ADNI Preprocessing Annotation and Tutorial

April 29, 2020

1 Softwares needed for preprocessing

- SPM12: Downloaded and available in Bernoulli at /array/ADNI/scripts/lib/spm12
- REST: Downloaded and available in Bernoulli at /array/ADNI/scripts/lib/REST_V1.8_130615
- Dicom to Nifti conversion codes: Downloaded and available in Bernoulli at /array/ADNI/scripts/lib/xiangruili-dicm2nii

  One needs to add these softwares/codes to MATLAB's path to be used by the actual preprocessing codes.

2 Downloaded Data

- fMRI: Downloaded images and metadata are available at /array/ADNI/FMRI
- MRI: Downloaded images and metadata are available at /array/ADNI/MRI

Downloading scripts for downloading ADNI data are available at /array/ADNI/scripts/downloading/ and subject specific metadata is available in this folder in the RData file fmriMetadata.RData and mriMetadata.RData.

3 Preprocessing Codes

All codes are available at /array/ADNI/scripts/preprocessing_v2/. The main codes are:

- adni_trialv2.m: Contains all the preprocessing steps for one single scan
- processSingleImage.m: Executes the processing for one single scan
- process_script.m: Executes the processing for all the scans

Rest of the codes are the modules to be used in the different steps of adni_trialv2.m. Just need to run the code process_script.m in server.
Make sure the folder tmp in /array/ADNI/scripts/preprocessing_v2/ is empty.

4 Output

The preprocessed images are all stored in: array/ADNI/processed/FMRI/processed_v2/. In this folder, the following folders are available:

- buck10_cube_avg: for each scan, a 10×136 dimensional time series whose rows correspond to the average BOLD signal of a $3 \times 3 \times 3$ cube around each of the 10 Buckner hub seed voxels.
- buck10_seed: for each scan, a 10×136 dimensional time series whose rows correspond to the BOLD signal from each of the 10 Buckner hubs.
- buck20_seed: for each scan, a 20×136 dimensional time series whose rows correspond to the BOLD signal from each of the 20 Buckner hubs.
- buck20_cube_avg: for each scan, a 20×136 dimensional time series whose rows correspond to the average BOLD signal of a $3 \times 3 \times 3$ cube around each of the 20 Buckner hub seed voxels.
- normalized: for each scan, the normalized versions are stored after applying all the previous steps till normalization as described in Section 6
- smoothed: for each scan, the smoothed scans are stored after normalization
- struct_files: for each scan, the structural file is stored which contains the normalized structural image, the whole brain mask and the tissue masks, resampled to the resolution of fMRI images and thresholded.
- motion_files: for each scan, this contains the motion parameters that are output from Motion Correction step in the preprocessing pipeline
- nuissance_covariates: for each scan, this contains the motion parameters, the average white matter and CSF signals and the linear trend which must be regressed out after ROI signal extraction.

These are the output from the preprocessing codes. Other folders present in array/ADNI/processed/FMRI/ were most likely created by users for their own research purpose and are not tied to preprocessing.

5 Resources for learning preprocessing steps

- `https://projecteuclid.org/euclid.ss/1242049389`
- `https://www.ernohermans.com/wp-content/uploads/2016/09/spm12_startersguide.pdf`
- `http://www.sbirc.ed.ac.uk/cyril/download/DTP_fMRI-preprocessing.pdf`
- `https://www.slideshare.net/paul_kyeong/fmri-preprocessing-steps-in-spm8`

6 Tutorial Illustration to Understand the Preprocessing Steps in MATLAB

The following is the illustration of the steps of the preprocessing on the scan with image ID 303069 corresponding to the subject 002_S_0295 which are stored in the folder 303069 locally on my computer.

6.1 Step 1

Add the necessary softwares described in Section 1 to MATLAB's path file.

6.2 Step 2

Set the paths of the folder containing the fMRI images and structural image and load the hub seed voxels

1. Create a root directory and name it rootdir. In my case:

   ```
   rootdir='~/Library/Mobile Documents/com~apple~CloudDocs/ADNIanalysis/ADNI2';
   ```

2. Inside root directory place the folder 303069. Set dcmdir to be the path of the this folder. In my case:

   ```
   dcmdir='/Users/pdubey/Library/Mobile Documents/com~apple~CloudDocs/ADNIanalysis
   /ADNI2/303069/ADNI/002_S_0295/Resting_State_fMRI/2012-05-10_15_42_37.0/S150058';
   ```

3. Specify the location of the structural file in structfile. In my case:

   ```
   structfile=strcat(rootdir,'/308078/ADNI/002_S_0295/MT1__N3m/2012-05-10_15_44_50.0/
   S10055/ADNI_002_S_0295_MR_MT1__N3m_Br_20120605092714994_S150055_I308078.nii');
   ```

4. Specify path of the normalization template in norm_template contained in spm12.

   ```
   norm_template='/Users/pdubey/Library/Mobile
   Documents/com~apple~CloudDocs/ADNIanalysis/spm12/tpm/TPM.nii';
   ```

5. Load the seed voxel locations in MATLAB. They can be found in the file hubs.mat which is available in /array/ADNI/scripts/preprocessing/

   ```
   load('/Users/pdubey/Library/Mobile
   Documents/com~apple~CloudDocs/ADNIanalysis/ADNI2/hubs.mat')
   ```

