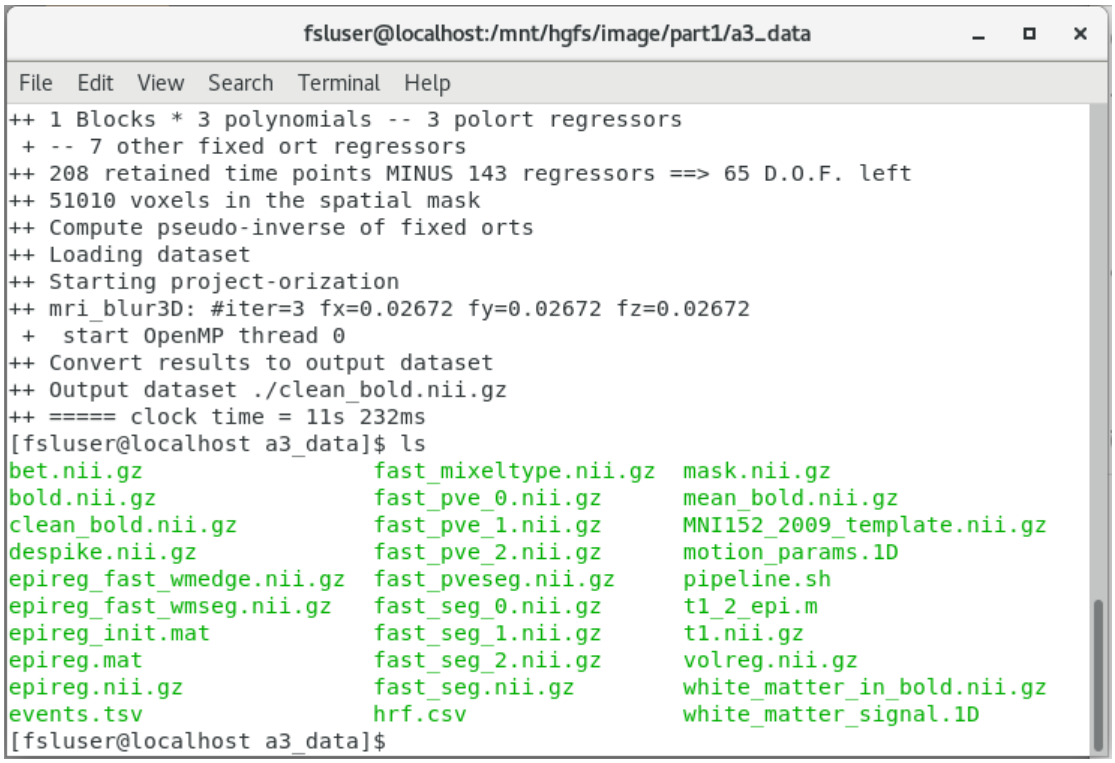
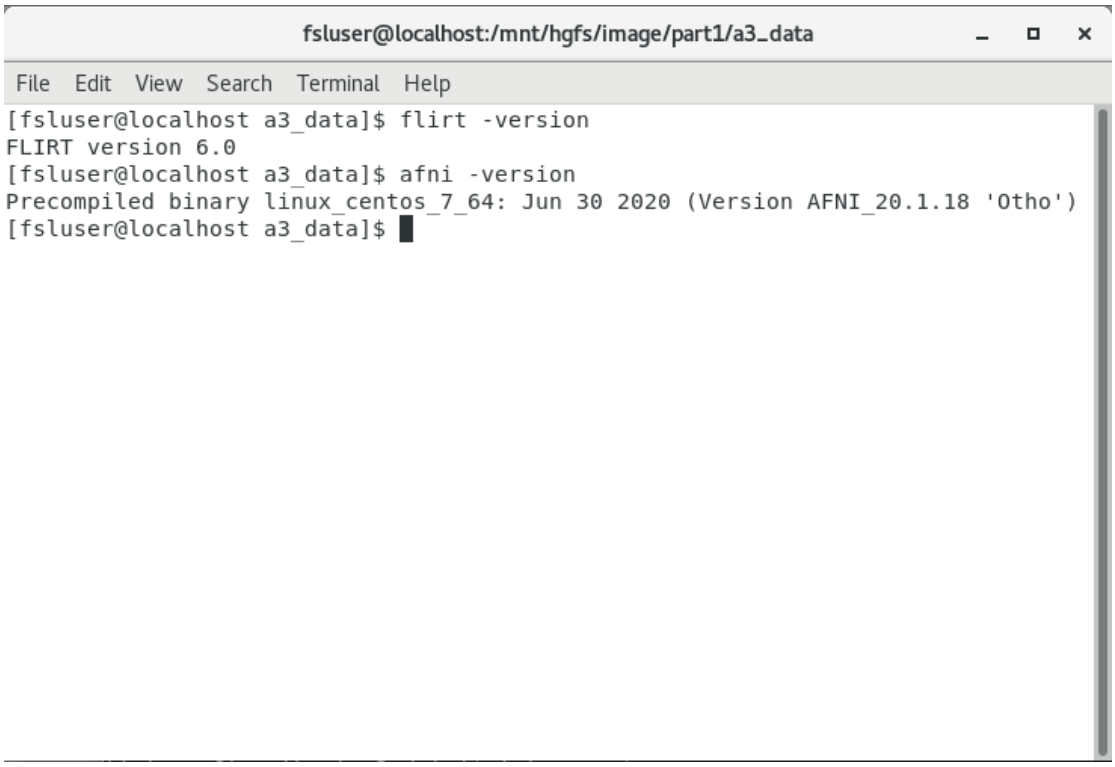
CS516 Assignment3 Report

Part 1：

We use centos operating system and VMware to do this part, here is the screenshot of my directory after running 'pipeline.sh' and typing 'ls' at the command line:



Here is the version of FSL and AFNI:



Part2:

We take the code from lecture 10 slide15, and change someplace to keep program run correct in our computer, here are the changes:

Load hrf file:

```
hrf=pd.read_csv('I:/image/part1/a3_data/hrf.csv',header=None).to_numpy()
```

Calculate convolve signal:

```
conved = signal.convolve(ts,hrf.reshape(-1),mode='full')   //we reshape hrf variable to keep it has same dimension with ts variable
```

Deal with nan value in corrs variable:

```
where_are_NaNs = np.isnan(corrs)
corrs[where_are_NaNs] = 0
```

