, and to treat each fMRI session as an independent for pre-processing, regardless of whether these are multiple visits for a single participant.

One result is that if there are multiple-tasks for a session, then the pipeline is to be run fully by task, this does result in a little bit of size inflation and process inflation as the derivation of normalization to MNI space is done for each task. This is due to the high-resolution image being co-registered to the BOLD data, and not the BOLD data being co-registered to the high-resolution image. Because each task is independently processed, this does provide flexibility that would not otherwise exist if all tasks were being processed in a single pipeline.

Currently this provides minimal QC, limited to motion parameters being plotted to a PNG file for viewing. Future versions will incorporate additionally QC metrics.

spm12Batch System
Installation

In addition to access to **SPM12**, you need in your bash **PATH** accessing **FSL**, **ANTs**, **AFNI**, and **ANIMA**. We leave it up to you to install these packages.

Also have the distribution folder for **spm12Batch** (downloadable from GIT) in you **PATH**. This has successfully worked with various version of **FSL** from version 5.x (only using small utilities in FSL for doing s-form and q-form manipulation, etcetera), and **ANTs** 2.3.5, **ANIMA** 4.X, and **AFNI** (21.0.20 “Titus”).

Edit **spm12Batch/spm12Batch_Global** to point to **MATLAB** and to **SPM12** code.

Once successfully configured, then run the command:

spm12Batch_CheckInstall

to see the list of commands do:

spm12Batch

Each command has built-in help. To see the help, issue the command without any options.

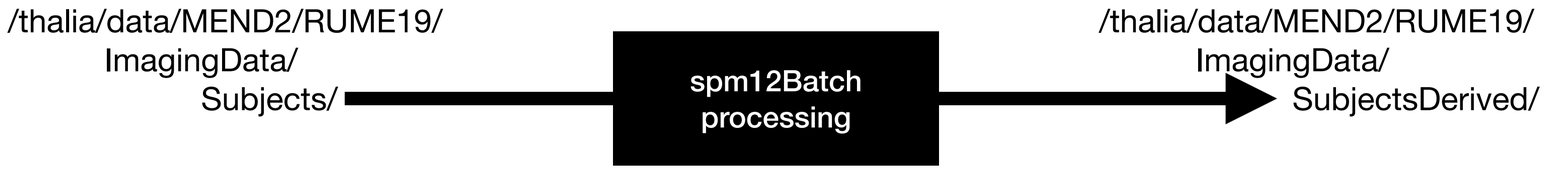
To successfully execute a command you must first run the command without any options as to let the command know you have minimally examined the help.

The system can be configured, via an expert, to allow for jobs to either email or text you on completion. Automated email at conclusion of a job, however, is problematic on some systems that are not configured with a static IP address and IP name in a DNS. The email will also be dependent on what firewall rules exist at your institution. Presently, the automated email is disabled. This has worked before on system using postfix. If you would like to attempt to use the automated email system, contact Robert for some small guidance, but you will need to have expertise in Linux/MAC OS X.

spm12Batch System
Usage

The general philosophy of this package is to automate each step of pre-processing, but to not necessarily do a full pipeline on a session. The system, however, does have flexibility to allow for full pipeline processing on a session. By allowing each step to be processed on a batch of sessions then allows for visual QC to occur on that batch of sessions prior to moving on to the next step. This is different than some published processing pipelines. I always encourage people to look at their data and to get close to it, and to do this examination at each step. By having the flexibility to process each step at a time allows for data examination. This permits erroneous issues to be resolved before moving on to the next processing step.

Data is organized such that all MR sessions reside under a a master directory, and the scripts are invoked above that directory. The default is for the master directory to be called **ImagingData/Subjects**. The typical usage of the **spm12Batch** system is to process data and have the processed (derivatives) data reside in a separate tree, which typically is called **ImagingData/SubjectsDerived**. The flexibility of the code is such that this can be specified if you so choose to keep all of your original data and processed data the same. I do, however, encourage using the **SubjectsDerived** folder as it permits ease of removing erroneously processed data without needing to be in the original data tree.



When the commands are invoked, scripts are written to a folder called **matlabScripts/spm12Batch**. Each command (such as **despikeANFI, distortionCorrect, etcetera**) and built scripts live in a tree with the year and month of creation. Scripts are fully unique, include the name of the command, the date/time of creation in the name, the user that created, and an abbreviation of the host system. An example for **distortionCorrect** is here:

distortionCorrect_210324_00_10_20_201_rcwelsh_mutis.sh

distortionCorrect_210324_00_10_20_201_rcwelsh_mutis.log

spm12Batch System
Expected Organization

The general organization is structured, and with generic names as to facilitate file name searches. An example is below. While the code is expecting the anatomy to reside in a folder called "**anatomy/**", this can be overridden with the "**-a**" flag. This is the same for task folder organization.

The **spm12Batch** system is flexible to file names, as long as they follow a generic pattern, and to folder names, also following a generic pattern. A typical organization is below. Default names are overridden using flags to the **spm12Batch** commands.

```
[session]/
     anatomy/
          hiRes.nii          <-- as long as this is the same name for all sessions to be processed, it can be anything (no spaces)
          loRes.nii          <-- this is for legacy processing when headers in NII are missing orientation.
     func/                   <-- this can be called something different than "func", but should be the same for all.
          [task]/            <-- you can have multiple task under your functional data directory
               fieldMap/     <-- for each task there is a fieldMap directory ("fieldMap" is the default)
                    forward.nii   <-- We use ANIMA for correction, which has a forward and backward image
                    backward.nii
               run_01/       <-- Multiple runs for a task are separated to reside in their own folder.
                    run_01.nii    <-- Generic name for a BOLD starts with "run" (default)
               run_02/
```

To get better sense of how the system are used, look at the example calls in **BatchExamples.** I provide additionally functionality in those scripts, which are to be executed in numeric order. Examine the to better understand how to operate within the **spm12Batch** system. The folder **Examples_wANTS** provided a full pipeline, when there is a high-resolution image, fieldmap files, and EPI BOLD data. An example of processing the full pipeline on a single session is given in **Examples_DaisyChained**. Daisy chaining is accomplished by indicating to **spm12Batch** to not sent jobs to the background (with **-B** flag). Instead, if you wish to process in the background you would submit a full pipeline command to the background. Processing following the SPM nomenclature of prepending the name of the BOLD data with the processing step. This is a parameter that can be specified with the **-on** flag, but have sensible defaults.