6.3 Step 3

Convert images in DICOM format to NIFTI in order to be used by SPM. Sometimes DICOM might be multiframe. This code accounts for both.

```
cd(rootdir)
mkdir 'temp1'
%% Read the data from the dicom folder
dcmRaw=dir(dcmdir);
filenames=cell(length(dcmRaw)-2,1);
for i=1:(length(dcmRaw)-2)
filenames{i}=strcat(strcat(dcmdir,'/'),dcmRaw(i+2).name);
end

%% Decomposing dcm to nifti files
if length(filenames) > 2

dicm2nii(dcmdir, strcat(rootdir,'/temp1'), '.nii 3D')

%% Extracting nifti and json files

mkdir 'NIIready'
niiList=dir(fullfile(strcat(rootdir,'/temp1'),'*.nii'));
A={niiList.name}';
for i=1:length(A)
copyfile(strcat(strcat(rootdir,'/temp1/'),A{i}),strcat(rootdir,'/NIIready'));
end

else
mkdir 'temp2'
outdir=strcat(rootdir,'/temp2');
hdr = spm_dicom_headers(filenames{2},'true');
spm_dicom_convert(hdr,'all','flat','nii',outdir)
nii4D=dir(fullfile(outdir,'*.nii'));
A={nii4D.name}';

matlabbatch = cell(1,1);
matlabbatch{1} = struct();
matlabbatch{1}.spm.util.split.vol = {strcat(strcat(outdir,'/'),A{1})};
matlabbatch{1}.spm.util.split.outdir = {strcat(rootdir,'/temp1')};
spm('defaults', 'FMRI');
spm_jobman('run', matlabbatch);


mkdir 'NIIready'
niiList=dir(fullfile(strcat(rootdir,'/temp1'),'*.nii'));
```

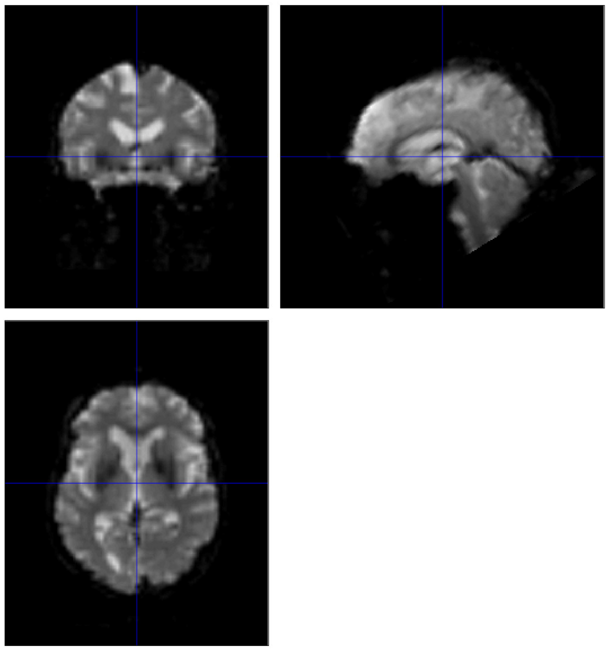
Figure 1: This is how the starting NIFTI files look. The visualization can be obtained by typing 'spm' in the command line of MATLAB and the opening the NIFTI file using the 'DISPLAY' option that pops up.

```
A={niiList.name}';
for i=1:length(A)
copyfile(strcat(strcat(rootdir,'/temp1/'),A{i}),strcat(rootdir,'/NIIready'));
end
rmdir('temp2','s')
end

rmdir('temp1','s')
```

6.4 Step 3

Next step is to reorient structural image and removing the skull from the structural image.

```
%% Reorient structrural images and skullstrip them

mkdir 'struct_stripped'
```

```
%reorient the image
nii_setOrigin(structfile,1)

%skull stripping
matlabbatch = cell(1,1);
matlabbatch{1} = struct();
matlabbatch{1}.spm.tools.MRI.MRTool_brain.res_dir =
{strcat(rootdir,'/struct_stripped')};
matlabbatch{1}.spm.tools.MRI.MRTool_brain.t1w = {structfile};
spm('defaults', 'FMRI');
spm_jobman('run', matlabbatch);

%find the location of the masked image in struct_stripped

cd(rootdir)
mkdir struct_new
cd(strcat(rootdir,'/struct_stripped'))
file=dir('*masked.nii');
file=strcat(rootdir,'/struct_stripped/',{file.name});
movefile(file{1},strcat(rootdir,'/struct_new'))
file=dir('*mask.nii');
file=strcat(rootdir,'/struct_stripped/',{file.name});
movefile(file{1},strcat(rootdir,'/struct_new'))

cd(rootdir)
```

6.5 Step 3.5

We reorient the functional images (set common origin) before proceeding with the rest of the steps of preprocessing. As is conventional, we remove the first 4 scans before preprocessing the functional images.

```
%% Discard the first 4 functional images before preprocessing
niiList=dir(fullfile(strcat(rootdir,'/NIIready'),'*.nii') );
p={niiList.name};
s_list=fullfile(strcat(rootdir,'/NIIready/'),p{1});
for i=2:length(p)
s_list=strvcat(s_list,fullfile(strcat(rootdir,'/NIIready/'),p{i}));
end
cd(rootdir)
nii_setOrigin(s_list,4)
```

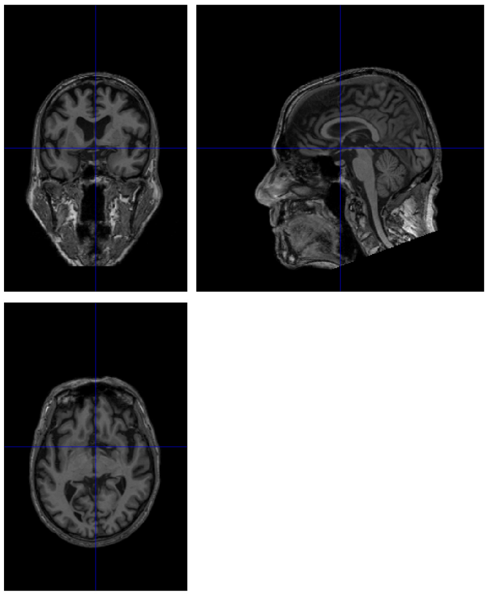

Figure 2: This is how the structural image looks like.