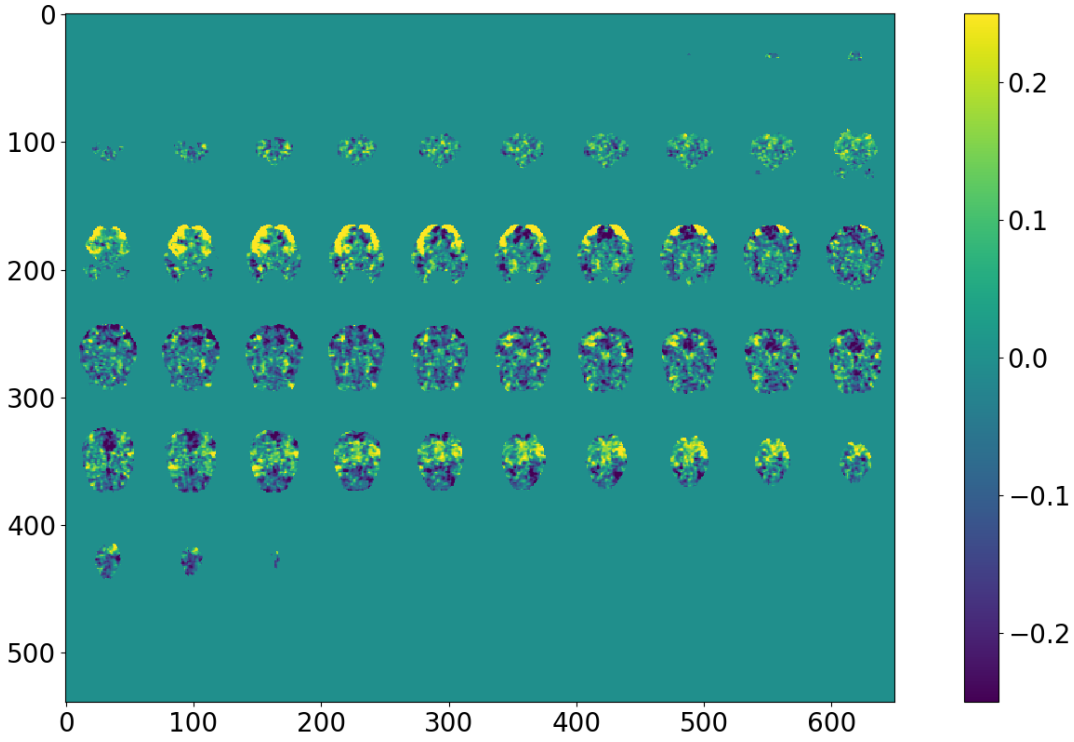
Visualize the correlation map:

```
corrs_all=np.zeros((77*7,65*10))
for i in range(0,corrs.shape[2]):
    h=i%10
    v=i//10
    corrs_all[v*77:v*77+77,h*65:h*65+65]=np.rot90(corrs[:,:,i])
plt.figure()
plt.imshow(corrs_all)
plt.colorbar()
```

here is our result:

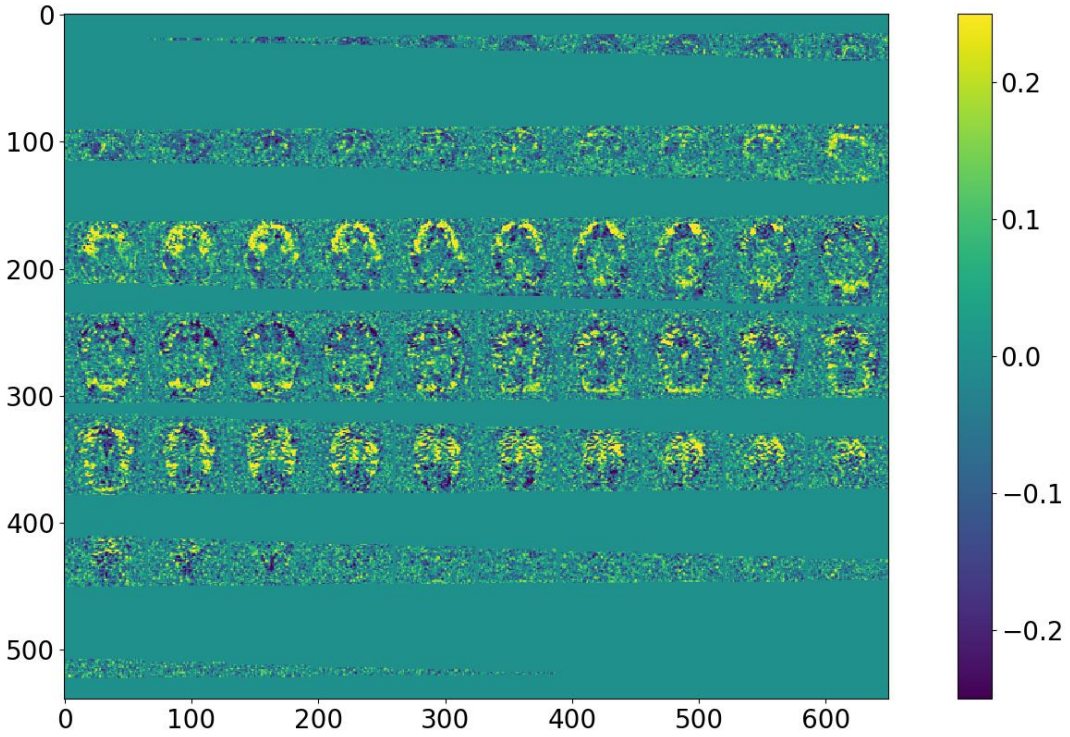
2a:

The image below shows all z-slices(total 67 slices) from the final activation map, there are 10 slices in one lines and there are 7 lines, from this figure, we can see in 21-27 slices there are clusters near the back of the brain with high correlation values.



2b:

In 2b, we use bold.nii.gz to replace clean_bold.nii.gz, the image below shows our result, from this figure, we can see there are lots of noise in each slice, and compare with 2a's result, 2b's result seems blurred, and there are high correlation values around the edge of brain.