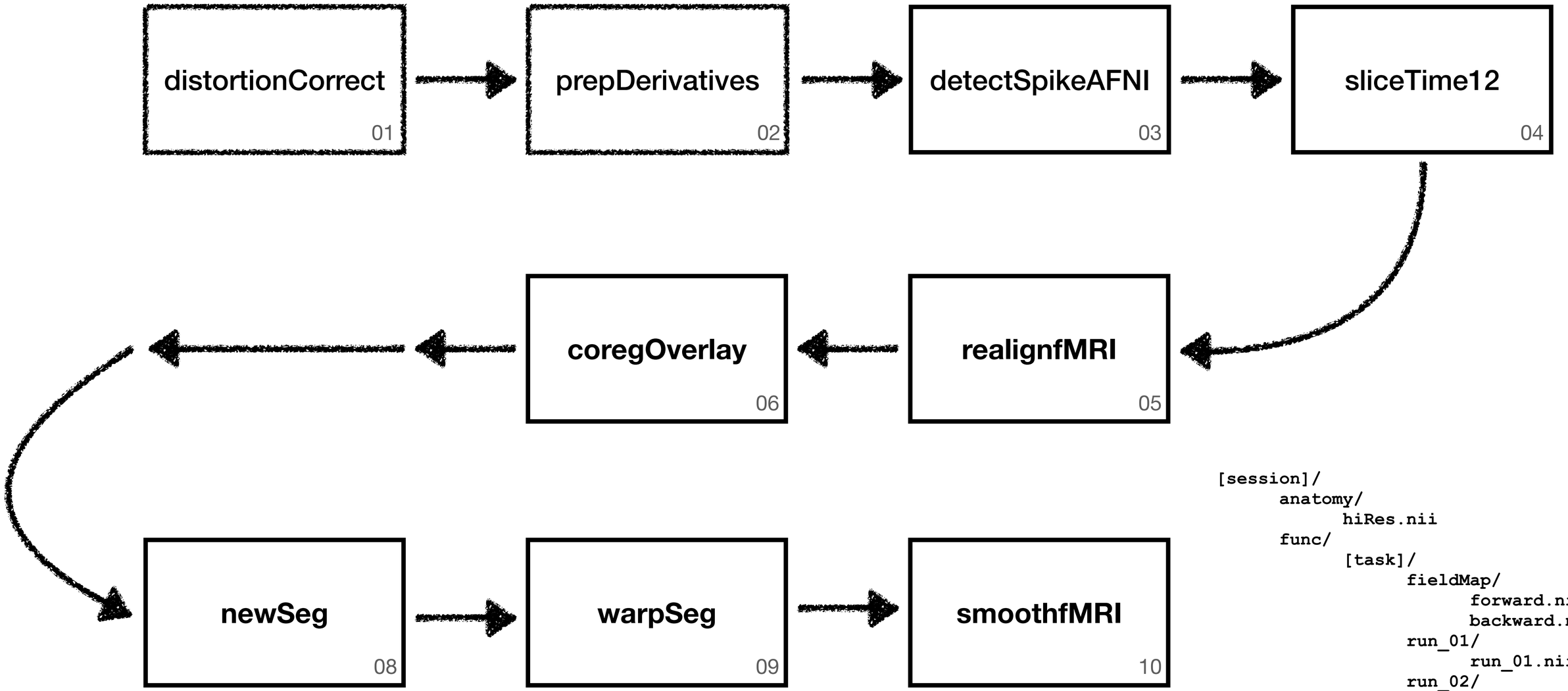
spm12Batch System
spm12Batch Commands

```
------------------------------------------------------------------------

Command            Purpose
----------         ------------------------------------------------------

distortionCorrect  A wrapper to ANIMA to correct EPI BOLD images.

prepDerivatives    A script to move data from canonical data to derived data.

sliceTime12        A wrapper to the SPM12 slice time correction routine.

sliceTimeMB        A wrapper to the SPM12 slice time correction routine.
                   Use this command for multiband data.

realignfMRI12      A wrapper to the SPM12 coregistration for time-series.

detectSpikeAFNI    A bash routine to find bad image slices using AFNI.

coregOverlay       A wrapper to the coregistration routine in SPM12.

coregHiRes         A wrapper to the coregistration routine in SPM12.

antsHiRes          A wrapper to ANTs functionality for doing warping of HIRES.

antsfMRI           A wrapper to ANTs functionality for warping fMRI data.

antsN4SS           A wrapper to ANTs functionality to N4 and skull strip HIRES.

newSeg             A wrapper to do MNI warping of hi-resolution image
                   to MNI the standard normalize function in SPM12.

pcafMRI            A wrapper to calculate COMPOR-like regressors.

erodeMask          A wrapper to FSL/fslmaths to erode masks.
                   Usually auto-called by newSeg/cat12 commands

warpSeg            A wrapper to do MNI warping of functional images
                   to MNI space based on either output of newSeg or cat12HiRes.

smoothfMRI         A wrapper to smooth images using SPM12 routines.


Copyright Robert C. Welsh and others, 2003-2021
```