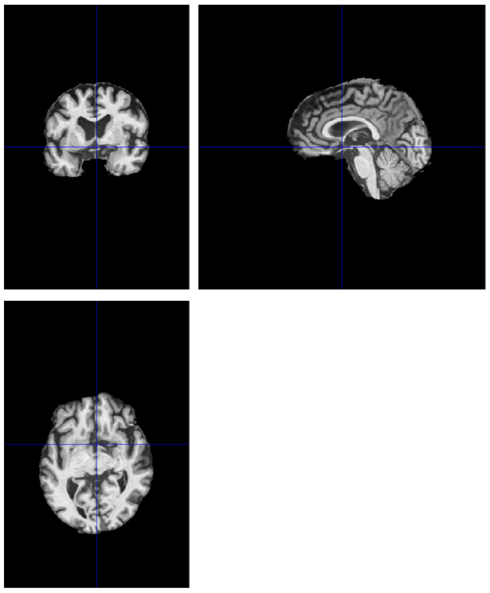

Figure 3: This is how the skull stripped structural image looks like.

```
cd(strcat(rootdir,'/NIIready'))
niiList=dir(fullfile(strcat(rootdir,'/NIIready'),'*.nii') );
p={niiList.name};
for i=1:4
delete(p{i})
end
cd(rootdir)
%now there are 136 fmri images which will be preprocessed
```

6.6 Step 4

We perform segmentation to get brain tissue maps and generate whole brain mask. The brain mask is obtained using the mask that is generated after the skull stripping processing and then thresholding it.

```
%% Segmentation of Brain tissues and generating whole brain mask

cd(strcat(rootdir,'/struct_new'))
stripped=dir('*masked.nii');
stripped=strcat(strcat(rootdir,'/struct_new/'),stripped.name);
cd(rootdir)


matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
matlabbatch{1}.spm.spatial.preproc.channel.vols = {stripped};
matlabbatch{1}.spm.spatial.preproc.channel.biasreg = 0.001;
matlabbatch{1}.spm.spatial.preproc.channel.biasfwhm = 60;
matlabbatch{1}.spm.spatial.preproc.channel.write = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(1).tpm = {strcat(norm_template,',1')};
matlabbatch{1}.spm.spatial.preproc.tissue(1).ngaus = 1;
matlabbatch{1}.spm.spatial.preproc.tissue(1).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(1).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(2).tpm = {strcat(norm_template,',2')};
matlabbatch{1}.spm.spatial.preproc.tissue(2).ngaus = 1;
matlabbatch{1}.spm.spatial.preproc.tissue(2).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(2).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(3).tpm = {strcat(norm_template,',3')};
matlabbatch{1}.spm.spatial.preproc.tissue(3).ngaus = 2;
matlabbatch{1}.spm.spatial.preproc.tissue(3).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(3).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(4).tpm = {strcat(norm_template,',4')};
matlabbatch{1}.spm.spatial.preproc.tissue(4).ngaus = 3;
```

```
matlabbatch{1}.spm.spatial.preproc.tissue(4).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(4).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(5).tpm = {strcat(norm_template,',5')};
matlabbatch{1}.spm.spatial.preproc.tissue(5).ngaus = 4;
matlabbatch{1}.spm.spatial.preproc.tissue(5).native = [1 0];
matlabbatch{1}.spm.spatial.preproc.tissue(5).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(6).tpm = {strcat(norm_template,',6')};
matlabbatch{1}.spm.spatial.preproc.tissue(6).ngaus = 2;
matlabbatch{1}.spm.spatial.preproc.tissue(6).native = [0 0];
matlabbatch{1}.spm.spatial.preproc.tissue(6).warped = [0 0];
matlabbatch{1}.spm.spatial.preproc.warp.mrf = 1;
matlabbatch{1}.spm.spatial.preproc.warp.cleanup = 1;
matlabbatch{1}.spm.spatial.preproc.warp.reg = [0 0.001 0.5 0.05 0.2];
matlabbatch{1}.spm.spatial.preproc.warp.affreg = 'mni';
matlabbatch{1}.spm.spatial.preproc.warp.fwhm = 0;
matlabbatch{1}.spm.spatial.preproc.warp.samp = 3;
matlabbatch{1}.spm.spatial.preproc.warp.write = [0 0];
spm_jobman('run',matlabbatch);

cd(strcat(rootdir,'/struct_new'))
file0=dir('*mask.nii');
file0=strcat(rootdir,'/struct_new/',file0.name);

matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
matlabbatch{1}.spm.util.imcalc.input = {
file0
};
matlabbatch{1}.spm.util.imcalc.output = 'output';
matlabbatch{1}.spm.util.imcalc.outdir = {strcat(rootdir,'/struct_new')};
matlabbatch{1}.spm.util.imcalc.expression = 'i1>0';
matlabbatch{1}.spm.util.imcalc.var = struct('name', {}, 'value', {});
matlabbatch{1}.spm.util.imcalc.options.dmtx = 0;
matlabbatch{1}.spm.util.imcalc.options.mask = 0;
matlabbatch{1}.spm.util.imcalc.options.interp = 1;
matlabbatch{1}.spm.util.imcalc.options.dtype = 4;

spm_jobman('run',matlabbatch);

cd(rootdir)
```