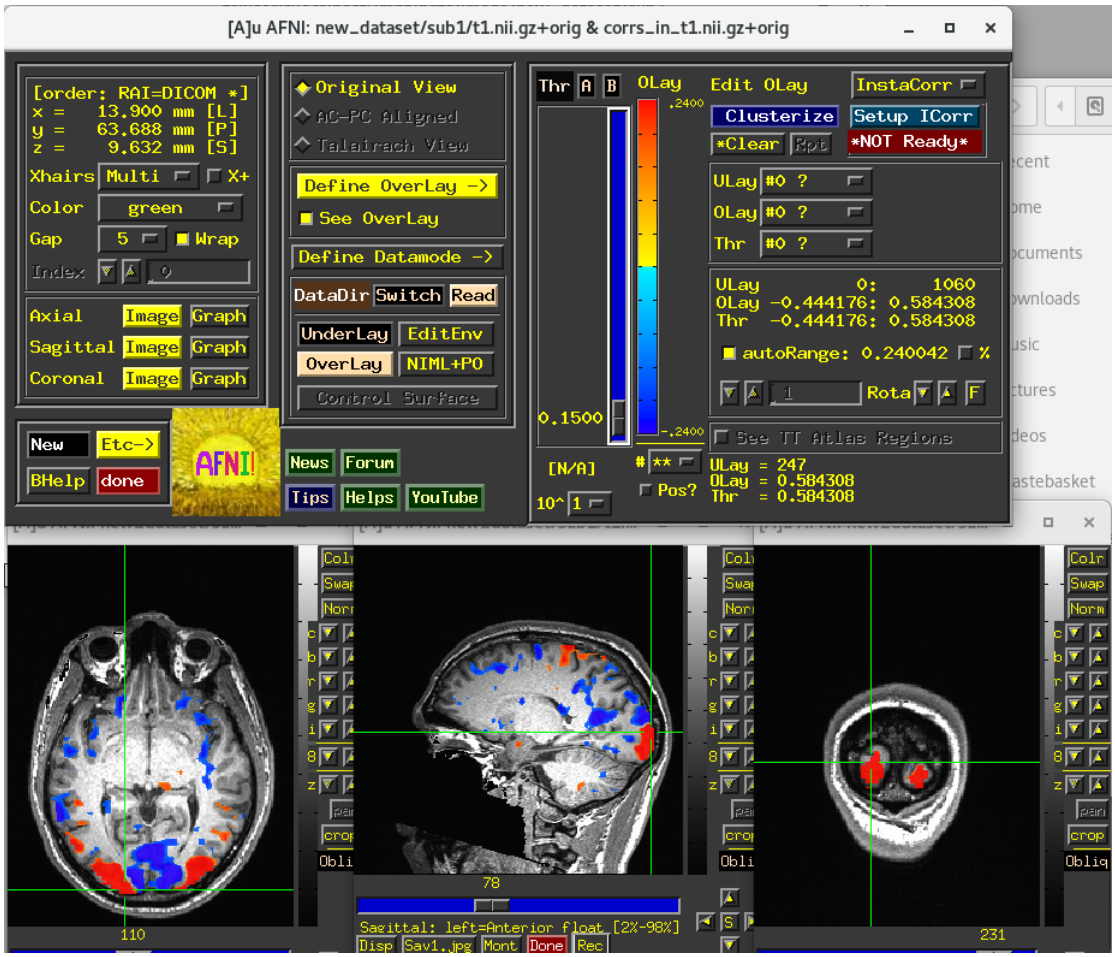


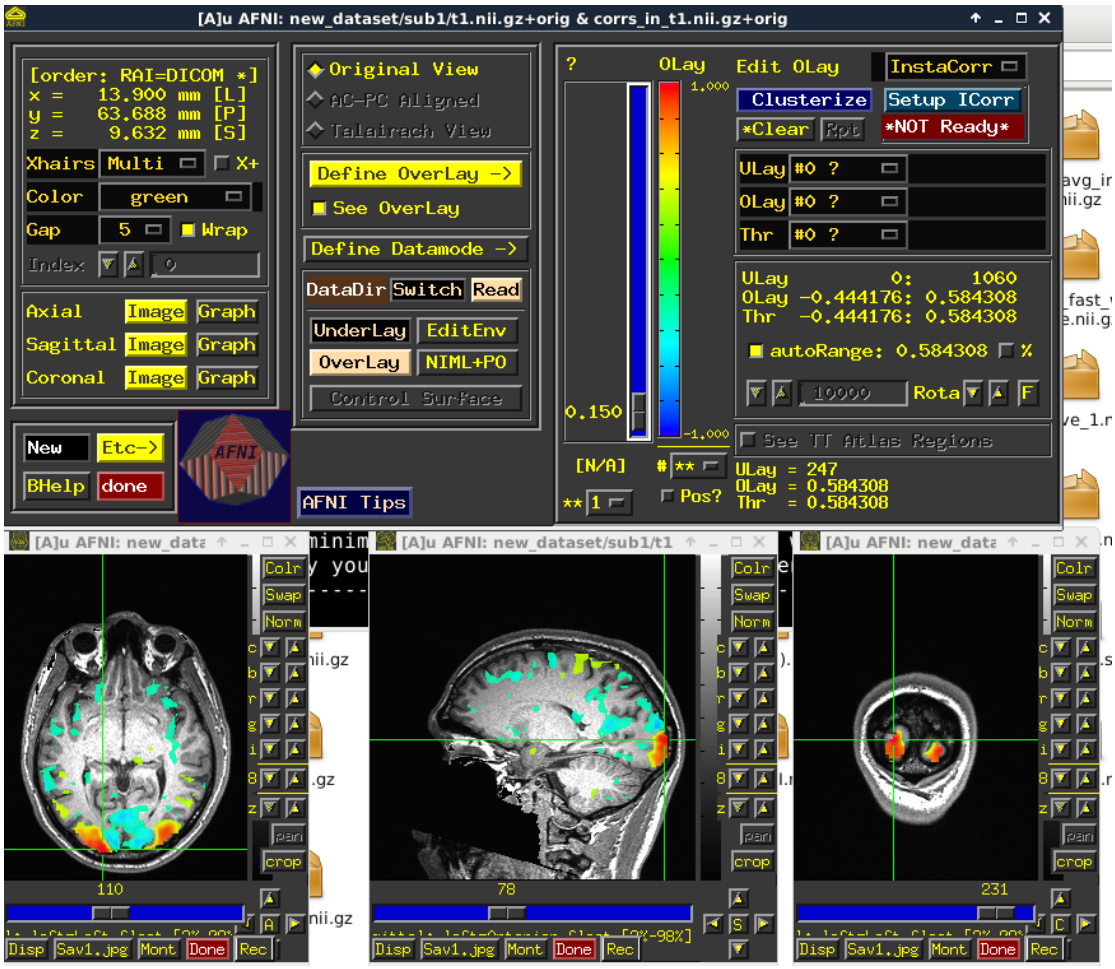
Comment on the difference between the correlation map produced with/without preprocessing. Are the correlations stronger or weaker when preprocessing is included? Why?

Answer: From 2a result (the correlation map produced with preprocessing) we can get the cluster with high correlation value, but 2b result (the correlation map produced without preprocessing) we can find high correlation value everywhere, even outside of the brain, so the correlation map produced with preprocessing has better result than the correlation map produced without preprocessing. And the correlation stronger when preprocessing is included because there are noises in correlation map without preprocessing, and those noise would influence the result of correlations, after we remove those noise, of course we can get a stronger correlation.

Part3:

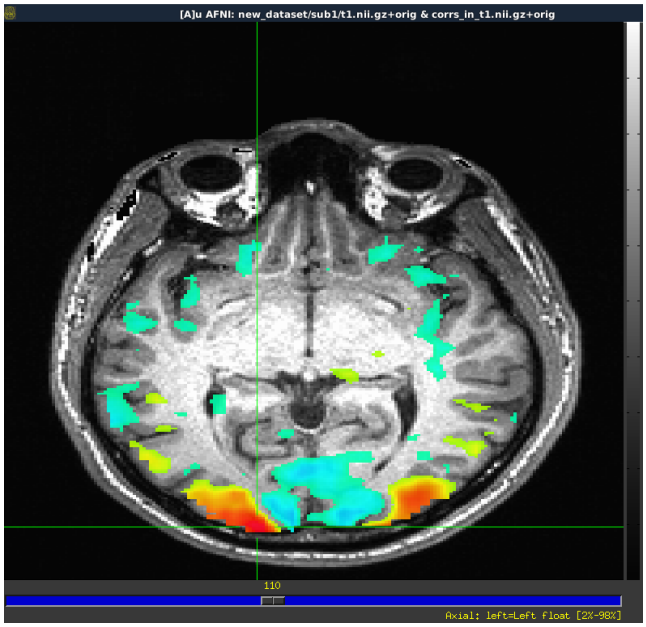
Because we use VMware and centos system rather than virtual box with NeuroDebian system, we install different version of FSL and AFNI, we found there are some different when we display result in new version AFNI, the result in old version AFNI is more clearly than new version AFNI, so we will use old version AFNI in NeuroDebian system to show our result. Here is an example to show the same result in different AFNI version.