spm12Batch System
Examples

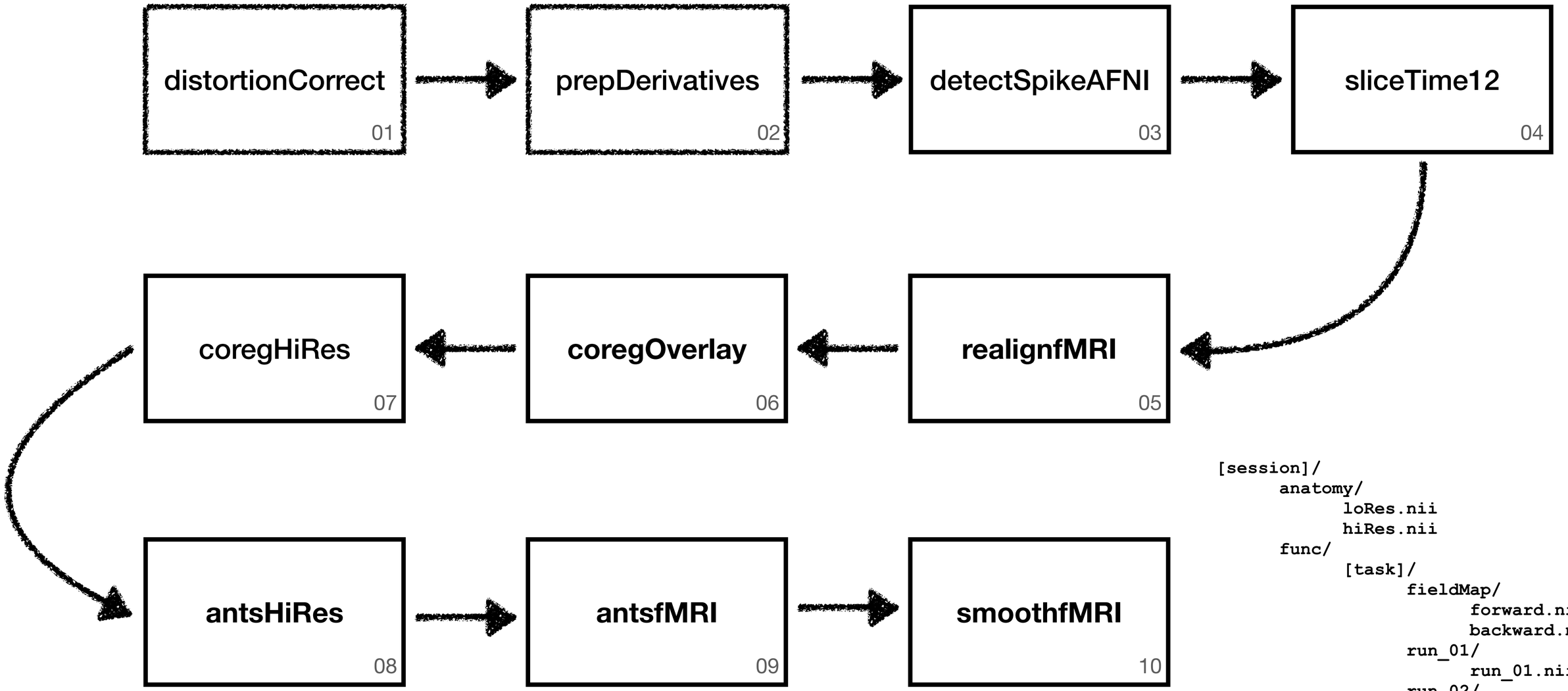
Following are diagrams illustrating the pipeline steps and different use cases.

spm12Batch System
spm12Batch processing pipeline without overlay image using SPM normalization



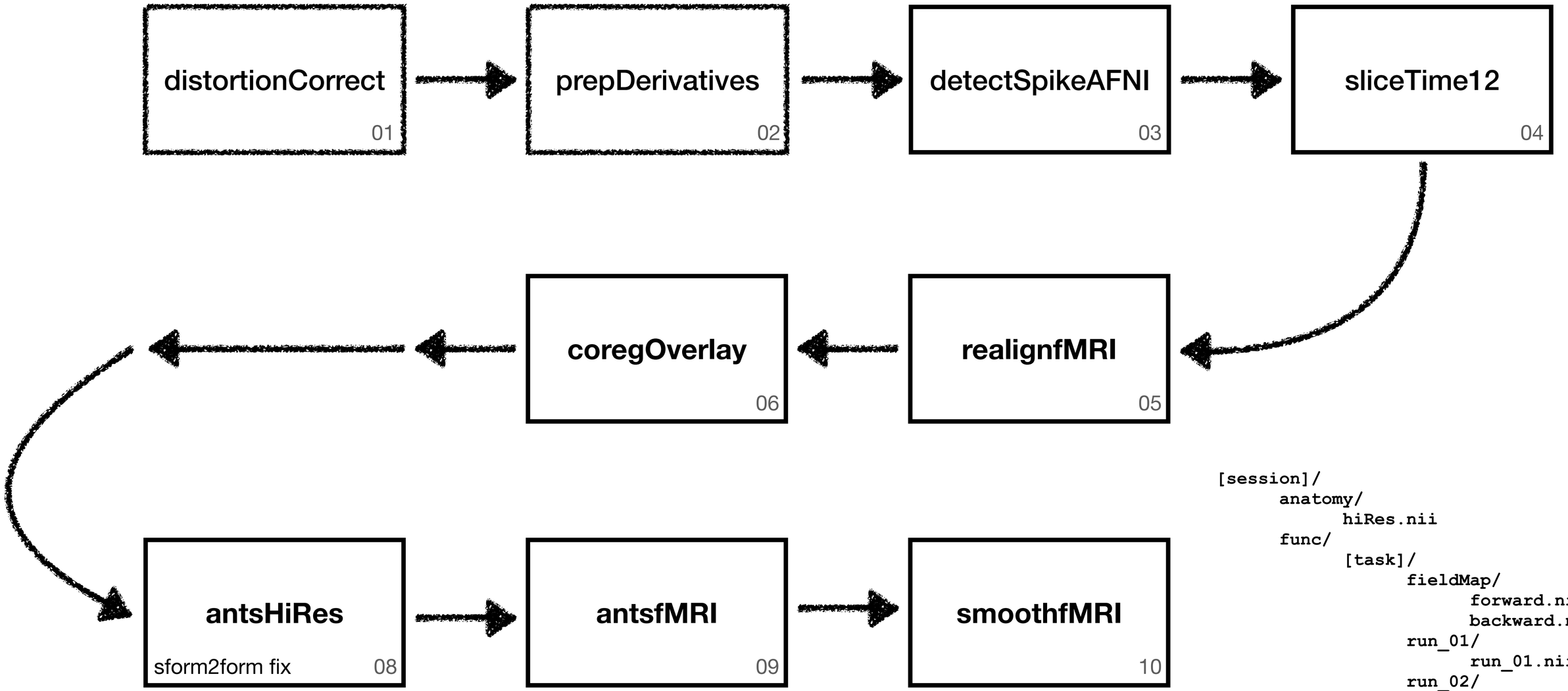

```
[session]/
      anatomy/
            hiRes.nii
      func/
            [task]/
                  fieldMap/
                        forward.nii
                        backward.nii
                  run_01/
                        run_01.nii
                  run_02/
                        …
```

spm12Batch System
spm12Batch processing pipeline with overlay image using ANTs normalization