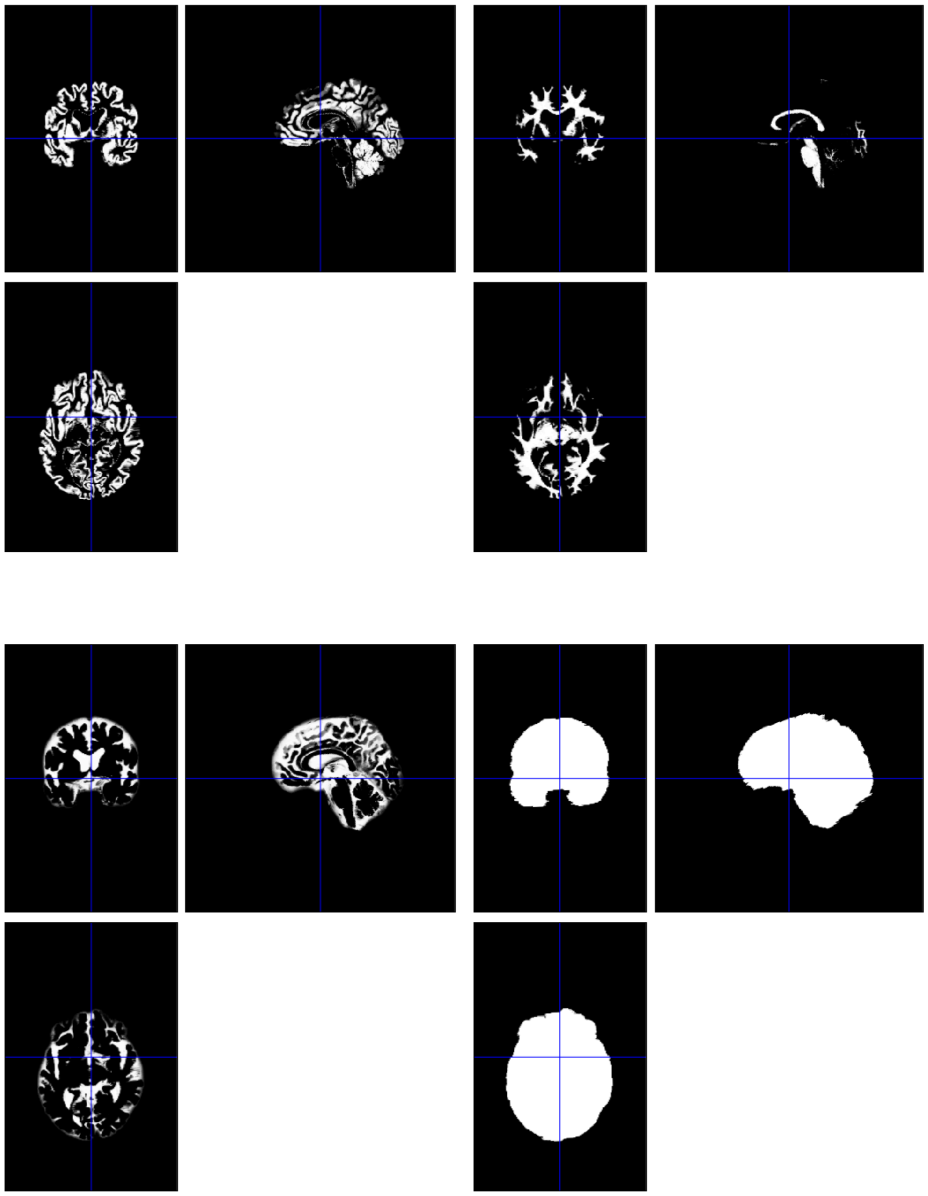

Figure 4: Gray matter, white matter, CSF and whole brain masks obtained after segmentation.

6.7 Step 5

We then carry out the following steps:

- Motion correction.
- Slice timing correction.
- Coregistration of the functional images with the structural image.
- Normalization of the structural image to the template and writing the corresponding transformation of the functional images, the tissue masks and the whole brain mask.
- Spatial smoothing using Gaussian kernel of the functional images.

```
%% Preprocessing Pipeline: Part I (Motion Correction,
Slice Timing Correction, Coregistration, Normalization)
niidir=fullfile(strcat(rootdir,'/NIIready'));
matlabbatch = cell(4,1);
matlabbatch{1} = struct(); matlabbatch{2} = struct();
matlabbatch{3} = struct(); matlabbatch{4} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
niiList = dir(fullfile(niidir,'*.nii'));
niiList = {niiList.name}';
niiList2 = strcat(strcat(niidir,'/'),strcat(niiList,',1'));
cd(strcat(rootdir,'/struct_new'))
%stripped=dir('ADNI*masked.nii');
%stripped=strcat(strcat(rootdir,'/struct_new/'),stripped.name);
mask=dir('output.nii');
mask=strcat(strcat(rootdir,'/struct_new/'),mask.name);
file0=dir('c1*.*');
file0=strcat(rootdir,'/struct_new/',file0.name);
file1=dir('c2*.*');
file1=strcat(rootdir,'/struct_new/',file1.name);
file2=dir('c3*.*');
file2=strcat(rootdir,'/struct_new/',file2.name);
cd(rootdir)




matlabbatch{1}.spm.spatial.realign.estwrite.data = {niiList2}';
matlabbatch{1}.spm.spatial.realign.estwrite.eoptions.quality = 0.9;
matlabbatch{1}.spm.spatial.realign.estwrite.eoptions.sep = 4;
matlabbatch{1}.spm.spatial.realign.estwrite.eoptions.fwhm = 5;
```

```
matlabbatch{1}.spm.spatial.realign.estwrite.eoptions.rtm = 1;
matlabbatch{1}.spm.spatial.realign.estwrite.eoptions.interp = 2;
matlabbatch{1}.spm.spatial.realign.estwrite.eoptions.wrap = [0 0 0];
matlabbatch{1}.spm.spatial.realign.estwrite.eoptions.weight = '';
matlabbatch{1}.spm.spatial.realign.estwrite.roptions.which = [2 1];
matlabbatch{1}.spm.spatial.realign.estwrite.roptions.interp = 4;
matlabbatch{1}.spm.spatial.realign.estwrite.roptions.wrap = [0 0 0];
matlabbatch{1}.spm.spatial.realign.estwrite.roptions.mask = 1;
matlabbatch{1}.spm.spatial.realign.estwrite.roptions.prefix = 'r';

niiList2 = strcat(strcat(strcat(niidir,'/r'),niiList),',1');

matlabbatch{2}.spm.temporal.st.scans = {niiList2}';
matlabbatch{2}.spm.temporal.st.nslices = 48;
matlabbatch{2}.spm.temporal.st.tr = 3;
matlabbatch{2}.spm.temporal.st.ta = 2.9375;
matlabbatch{2}.spm.temporal.st.so = [2 4 6 8 10 12 14 16 18 20 22
24 26 28 30 32 34 36 38 40 42 44 46 48 1 3 5 7 9 11 13 15 17 19 21
23 25 27 29 31 33 35 37 39 41 43 45 47];
matlabbatch{2}.spm.temporal.st.refslice = 1;
matlabbatch{2}.spm.temporal.st.prefix = 'a';

niiList2 = strcat(strcat(strcat(niidir,'/mean'),niiList{1}),',1');
matlabbatch{3}.spm.spatial.coreg.estwrite.ref = {stripped};
matlabbatch{3}.spm.spatial.coreg.estwrite.source = {niiList2};
niiList2 = strcat(strcat(strcat(niidir,'/ar'),niiList),',1');

matlabbatch{3}.spm.spatial.coreg.estwrite.other=niiList2;
matlabbatch{3}.spm.spatial.coreg.estwrite.eoptions.cost_fun = 'nmi';
matlabbatch{3}.spm.spatial.coreg.estwrite.eoptions.sep = [4 2];
matlabbatch{3}.spm.spatial.coreg.estwrite.eoptions.tol = [0.02 0.02
0.02 0.001 0.001 0.001 0.01 0.01 0.01 0.001 0.001 0.001];
matlabbatch{3}.spm.spatial.coreg.estwrite.eoptions.fwhm = [7 7];
matlabbatch{3}.spm.spatial.coreg.estwrite.roptions.interp = 0;
matlabbatch{3}.spm.spatial.coreg.estwrite.roptions.wrap = [0 0 0];
matlabbatch{3}.spm.spatial.coreg.estwrite.roptions.mask = 0;
matlabbatch{3}.spm.spatial.coreg.estwrite.roptions.prefix = 'r';

matlabbatch{4}.spm.spatial.normalise.estwrite.subj.vol = {stripped};
niiList2 = strcat(strcat(strcat(niidir,'/rar'),niiList),',1');
matlabbatch{4}.spm.spatial.normalise.estwrite.subj.resample =
[niiList2; stripped; file0; file1; file2; mask];
matlabbatch{4}.spm.spatial.normalise.estwrite.eoptions.biasreg = 0.0001;
matlabbatch{4}.spm.spatial.normalise.estwrite.eoptions.biasfwhm = 60;
matlabbatch{4}.spm.spatial.normalise.estwrite.eoptions.tpm = {norm_template};
```

```
matlabbatch{4}.spm.spatial.normalise.estwrite.eoptions.affreg = 'mni';
matlabbatch{4}.spm.spatial.normalise.estwrite.eoptions.reg = [0 0.001 0.5 0.05 0.2];
matlabbatch{4}.spm.spatial.normalise.estwrite.eoptions.fwhm = 0;
matlabbatch{4}.spm.spatial.normalise.estwrite.eoptions.samp = 3;
matlabbatch{4}.spm.spatial.normalise.estwrite.woptions.bb = [-78 -112 -70
78 76 85];
matlabbatch{4}.spm.spatial.normalise.estwrite.woptions.vox = [2 2 2];
matlabbatch{4}.spm.spatial.normalise.estwrite.woptions.interp = 4;
matlabbatch{4}.spm.spatial.normalise.estwrite.woptions.prefix = 'w';

spm_jobman('run',matlabbatch);

% Moving normalized images to a different folder for rest of preprocessing
% pipeline
mkdir normalized
Nimages=dir( fullfile(niidir,'w*.*') );
Nimages={Nimages.name};
Nimages = strcat(niidir,'/',Nimages);
for j = 1:length(Nimages)
copyfile(Nimages{j},strcat(rootdir,'/normalized'));
end

motion=dir( fullfile(niidir,'rp*.*'));
motion = strcat(niidir,'/',motion.name);
copyfile(motion,rootdir);
motion=dir( fullfile(rootdir,'rp*.*'));
movefile(strcat(rootdir,'/',motion.name),strcat(rootdir,'/motion.txt'));


matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
Nimages=dir( fullfile(strcat(rootdir,'/normalized'),'w*.*') );
Nimages={Nimages.name};
s_list=cell(136,1);
for i=1:length(Nimages)
s_list{i}=strcat(fullfile(strcat(rootdir,'/normalized/')), Nimages{i});
end

matlabbatch{1}.spm.spatial.smooth.data = s_list;
matlabbatch{1}.spm.spatial.smooth.fwhm = [4 4 4];
matlabbatch{1}.spm.spatial.smooth.dtype = 0;
matlabbatch{1}.spm.spatial.smooth.im = 0;
matlabbatch{1}.spm.spatial.smooth.prefix = 's';
```

```
spm_jobman('run',matlabbatch);


mkdir smoothed
Nimages=dir( fullfile(strcat(rootdir,'/normalized'),'s*.*') );
Nimages={Nimages.name};
Nimages = strcat(strcat(rootdir,'/normalized'),'/',Nimages);
for j = 1:length(Nimages)
copyfile(Nimages{j},strcat(rootdir,'/smoothed'));
end
```

6.8 Step 6

Resampling and thresholding the masks: We resample all the normalized masks (the tissue masks and also the whole brain mask) from the last step to the resolution of the fMRI images. We also threshold the white matter and the CSF masks at higher probabilities (0.75) to make sure that we regress out only the white matter and CSF signals with greater confidence.

```
%% Resampling to functional images

cd(strcat(rootdir,'/struct_new'))
file0=dir('wc1*.*');
file0=strcat(rootdir,'/struct_new/',file0.name);
file1=dir('wc2*.*');
file1=strcat(rootdir,'/struct_new/',file1.name);
file2=dir('wc3*.*');
file2=strcat(rootdir,'/struct_new/',file2.name);
file3=dir('woutput.nii');
file3=strcat(rootdir,'/struct_new/',file3.name);

refImages=dir(fullfile(strcat(rootdir,'/normalized'),'w*.*'));
refList={refImages.name};
ref=strcat(rootdir,'/normalized/',refList{1});


matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
matlabbatch{1}.spm.spatial.coreg.estwrite.ref = {ref};
matlabbatch{1}.spm.spatial.coreg.estwrite.source = {file0};
matlabbatch{1}.spm.spatial.coreg.estwrite.other = {
file0
file1
```

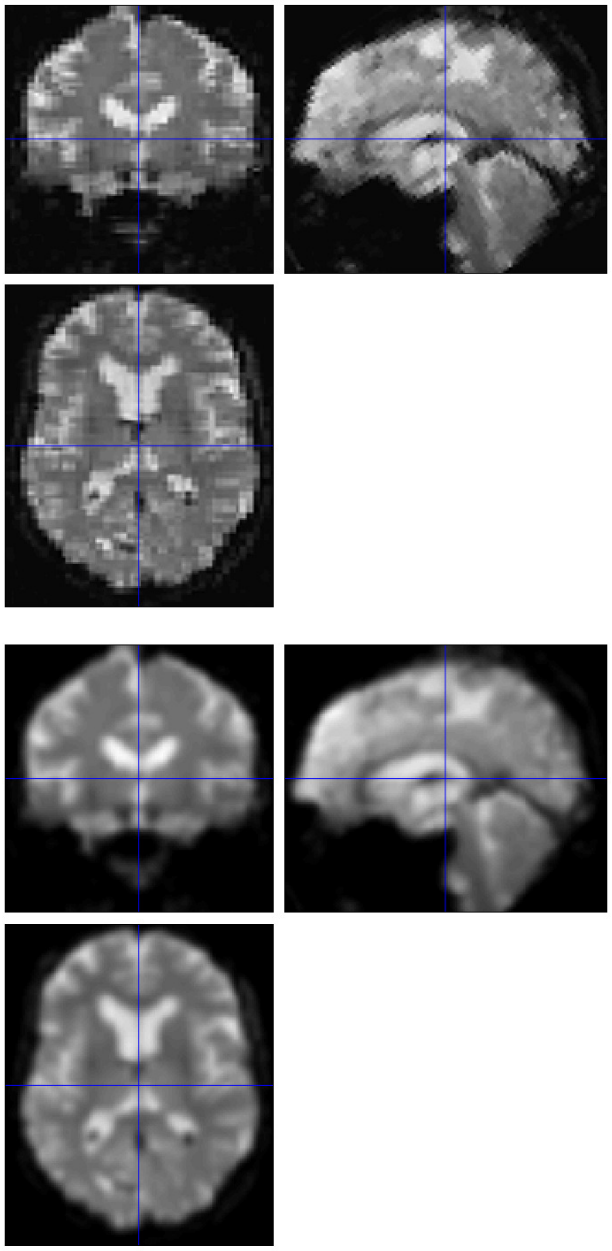

Figure 5: The top one is how the normalized image looks like and the bottom one is the smoothed image.

```
file2
file3
};
matlabbatch{1}.spm.spatial.coreg.estwrite.eoptions.cost_fun = 'nmi';
matlabbatch{1}.spm.spatial.coreg.estwrite.eoptions.sep = [4 2];
matlabbatch{1}.spm.spatial.coreg.estwrite.eoptions.tol = [0.02 0.02
0.02 0.001 0.001 0.001 0.01 0.01 0.01 0.001 0.001 0.001];
matlabbatch{1}.spm.spatial.coreg.estwrite.eoptions.fwhm = [7 7];
matlabbatch{1}.spm.spatial.coreg.estwrite.roptions.interp = 0;
matlabbatch{1}.spm.spatial.coreg.estwrite.roptions.wrap = [0 0 0];
matlabbatch{1}.spm.spatial.coreg.estwrite.roptions.mask = 0;
matlabbatch{1}.spm.spatial.coreg.estwrite.roptions.prefix = 're';


spm_jobman('run',matlabbatch);

cd(rootdir)


%% Thresholding the masks

cd(strcat(rootdir,'/struct_new'))
file1=dir('rewc2*.*');
file1=strcat(rootdir,'/struct_new/',file1.name);
file2=dir('rewc3*.*');
file2=strcat(rootdir,'/struct_new/',file2.name);
file3=dir('rewoutput.nii');
file3=strcat(rootdir,'/struct_new/',file3.name);

matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
matlabbatch{1}.spm.util.imcalc.input = {file1};
matlabbatch{1}.spm.util.imcalc.output = 'white';
matlabbatch{1}.spm.util.imcalc.outdir = {strcat(rootdir,'/struct_new')};
matlabbatch{1}.spm.util.imcalc.expression = 'i1>0.75';
matlabbatch{1}.spm.util.imcalc.var = struct('name', {}, 'value', {});
matlabbatch{1}.spm.util.imcalc.options.dmtx = 0;
matlabbatch{1}.spm.util.imcalc.options.mask = 0;
matlabbatch{1}.spm.util.imcalc.options.interp = 1;
matlabbatch{1}.spm.util.imcalc.options.dtype = 4;

spm_jobman('run',matlabbatch);
```

```
matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
matlabbatch{1}.spm.util.imcalc.input = {file2};
matlabbatch{1}.spm.util.imcalc.output = 'csf';
matlabbatch{1}.spm.util.imcalc.outdir = {strcat(rootdir,'/struct_new')};
matlabbatch{1}.spm.util.imcalc.expression = 'i1>0.75';
matlabbatch{1}.spm.util.imcalc.var = struct('name', {}, 'value', {});
matlabbatch{1}.spm.util.imcalc.options.dmtx = 0;
matlabbatch{1}.spm.util.imcalc.options.mask = 0;
matlabbatch{1}.spm.util.imcalc.options.interp = 1;
matlabbatch{1}.spm.util.imcalc.options.dtype = 4;

spm_jobman('run',matlabbatch);

cd(rootdir)

matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
matlabbatch{1}.spm.util.imcalc.input = {file3};
matlabbatch{1}.spm.util.imcalc.output = 'wb';
matlabbatch{1}.spm.util.imcalc.outdir = {strcat(rootdir,'/struct_new')};
matlabbatch{1}.spm.util.imcalc.expression = 'i1>0.1';
matlabbatch{1}.spm.util.imcalc.var = struct('name', {}, 'value', {});
matlabbatch{1}.spm.util.imcalc.options.dmtx = 0;
matlabbatch{1}.spm.util.imcalc.options.mask = 0;
matlabbatch{1}.spm.util.imcalc.options.interp = 1;
matlabbatch{1}.spm.util.imcalc.options.dtype = 4;

spm_jobman('run',matlabbatch);

cd(rootdir)
```

6.9 Step 6

Next we extract the seed voxel signals and the average of a $3 \times 3 \times 3$ cube around the seed voxel. This signal is then detrended, followed by regressing out motion parameters and the white matter and CSF signals. Finally we carry out bandpass filtering.

```
%% Preprocessing Pipeline: Part II (Find the covariates to be regressed out)
```

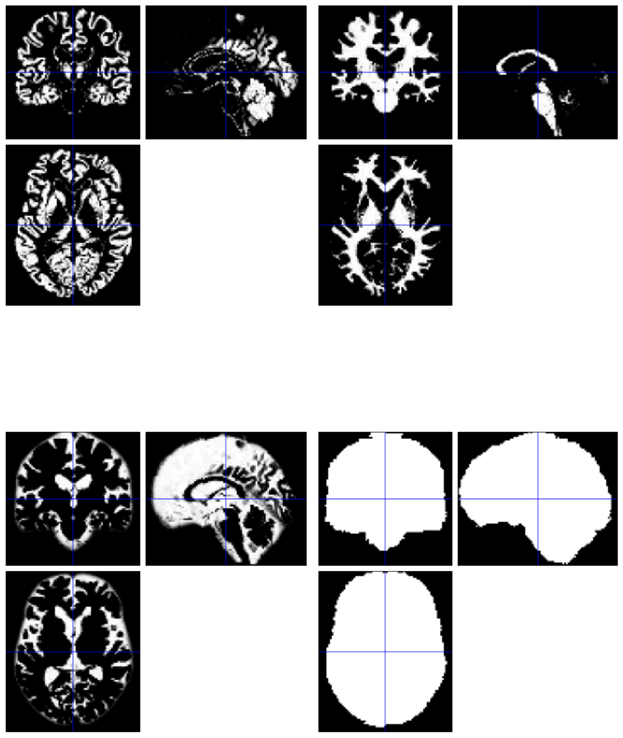

Figure 6: This is how the final masks looks like.

```
DataDir=strcat(rootdir,'/smoothed');
cd(strcat(rootdir,'/struct_new'))
file1=dir('white.nii');
file1=strcat(rootdir,'/struct_new/',file1.name);
file2=dir('csf.nii');
file2=strcat(rootdir,'/struct_new/',file2.name);
cd(rootdir)

cd(strcat(rootdir,'/NIIready'))
file4=dir('rp*.*');
file4=strcat(rootdir,'/NIIready/',file4.name);
cd(rootdir)

WM_CSF_masks=cell(2,1);
WM_CSF_masks{1}=file1;
WM_CSF_masks{2}=file2;


CovariablesDef=struct;
CovariablesDef.polort=1;
CovariablesDef.ort_file=file4;
CovariablesDef.CovMask=WM_CSF_masks;

[AllVolume,~,~,~,~] =rest_to4d(DataDir);
[~,~,~,nDim4]=size(AllVolume);

theCovariables=[];
if isfield(CovariablesDef,'polort')
thePolOrt=[];
if CovariablesDef.polort>=0
thePolOrt =(1:nDim4)';
thePolOrt =repmat(thePolOrt, [1, (1+CovariablesDef.polort)]);
for x=1:(CovariablesDef.polort+1)
thePolOrt(:, x) =thePolOrt(:, x).^(x-1) ;
end
end
theCovariables =[theCovariables,thePolOrt];
end

if isfield(CovariablesDef,'ort_file')
if exist(CovariablesDef.ort_file, 'file')==2
theCovariablesFromFile =load(CovariablesDef.ort_file);
theCovariables =[theCovariables,theCovariablesFromFile];
else
```

```
warning(sprintf('\n\nCovariables definition text file "%s" doesn''t
exist, please check! This covariables will not be regressed out this time.',
CovariablesDef.ort_file));
end
end



if isfield(CovariablesDef,'CovMask')
for iMask=1:length(CovariablesDef.CovMask)
[iMaskData,~,~]=rest_readfile(CovariablesDef.CovMask{iMask});

TempTC=reshape(AllVolume,[],nDim4);
TempTC = mean(TempTC(find(iMaskData),:))';

theCovariables=[theCovariables,TempTC];
end
end

covariables=theCovariables;

%Can add more covariates here

save('cova.mat','covariables');

%% Extracting hub data, regressing out the covariates and applying
bandpass filter for the window [0.01, 0.1]

matlabbatch = cell(1,1);
matlabbatch{1} = struct();
spm('defaults', 'fmri');
spm_jobman('initcfg');
Nimages=dir( fullfile(strcat(rootdir,'/smoothed'),'s*.*') );
Nimages={Nimages.name};
s_list=cell(136,1);
for i=1:length(Nimages)
s_list{i}=strcat(fullfile(strcat(rootdir,'/smoothed/')), Nimages{i});
end

matlabbatch{1}.spm.util.cat.vols = s_list;
matlabbatch{1}.spm.util.cat.name = 'smoothed4D.nii';
matlabbatch{1}.spm.util.cat.dtype = 0;

spm_jobman('run',matlabbatch);
```

```
[Data ,Header]=y_Read(fullfile(strcat(rootdir,'/smoothed/smoothed4D.nii')),'all');
voxels10=inv(Header.mat)*[buck10hubs; ones(1,10)]; voxels10=round(voxels10(1:3,:));
voxels20=inv(Header.mat)*[buck20hubs; ones(1,20)]; voxels20=round(voxels20(1:3,:));
buck10=zeros(10,136); buck20=zeros(20,136);


for i=1:10
temp=reshape(Data(voxels10(1,i),voxels10(2,i),voxels10(3,i),:),1,[]);
[~,~,buck10(i,:)]=regress(temp',covariables);
buck10(i,:)=y_IdealFilter(buck10(i,:)',3,[0.01,0.1]);
end
for i=1:20
temp=reshape(Data(voxels20(1,i),voxels20(2,i),voxels20(3,i),:),1,[]);
[~,~,buck20(i,:)]=regress(temp',covariables);
buck20(i,:)=y_IdealFilter(buck20(i,:)',3,[0.01,0.1]);
end


%3 X 3 X 3 cube around seed voxel
buck10_cube=cell(10,2); buck20_cube=cell(20,2);
buck10_cube_avg=zeros(10,136); buck20_cube_avg=zeros(20,136);
for i=1:10
v1=voxels10(1,i); v2=voxels10(2,i); v3=voxels10(3,i); c1=[v1-1 v1 v1+1];
c2=[v2-1 v2 v2+1];  c3=[v3-1 v3 v3+1];
k=1; temp=zeros(27,3);
for r=1:3
for s=1:3
for t=1:3
temp(k,:)=[c1(r) c2(s) c3(t)];
k=k+1;
end
end
end
hub=cell(27,1);
for k=1:27
temp1(k,:)=reshape(Data(temp(k,1),temp(k,2),temp(k,3),:),1,[]);
[~,~,hub{k}]=regress(temp1(k,:)',covariables);
hub{k}=y_IdealFilter(hub{k}',3,[0.01,0.1]);
end
[~,~,buck10_cube_avg(i,:)]=regress(mean(temp1,1)',covariables);
buck10_cube_avg(i,:)=y_IdealFilter(buck10_cube_avg(i,:)',3,[0.01,0.1]);
buck10_cube{i,1}=hub;
buck10_cube{i,2}=temp;
end
```

```
for i=1:20
v1=voxels20(1,i); v2=voxels20(2,i); v3=voxels20(3,i); c1=[v1-1 v1 v1+1];
c2=[v2-1 v2 v2+1]; c3=[v3-1 v3 v3+1];
k=1; temp=zeros(27,3);
for r=1:3
for s=1:3
for t=1:3
temp(k,:)=[c1(r) c2(s) c3(t)];
k=k+1;
end
end
end
hub=cell(27,1);
for k=1:27
temp1(k,:)=reshape(Data(temp(k,1),temp(k,2),temp(k,3),:),1,[]);
[~,~,hub{k}]=regress(temp1(k,:)',covariables);
hub{k}=y_IdealFilter(hub{k}',3,[0.01,0.1]);
end
[~,~,buck20_cube_avg(i,:)]=regress(mean(temp1,1)',covariables);
buck20_cube_avg(i,:)=y_IdealFilter(buck20_cube_avg(i,:)',3,[0.01,0.1]);
buck20_cube{i,1}=hub;
buck20_cube{i,2}=temp;
end


%% Saving Extracted Data

save('buck10_seed.mat','buck10');
save('buck20_seed.mat','buck20');
save('buck10_cube.mat','buck10_cube');
save('buck20_cube.mat','buck20_cube');
save('buck10_cubeAvg.mat','buck10_cube_avg');
save('buck20_cubeAvg.mat','buck20_cube_avg');
```