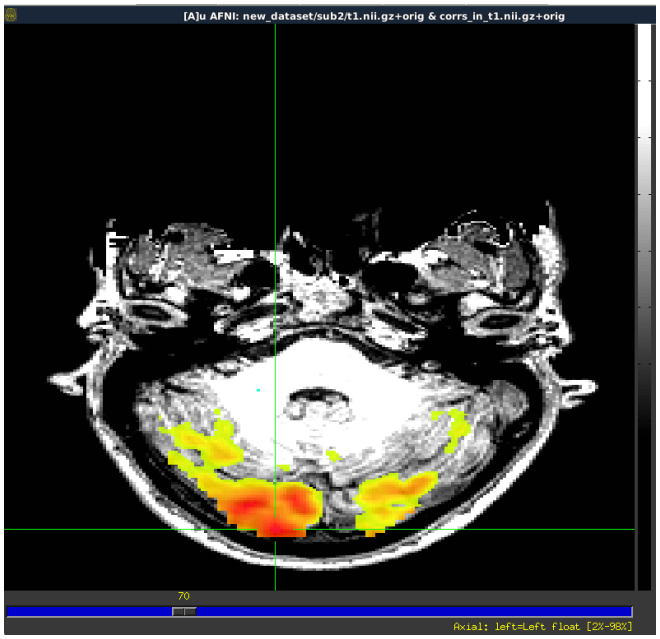


Here are the images of correlation maps overlayed on t1 in afni with a threshold of 0.15:(only show axial view)

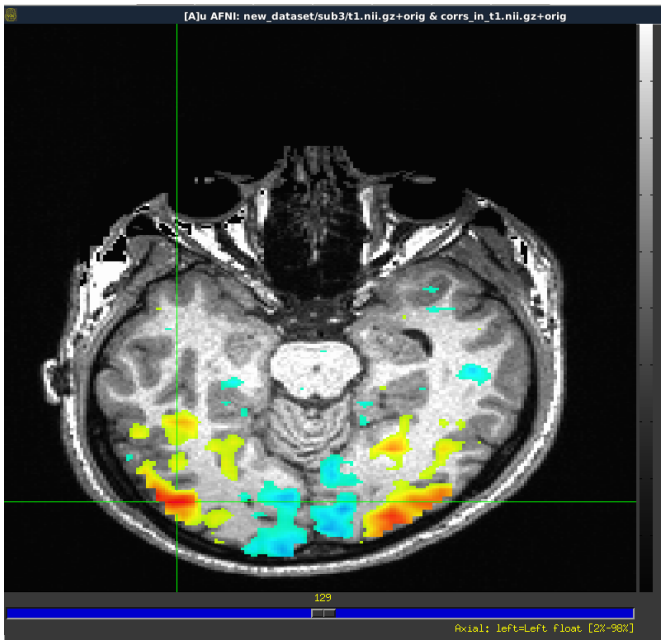
Sub1:



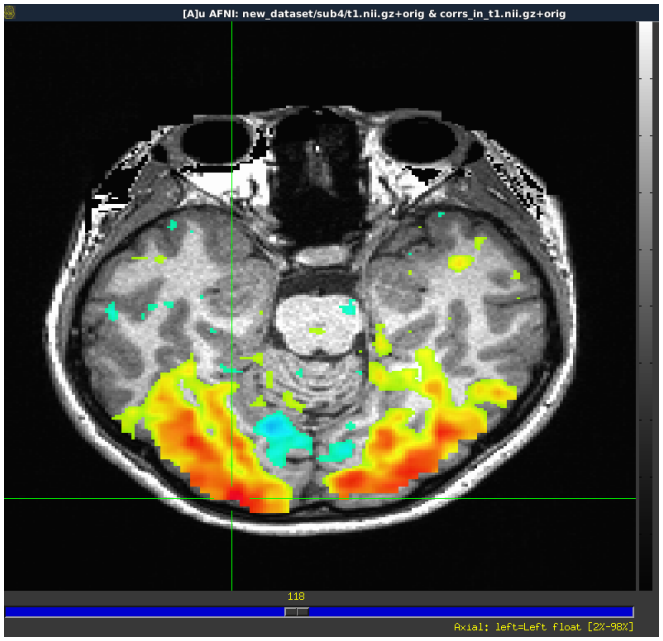
Sub2:


[A]u AFNI: new_dataset/sub2/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
70
Axial: left=Left float [2%-98%]

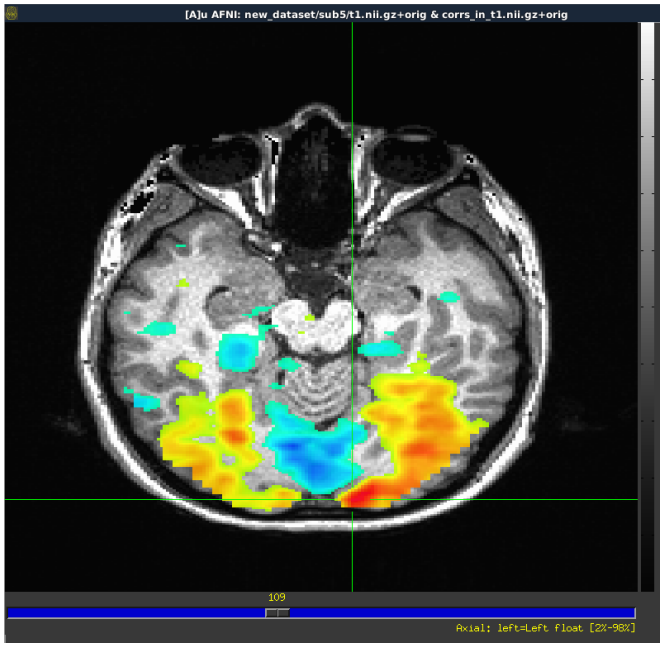
Sub3:


[A]u AFNI: new_dataset/sub3/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
129
Axial: left=Left float [2%-98%]