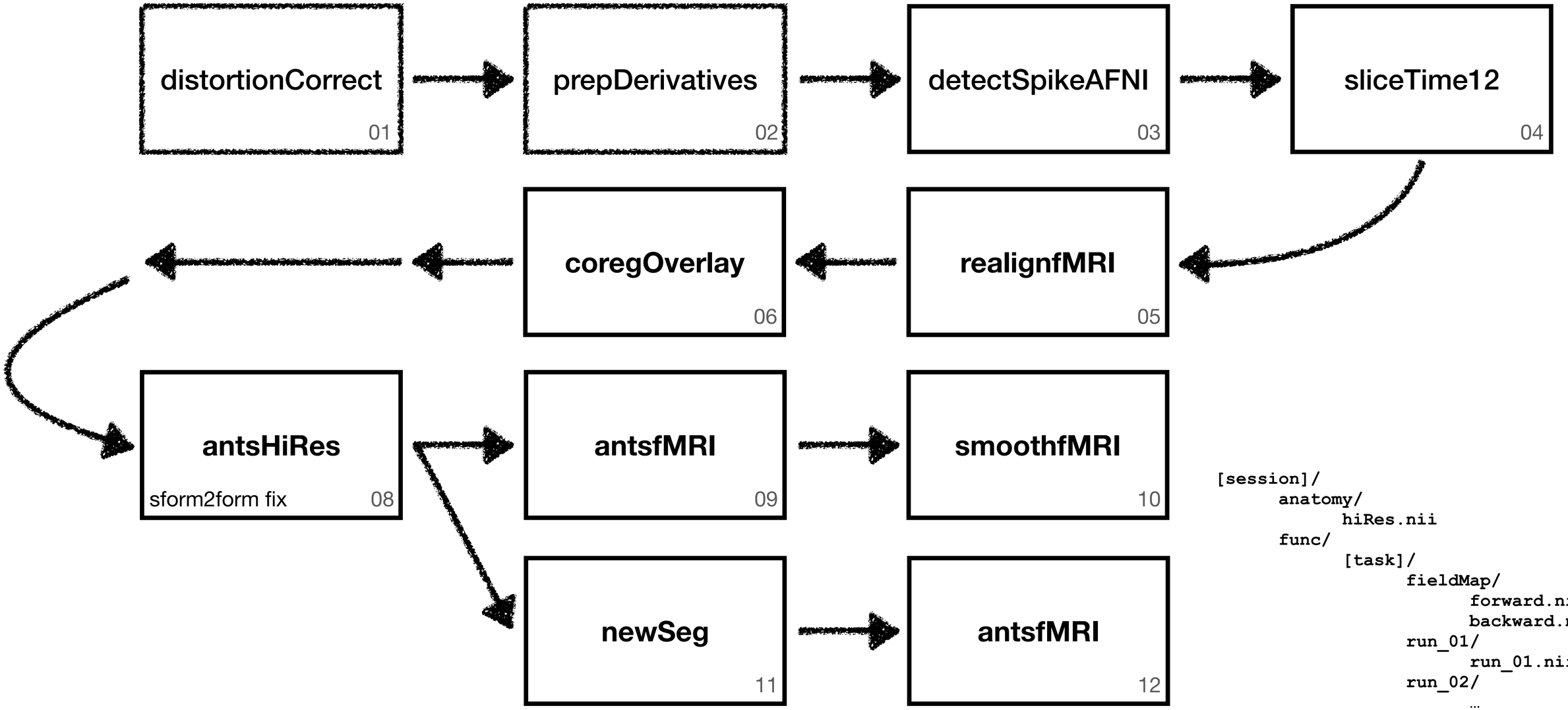

```
[session]/
      anatomy/
            loRes.nii
            hiRes.nii
      func/
            [task]/
                  fieldMap/
                        forward.nii
                        backward.nii
                  run_01/
                        run_01.nii
                  run_02/
                        …
```

spm12Batch System
spm12Batch processing pipeline without overlay image using ANTs normalization




```
[session]/
      anatomy/
            hiRes.nii
      func/
            [task]/
                  fieldMap/
                        forward.nii
                        backward.nii
                  run_01/
                        run_01.nii
                  run_02/
                        …
```

spm12Batch System
spm12Batch processing pipeline without overlay image using ANTs normalization and SPM segmentation for CSF/WM




```
[session]/
        anatomy/
                hiRes.nii
        func/
                [task]/
                        fieldMap/
                                forward.nii
                                backward.nii
                        run_01/
                                run_01.nii
                        run_02/
                                …
```

spm12Batch System
Note on derivatives folder

Most commands will operate on data and put resulting data back into the same tree. However, the earlier stages can be directed to output data to a new parent tree.

distortationCorrect can take input data with **-M** flag and put into a new tree with the **-MO** flag.

prepDeratives will copy input data with **-M** flag and put into a new tree with the **-MO** flag. This is good for copying over anatomy/ folder.

despikeAFNI will copy input data with **-M** flag and put into a new tree with the **-MO** flag.

If you've used **distortionCorrect** with the **-MO** flag, then for **despikeAFNI** you will need to specify both **-M** and **-MO** with the same parameter since **distortionCorrect** has already put output into new tree, and **despikeAFNI** will need to know that. You have to also specify the **-MO** to ensure you're putting data were you expect it to go. Examine the examples of how this works.

ImagingData/Subjects
- [subject]/
- [subject]/
- …

ImagingData/SubjectsDerived
- [subject]/
- [subject]/
- …

spm12Batch expected organization

```
[session]/
    anatomy/
        hiRes.nii
    func/
        [task]/
            fieldMap/
                forward.nii
                backward.nii
            run_01/
                run_01.nii
            run_02/
                …
```

```
[session]/
    anatomy/
        hiRes.nii
    func/
        [task]/
            fieldMap/
                forward.nii
                backward.nii
            run_01/
                run_01.nii
            run_02/
                …
            coReg/
                ANTs/
```

```
[session]/
    anatomy/
        hiRes.nii
    func/
        [task]/
            fieldMap/
                forward.nii
                backward.nii
            run_01/
                run_01.nii
            run_02/
                …
            coReg/
                newSeg/
```

```
[session]/
    anatomy/
        hiRes.nii
    func/
        [task]/
            fieldMap/
                forward.nii
                backward.nii
            run_01/
                run_01.nii
            run_02/
                …
            coReg/
                ANTs/
                    newSeg/
```

spm12Batch System
Daisy chaine example

```
#!/bin/bash

DATEStart=`date`
DATE=`date`
echo ${DATE} Step 01
distortionCorrect -B -MO ImagingData/SubjectsDerived -bi FM_AP -fi FM_PA -fm fieldMap12 -f func/Rest -v run_01 5502_02
DATE=`date`
echo ${DATE} Step 01
distortionCorrect -B -MO ImagingData/SubjectsDerived -bi FM_AP -fi FM_PA -fm fieldMap12 -f func/Rest -v run_02 5502_02
DATE=`date`
echo ${DATE} Step 02
prepDerivatives -B -MO ImagingData/SubjectsDerived -inc anatomy 5502_02
DATE=`date`
echo ${DATE} Step 03
despikeAFNI -B -M ImagingData/SubjectsDerived -MO ImagingData/SubjectsDerived -f func/Rest -on ds_ -v dc_run 5502_02
DATE=`date`
echo ${DATE} Step 04
sliceTimeMB -B -M ImagingData/SubjectsDerived  -v ds_dc_run -f func/Rest 5502_02
DATE=`date`
echo ${DATE} Step 05
realignfMRI12 -B -M ImagingData/SubjectsDerived -v aMBds_dc_run -f func/Rest 5502_02
DATE=`date`
echo ${DATE} Step 06
coregOverlay -B -M ImagingData/SubjectsDerived/ -o mprage -f func/rest -v r12aMBds_dc_run 5502_02
DATE=`date`
echo ${DATE} Step 07
antsHiRes -a func/Rest/coReg -B -h mprage -w func/Rest/coReg/ANTs_New -M ImagingData/SubjectsDerived/ 5502_02
DATE=`date`
echo ${DATE} Step 08
antsfMRI -B -f func/Rest -h mprage -M ImagingData/SubjectsDerived -on ants -v r12aMBds_dc_ -w coReg/ANTs_New 5502_02
DATE=`date`
echo ${DATE} Step 09
smoothfMRI -B -f func/Rest -M ImagingData/SubjectsDerived -v antsr12aMBds_dc_ 5502_02
DATE=`date`
echo ${DATE} Step 10
newSeg -B -a func/Rest/coReg/ANTs_New -h N4_mprage -w func/Rest/coReg/ANTs_New/newSeg -M ImagingData/SubjectsDerived/ 5502_02
DATE=`date`
echo ${DATE} Step 11
antsfMRI -B -h mprage -M ImagingData/SubjectsDerived -f func/Rest -w coReg/ANTs_New -v c2N4_mprage -in ants -on WM_ants_ 5502_02
DATE=`date`
echo ${DATE} Step 12
antsfMRI -B -h mprage -M ImagingData/SubjectsDerived -f func/Rest -w coReg/ANTs_New -v c3N4_mprage -in ants -on CSF_ants_ 5502_02
DATEStop=`date`

echo
echo Process is all completed and ran from

echo ${DATEStart}
echo ${DATEStop}
```

Tells spm12Batch to run the command actually in the foreground

because the 4 runs of resting state are under “Rest/“, but have different field maps, then the runs my be explicitly corrected one at a time.

copy the anatomy over to the SubjectsDerived/ tree

despike the data

slice time correct the data

motion correct the data

coregister the mprage to the motion corrected data.

using ANTs to calculate the warp to normalized space

using ANTs to take the BOLD data to normal space

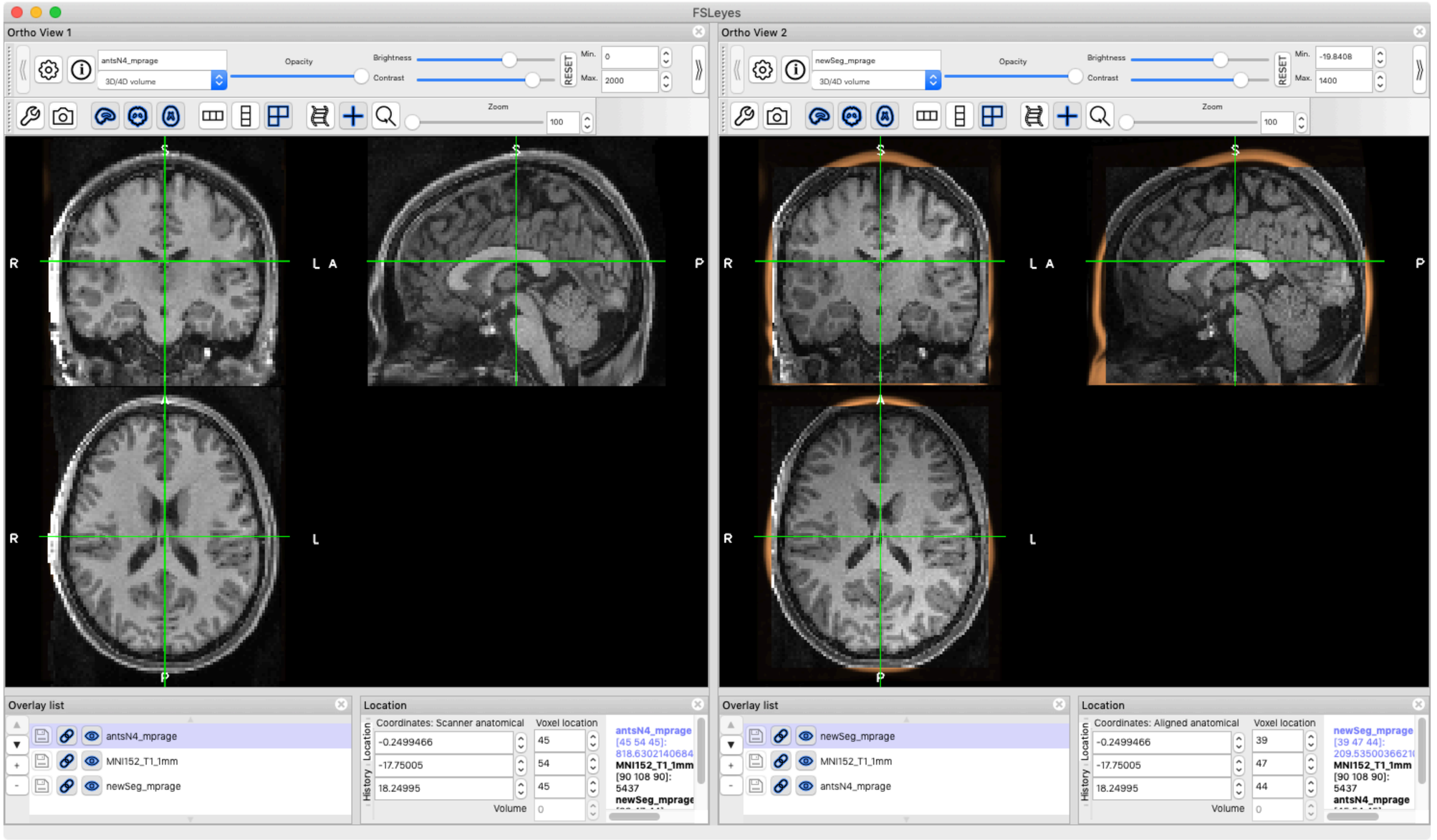
smooth the data

use SPM to do a segmentation, but in the ANTs tree

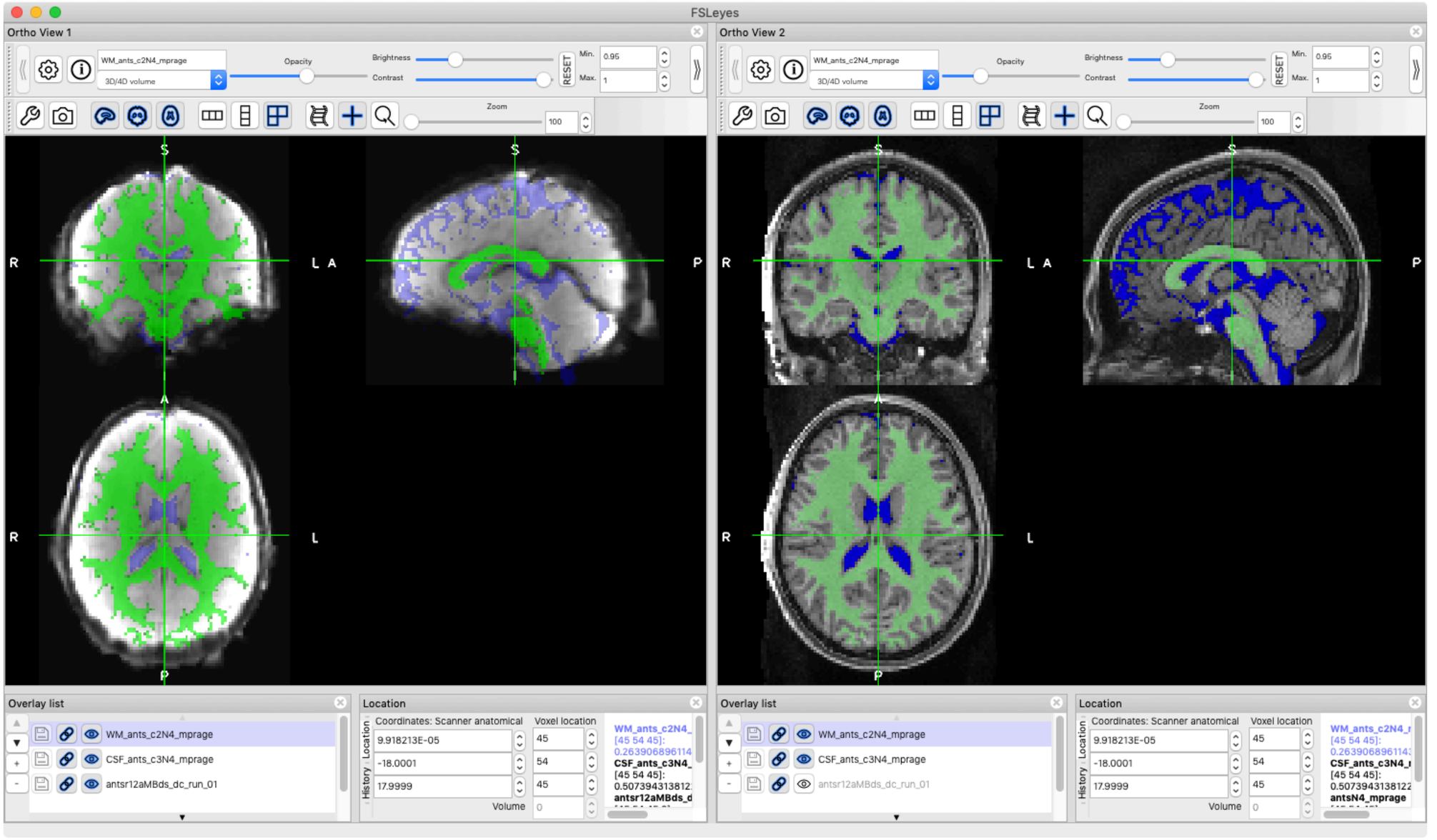
use ANTs to normalize the WM segment from SPM

use ANTs to normalize the CSF segment from SPM

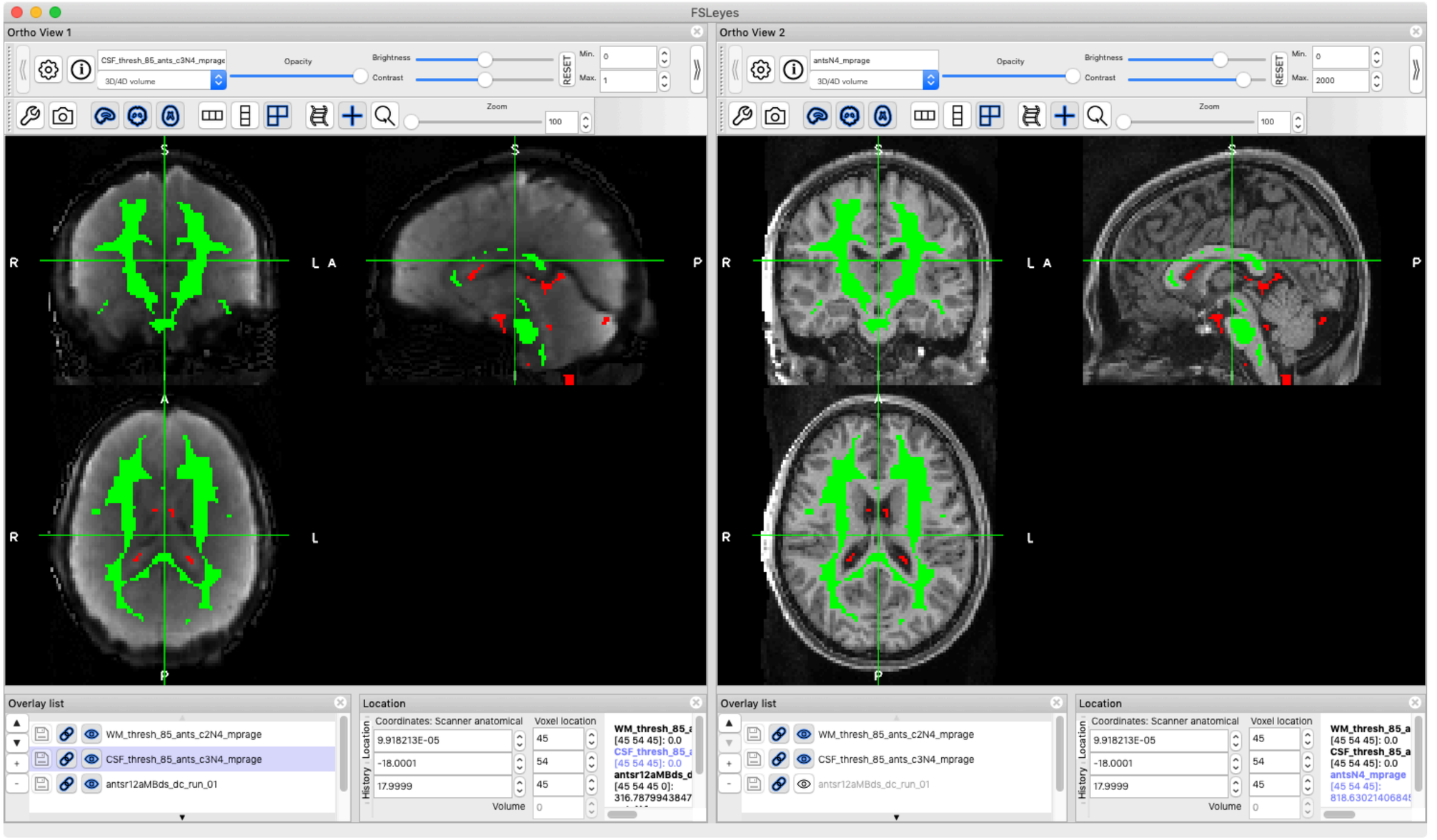
ANTs versus SPM normalization



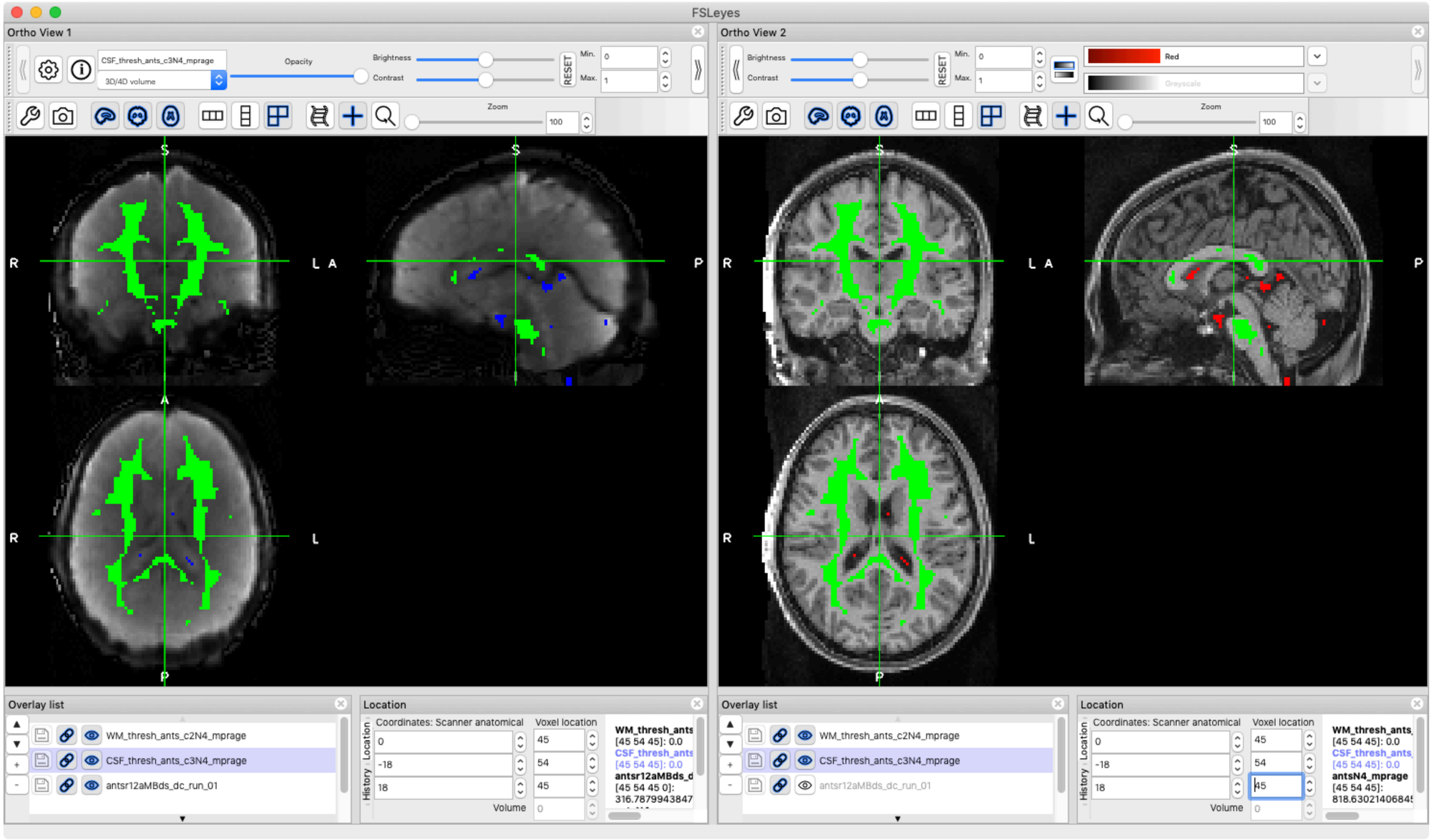
ANTs normalization with SPM tissue segmentation



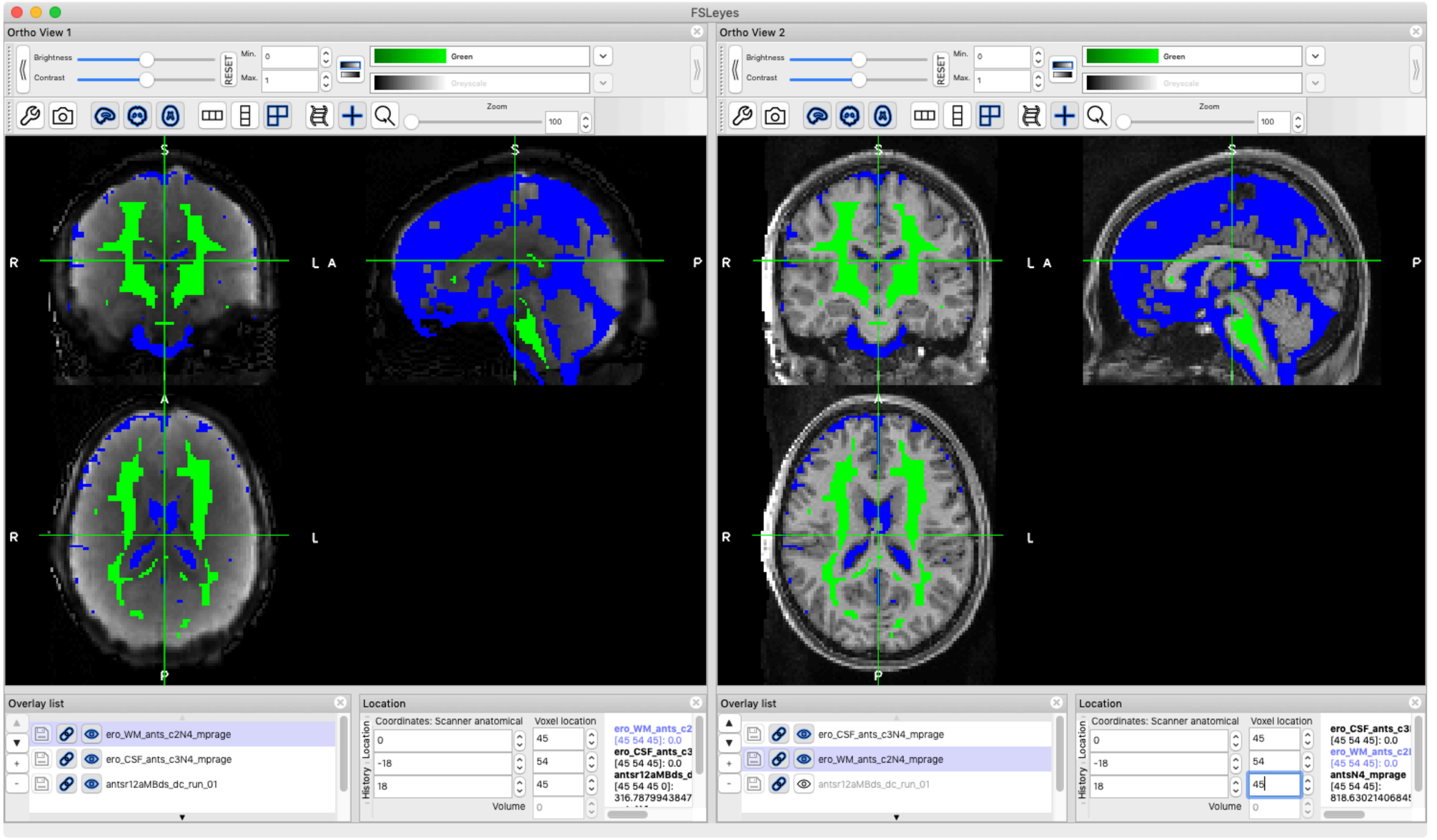
ANTs normalization with SPM tissue segmentation (thresholded at .85, binarized, and eroded 1)



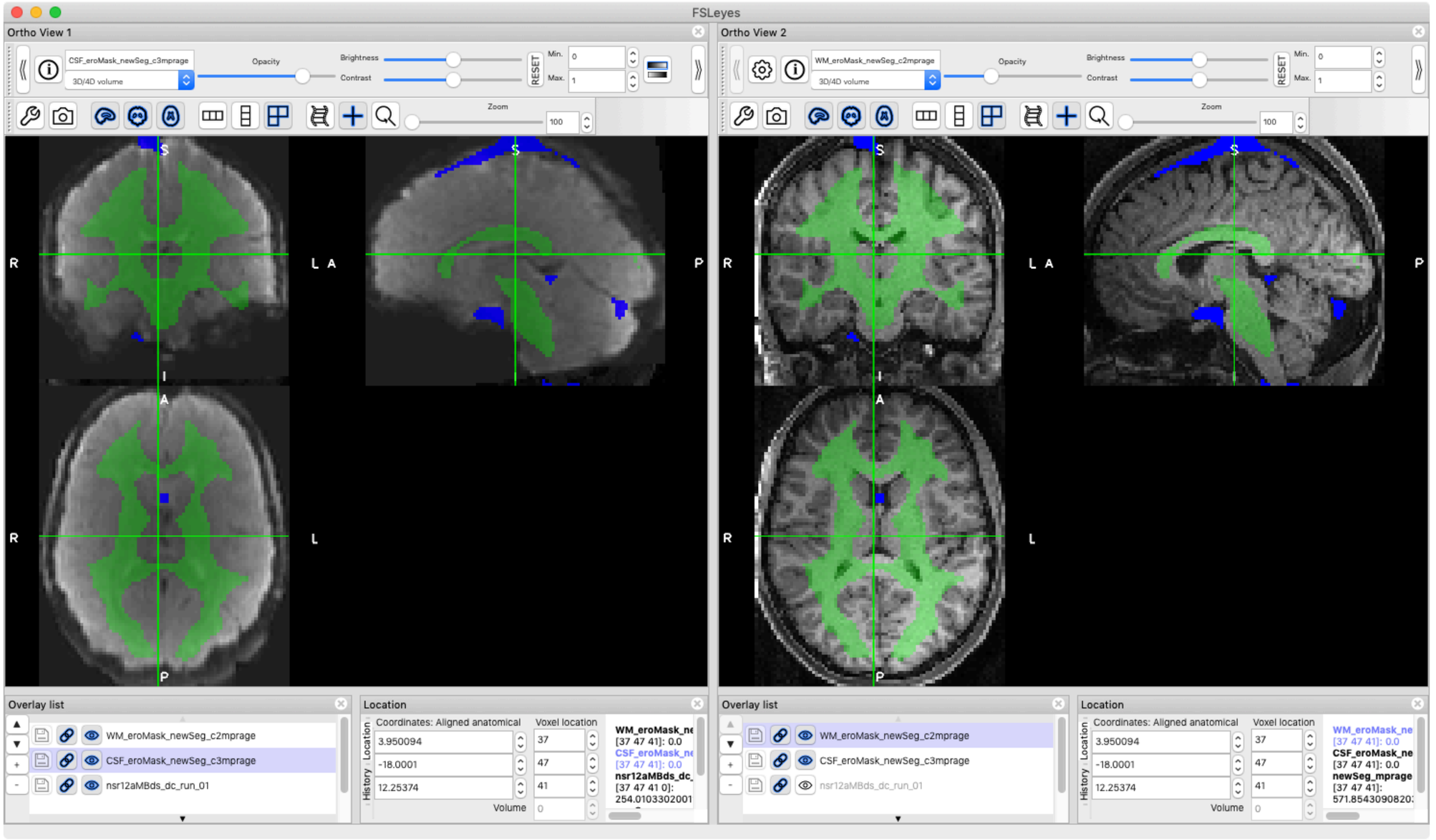
ANTs normalization with SPM tissue segmentation (thresholded at .95, binarized, and eroded 1)



ANTs normalization with SPM tissue segmentation (binarized, then eroded — twice for WM)



SPM normalization with SPM tissue segmentation



ANTs versus SPM normalization