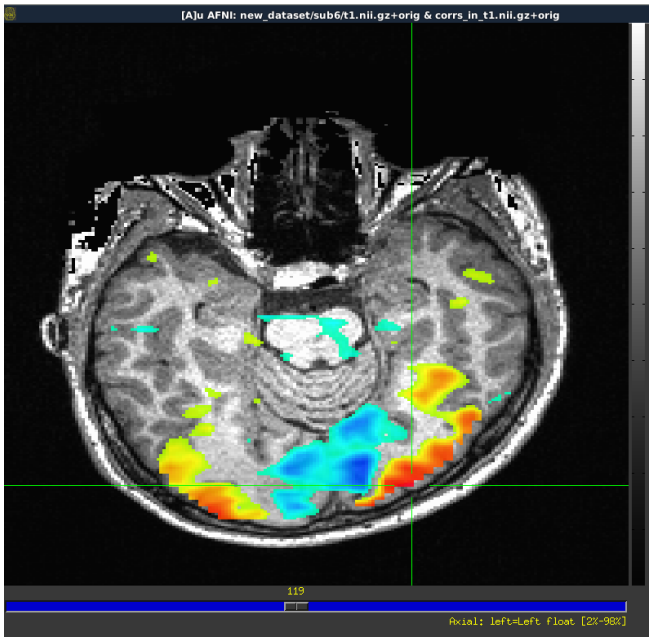
Sub4:


[A]u AFNI: new_dataset/sub4/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
118
Axial: left=Left float [2%-98%]

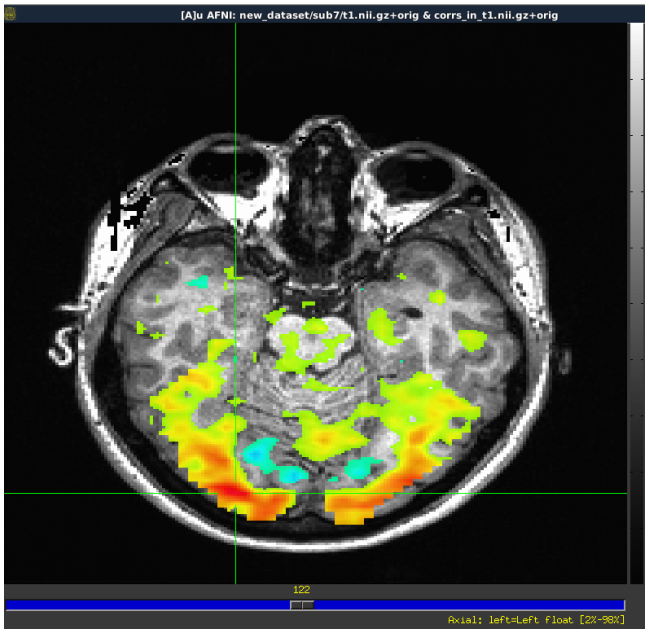
Sub5:


[A]u AFNI: new_dataset/sub5/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
109
Axial: left=Left float [2%-98%]

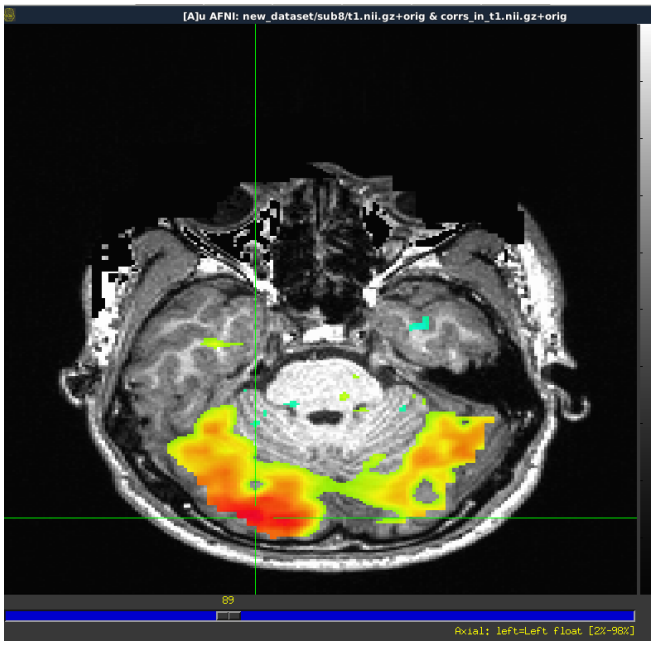
Sub6:


[A]u AFNI: new_dataset/sub6/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
119
Axial: left=Left float [2%-98%]

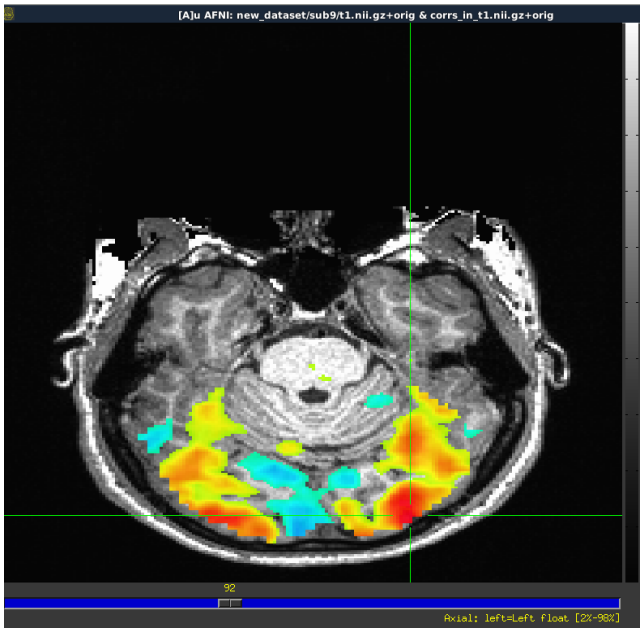
Sub7:


[A]u AFNI: new_dataset/sub7/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
122
Axial: left=Left float [2%-98%]

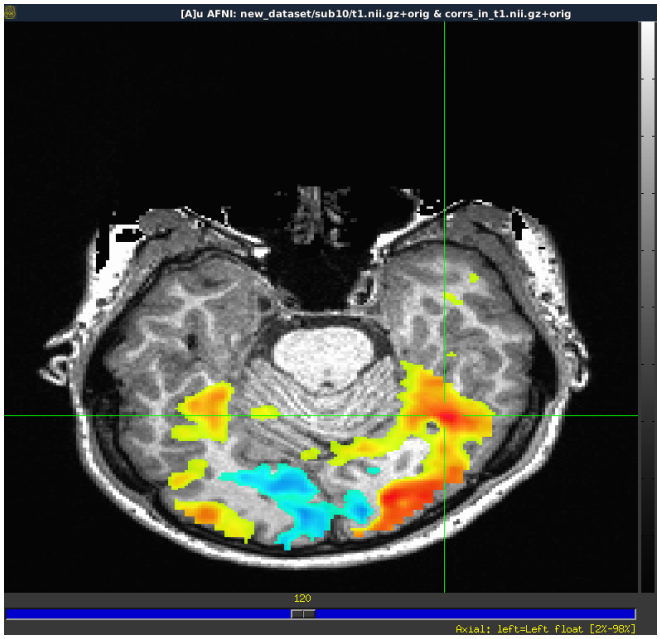
Sub8:


[A]u AFNI: new_dataset/sub8/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
89
Axial: left=Left float [2%-98%]

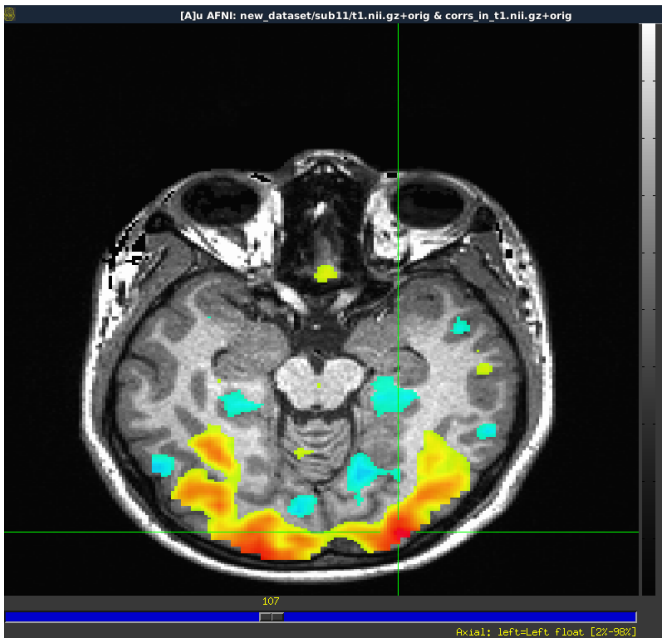
Sub9:


[A]u AFNI: new_dataset/sub9/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
92
Axial: left=Left float [2%-98%]

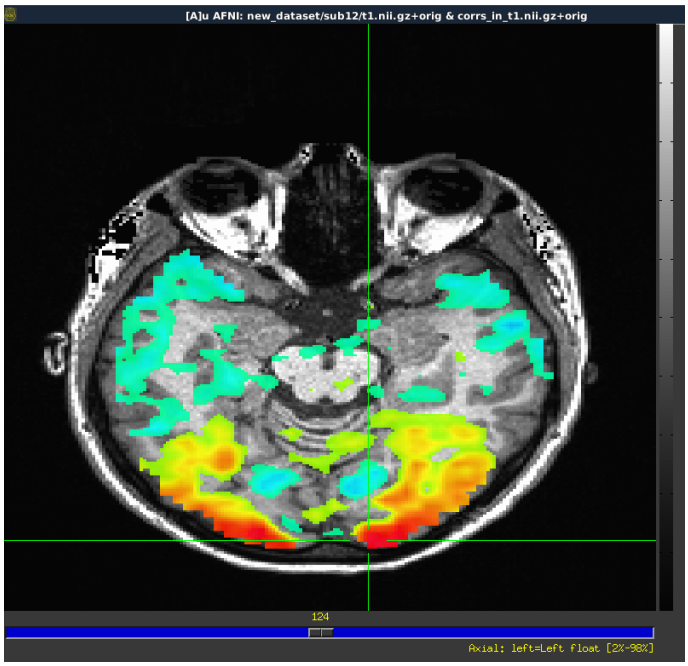
Sub10:


[A]u AFNI: new_dataset/sub10/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
120
Axial: left=Left float [2%-98%]

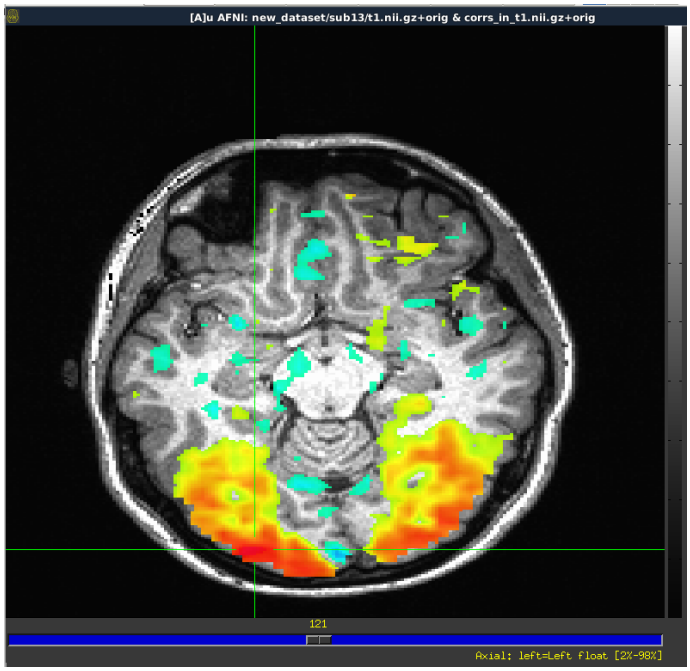
Sub11:


[A]u AFNI: new_dataset/sub11/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
107
Axial: left=Left float [2%-98%]

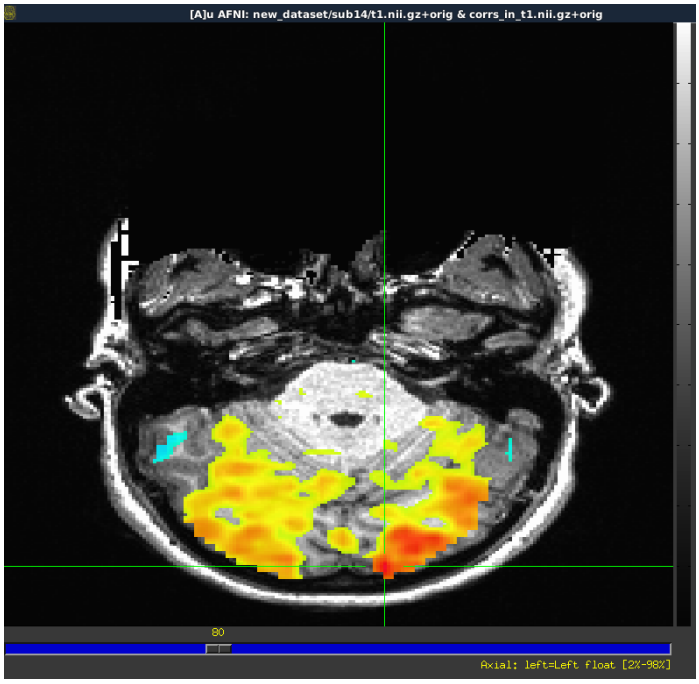
Sub12:


[A]u AFNI: new_dataset/sub12/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
124
Axial: left=Left float [2%-98%]

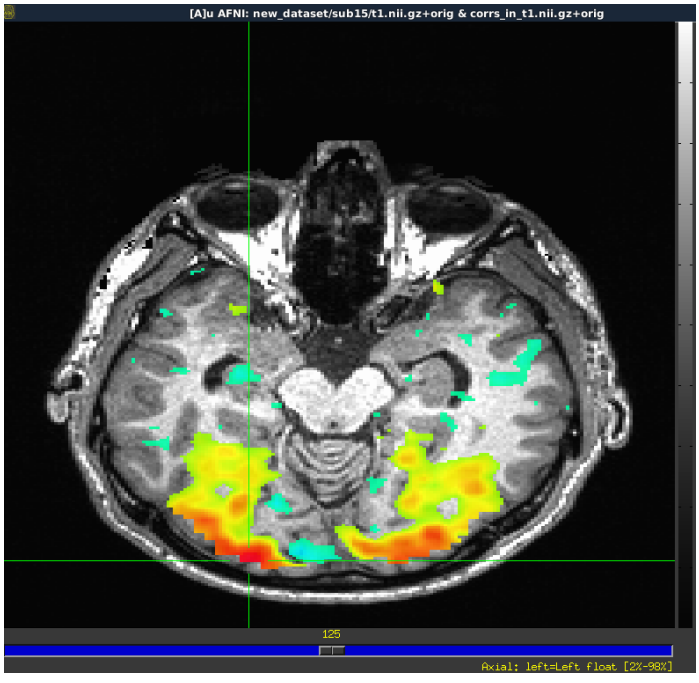
Sub13:


[A]u AFNI: new_dataset/sub13/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
121
Axial: left=Left float [2%-98%]

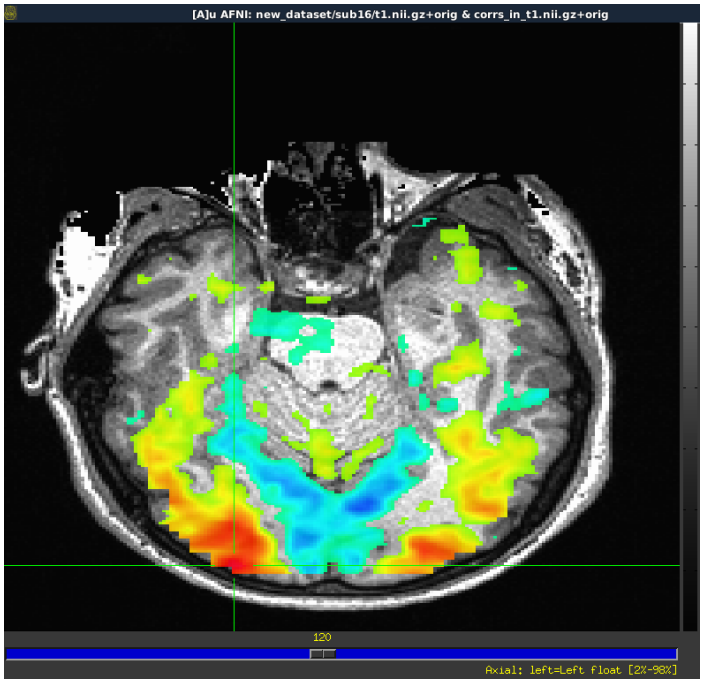
Sub14:


[A]u AFNI: new_dataset/sub14/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
80
Axial: left=Left float [2%-98%]

Sub15:


[A]u AFNI: new_dataset/sub15/t1.nii.gz+orig & corrs_in_t1.nii.gz+orig
125
Axial: left=Left float [2%-98%]

Sub16:



Group average:

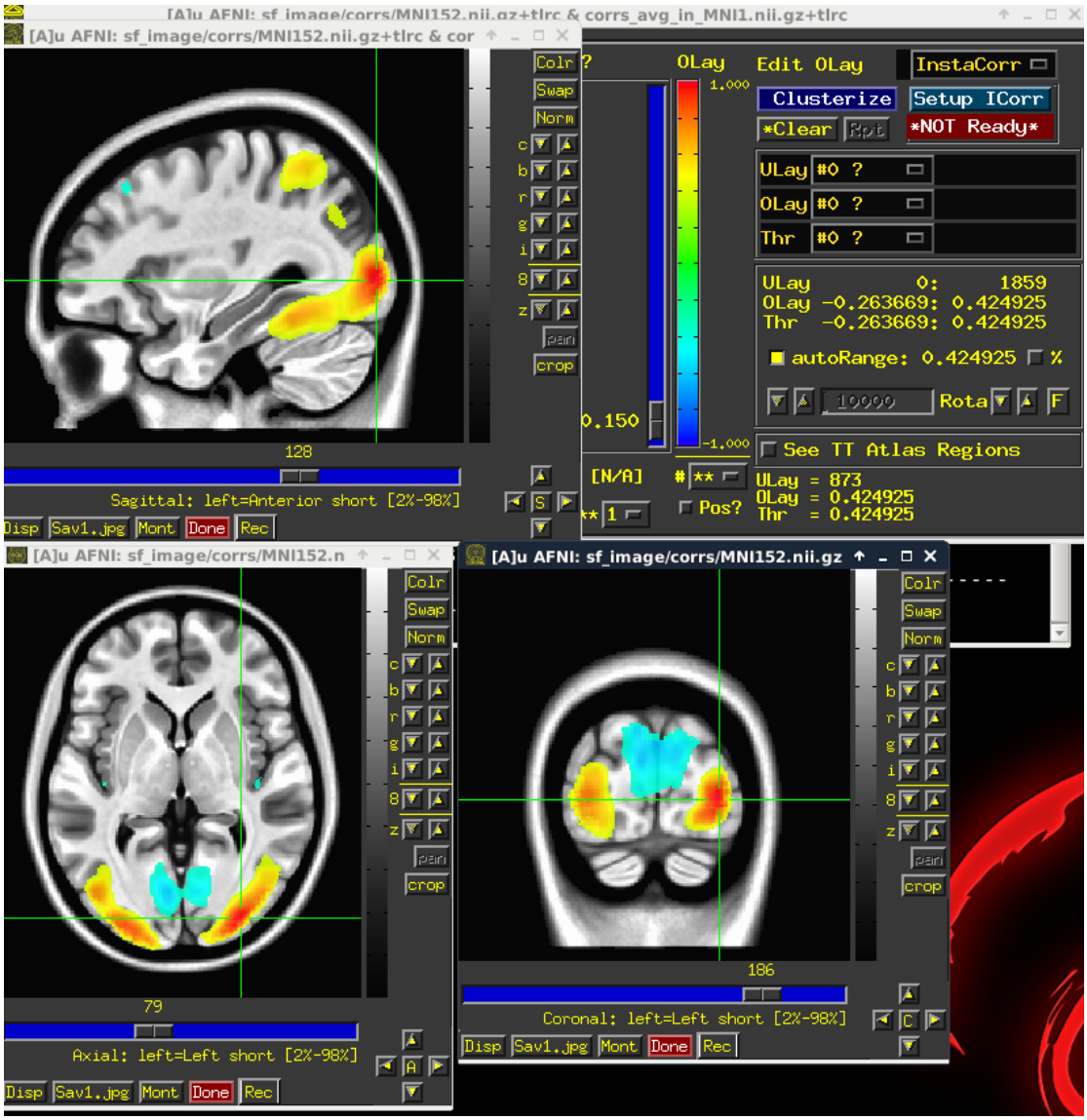
In this part, we need use template brain " MNI152_2009_template.nii.gz" to register t1 on this image, and I use python code to extract the image with skull in MNI152_2009_template.nii.gz file, here is my code to extract the template brain with skull:

```
fmri = nib.load('I:/image/part1/a3_data/MNI152_2009_template.nii.gz');
img = fmri.get_data()
img1=img[:,:,:,0,1]
corrs_nifti=nib.Nifti1Image(img1, fmri.affine)
nib.save(corrs_nifti,'I:/image/MNI152.nii.gz')
```

here is the code to calculate grand average correlation map in python:

```
imgs=np.zeros((193,229,193,16))    //(193,229,193) is the shape of correlation map, 16 represent 16 images
for i in range(0,16):
    sub1=nib.load('I:/image/corrs/corrs_in_MNI'+str(i+1)+'.nii.gz');
    img=sub1.get_data()
    where_are_NaNs = np.isnan(img)
    img[where_are_NaNs] = 0
    imgs[:,:,:,i]=img
img_avg=np.average(imgs,axis=3)
corrs_nifti=nib.Nifti1Image(img_avg, sub1.affine)
nib.save(corrs_nifti,'I:/image/corrs/corrs_avg_in_MNI1.nii.gz')
plt.imshow(img_avg[:,:,90])
```

here is our result image (grand average correlation map overlayed on MNI152):



Bonus1:

We use sub1 data to do this bonus, first we need to find images at special time(after 4.5 sceonds after famous and unfamilar face showed), here is the code we use to find those images:

```
fmri = nib.load('I:/image/new_dataset/sub1/clean_bold.nii.gz');
events = pd.read_csv('I:/image/new_dataset/sub1/events.tsv',delimiter='\t');
events = events.to_numpy()
img = fmri.get_data()
flag1=0//record the times of famous face showed
flag2=0// record the times of unfamiliar face showed
for i in np.arange(0,events.shape[0]):
    if events[i,3]=='FAMOUS':
        flag1+=1
    elif events[i,3]=='UNFAMILIAR':
        flag2+=1
```

```
f1img=np.zeros((65,77,67,flag1+1)) //(65,77,67) is the shape of img, 32 is the total times of famous face showed
f2img=np.zeros((65,77,67,flag2+1))
flag1=0
flag2=0
for i in np.arange(0,events.shape[0]):
    if events[i,3]=='FAMOUS':
        flag1+=1
        f1img[:,:,:,flag1]=img[:,:,:,int((events[i,0]+4.5)/2)]
    elif events[i,3]=='UNFAMILIAR':
        flag2+=1
        f2img[:,:,:,flag2]=img[:,:,:,int((events[i,0]+4.5)/2)]
```

Then, we do a t-test,here is our code to do this:

```
t=stats.ttest_ind(f1img,f2img,axis=3)
where_are_NaNs = np.isnan(t[0])
t[0][where_are_NaNs] = 0 //set nan to zero
where_are_NaNs = np.isnan(t[1])
t[1][where_are_NaNs] = 0 //set nan to zero
```

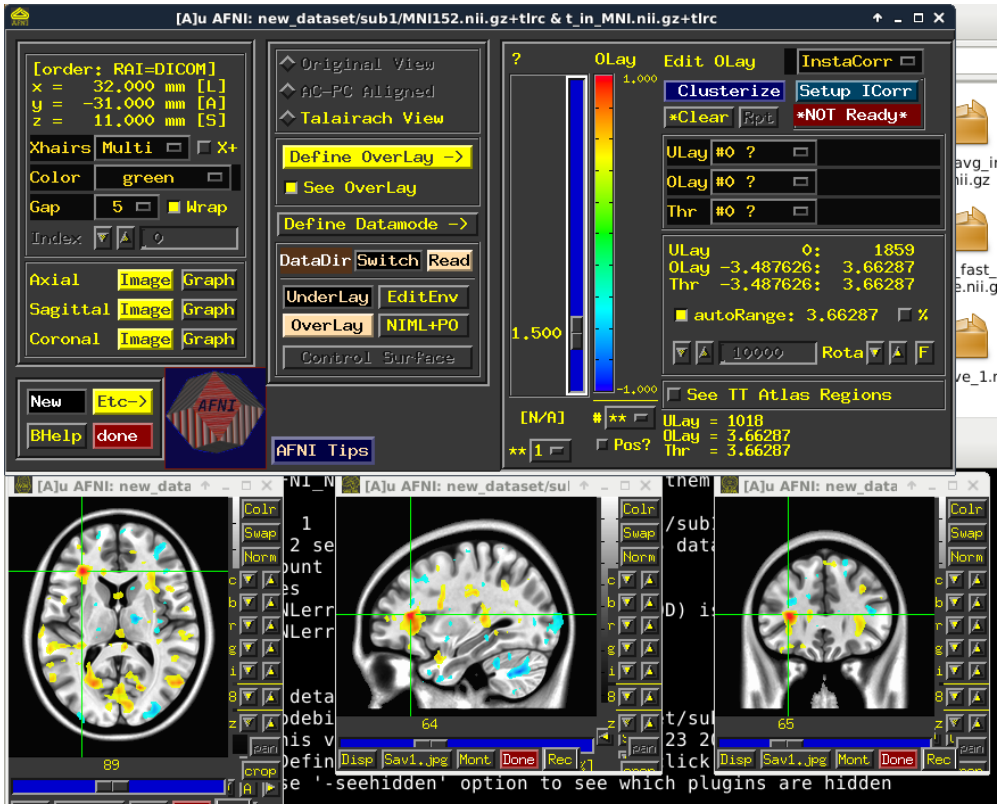
at last, we save t-map into file:

```
t_nifti=nib.Nifti1Image(t[0], fmri.affine)
nib.save(t_nifti,'I:/image/new_dataset/sub1/t.nii.gz')
```

After that, we use FSL to transform our result to MNI152 image, here is the command(use some file from part3 files):

```
flirt -in t.nii.gz -ref t1.nii.gz -applyxfm -init epireg.mat -out t_in_t1.nii.gz
flirt -in t_in_t1.nii.gz -ref t1_in_MNI.nii.gz -applyxfm -init transform.mat -out t_in_MNI.nii.gz
```

here is our final result:



Bonus3:

We try to solve this question, here is the code we use to calculate the mutual information, but I guess we got a wrong result:

```
from sklearn import metrics
img_new=np.zeros((img.shape[0],img.shape[1],img.shape[2]))
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        for k in range(img.shape[2]):
            img_new[i,j,k]=metrics.mutual_info_score(conved, img[i,j,k,:])
```

here is our final result: