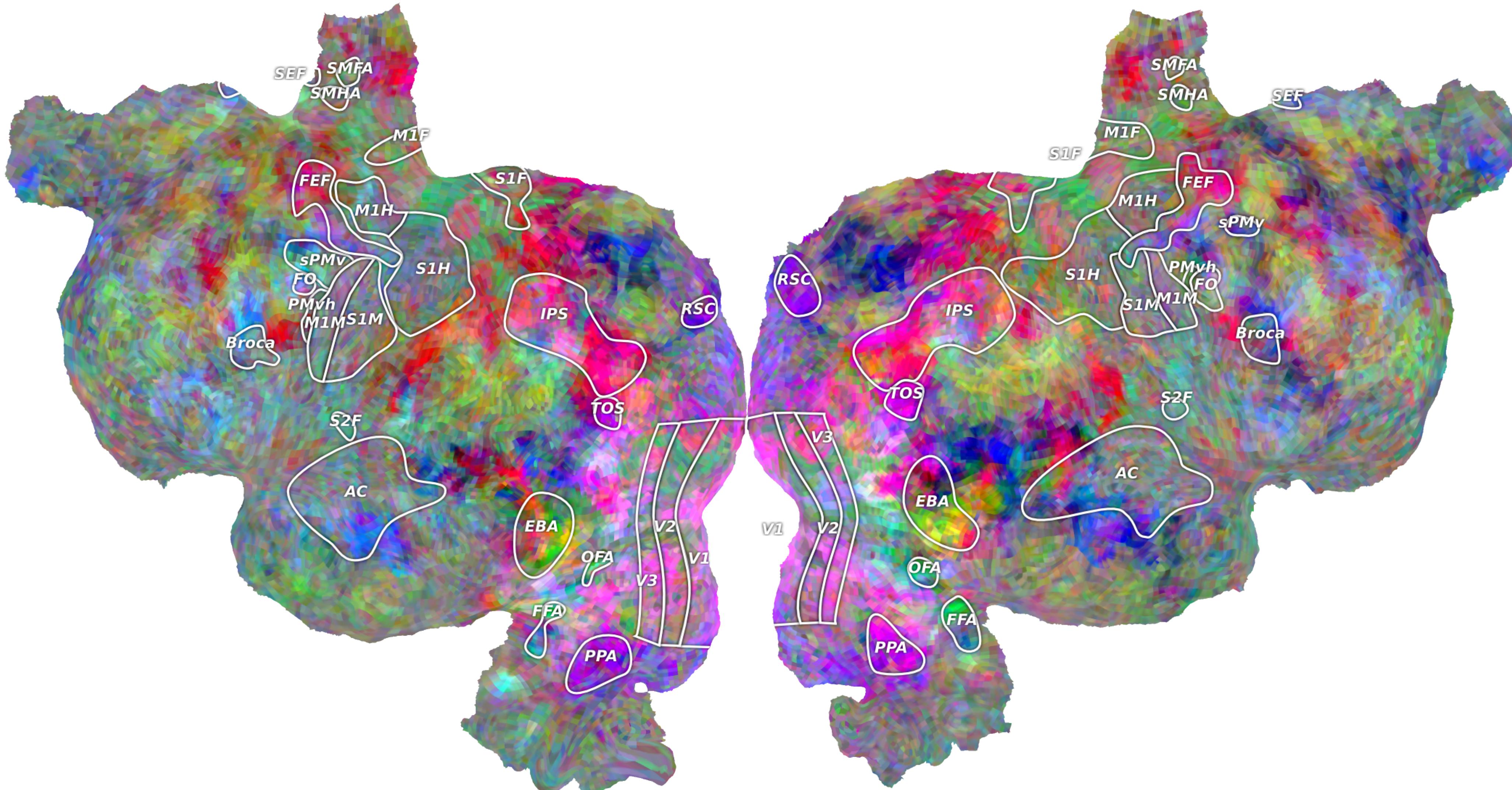




Neuroimaging data visualization with python

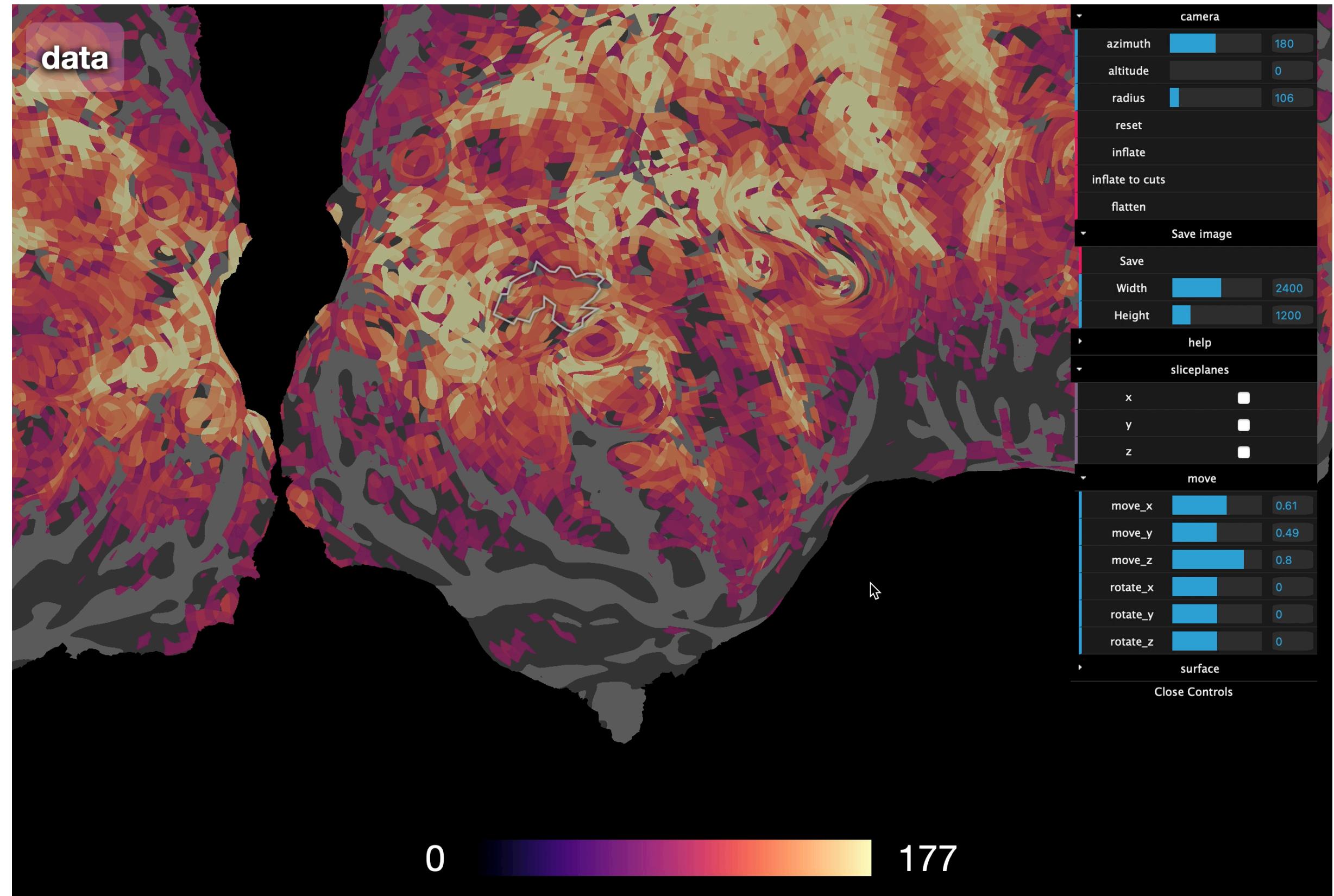
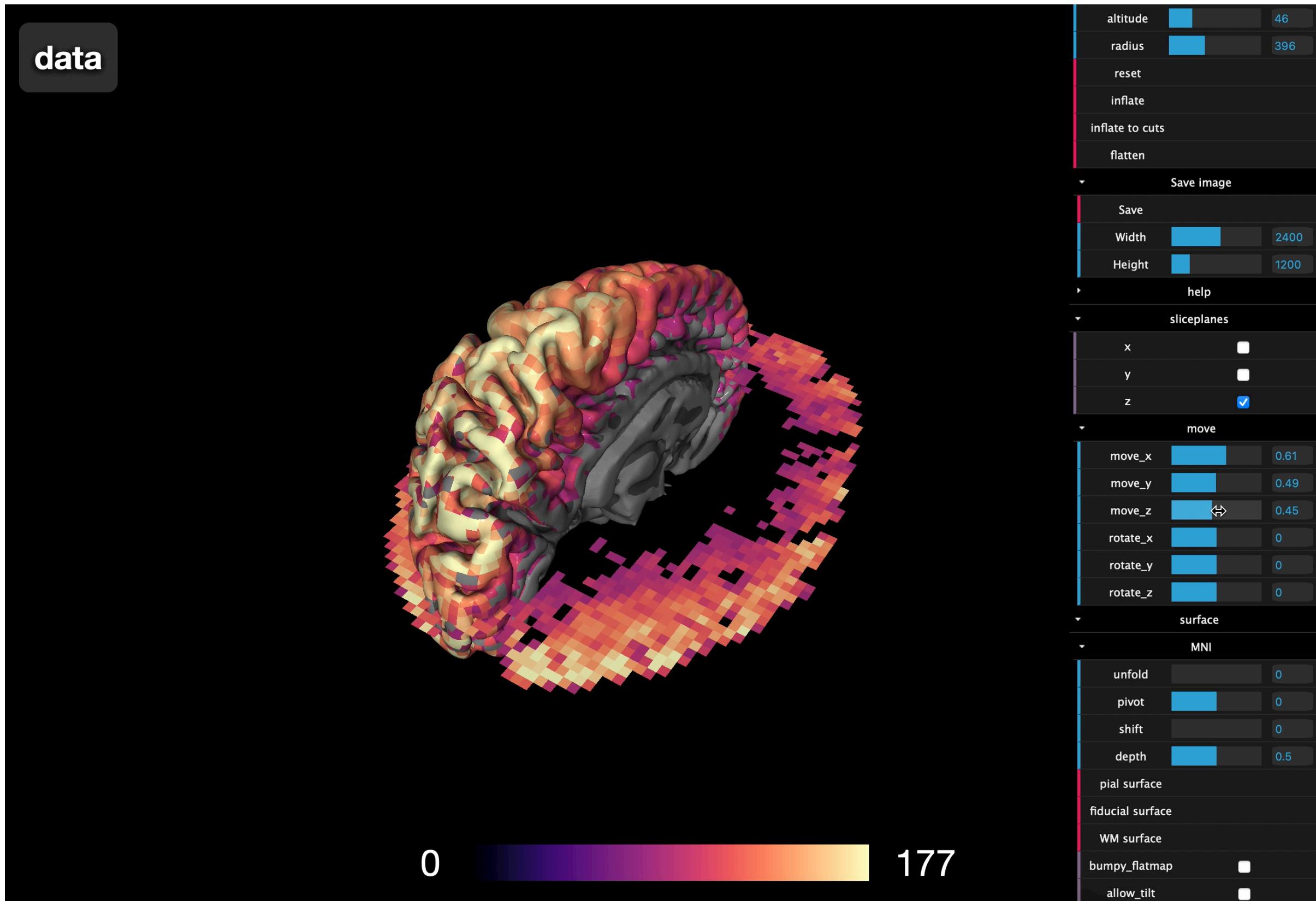
성균관대학교 지각인지신경과학연구실 박지웅

Pycortex: python-based toolkit for surface visualization



source: <https://github.com/gallantlab/pycortex>

Showcase



PyCortex는 WebGL 기반의 interactive, customizable surface visualization을 제공합니다.
매우 가볍고, 기기 종속성이 낮으며, Python 기반의 분석 환경에 가장 잘 맞는 neuroimaging visualizer입니다.

Showcase

```
def export_figure(statmap, cmap, cmap_label, cmin, cmax):
    vol_data = cortex.Volume(statmap.transpose(2,1,0), 'MNI', 'fmriprep_3mm',
                             cmap=cmap, colorbar=True, vmax=cmax, vmin=cmin)

    filenames = cortex.export.save_3d_views(
        vol_data, base_name=base_name,
        list_angles=['medial_pivot', 'lateral_pivot'],
        list_surfaces=['inflated', 'inflated'],
        size=(1024 * 4, 768 * 2), trim=True,)

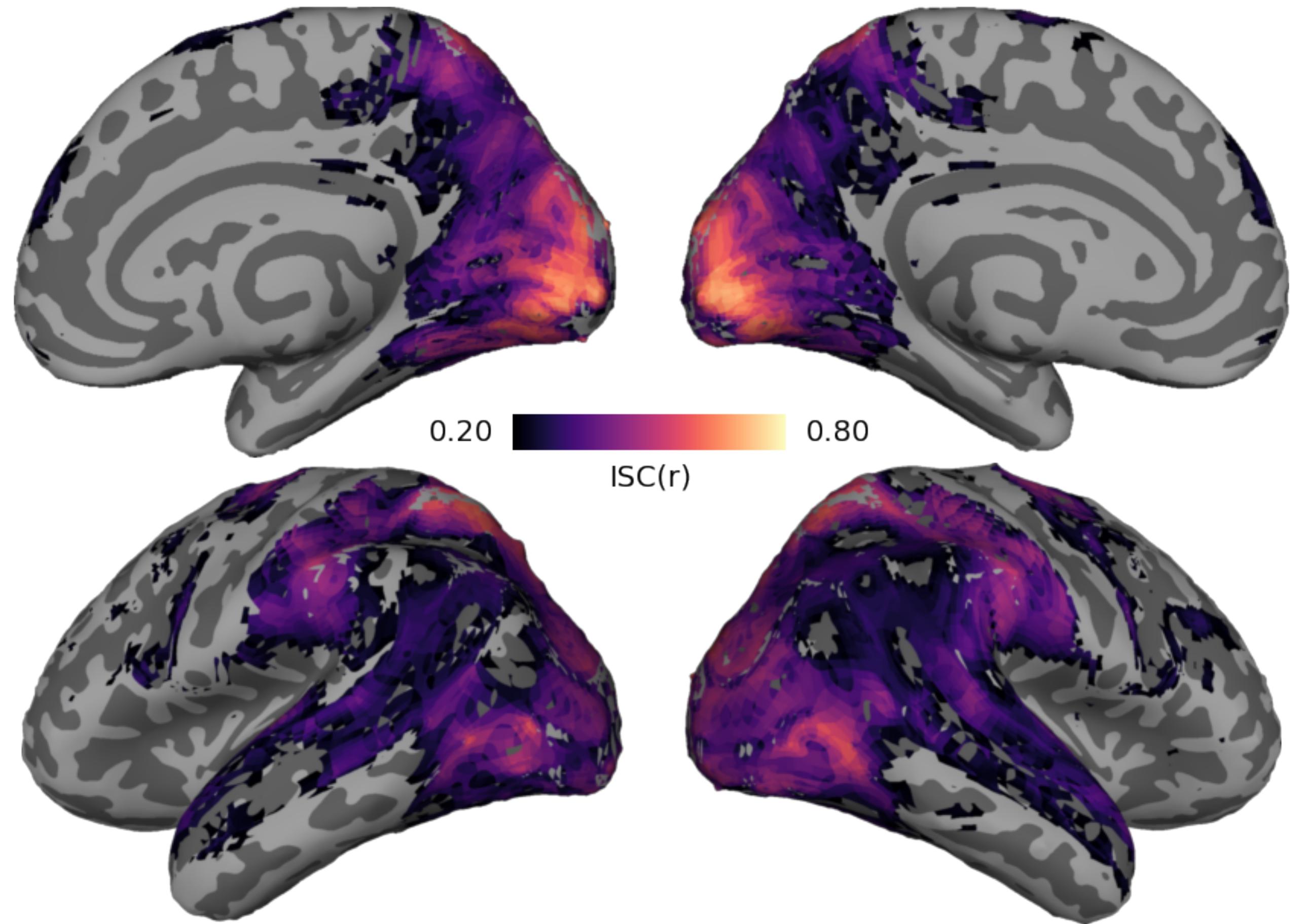
    fig = plt.figure(constrained_layout=True, dpi=200, figsize=(6,4))
    gs = fig.add_gridspec(2, 1, height_ratios=[4.95, 5.05])

    ax1 = fig.add_subplot(gs[0])
    ax1.imshow(plt.imread(filenames[0]), interpolation='gaussian')
    ax1.axis('off')

    cax = fig.add_axes([0.437, 0.515, 0.15, 0.03])
    gradient = np.linspace(0, 1, 256)
    gradient = np.vstack((gradient, gradient))
    cax.imshow(gradient, aspect='auto', cmap=cmap)
    pos = list(cax.get_position().bounds)
    x_text_left = pos[0] - 0.01
    x_text_center = pos[0] + pos[2]/2.
    x_text_right = pos[0] + pos[2] + 0.01
    y_text = pos[1] + pos[3]/2.
    y_text_bottom = pos[1] - 0.01

    fig.text(x_text_left, y_text, '{:0.2f}'.format(cmin), va='center', ha='right', fontsize=7)
    fig.text(x_text_right, y_text, '{:0.2f}'.format(cmax), va='center', ha='left', fontsize=7)
    fig.text(x_text_center, y_text_bottom, cmap_label, va='top', ha='center', fontsize=7)
    cax.axis('off')

    ax2 = fig.add_subplot(gs[1])
    ax2.imshow(plt.imread(filenames[1]), interpolation='gaussian')
    ax2.axis('off')
    plt.subplots_adjust(hspace=0.01)
```



Jupyter notebook 위에서 분석중인 brain data, stat map 등을 원하는 custom layout으로 바로 visualize 할 수 있습니다.
Python package 답게 customize 할 수 있는 부분이 많고, Jupyter notebook과 함께 매끄럽게 동작합니다.

Installation

```
import cortex
```

```
ModuleNotFoundError Traceback (most recent call last)
<ipython-input-1-abeddb261fa9> in <module>
----> 1 import cortex
```

```
ModuleNotFoundError: No module named 'cortex'
```

```
!pip install -U pycortex
```

```
Collecting pycortex
```

```
  Downloading pycortex-1.2.4.tar.gz (37.2 MB)
```

```
 |██████████| 37.2 MB 56.2 MB/s eta 0:00:01
```

```
Installing build dependencies ... done
```

```
Getting requirements to build wheel ... done
```

```
Preparing wheel metadata ... done
```

```
import cortex
```

설치 전

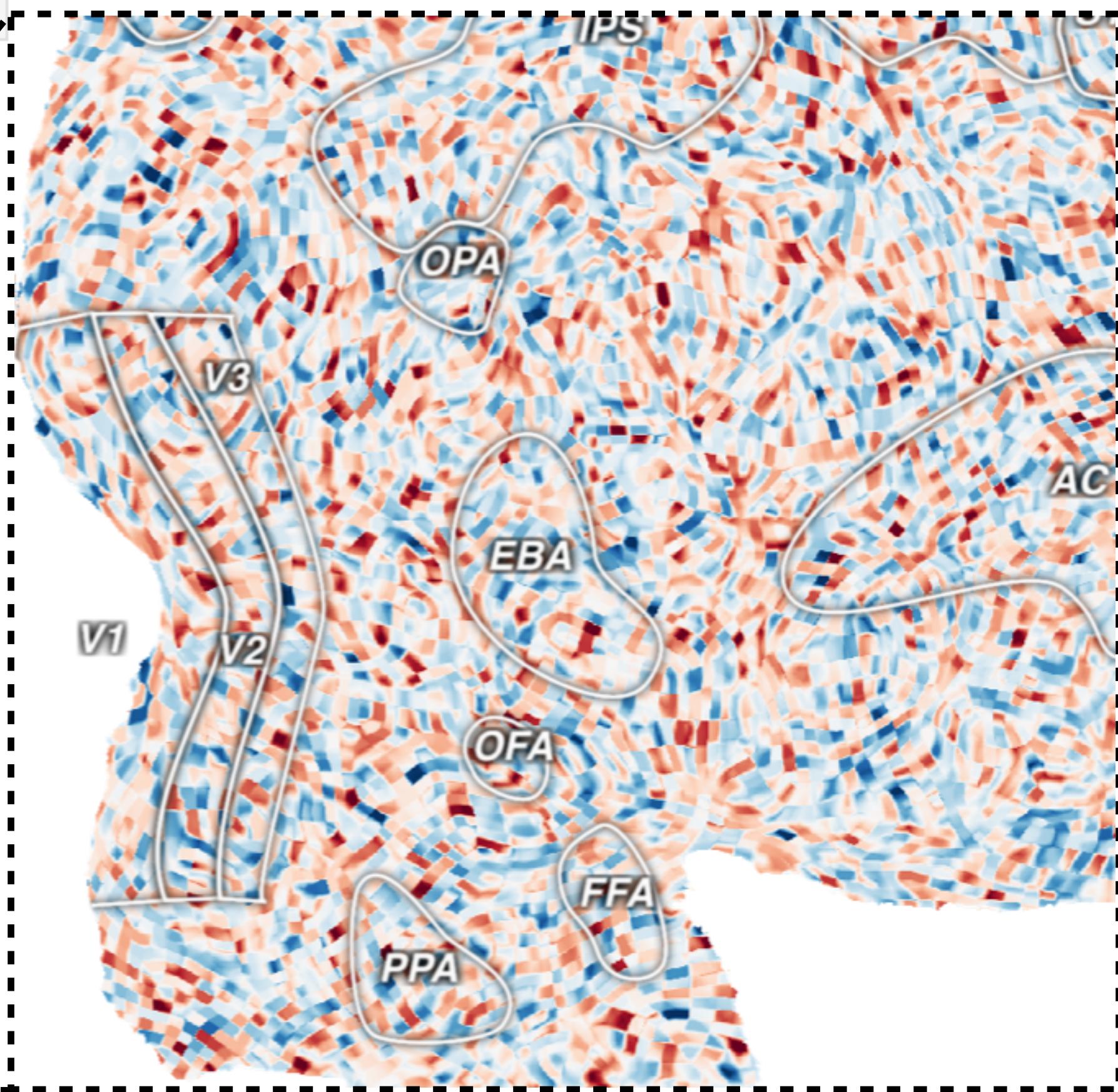
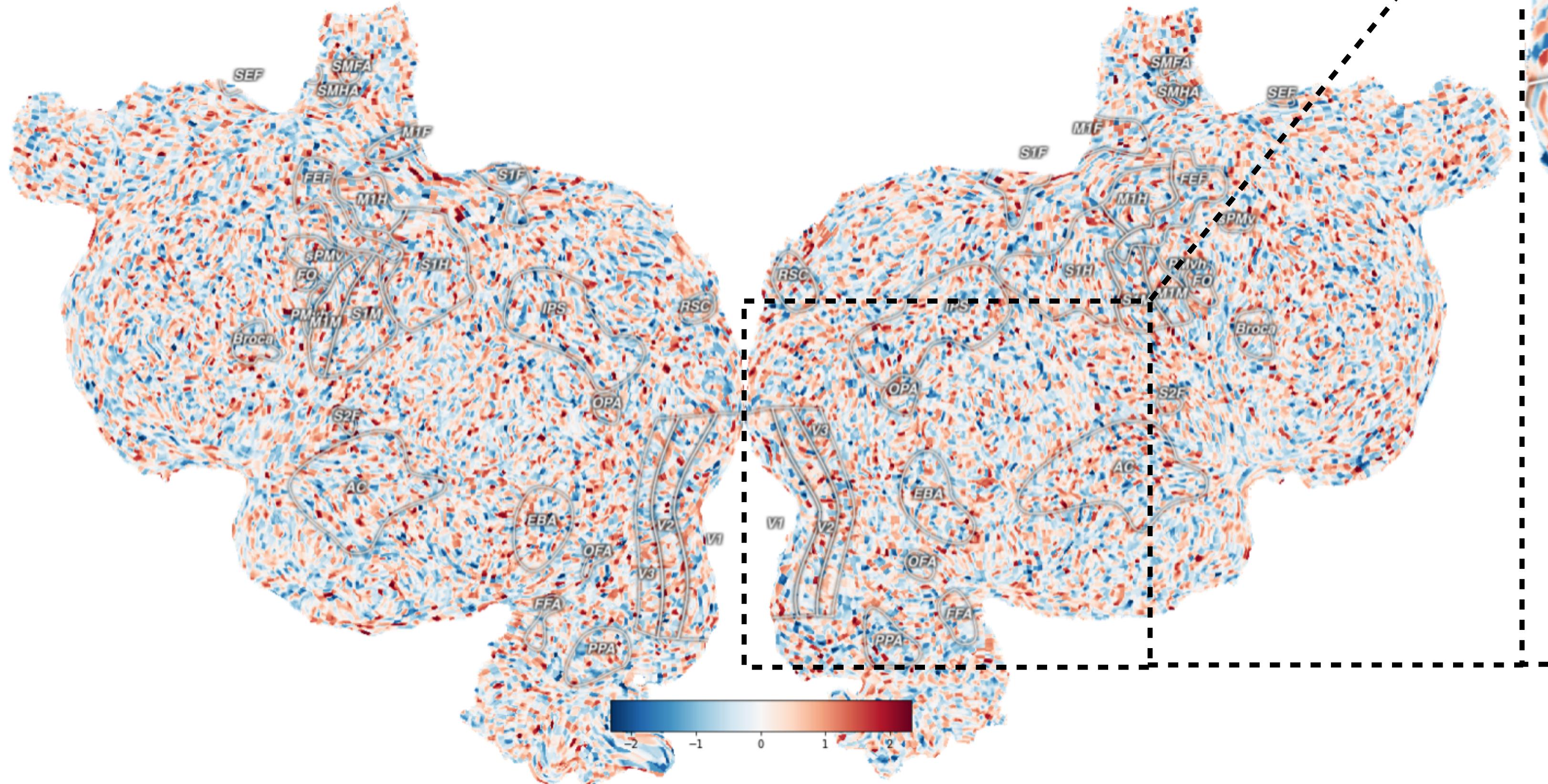
설치 후

Pycortex의 설치는 PyPI repository를 통해 쉽게 설치 할 수 있습니다.

Demo visualization

```
import cortex  
import matplotlib.pyplot as plt  
cortex.quickshow(cortex.Volume.random("S1", "fullhead"))  
plt.show()
```

Fontconfig warning: ignoring UTF-8: not a valid region tag
Background RRGGBBAA: ffffff00
Area 0:0:1960.5:1024 exported to 1960 x 1024 pixels (96 dpi)

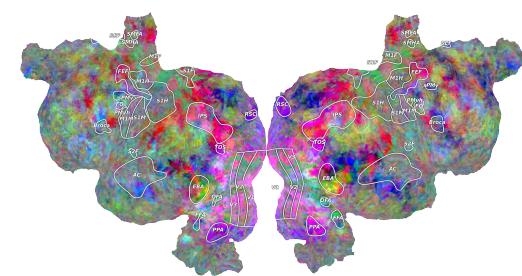


고해상도로 figure를 export 할 수 있습니다.

Software dependencies

소프트웨어의 작동을 보장하는 의존성

여러분들의 개발 환경



PyCortex



INKSCAPE



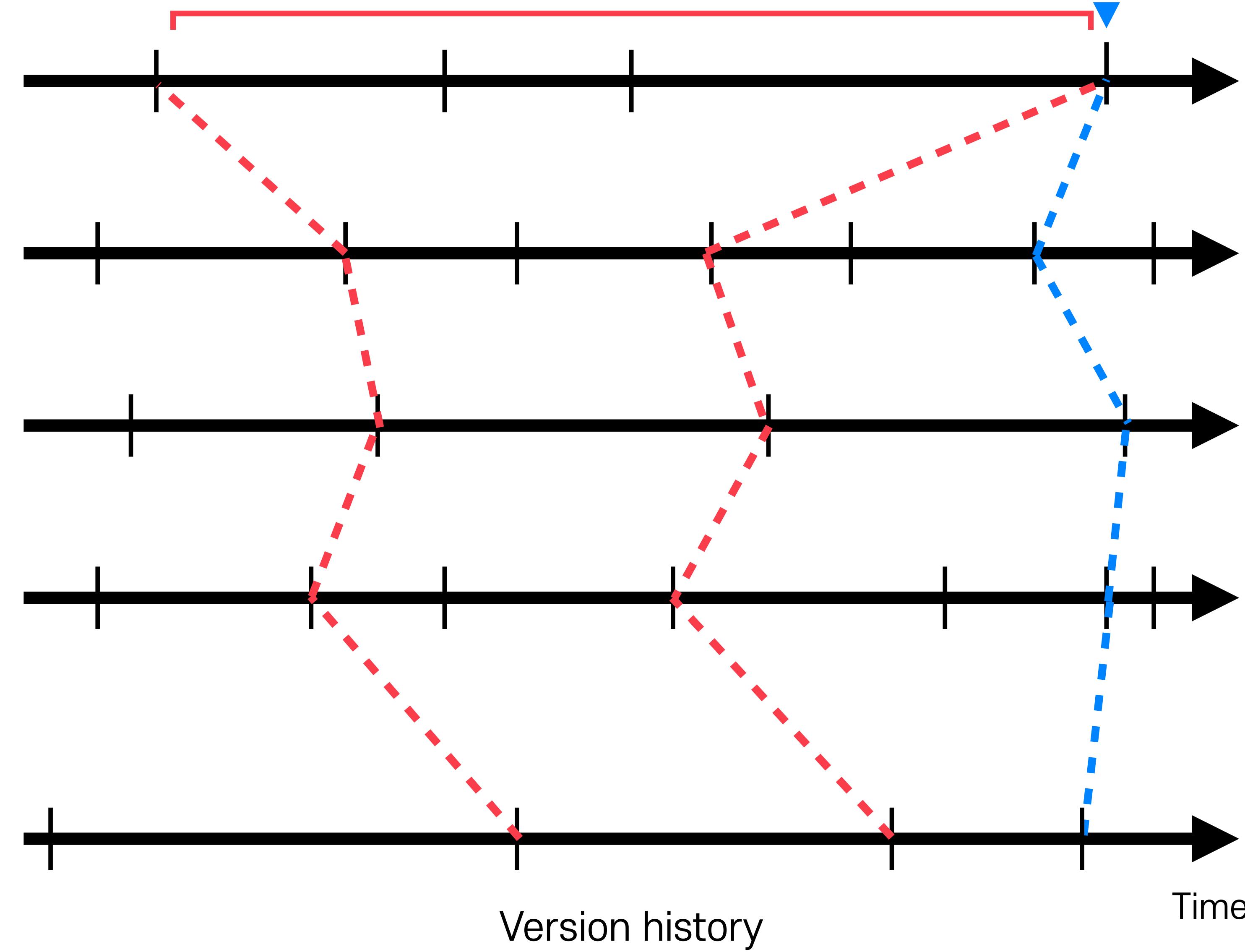
FreeSurfer



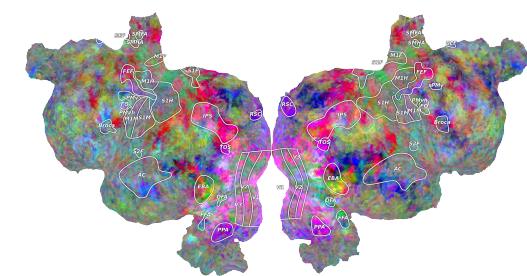
macOS



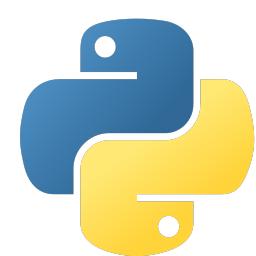
Ubuntu



Software dependencies



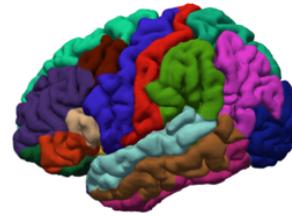
Pycortex



python™



INKSCAPE



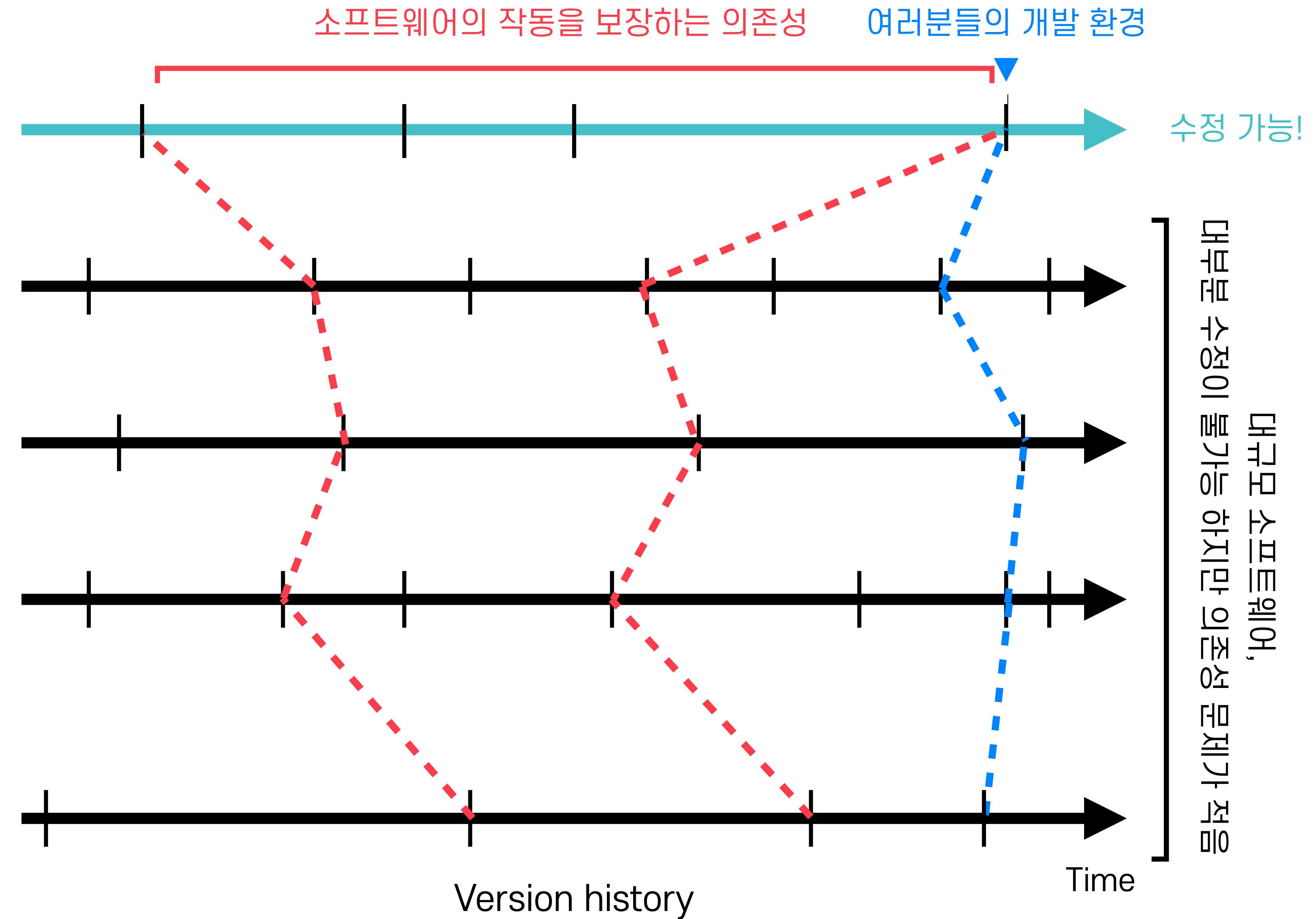
FreeSurfer



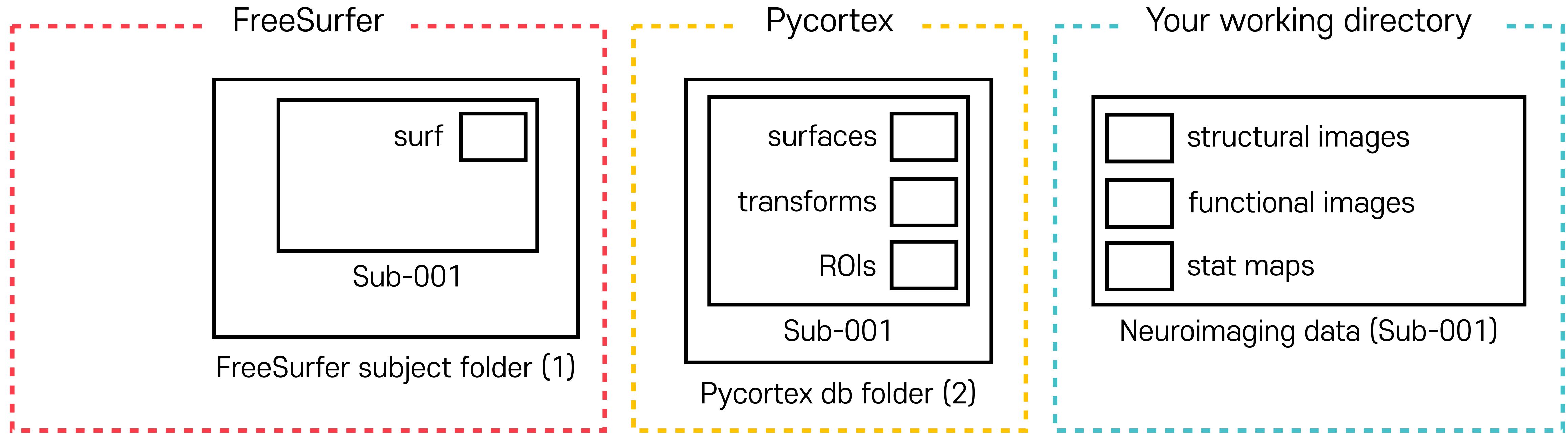
macOS



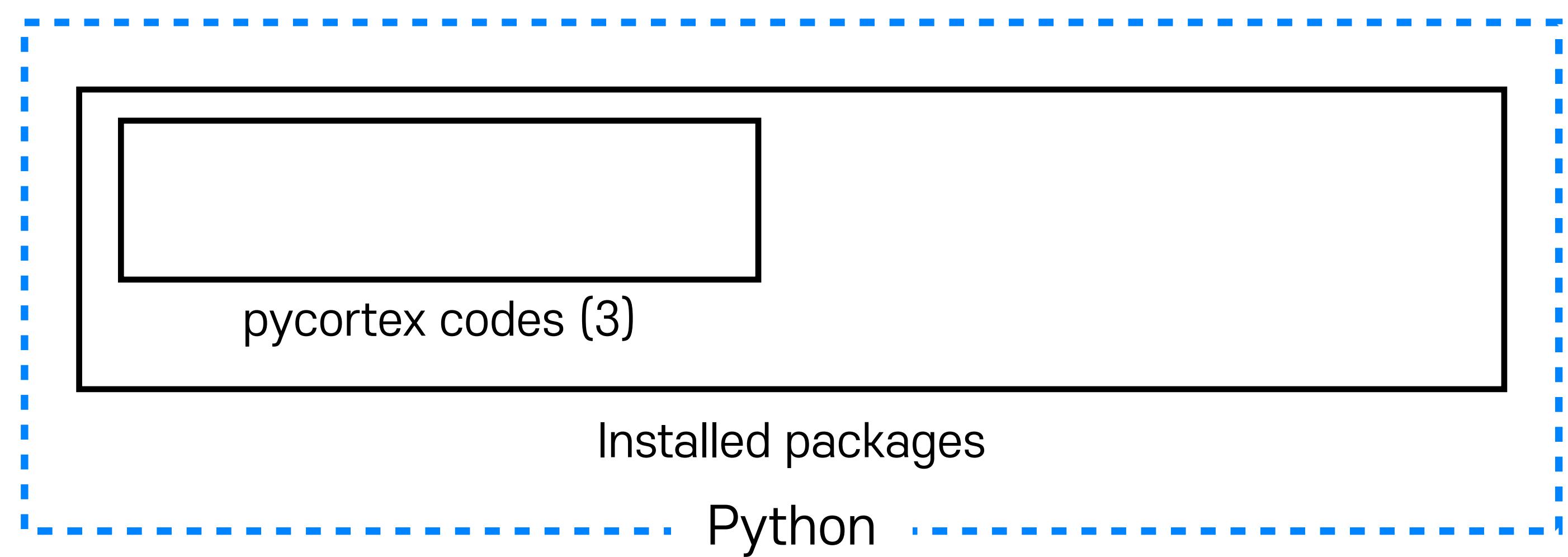
Ubuntu



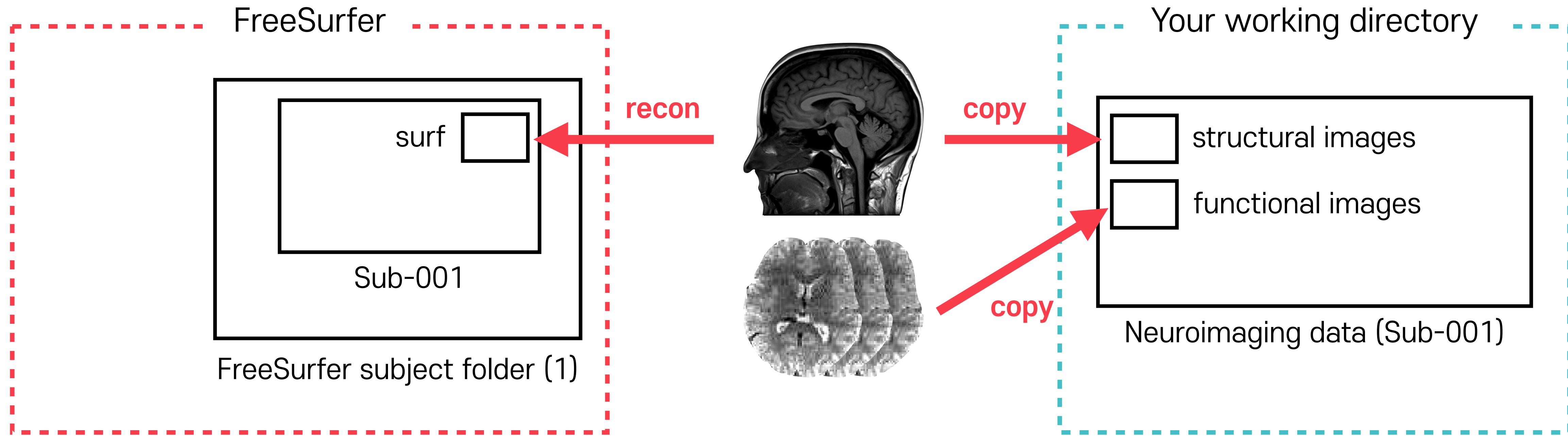
Data structure



여러분들은 이번 세미나에서
이 그림에 보이는 data structure를 이해하고,
여러분이 가진 데이터를 어떻게
Pycortex를 이용해 visualize 할 수 있는지
배우게 됩니다.



Importing data into FreeSurfer



FreeSurfer를 이용하여 structural image의 surface segmentation:

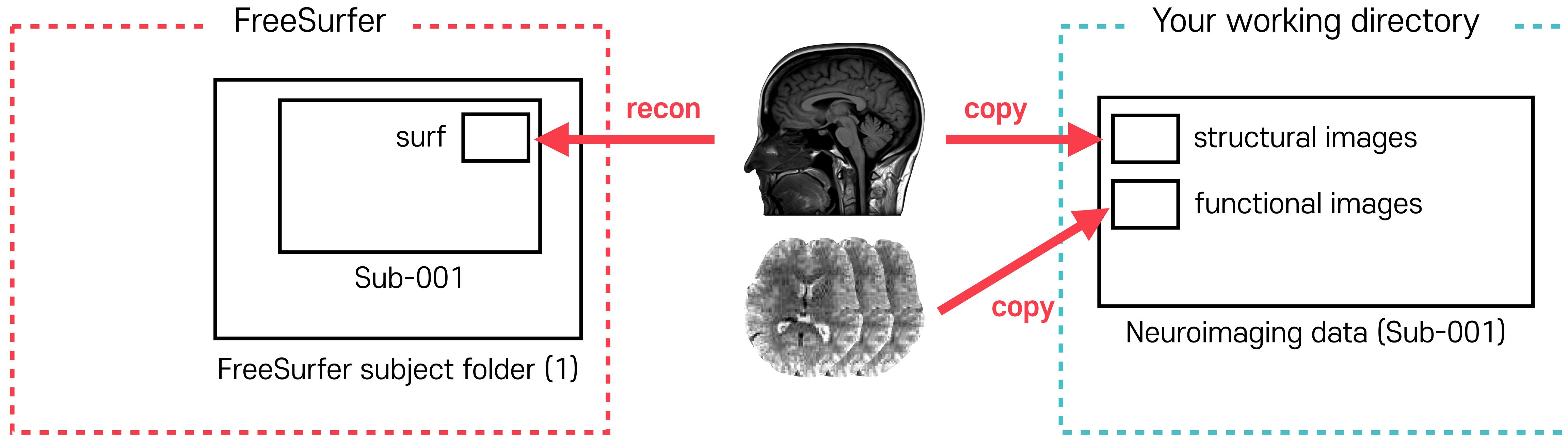
```
recon-all -i T1.nii -s <subject_name>
```

surface segmentation 과정은 대략 6~7시간 정도 소요 됩니다.

만약 주로 MNI space(MNI)에서 작업하시는 경우엔 고해상도 MNI template를 한 번만 recon 해 주시면 됩니다.

제가 공유한 드라이브의 freesurfer_subjects 폴더의 <MNI> 파일을 다운로드 해서 subject folder에 넣어주셔도 됩니다.

Importing data into FreeSurfer



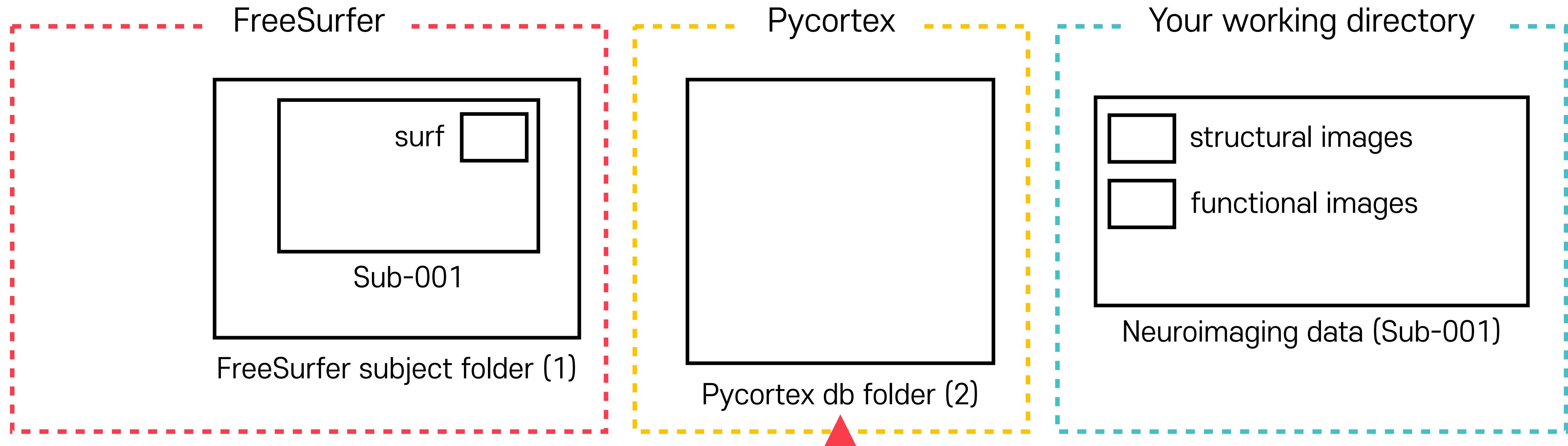
```
----- freesurfer-darwin-macOS-7.2.0-20210713-aa8f76b -----
Setting up environment for FreeSurfer/FS-FAST (and FSL)
FREESURFER_HOME  /Applications/freesurfer/7.2.0
FSFAST_HOME       /Applications/freesurfer/7.2.0/fsfast
FSF_OUTPUT_FORMAT nii.gz
SUBJECTS_DIR      /Applications/freesurfer/7.2.0/subjects
MNI_DIR           /Applications/freesurfer/7.2.0/mni
(base) jwpark@JWPARK ~ $ cd $SUBJECTS_DIR
(base) jwpark@JWPARK /Applications/freesurfer/7.2.0/subjects$ tree -L 1
```

이 폴더들은 각각
FreeSurfer에 등록한
subject에 해당합니다.

FreeSurfer가 문제 없이 설치가 되었다면 shell을 실행 시
FREESURFER_HOME 폴더와 SUBJECT_DIR 폴더의 경로를
확인 할 수 있습니다.

제 시스템에서는 “/Applications/freesurfer/7.2.0/subjects” 폴더가
FreeSurfer subject folder (1)에 해당합니다.
(Ubuntu의 경우 “/usr/local/freesurfer/subjects”)

Import subject from FreeSurfer



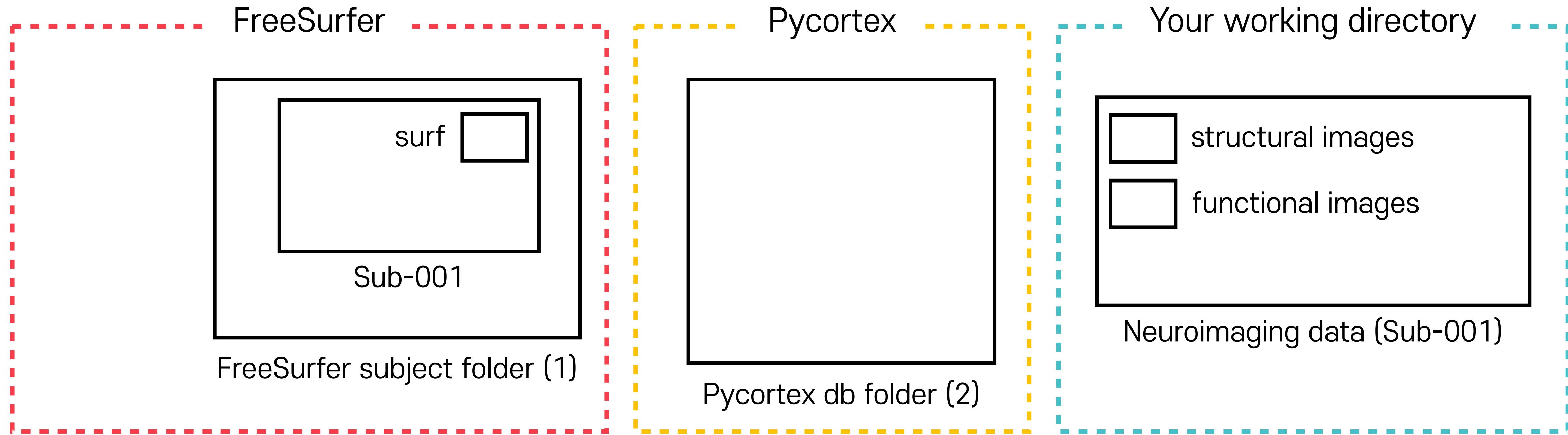
```
import cortex
print(cortex.database.default_filestore)

/usr/local/share/pycortex/db
```

Pycortex가 설치 된 후, pycortex의 db folder를 확인 할 수 있습니다. 이 코드를 실행해야 생성되는게 아님

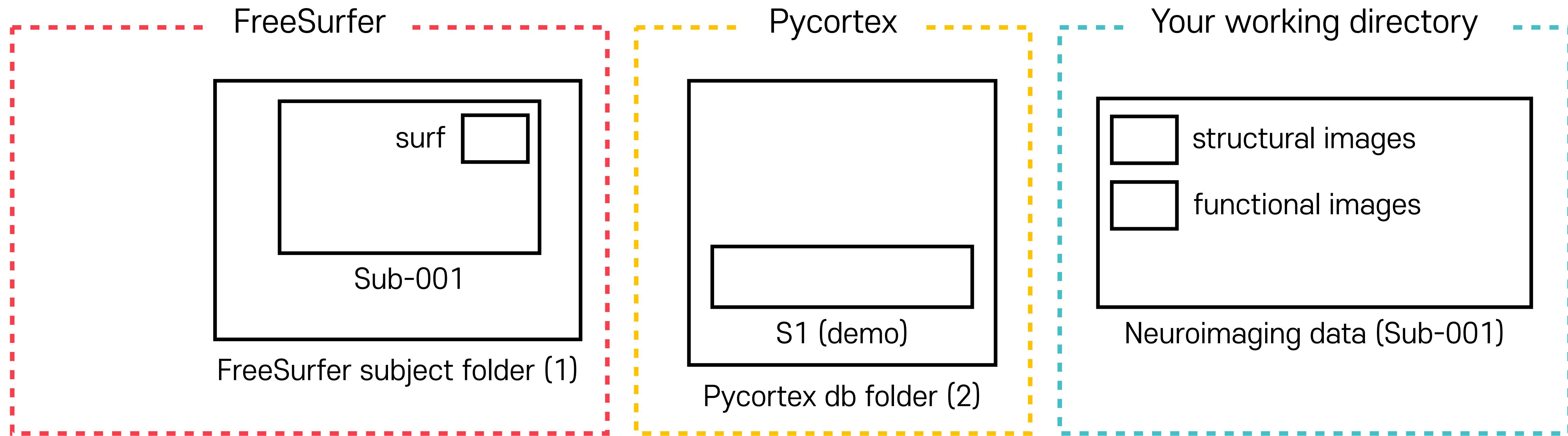
가끔 처음 Pycortex를 설치 할 때, 해당 폴더가 임시 폴더(*tmp*)로 지정 될 수 있습니다.
이 경우엔 user config 파일을 수정하여 경로를 변경해 주어야 합니다.

Import subject from FreeSurfer



사용자 설정(user config)파일의 위치를 확인 한 후, 해당 파일에서 filestore 부분의 경로를 수정해 주시면 됩니다.

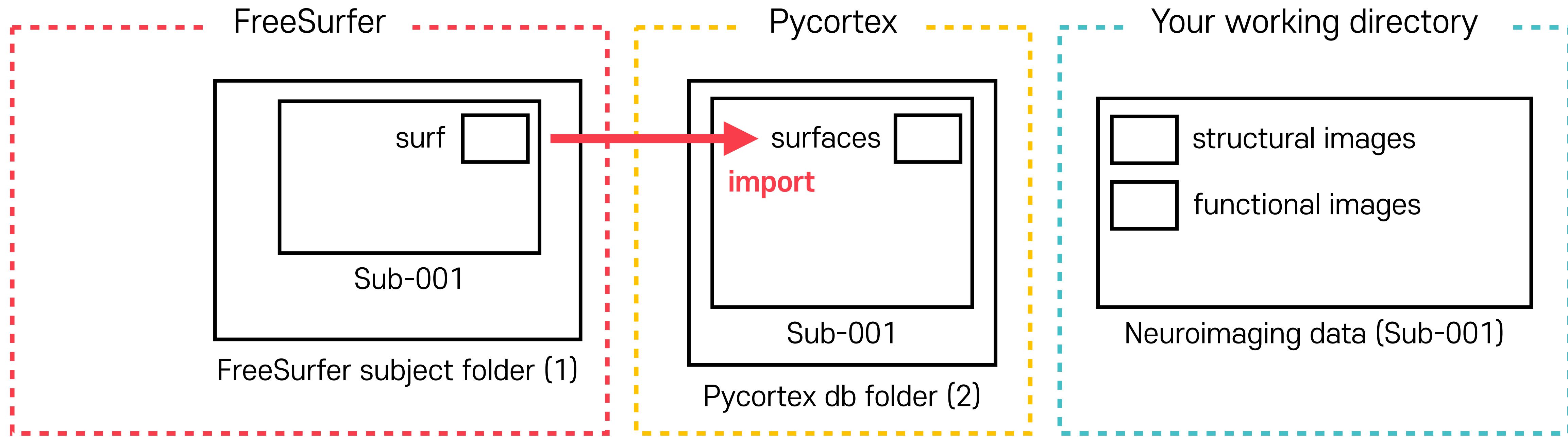
Import subject from FreeSurfer



```
(base) jwpark@JWPARK /usr/local/share/pycortex/db$ tree -L 2
.
└── S1
    ├── anatomicals
    ├── cache
    ├── overlays.svg
    ├── surface-info
    ├── surfaces
    └── transforms
        └── views
```

Pycortex db folder에는 default로
demo visualization을 위한
Example subject(S1)가 들어 있습니다.
(앞으로 사용하지는 않습니다)

Import subject from FreeSurfer



이제 FreeSurfer의 subject folder (1)에 있는 <MNI> subject를 Pycortex의 db folder (2)로 불러 옵니다.

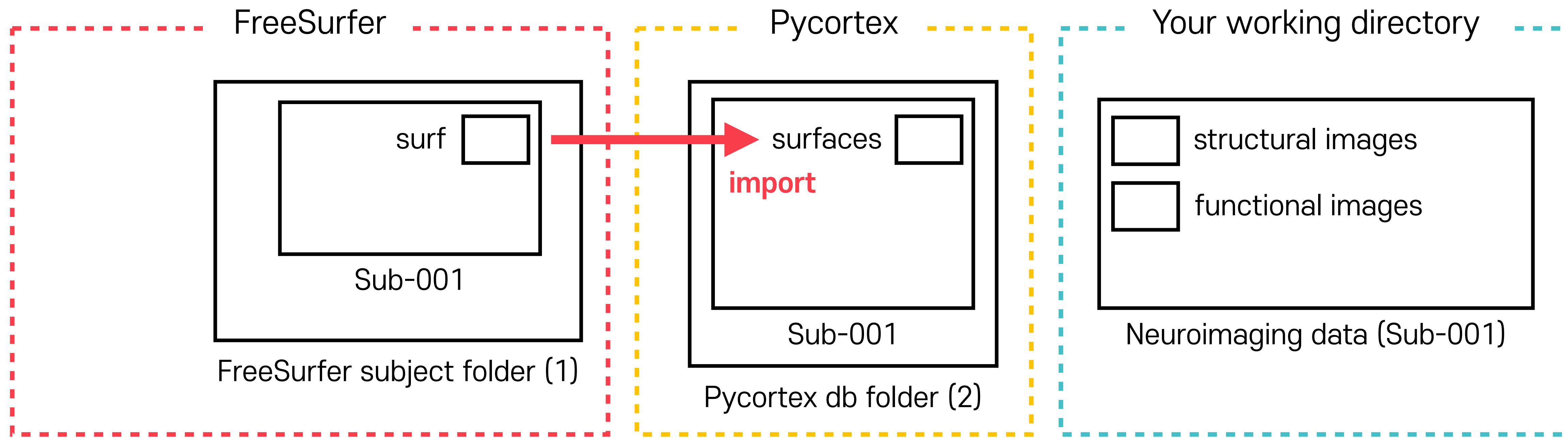
```
cortex.freesurfer.import_subj('MNI')
```

```
b'created by jwpark on Tue Sep  8 11:14:25 2020\n'  
b'created by jwpark on Tue Sep  8 12:09:57 2020\n'  
b'created by jwpark on Tue Sep  8 11:14:29 2020\n'  
b'created by jwpark on Tue Sep  8 12:19:26 2020\n'  
b'created by jwpark on Tue Sep  8 11:14:25 2020\n'  
b'created by jwpark on Tue Sep  8 11:14:29 2020\n'  
b'created by jwpark on Tue Sep  8 12:09:57 2020\n'  
b'created by jwpark on Tue Sep  8 12:19:26 2020\n'  
b'created by jwpark on Tue Sep  8 11:14:41 2020\n'  
b'created by jwpark on Tue Sep  8 11:14:54 2020\n'
```

```
(base) jwpark@JWPARK /usr/local/share/pycortex/db$ tree -L 2  
.  
└── MNI  
    ├── anatomicals  
    ├── cache  
    ├── surface-info  
    └── surfaces  
        └── transforms  
            └── views  
.  
└── S1  
    ├── anatomicals  
    ├── cache  
    ├── overlays.svg  
    ├── surface-info  
    └── surfaces  
        └── transforms  
            └── views
```

MNI subject가
생성되었습니다.
아직 transforms 폴더는
비어있습니다.

Import subject from FreeSurfer

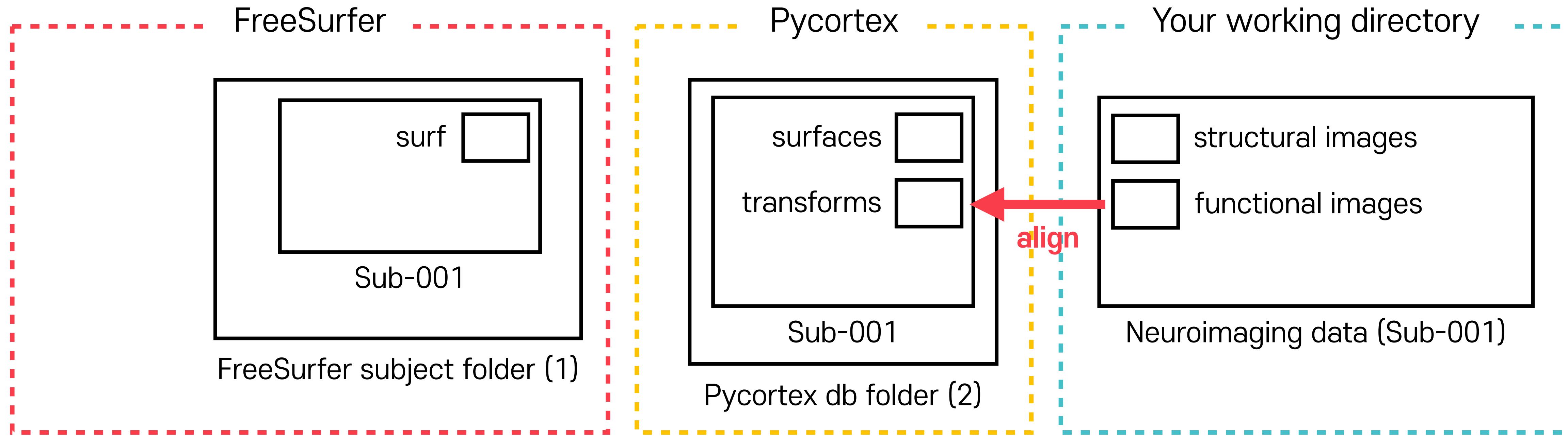


이렇게 import를 하고 난 후에는 FreeSurfer를 사용하지 않습니다.

따라서 제가 공유한 드라이브의 `pycortex_subjects` 폴더에 있는 MNI 폴더를 db folder (2)에 복사하시면, FreeSurfer가 설치되어 있지 않아도 Pycortex를 이용하여 surface visualization을 할 수 있습니다.

이렇게 불러온 subject는 functional image의 spatial resolution이나 FoV와는 무관합니다.

Functional image alignment



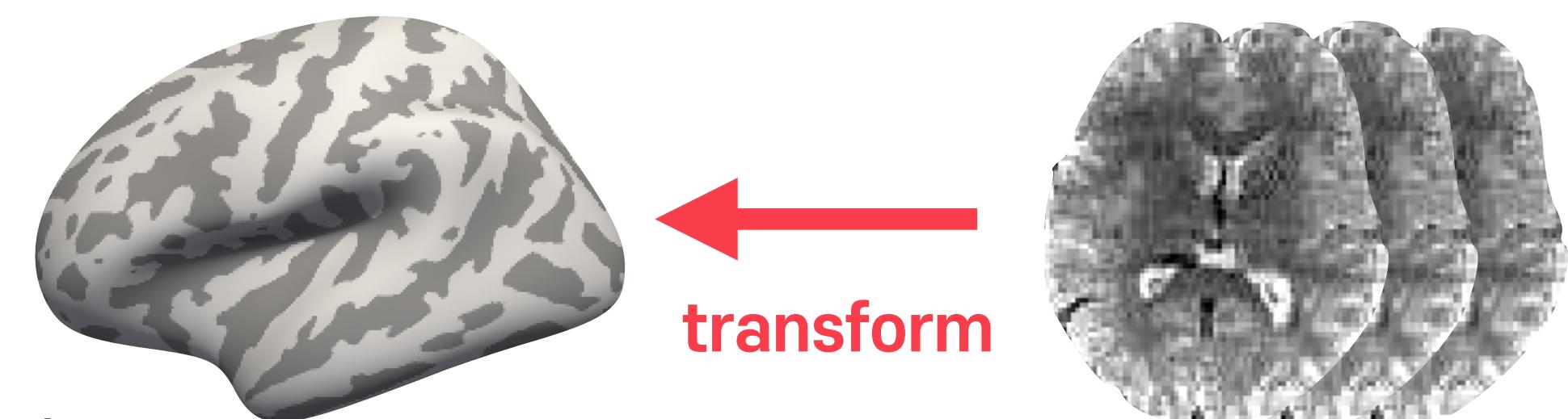
가진 Functional image를 surface에 align 합니다.

이때 voxel-to-vertex transform의 이름을 정의합니다.

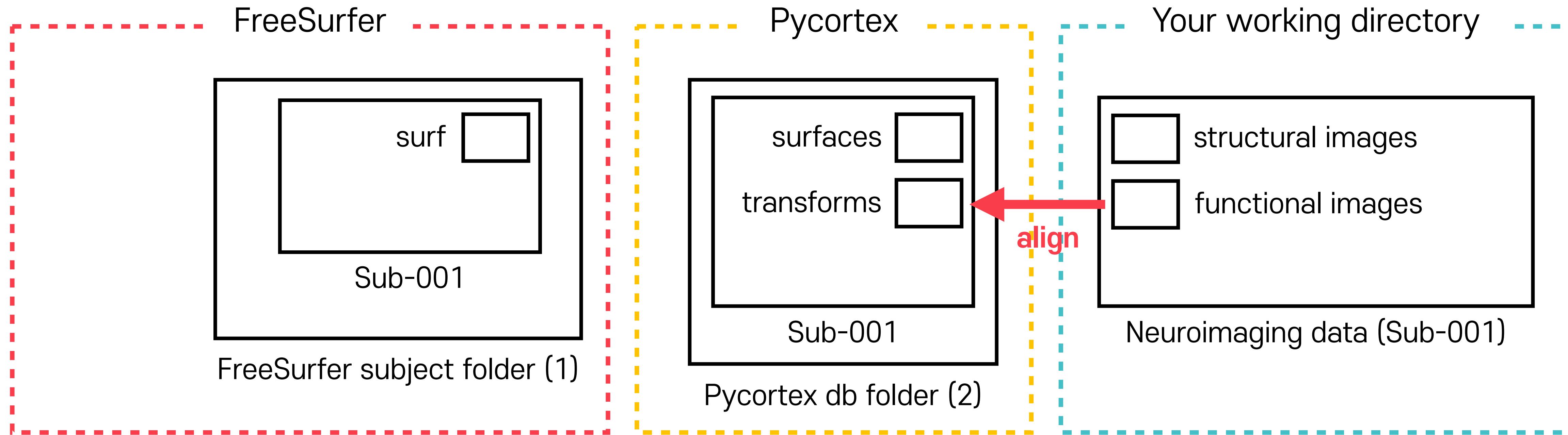
만들어진 transform은 같은 FoV, resolution을 가진

Functional image의 visualization에 앞으로도 사용 할 수 있습니다.

저는 해당 functional image를 3mm resolution으로 찍고, fMRIprep을 이용하여 preprocessing 했기 때문에
voxel-to-vertex transform의 이름을 <fmriprep_3mm> 라고 지정하겠습니다.



Functional image alignment



```
cortex.align.automatic('MNI',
                      'fmriprep_3mm',
                      'example_EPI.nii.gz', use_fs_bbr=True)
```

Running freesurfer BBR

FreeSurfer의 BBR(Boundary-based registration) 알고리즘을 사용하여 automatic alignment를 진행합니다.

<MNI> : Pycortex DB에 저장된 subject의 이름

<fmriprep_3mm> : voxel-to-vertex transform의 이름

<example_EPI.nii.gz> : align을 원하는 voxel data (3D or 4D)

Troubleshooting

```
ValueError                                Traceback (most recent call last)
<ipython-input-18-d364d41174f9> in <module>
      1 cortex.align.automatic('MNI',
      2                         'fmriprep_3mm',
      3                         'example_EPI.nii.gz')

~/opt/anaconda3/lib/python3.8/site-packages/cortex/align.py in automatic(subject, xfmname, reference, noclean, bbrtype, pre_flirt_args, use_fs_bbr, epi_mask, intermediate)
    312         raise IOError('Error calling freesurfer BBR!')
    313
--> 314     xfm = Transform.from_freesurfer(os.path.join(cache, "register.dat"), absreference, subject)
    315     else:
    316         raw = db.get_anat(subject, type='raw').get_filename()

~/opt/anaconda3/lib/python3.8/site-packages/cortex/xfm.py in from_freesurfer(cls, fs_register, func_nii, subject, freesurfer_subject_dir)
    287         if len(L) == 5:
    288             L = L[1:]
--> 289         func_tkrvox2ras = np.array([[np.float(s) for s in ll.split() if s] for ll in L])
    290     except OSError:
    291         print ("Error occured while executing:\n{}\n".format(' '.join(cmd)))

~/opt/anaconda3/lib/python3.8/site-packages/cortex/xfm.py in <listcomp>(.0)
    287         if len(L) == 5:
    288             L = L[1:]
--> 289         func_tkrvox2ras = np.array([[np.float(s) for s in ll.split() if s] for ll in L])
    290     except OSError:
    291         print ("Error occured while executing:\n{}\n".format(' '.join(cmd)))
```

제가 겪은 에러는 다음과 같습니다.
아마 transformation을 위한
affine matrix를 생성하는 부분이
오류가 난 것 같습니다.

Pycortex 내부의 코드를 실행하는 도중 여러 종류의 에러가 발생 할 수 있습니다.
이때는 설치된 Python package 내에서 pycortex(cortex) 폴더를 찾아서
Pycortex의 core 코드를 적당히 수정 한 뒤, 다시 import 하여 사용하시면 됩니다.

```
import cortex
print(cortex.__file__)
```

/Users/jwpark/opt/anaconda3/lib/python3.8/site-packages/cortex/__init__.py

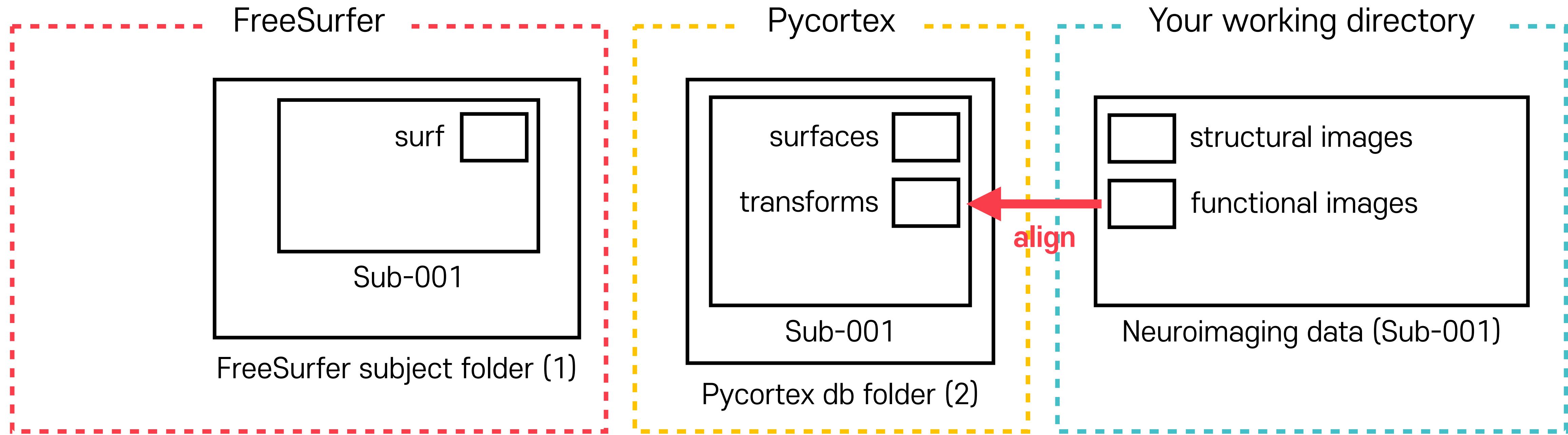
pycortex가 설치 된 폴더(3)는 위 명령어를 통해 찾을 수 있습니다.
비단 troubleshooting 뿐만 아니라, 여려모로 연구에 적합하게 튜닝하는 과정에서
자주 방문하여 코드를 수정할 일이 있으니 숙지해 두시면 좋을것 같습니다.

pycortex codes (3)

Installed packages

Python

Functional image alignment



```
cortex.align.automatic('MNI',
                      'fmriprep_3mm',
                      'example_EPI.nii.gz')
```

```
/Users/jwpark/opt/anaconda3/lib/python3.8/site-packages/cortex/align.py:282:
UserWarning: Defaults changed in pycortex 1.3. Now automatic alignment uses
Freesurfer's bbregister and mri_coreg for initialization.
  warnings.warn("Defaults changed in pycortex 1.3. Now automatic alignment "
Running freesurfer BBR
Success
```

예상대로 해당 코드를 수정하니 잘 작동하는 모습을 볼 수 있습니다.

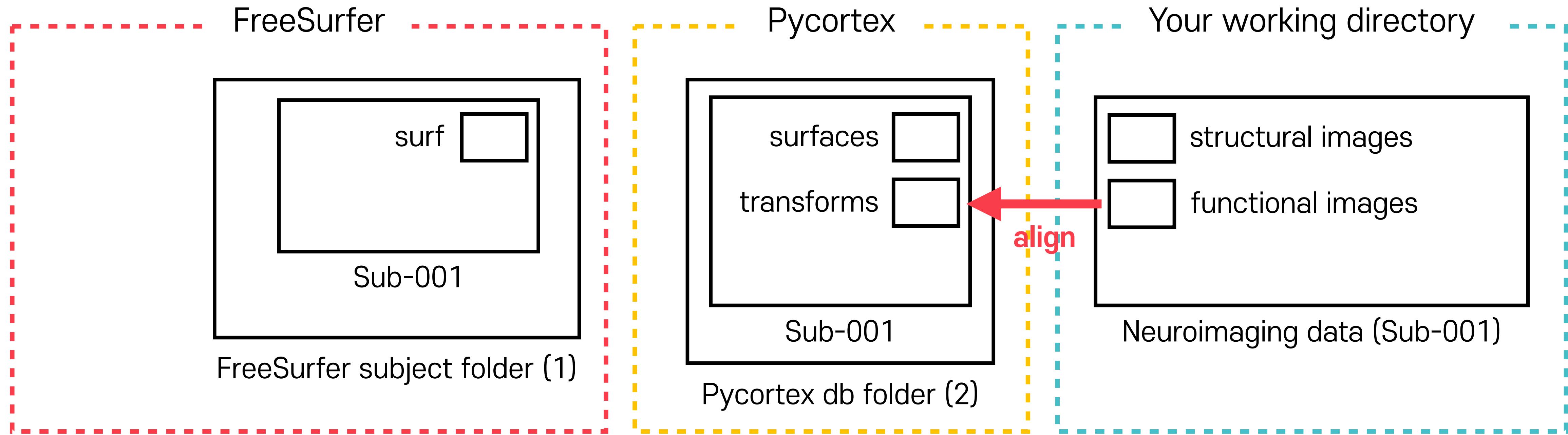
~/opt/anaconda3/lib/python3.8/site-packages/cortex/xfm.py

```
# Original code
# if len(L) == :
#     L = L[1:]
# Edited code
if len(L) > 4:
    L = L[-4:]
```

```
func_tkrvox2ras = np.array([[np.float(s) \
                           for s in ll.split() if s] \
                           for ll in L])
```

Affine transformation을
가공하는 코드를 수정하였습니다.
(issue report 함)

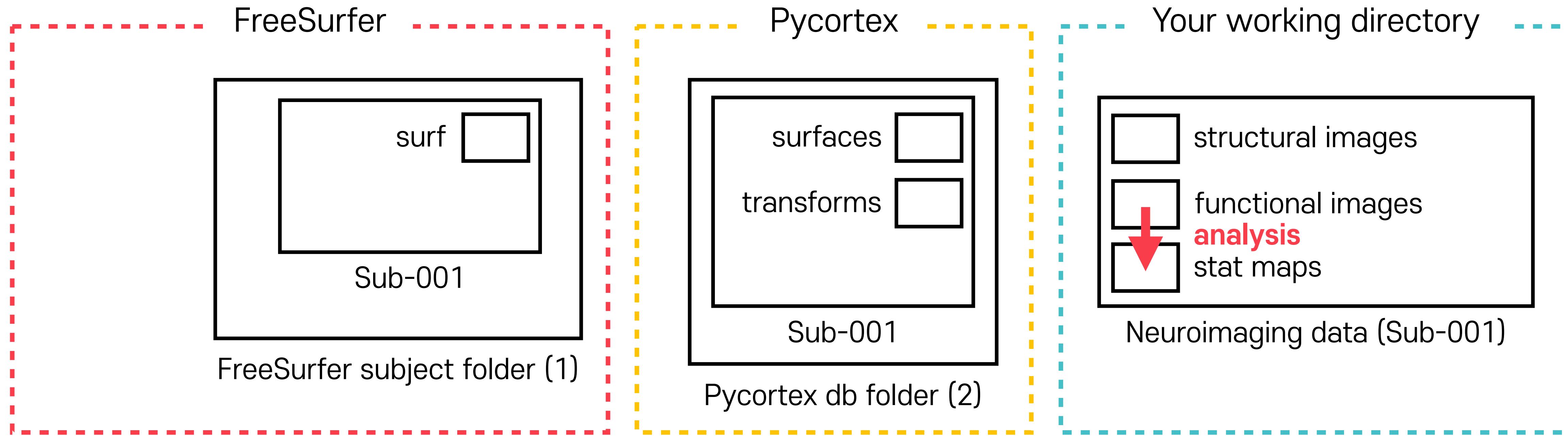
Functional image alignment



```
(base) jwpark@JWPARK /usr/local/share/pycortex/db/MNI$ tree transforms -L 2
transforms
└── fmriprep_3mm
    ├── matrices.xfm
    └── reference.nii.gz
```

DB의 transforms 폴더에 들어가 보면 <fmriprep_3mm>라는 이름의 transform이 생성 된 것을 볼 수 있습니다.
이제 해당 transform을 이용해서 [65 x 77 x 65] size의 임의의 3D array를 손쉽게 visualize 할 수 있습니다.
즉, 3D array를 NIFTI format으로 변경할 필요가 없습니다.

Example surface visualization



예시로 사용하기 위해 <Example_EPI.nii.gz> 데이터의 tSNR 값을 구하고, 이를 visualize 해 보겠습니다.

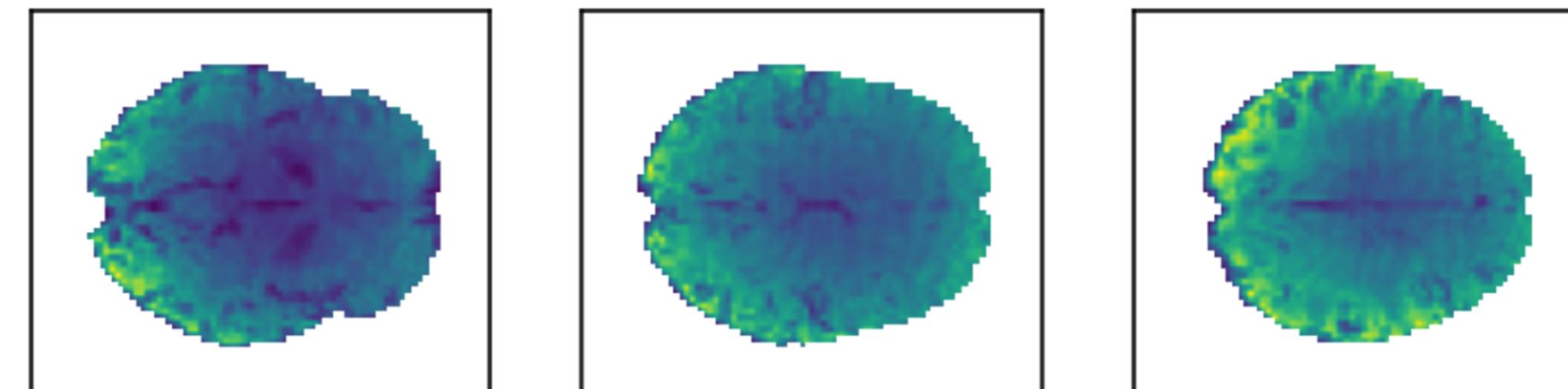
```
import numpy as np
import nibabel as nib
import matplotlib.pyplot as plt

example_EPI = nib.load("example_EPI.nii.gz").get_fdata()
example_tSNR = np.divide(np.mean(example_EPI, axis=-1),
                        np.std(example_EPI, axis=-1))
print(example_tSNR.shape)
```

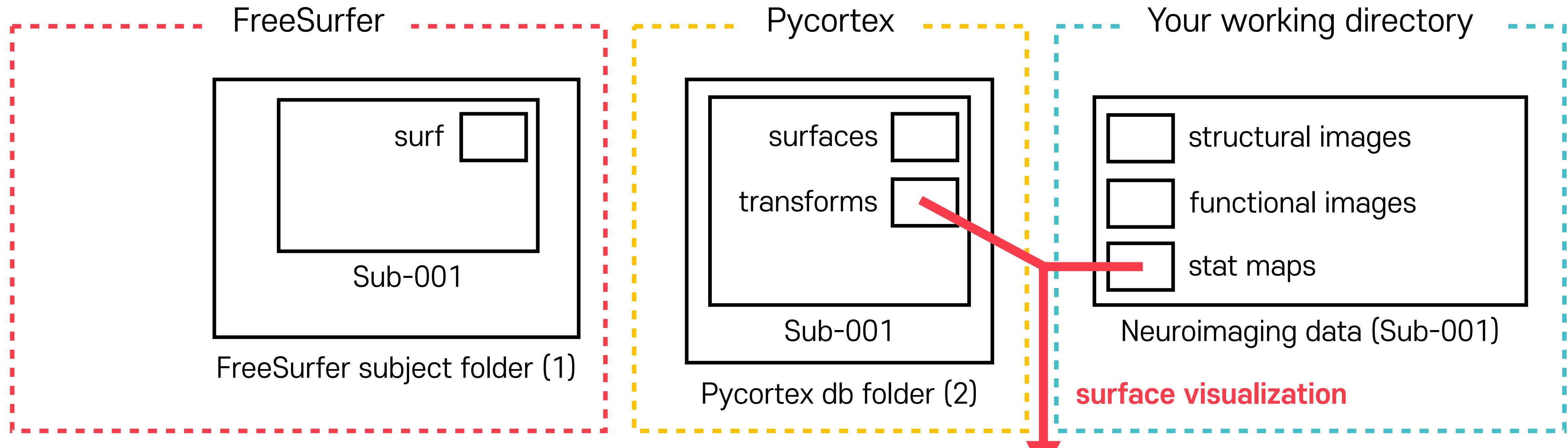
(65, 77, 65)

만들어진 example_tSNR은 각 voxel의 tSNR 값을 가진 3D array입니다.

```
fig, axes = plt.subplots(1,3, dpi=150, sharex=True, sharey=True)
for i, z in enumerate([25, 30, 35]):
    axes[i].imshow(example_tSNR[:, :, z])
    axes[i].set_xticks([])
    axes[i].set_yticks([])
```

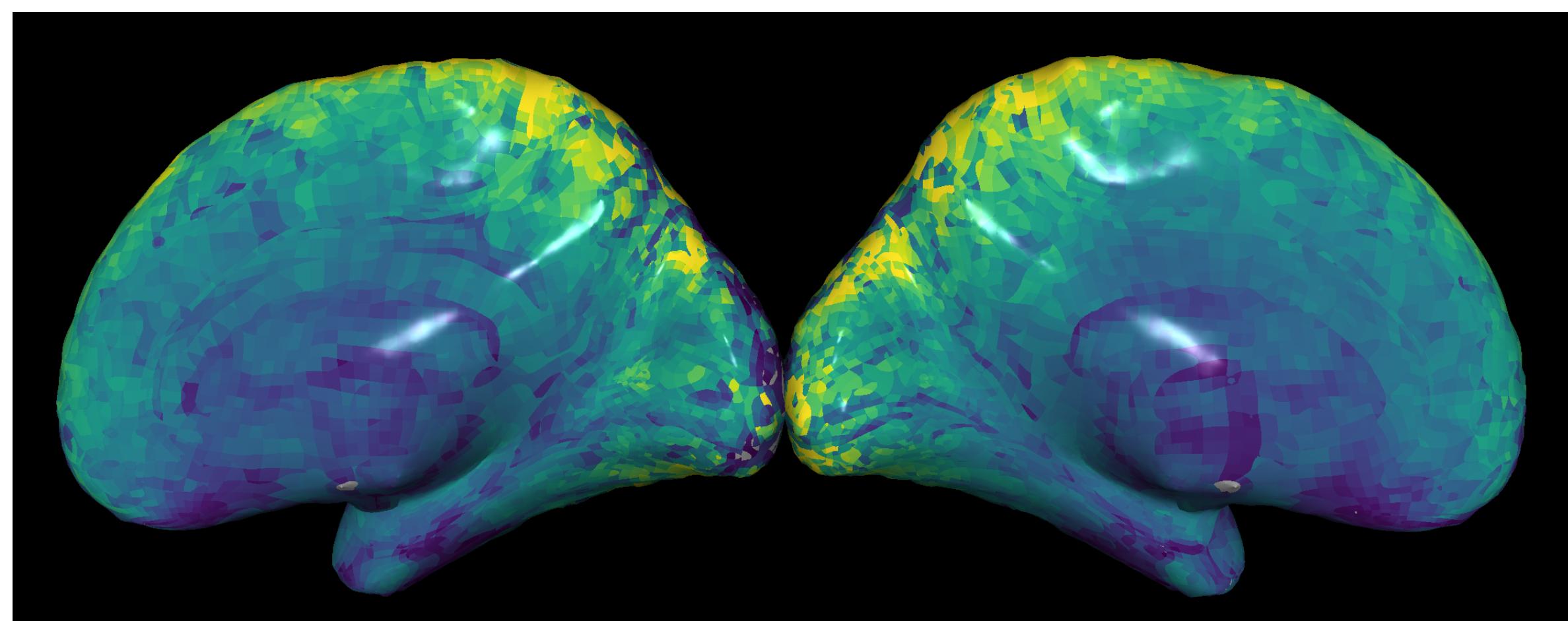


Example surface visualization



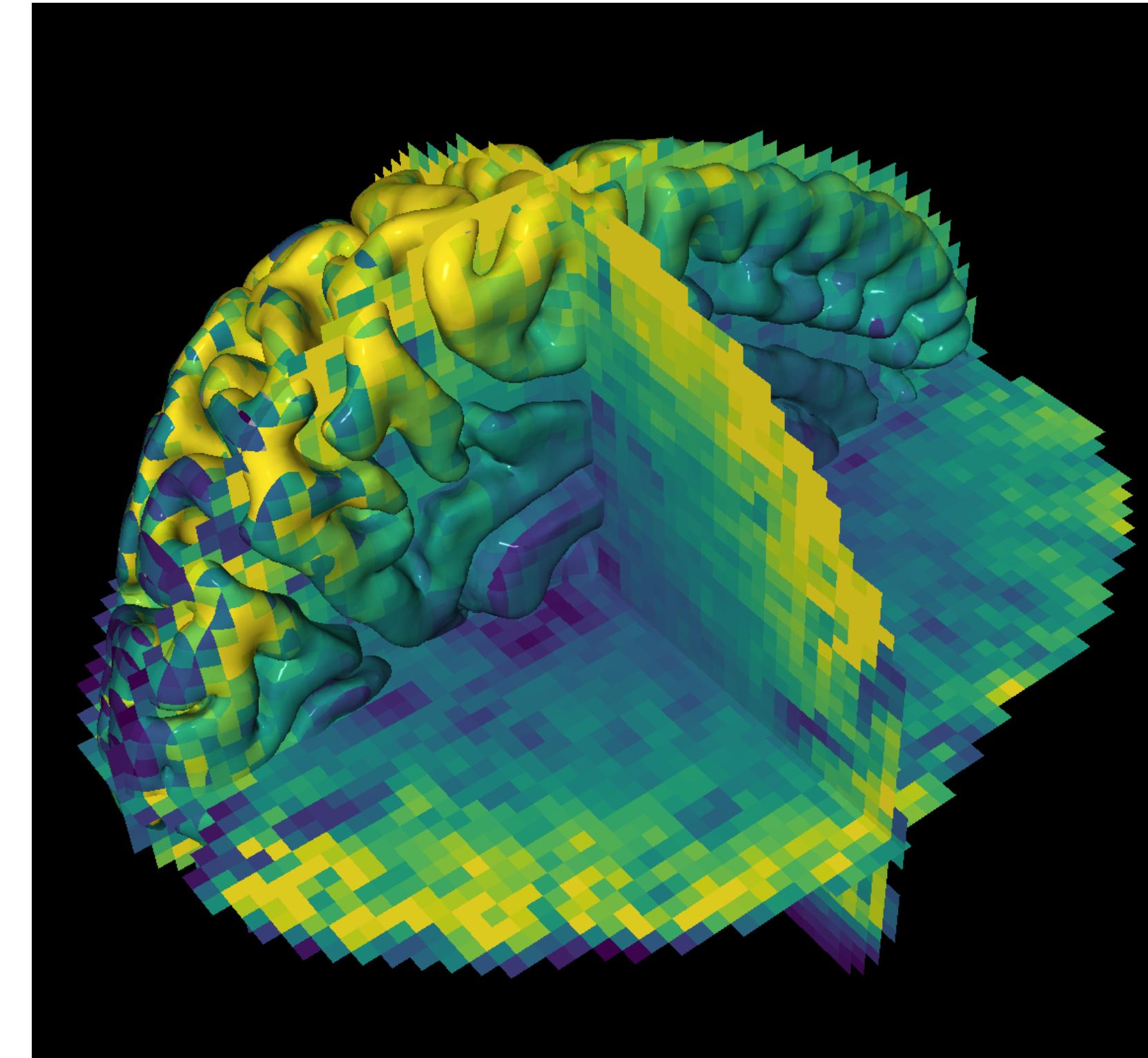
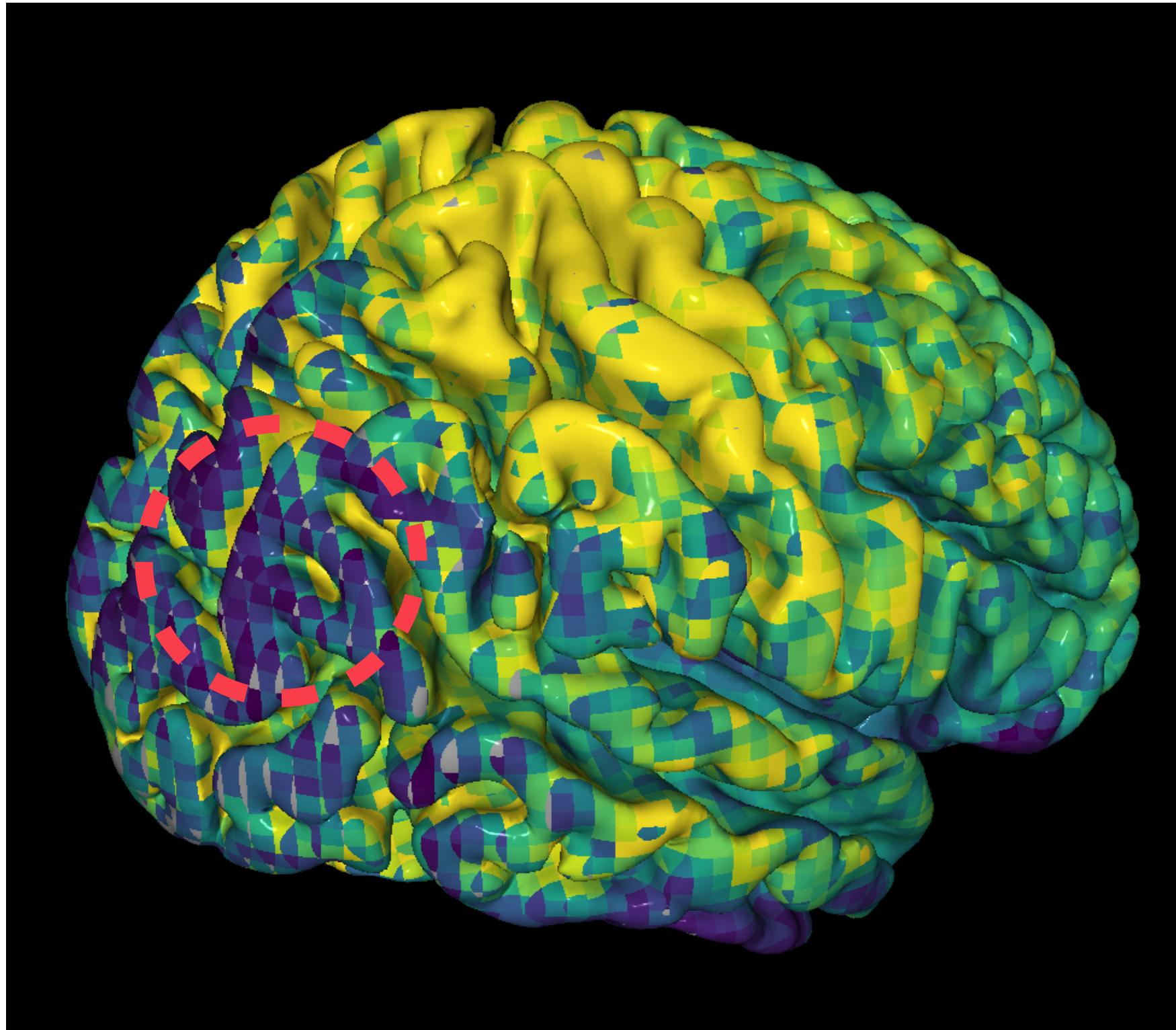
```
tSNR = cortex.Volume(example_tSNR.transpose(2,1,0),  
                      'MNI', 'fmriprep_3mm',  
                      cmap='viridis')  
cortex.webshow(tSNR)
```

Pycortex의 data axis는 Z, Y, X 축이기 때문에
.transpose(2,1,0) 메서드를 사용하여 축을 바꿔 줍니다.
<MNI>라는 surface에 <fmriprep_3mm>라는
transformation을 적용하여 visualize 합니다.



Manual alignment

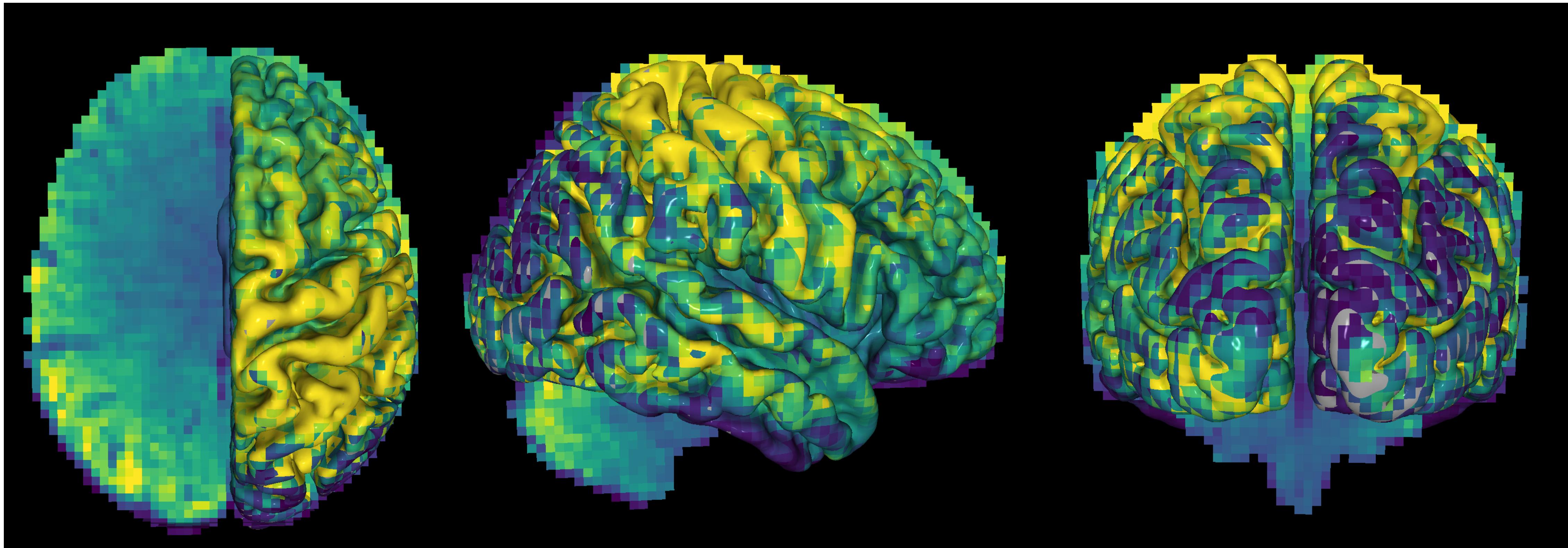
다음은 실제로 <MNI>라는 brain surface에 <fmriprep_3mm>라는 voxel-to-vertice transform의 automatic alignment가 잘 되었는지 quality check을 해야 합니다. 만약 문제가 있는 경우 조금씩 수정하여 고칠 수 있습니다.



inflate 하기 전의 surface를 보면 대략적으로 문제를 알 수 있습니다.
빨간색 원으로 표시한 occipital lobe의 tSNR 값이 낮을 수가 있는데,
아마 해당 부분의 alignment에 문제가 있는 것으로 예상 됩니다.

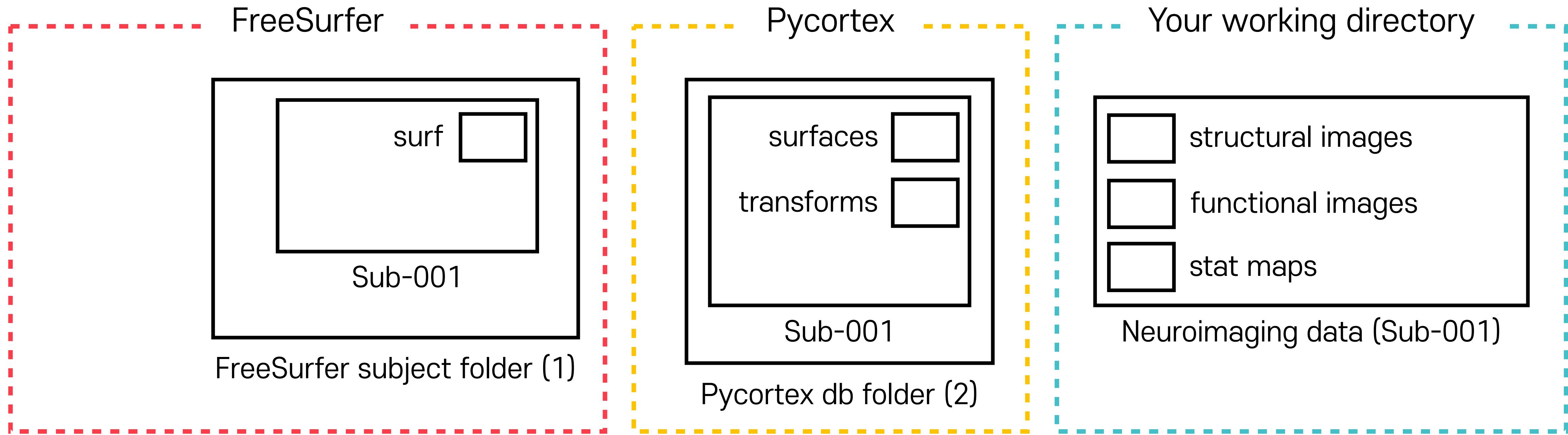
Manual alignment

다음은 실제로 <MNI>라는 brain surface에 <fmriprep_3mm>라는 voxel-to-vertice transform의 automatic alignment가 잘 되었는지 quality check을 해야 합니다. 만약 문제가 있는 경우 조금씩 수정하여 고칠 수 있습니다.



Alignment가 아주 문제가 있는것 같지는 않고, 예상대로 Occipital lobe쪽 margin이 살짝 부족한 것 같습니다.
transformation matrix의 scale을 살짝 조정하면 해결이 되지 않을까 싶습니다.

Manual alignment

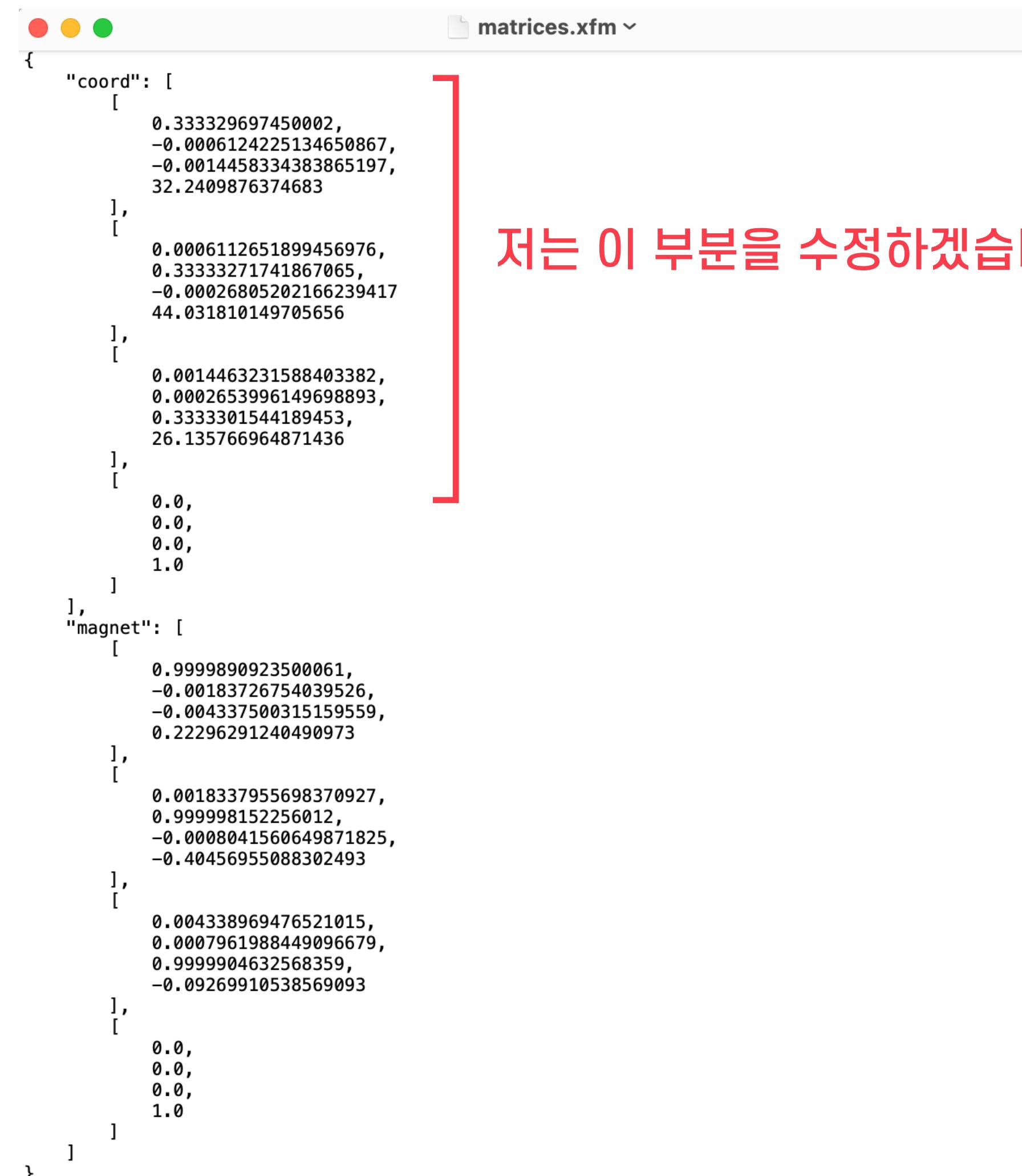


```
(base) jwpark@JWPARK /usr/local/share/pycortex/db/MNI$ tree transforms -L 2
transforms
└── fmriprep_3mm
    └── matrices.xfm ← 이 행렬을 수정하시면 됩니다.
        └── reference.nii.gz
```

Transformation matrix는 transforms 폴더의 <matrices.xfm>을 수정하시면 됩니다.
하지만 **수정하기 전에 <fmriprep_3mm> 폴더를 back up 해 두시는 것을 추천 드립니다.**
저는 <fmriprep_3mm_edit>이라는 폴더로 복사하여 이를 수정하도록 하겠습니다.

Manual alignment

<fmriprep_3mm_edit> transform 폴더의 <matrices.xfm> 파일을 텍스트 에디터로 열어 값을 좀 수정해 줍니다.



```
{  
  "coord": [  
    [ 0.33332969745002,  
      -0.0006124225134650867,  
      -0.0014458334383865197,  
      32.2409876374683  
    ],  
    [ 0.0006112651899456976,  
      0.3333271741867065,  
      -0.00026805202166239417  
      44.031810149705656  
    ],  
    [ 0.0014463231588403382,  
      0.0002653996149698893,  
      0.333301544189453,  
      26.135766964871436  
    ],  
    [ 0.0,  
      0.0,  
      0.0,  
      1.0  
    ],  
    "  
  "magnet": [  
    [ 0.999989092350061,  
      -0.00183726754039526,  
      -0.004337500315159559,  
      0.22296291240490973  
    ],  
    [ 0.0018337955698370927,  
      0.99998152256012,  
      -0.0008041560649871825,  
      -0.40456955088302493  
    ],  
    [ 0.004338969476521015,  
      0.0007961988449096679,  
      0.9999904632568359,  
      -0.09269910538569093  
    ],  
    [ 0.0,  
      0.0,  
      0.0,  
      1.0  
    ]  
  ]  
}
```

변경 전

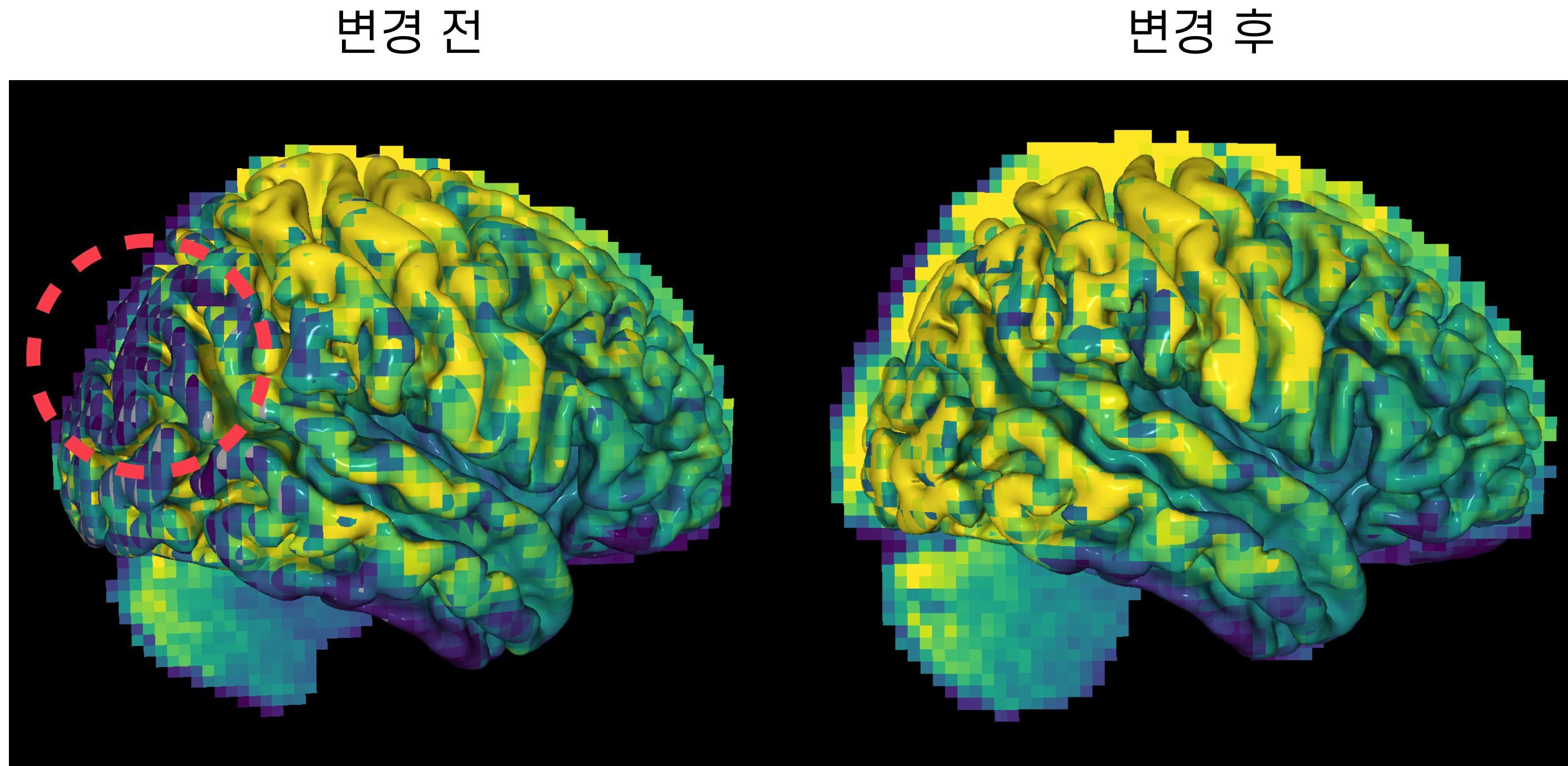
```
"coord": [  
  [ 0.33332969745002,  
    -0.0006124225134650867,  
    -0.0014458334383865197,  
    32.2409876374683  
  ],  
  [ 0.0006112651899456976,  
    0.3333271741867065,  
    -0.00026805202166239417,  
    44.031810149705656  
  ],  
  [ 0.0014463231588403382,  
    0.0002653996149698893,  
    0.333301544189453,  
    26.135766964871436  
  ],  
  [ 0.0,  
    0.0,  
    0.0,  
    1.0  
  ],  
  ]
```

변경 후

```
"coord": [  
  [ 0.32,  
    -0.0006124225134650867,  
    -0.0014458334383865197,  
    32.5  
  ],  
  [ 0.0006112651899456976,  
    0.32,  
    -0.00026805202166239417,  
    45.0  
  ],  
  [ 0.0014463231588403382,  
    0.0002653996149698893,  
    0.32,  
    26.135766964871436  
  ],  
  [ 0.0,  
    0.0,  
    0.0,  
    1.0  
  ],  
  ]
```

Manual alignment

<fmriprep_3mm_edit> transform 폴더의 <matrices.xfm> 파일을 텍스트 에디터로 열어 값을 좀 수정해 줍니다.



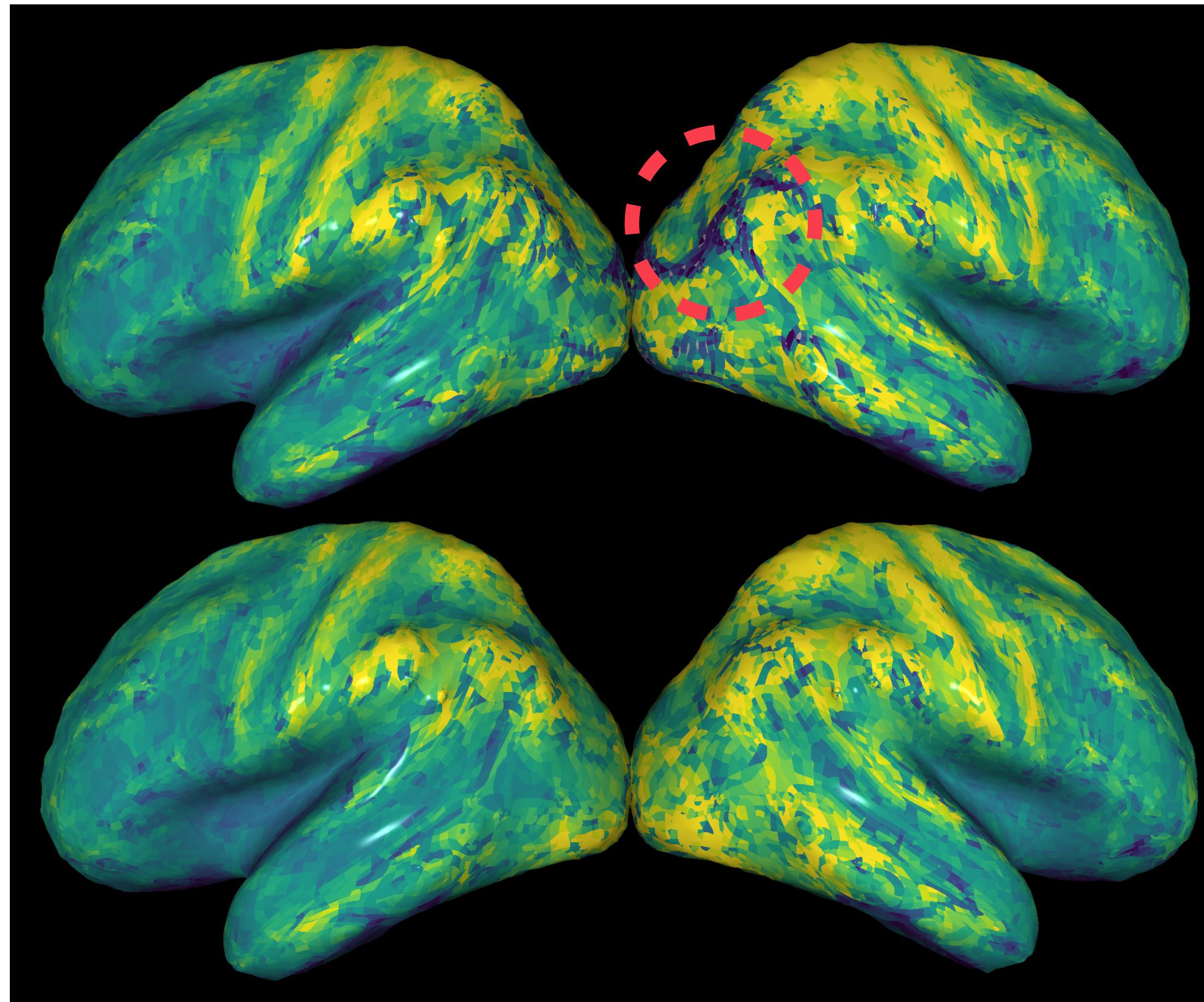
Scale factor와 translation factor를 살짝 수정하여 앞뒤로 voxel들이 surface에 잘 들어올 수 있게 조정하였습니다.
전반적으로 margin을 주게 되면 깔끔한 surface plot을 얻을 수 있지만 반대로 너무 큰 margin을 주게 되면
brain 안쪽(white matter or sulcus)의 mapping이 정확하지 않게 됩니다.

여러분이 Pycortex를 실제 연구용으로 사용하신다면 훨씬 더 신경 써서 alignment를 수정하시길 권장합니다.

Manual alignment

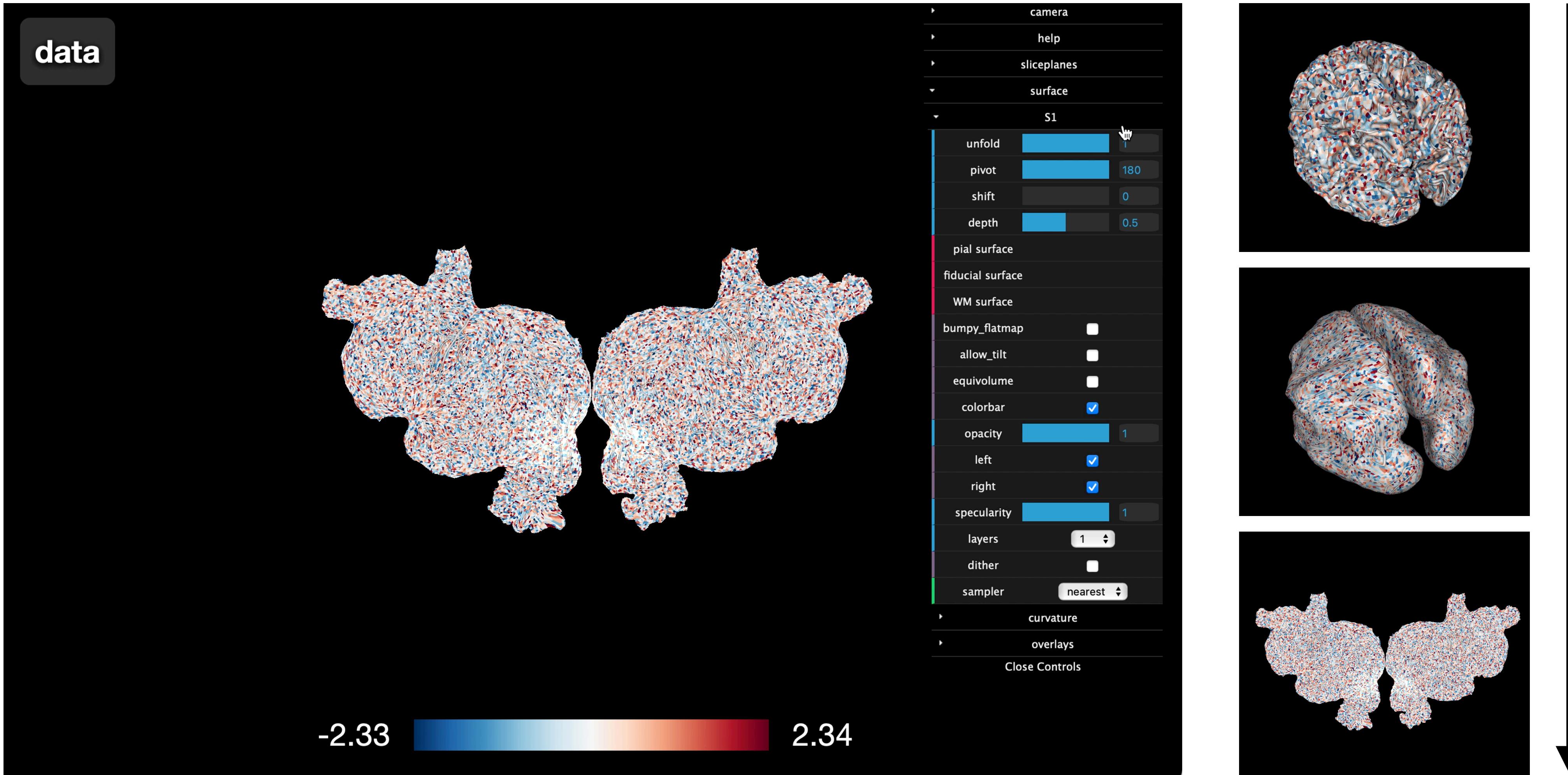
<fmriprep_3mm_edit> transform 폴더의 <matrices.xfm> 파일을 텍스트 에디터로 열어 값을 좀 수정해 줍니다.

변경 전



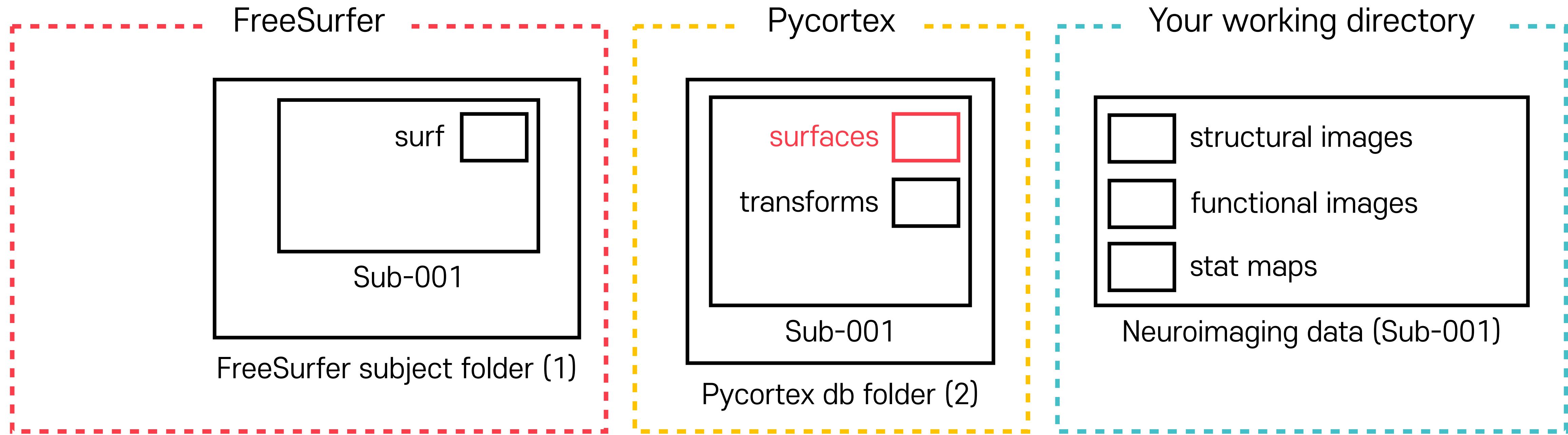
돌출된 Gyrus를 따라 발생하는 전형적인 alignment error가 수정된 것을 볼 수 있습니다. (surface가 너무 작아서 생김)
반대로 margin이 너무 커서 발생하는 alignment error는 sulcus에서 볼 수 있습니다. (surface가 너무 커서 생김)

Cutting and Flattening



Folded brain을 inflated surface로 바꾸는 것은 FreeSurfer의 default surface segmentation(recon-all) 기능으로 가능하지만, cortical surface를 잘라 평면으로 만드는 것은 추가적인 작업이 필요합니다.

Cutting and Flattening



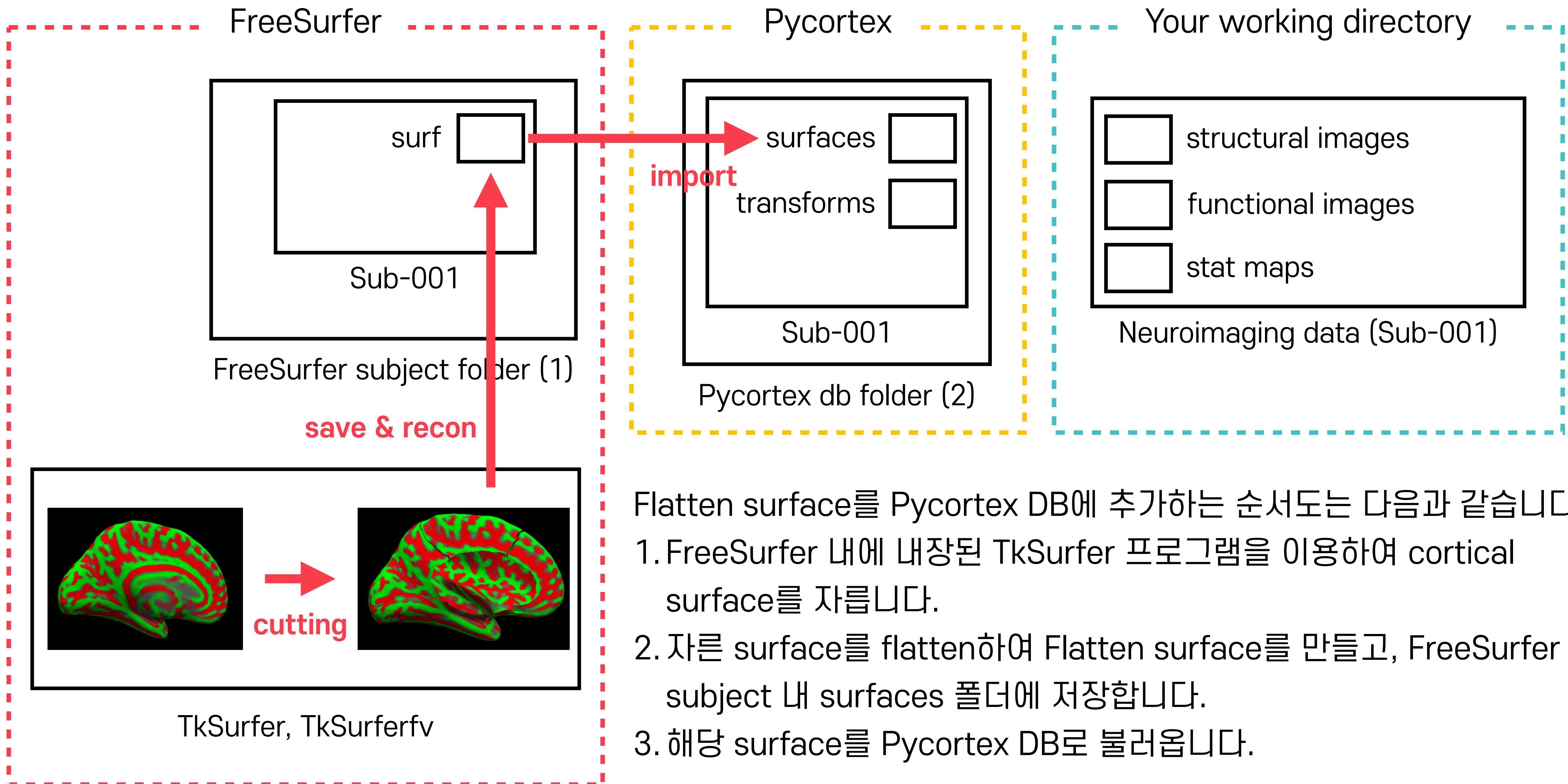
Pycortex db의 surfaces 폴더에 <flat_lh.gii>, <flat_rh.gii> 파일이 없으면 flatten 되지 않습니다.

```
(base) jwpark@JWPARK /usr/local/share/pycortex/db$ tree S1/surfaces
S1/surfaces
├── flat_lh.gii
├── flat_rh.gii
├── inflated_lh.gii
├── inflated_rh.gii
├── pia_lh.gii
├── pia_rh.gii
└── wm_lh.gii
    └── wm_rh.gii
```

Example subject인
S1의 경우 surfaces 폴더에
<flat_lh.gii>, <flat_rh.gii>
파일이 존재합니다.

```
(base) jwpark@JWPARK /usr/local/share/pycortex/db$ tree MNI/surfaces
MNI/surfaces
├── inflated_lh.gii
├── inflated_rh.gii
├── pia_lh.gii
├── pia_rh.gii
└── wm_lh.gii
    └── wm_rh.gii
```

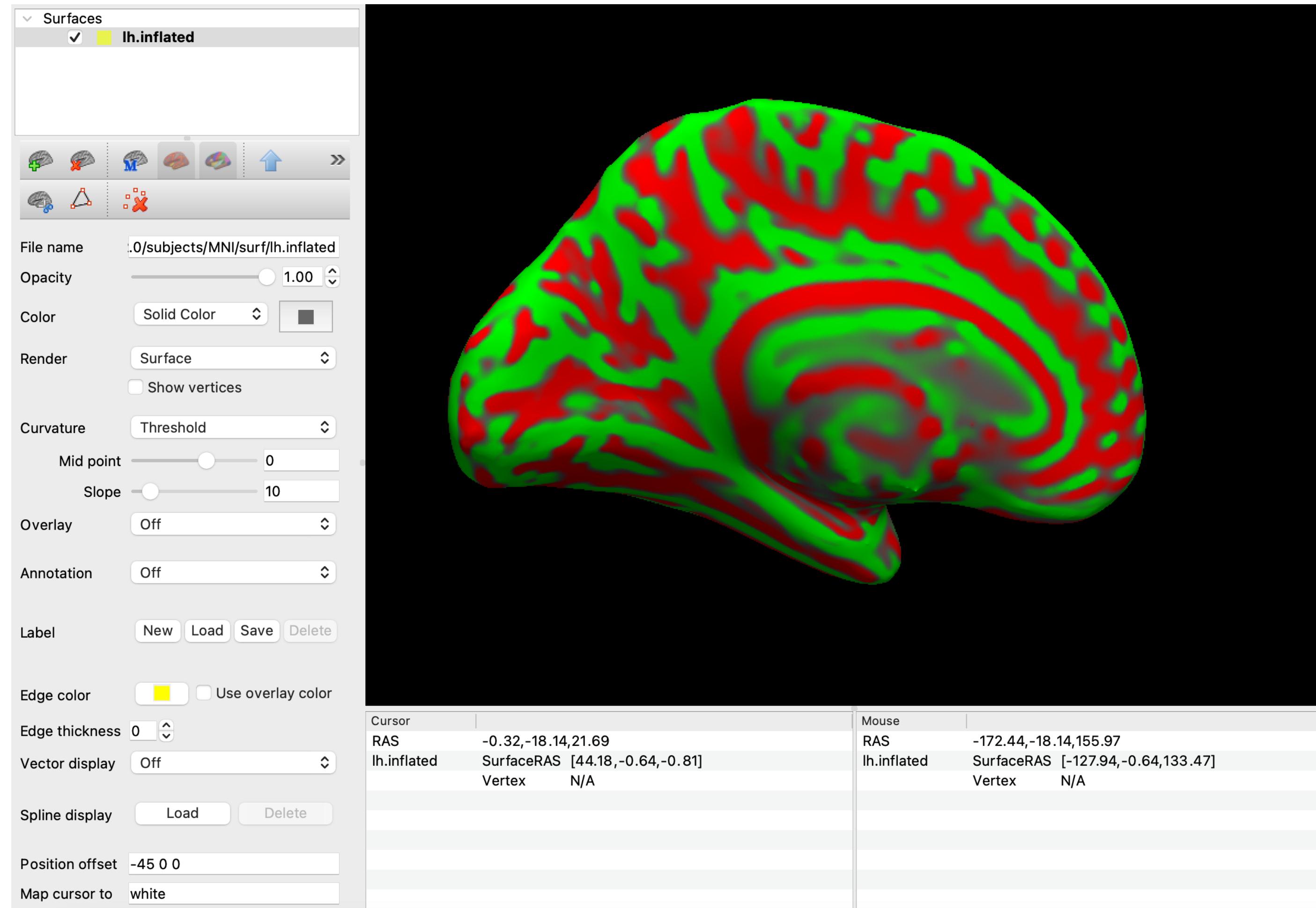
Cutting and Flattening



Cutting and Flattening

macOS의 FreeSurfer에서는 더이상 TkSurfer를 제공하지 않고, TkSurferfv 프로그램을 제공합니다.

tksurferfv MNI lh inflated 명령어를 입력하여 <MNI> subject의 Left inflated surface를 불러 올 수 있습니다.

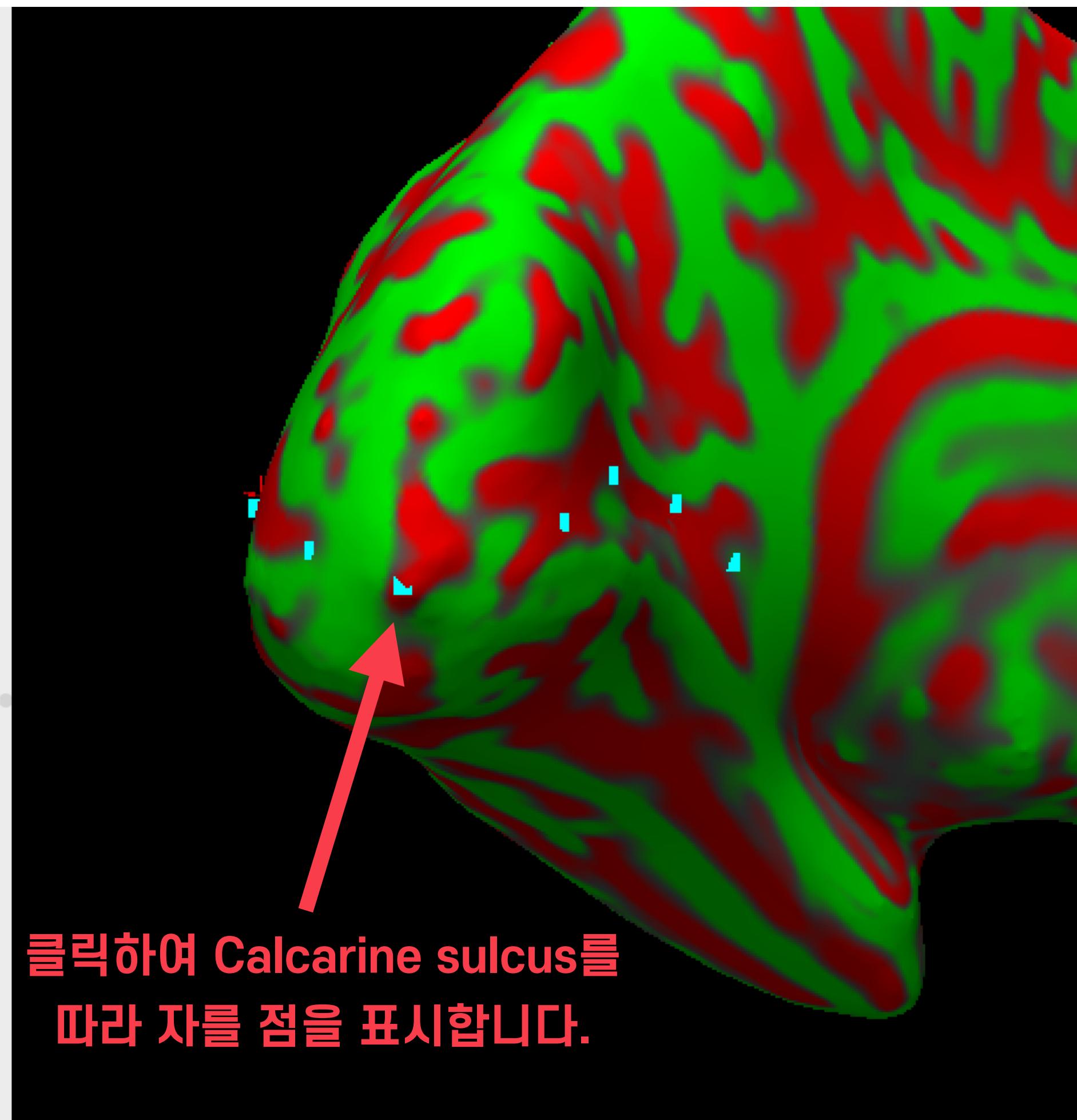
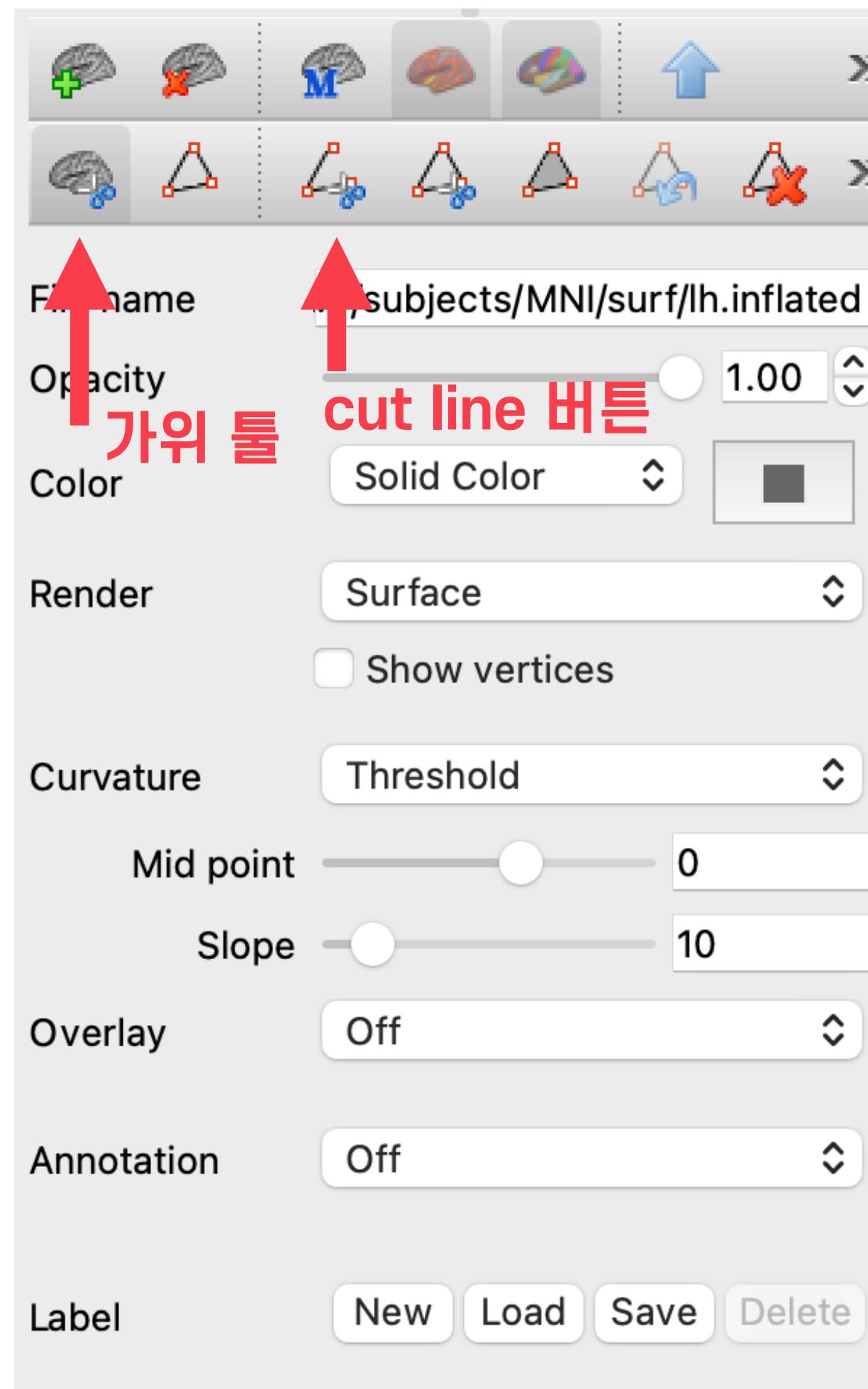


Cutting and Flattening

먼저 calcarine sulcus를 따라 surface를 잘라 보겠습니다.

가위 툴을 선택 후, surface에 몇개의 점을 표시합니다. (가위 툴을 선택하면 점이 하늘색으로 찍힘)

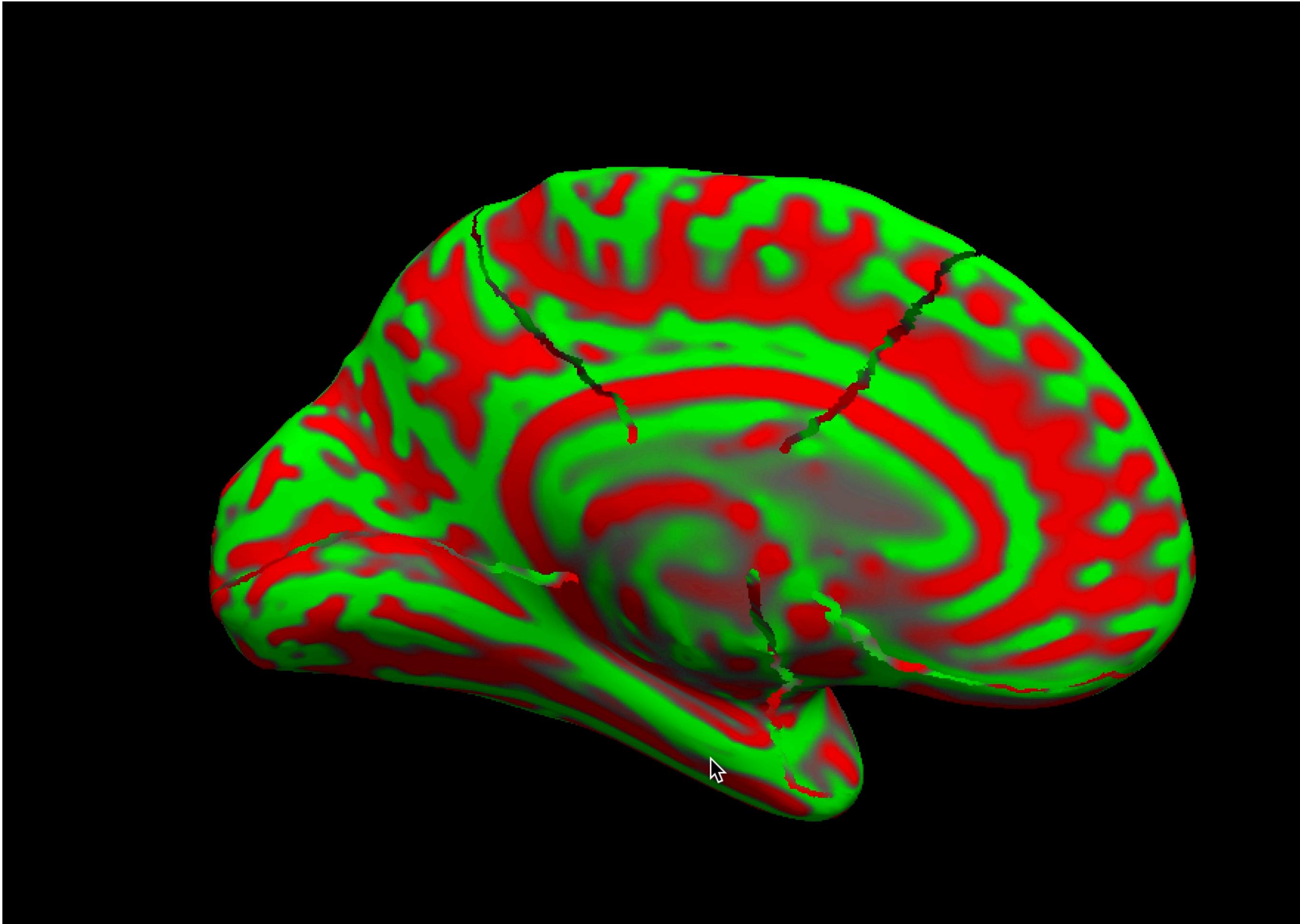
이후 <cut line> 버튼을 클릭하면 해당 선을 따라 자를 수 있습니다.



Cutting and Flattening

같은 방식으로 corpus callosum을 가로질러 mid brain 영역을 잘라 줍니다.

자르는 곳이 calcarine sulcus 만큼 명확하지는 않은데, 몇 번 잘라 보시면 감이 옵니다… 😅😅😅😅

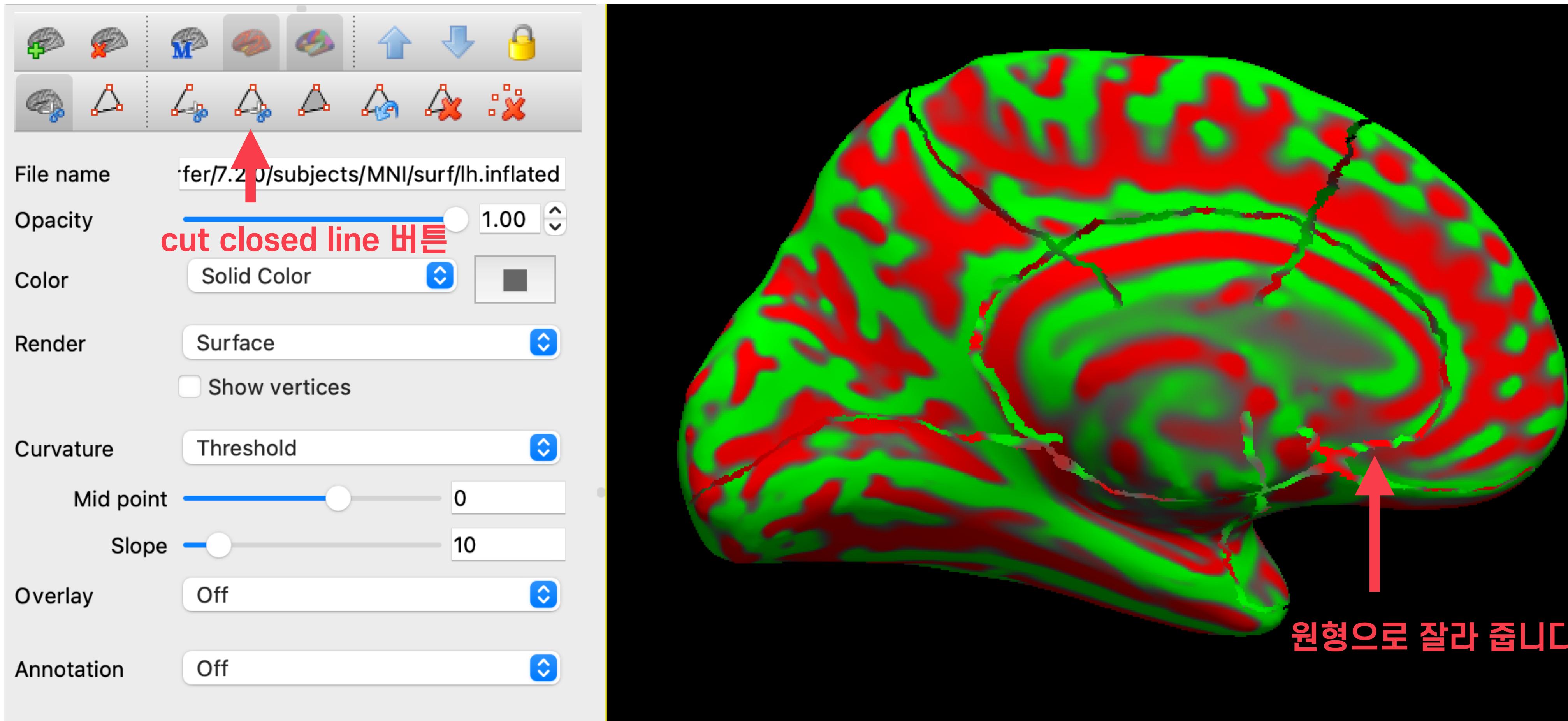


너무 medial 쪽만 cutting 하지 마시고 바깥쪽을 같이 잘라 주시면 예쁘게 잘립니다.

Cutting and Flattening

다음은 corpus callosum을 둘러 mid brain 안쪽을 잘라 줍니다.

이번엔 선으로 자르는 것이 아니기 때문에 <cut line>이 아닌 <cut closed line> 버튼을 사용합니다.

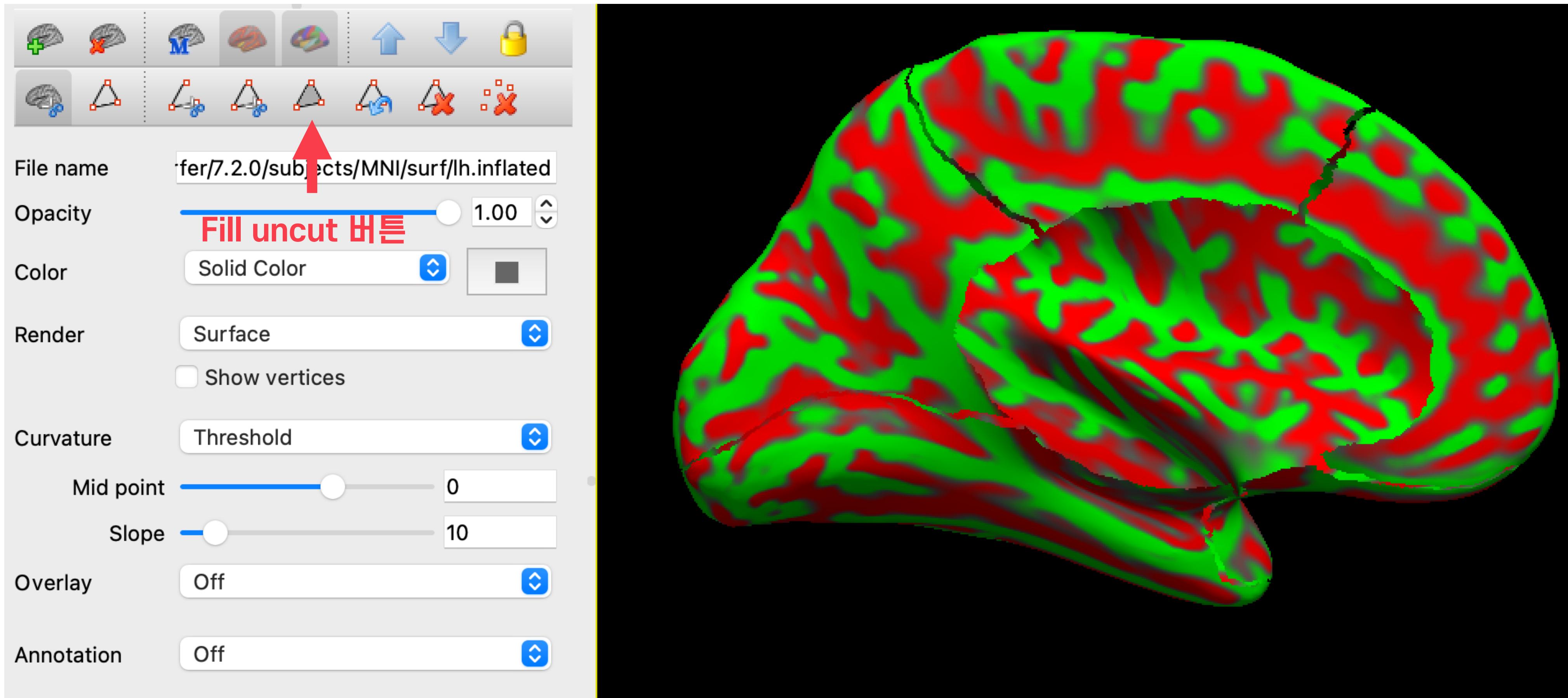


원형으로 자를때 이미 잘린 부분을 클릭하지 않도록 주의하세요, 반대쪽(lateral) surface가 클릭 됩니다.

Cutting and Flattening

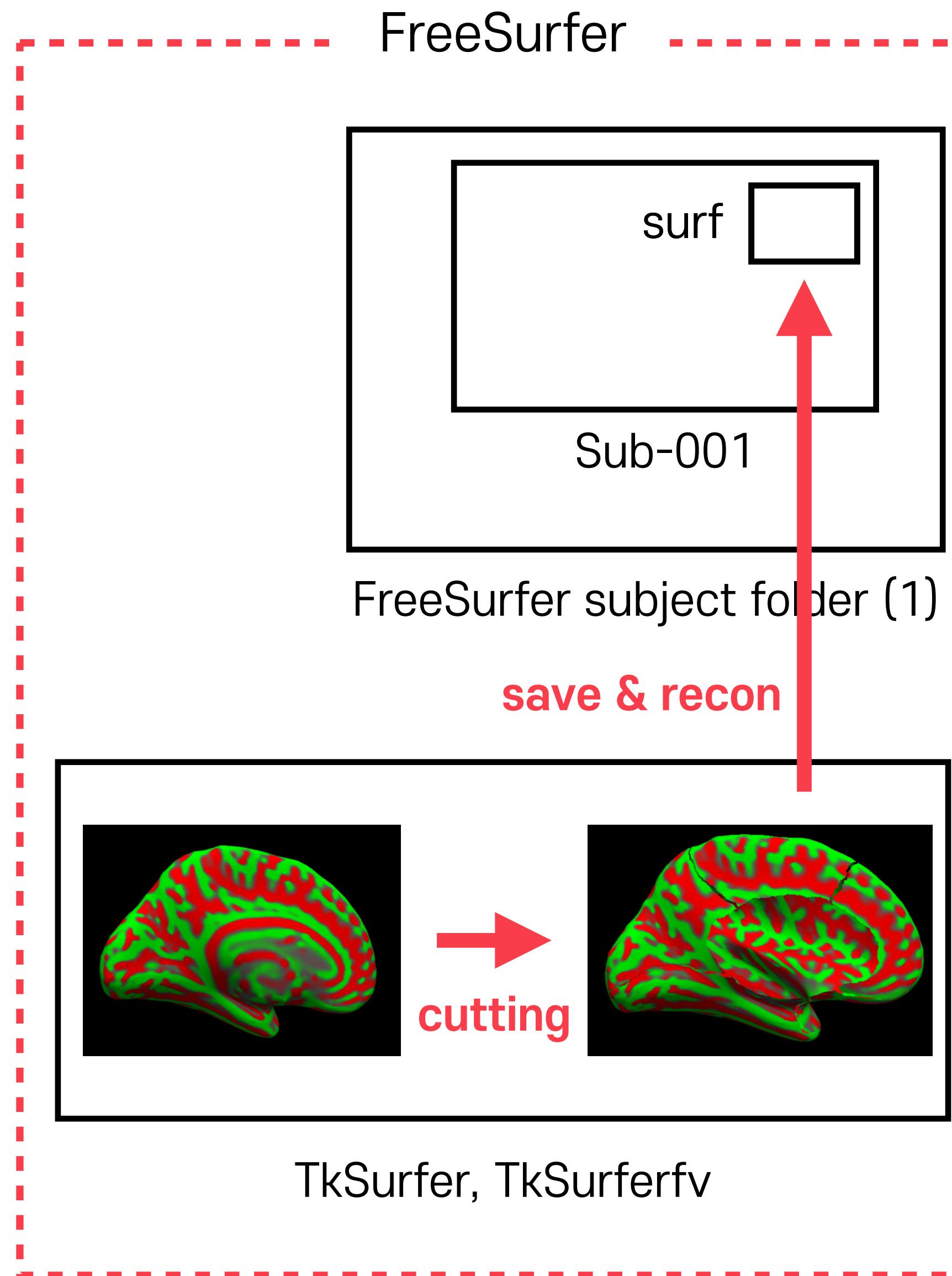
다음은 corpus callosum을 둘러 mid brain 안쪽을 잘라 줍니다.

<Fill uncut> 버튼을 눌러 closed line을 면으로 제거합니다.



너무 medial 쪽만 cutting 하지 마시고 살짝 바깥쪽을 같이 잘라 주시면 예쁘게 잘립니다.

Cutting and Flattening



이제 자른 surface를 <lh.full.patch.3d> 라는 이름으로
FreeSurfer의 surf 폴더에 저장 해 줍니다.
같은 방법으로 Right hemisphere도 cutting 해서 저장 합니다.



Cutting and Flattening

Cutting 후 저장한 <lh.full.patch.3d>, <rh.full.patch.3d> 파일을 FreeSurfer에 내장된 <mris_flatten> 함수를 사용하여 <lh.full.flat.patch.3d>와 <rh.full.flat.patch.3d> 파일로 변환해 줍니다.

```
mris_flatten /usr/local/freesurfer/subjects/MNI/surf/lh.full.patch.3d \
              /usr/local/freesurfer/subjects/MNI/surf/lh.full.flat.patch.3d
```

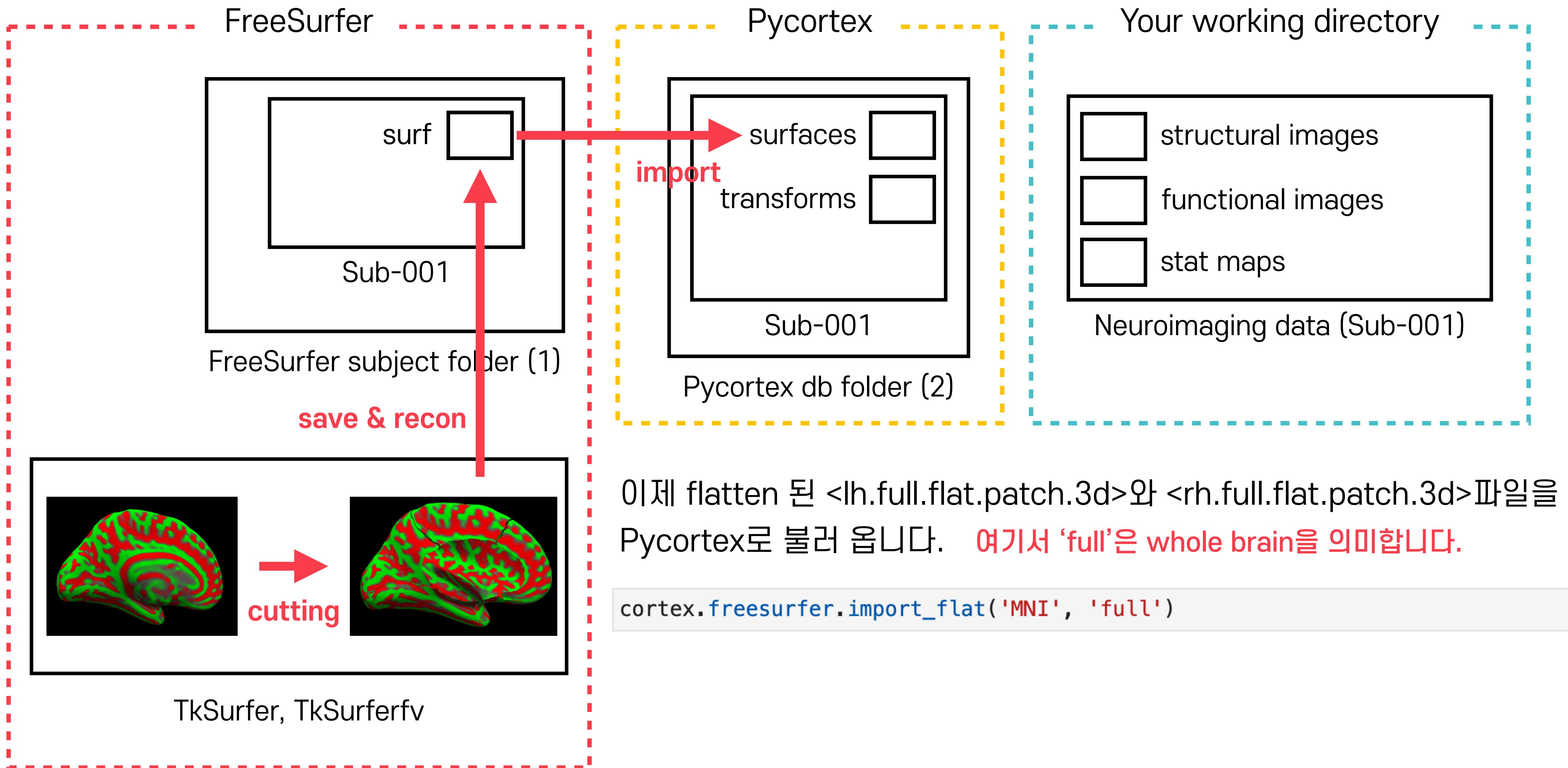
```
mris_flatten /usr/local/freesurfer/subjects/MNI/surf/rh.full.patch.3d \
              /usr/local/freesurfer/subjects/MNI/surf/rh.full.flat.patch.3d
```

<mris_flatten> 작업은 각 hemisphere 당 약 2시간 정도가 소요됩니다.

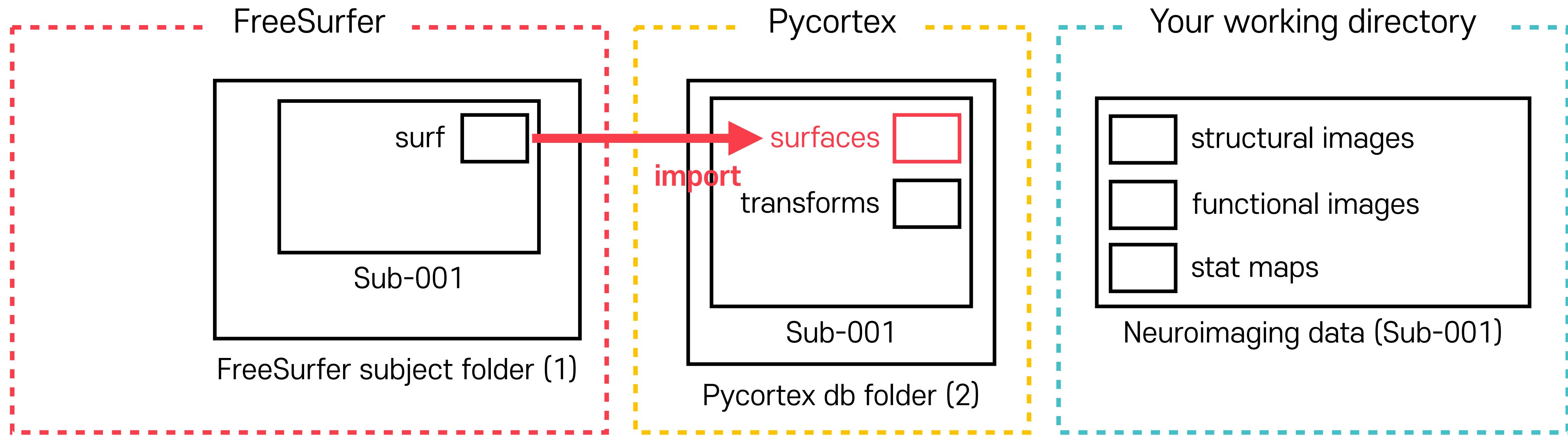
```
(base) jwpark@JWPARK /Applications/freesurfer/7.2.0/subjects/MNI/surf$ ll *patch.3d
-rw-r--r--@ 1 jwpark  staff  2.2M Feb  8  2021 lh.full.flat.patch.3d
-rw-r--r--@ 1 jwpark  staff  2.2M Feb  8  2021 lh.full.patch.3d
-rw-r--r--@ 1 jwpark  staff  2.3M Feb  8  2021 rh.full.flat.patch.3d
-rw-r--r--@ 1 jwpark  staff  2.3M Feb  8  2021 rh.full.patch.3d
```

FreeSurfer subject 폴더에 flatten surface들이 생성 된 것을 볼 수 있습니다.

Cutting and Flattening



Cutting and Flattening



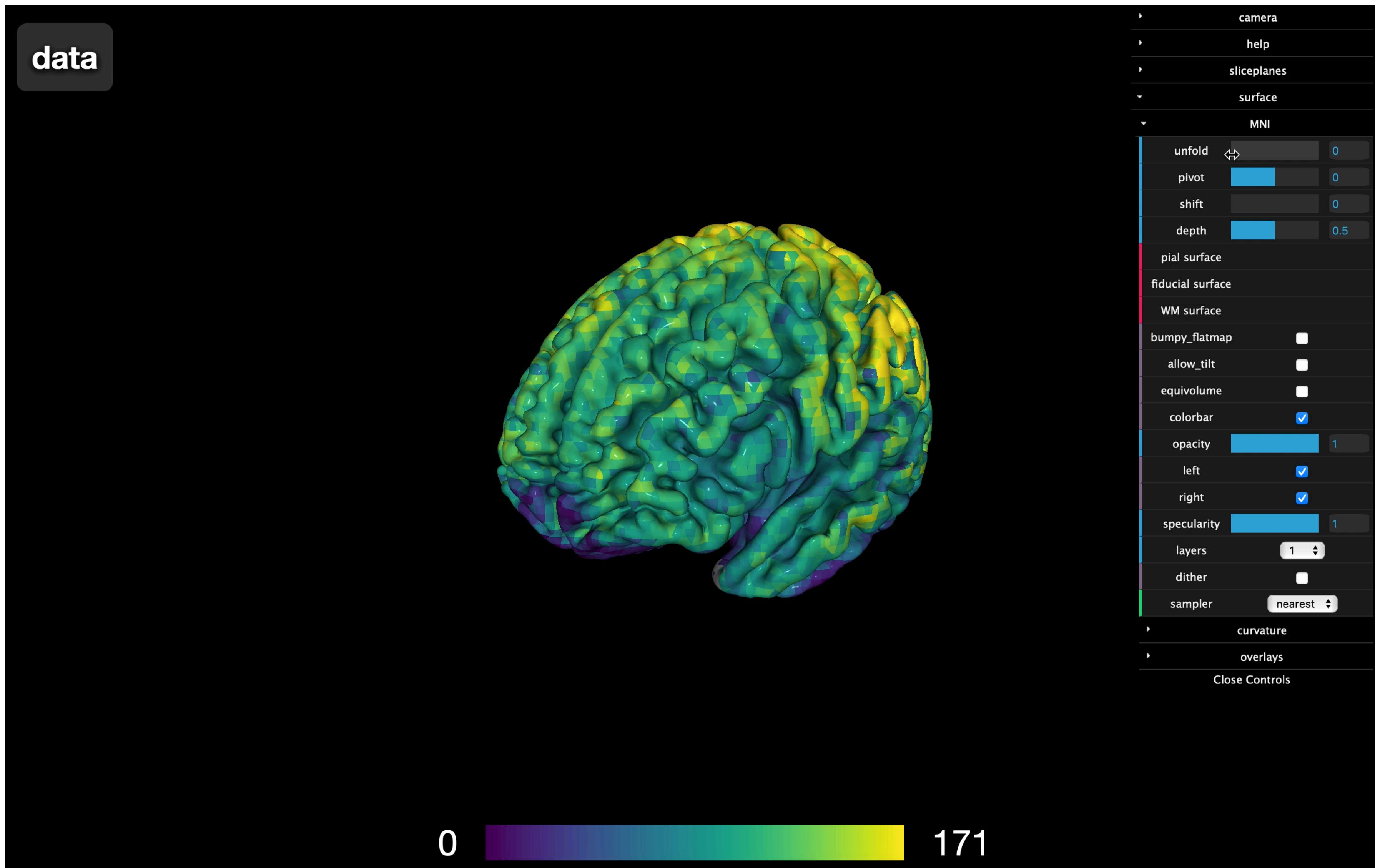
Pycortex db의 surfaces 폴더에 <flat_lh.gii>, <flat_rh.gii> 파일이 생성 된 것을 확인합니다.

```
(base) jwpark@JWPARK /usr/local/share/pycortex/db$ tree MNI/surfaces
MNI/surfaces
├── inflated_lh.gii
├── inflated_rh.gii
├── pia_lh.gii
├── pia_rh.gii
└── wm_lh.gii
    └── wm_rh.gii
```

```
(base) jwpark@JWPARK /usr/local/share/pycortex/db$ tree MNI/surfaces
MNI/surfaces
├── flat_lh.gii
├── flat_rh.gii
├── inflated_lh.gii
├── inflated_rh.gii
├── pia_lh.gii
├── pia_rh.gii
└── wm_lh.gii
    └── wm_rh.gii
```

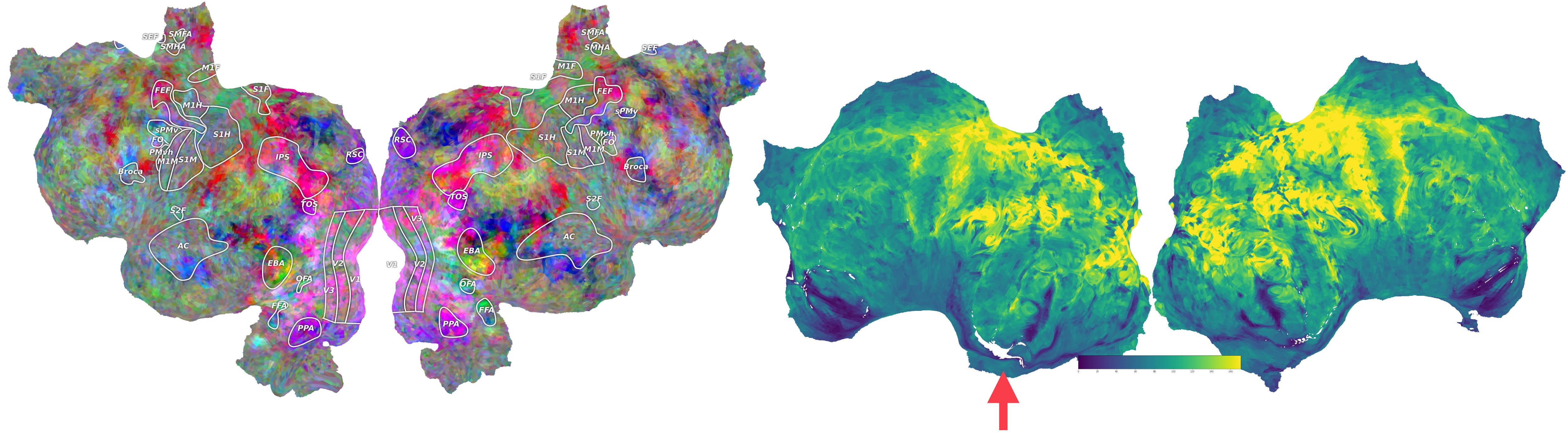
MNI subject에
<flat_lh.gii>, <flat_rh.gii>
파일이 생성 된 것을 볼 수 있습니다.

Cutting and Flattening



이제 unfold slider를 최대로 올리면 flatten surface로 visualization 할 수 있습니다.

Defining surface ROIs



alignment가 살짝 안맞는 부분
(값이 정의되지 않아서 투명하게 나옴)

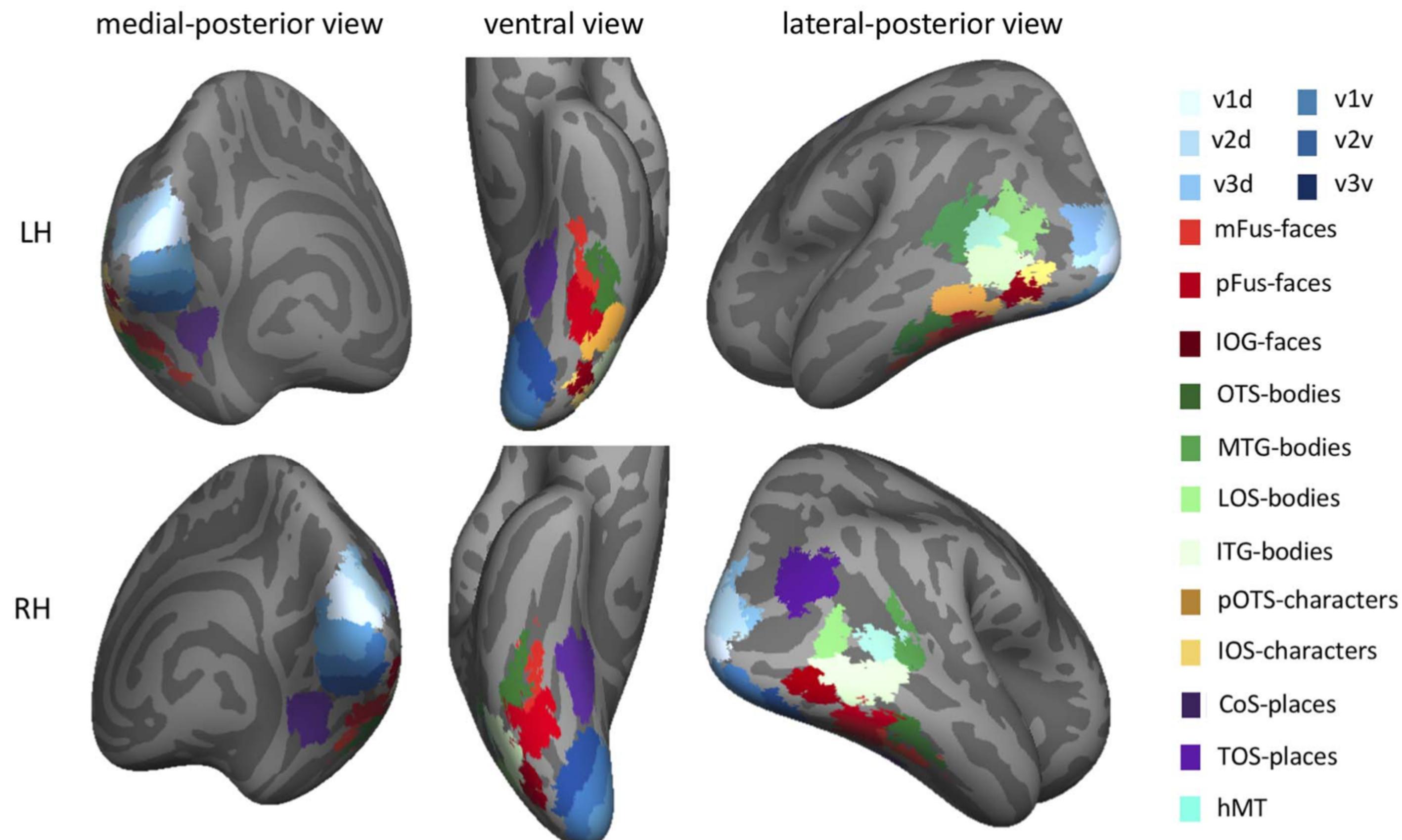
Pycortex의 Example subject(S1)의 flatten map과 비교해 보면 몇 가지 문제점이 있습니다.

1. 자른 모양이 균일하지 않습니다. (temporal lobe에 왜곡이 있네요)
2. Temporal lobe 쪽에 아까 한 manual alignment가 살짝 맞지 않는 부분이 있네요.
3. ROI가 표시되지 않습니다.

Defining surface ROIs

Pycortex에서 surface ROI를 define하는 부분은 FreeSurfer의 ROI format인 <annot>이나 <label> 파일을 바로 불러 올 수 없습니다. volume-ROI를 surface에 올린 뒤, vector-drawing tool인 Inkscape를 이용해 직접 그려야 합니다.

오늘은 예시로 Mona Rosenke (2021) 연구에 소개된 visfAtlas의 일부 ROI를 그려 보겠습니다.



Mona Rosenke et al. (2021); download.brainvoyager.com/data/visfAtlas.zip

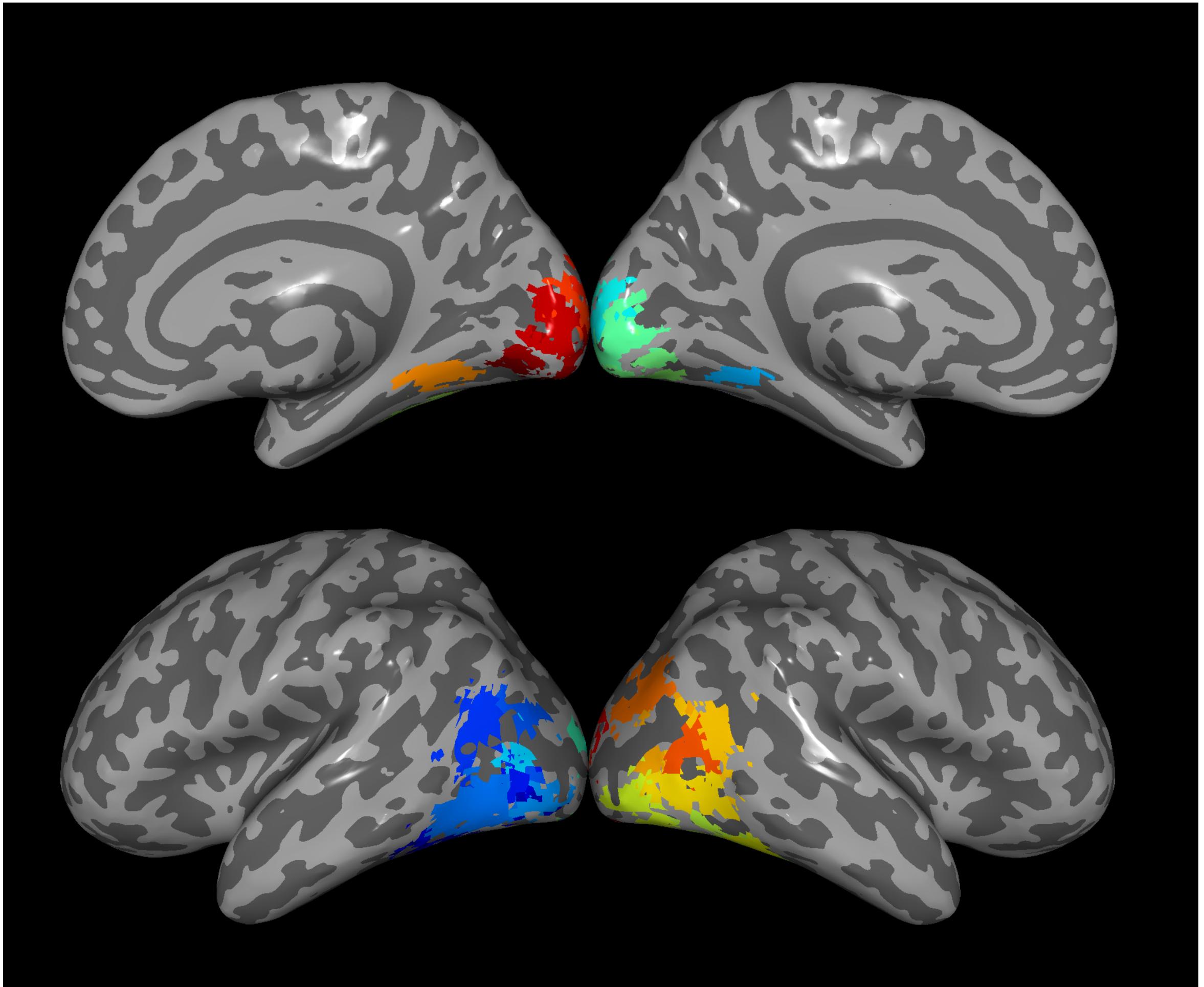
Defining surface ROIs

visfAtlas를 분석하고자 하는 EPI의 FoV, spatial resolution에 맞게 resample 하고 난 후,
이를 Pycortex로 visualize 해 봅니다.

```
import numpy as np
import nibabel as nib
import matplotlib.pyplot as plt

visfAtlas = nib.load("visfAtlas_fmriprep.nii.gz").get_fdata()
visfAtlas[visfAtlas==0.0] = np.nan

visfAtlas_volume = cortex.Volume(visfAtlas.transpose(2,1,0),
                                 'MNI', 'fmriprep_3mm_edit',
                                 cmap='jet', vlim=1, vmax=33)
cortex.webshow(visfAtlas_volume)
```



저해상도로 resampling을 안했으면 보기에 더 좋았을텐데,
그럼 고해상도 alignment를 다시 해야 해서 그냥 이렇게 하겠습니다.

Defining surface ROIs

index가 27번인 right hemisphere의 MT ROI를 그려 보겠습니다. <add_roi> 메서드를 사용합니다.

```
visfAtlas_MT_rh = np.zeros(visfAtlas.shape)
visfAtlas_MT_rh[visfAtlas==27] = 1.0

visfAtlas_MT_rh_volume = cortex.Volume(visfAtlas_MT_rh.transpose(2,1,0),
                                         'MNI', 'fmriprep_3mm_edit')

cortex.utils.add_roi(visfAtlas_MT_rh_volume, name='MT')
```

index가 26번이라 표기되어 있는데
이는 기준이 0번부터 시작하니
27번을 사용합니다.

```
<label index="0" x="67" y="38" z="27">lh_mFus_faces</label>
<label index="1" x="64" y="40" z="27">lh_pFus_faces</label>
<label index="2" x="64" y="34" z="27">lh_IOG_faces</label>
<label index="3" x="63" y="22" z="29">lh_OTS_bodies</label>
<label index="4" x="67" y="38" z="27">lh_ITG_bodies</label>
<label index="5" x="68" y="26" z="35">lh_MTG_bodies</label>
<label index="6" x="69" y="32" z="41">lh_LoS_bodies</label>
<label index="7" x="65" y="23" z="41">lh_pOTS_characters</label>
<label index="8" x="66" y="31" z="30">lh_IoS_characters</label>
<label index="9" x="63" y="21" z="31">lh_CoS_places</label>
<label index="10" x="58" y="39" z="32">lh_hMT_motion</label>
<label index="11" x="66" y="27" z="39">lh_v1d_retinotopic</label>
<label index="12" x="48" y="15" z="36">lh_v2d_retinotopic</label>
<label index="13" x="52" y="12" z="41">lh_v3d_retinotopic</label>
<label index="14" x="57" y="15" z="41">lh_v1v_retinotopic</label>
<label index="15" x="47" y="19" z="33">lh_v2v_retinotopic</label>
<label index="16" x="49" y="22" z="30">lh_v3v_retinotopic</label>
<label index="17" x="54" y="23" z="29">rh_mFus_faces</label>
<label index="18" x="24" y="43" z="26">rh_pFus_faces </label>
<label index="19" x="24" y="33" z="28">rh_IOG_faces</label>
<label index="20" x="25" y="22" z="30">rh_OTS_bodies</label>
<label index="21" x="23" y="39" z="28">rh_ITG_bodies</label>
<label index="22" x="21" y="28" z="33">rh_MTG_bodies</label>
<label index="23" x="19" y="32" z="40">rh_LoS_bodies</label>
<label index="24" x="21" y="26" z="40">rh_CoS_places</label>
<label index="25" x="31" y="40" z="32">rh_ToS_places</label>
<label index="26" x="27" y="22" z="46">rh_hMT_motion</label>
<label index="27" x="22" y="29" z="38">rh_v1d_retinotopic</label>
<label index="28" x="40" y="15" z="37">rh_v2d_retinotopic</label>
<label index="29" x="38" y="14" z="42">rh_v3d_retinotopic</label>
<label index="30" x="32" y="15" z="44">rh_v1v_retinotopic</label>
<label index="31" x="41" y="20" z="35">rh_v2v_retinotopic</label>
<label index="32" x="40" y="22" z="32">rh_v3v_retinotopic</label>
```

<visfAtlas_FSL.xml>

Defining surface ROIs

index가 27번인 right hemisphere의 MT ROI를 그려 보겠습니다. <add_roi> 메서드를 사용합니다.

```
visfAtlas_MT_rh = np.zeros(visfAtlas.shape)
visfAtlas_MT_rh[visfAtlas==27] = 1.0

visfAtlas_MT_rh_volume = cortex.Volume(visfAtlas_MT_rh.transpose(2,1,0),
                                         'MNI', 'fmriprep_3mm_edit')

cortex.utils.add_roi(visfAtlas_MT_rh_volume, name='MT')
```

```
-----  
AttributeError                                Traceback (most recent call last)  
<ipython-input-34-67ed5a9d609f> in <module>  
      5                                         'MNI', 'fmriprep_3mm_edit')  
      6  
--> 7 cortex.utils.add_roi(visfAtlas_V1v_volume, name='V1v')  
  
~/opt/anaconda3/lib/python3.8/site-packages/cortex/utils.py in add_roi(data,  
name, open_inkscape, add_path, overlay_file, **kwargs)  
    283     if open_inkscape:  
    284         inkscape_cmd = config.get('dependency_paths', 'inkscape')  
--> 285         if LooseVersion(INKSCAPE_VERSION) < LooseVersion('1.0'):  
    286             cmd = [inkscape_cmd, '-f', svg.svgfile]  
    287         else:  
  
~/opt/anaconda3/lib/python3.8/distutils/version.py in __lt__(self, other)  
    50  
    51     def __lt__(self, other):  
--> 52         c = self._cmp(other)  
    53         if c is NotImplemented:  
    54             return c  
  
~/opt/anaconda3/lib/python3.8/distutils/version.py in _cmp(self, other)  
    333         other = LooseVersion(other)  
    334  
--> 335         if self.version == other.version:  
    336             return 0  
    337         if self.version < other.version:  
  
AttributeError: 'LooseVersion' object has no attribute 'version'
```

또 Inkscape 관련 에러가 나네요 ...

pycortex의 implementation code를 고치려 또 (3)번 풀더로 갑니다.

pycortex codes (3)

Installed packages

Python

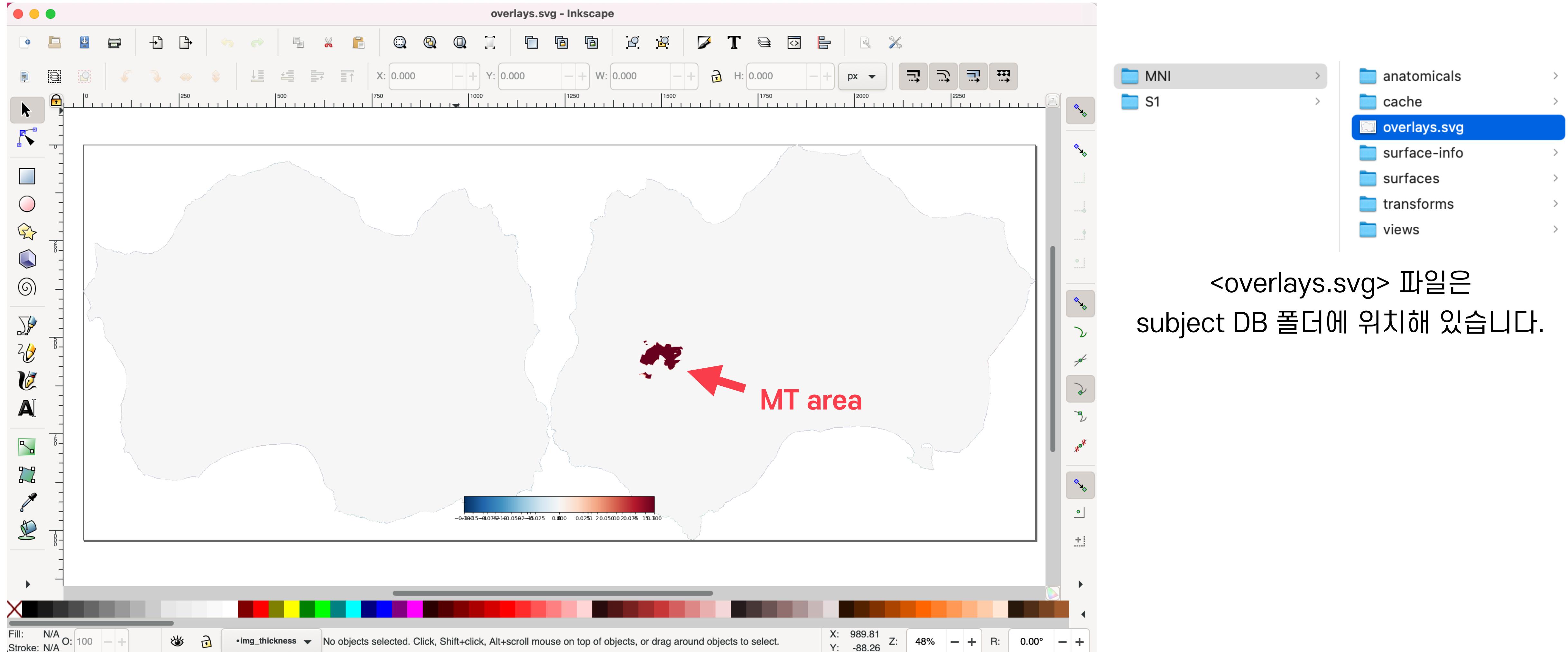
Defining surface ROIs

```
if open_inkscape:  
    # original code  
    # inkscape_cmd = config.get('dependency_paths', 'inkscape')  
    # if LooseVersion(INKSCAPE_VERSION) < LooseVersion('1.0'):  
    #     cmd = [inkscape_cmd, '-f', svg.svgfile]  
    # else:  
    #     cmd = [inkscape_cmd, svg.svgfile]  
  
    # edited code  
    inkscape_cmd = "/Applications/Inkscape.app/Contents/MacOS/inkscape"  
    cmd = [inkscape_cmd, svg.svgfile]  
  
return sp.call(cmd)
```

Version checker와 inkscape를 call 하는 cmd를 좀 수정해 줍니다.

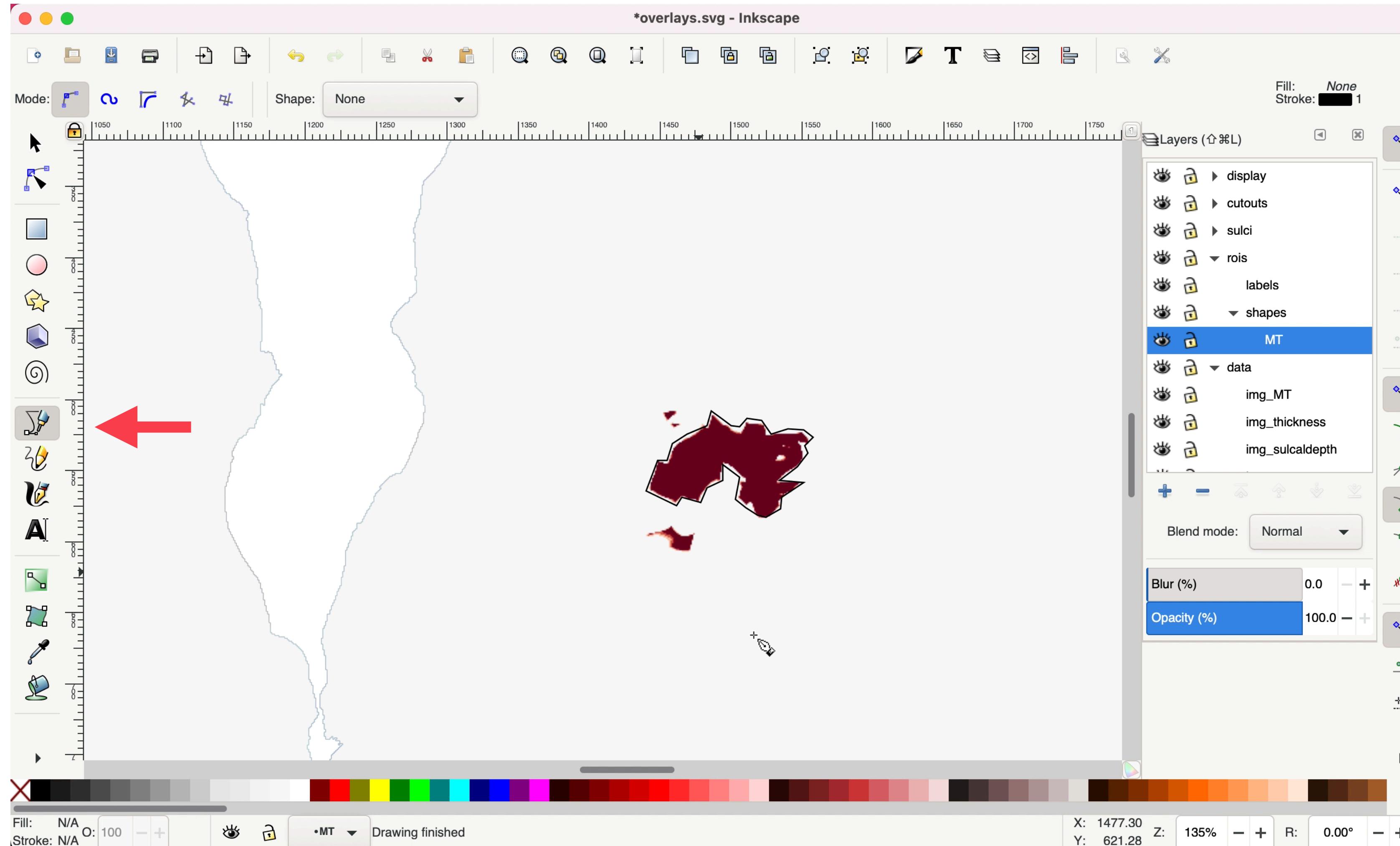
Defining surface ROIs

정상적으로 실행 될 경우, inkscape가 `<overlays.svg>` 파일을 엽니다. Flatten map 위로 MT ROI가 보이네요.



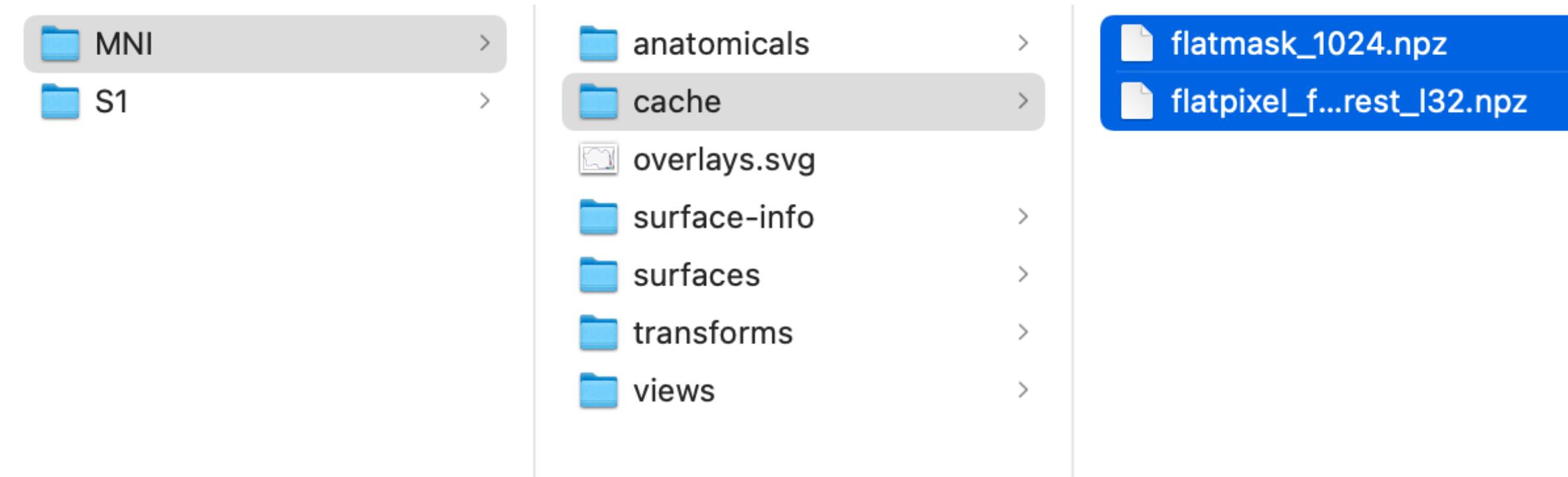
Defining surface ROIs

정상적으로 실행 될 경우, inkscape가 `<overlays.svg>` 파일을 엽니다. Flatten map 위로 MT ROI가 보이네요.



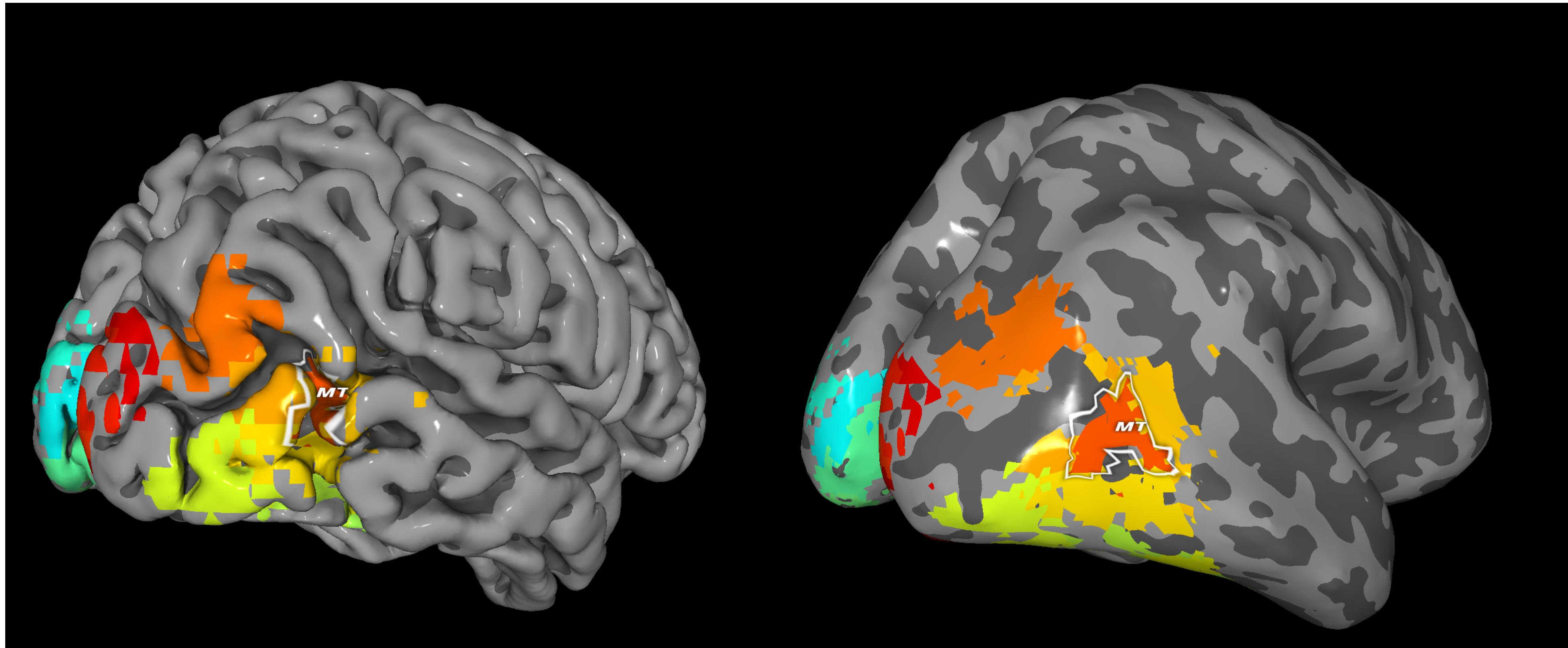
Layers > rois > shapes > MT layer를 선택하고, Pen tool로 그리시면 됩니다.

Defining surface ROIs

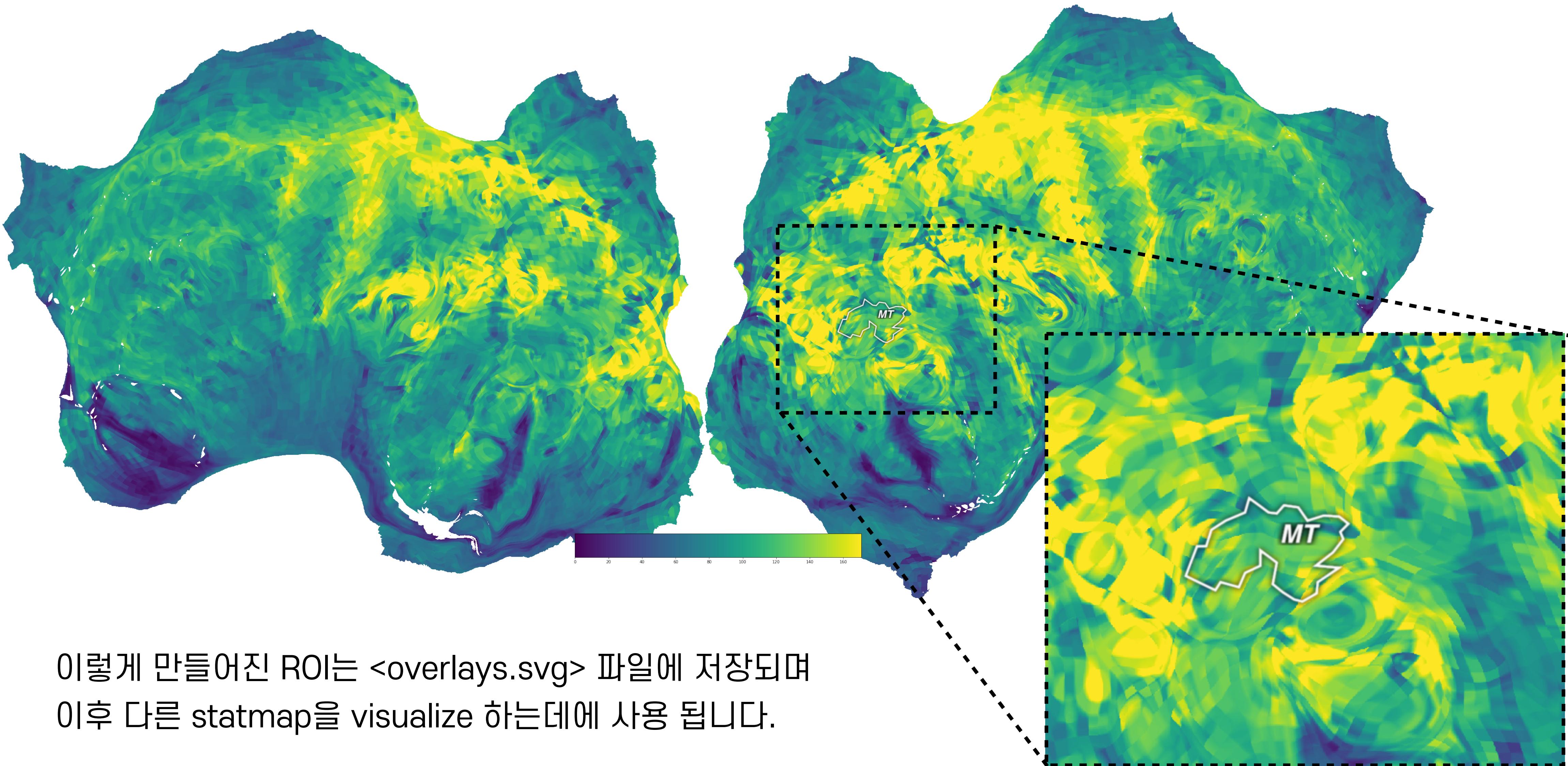


ROI를 추가하거나 제거한 후 subject DB 속 cache 폴더 안의 파일들을 삭제해야 정상적으로 표기가 됩니다.

cache 파일을 제거한 후 visualize 해 보면 정상적으로 MT ROI가 3D surface 위로 표시 되는 것을 볼 수 있습니다.



Defining surface ROIs



Take-home message 😊

1. 오늘 한 모든 과정을 매번 반복할 필요가 없습니다.

Pycortex의 DB가 제대로 갖추어 지고 나면 앞으로는 FreeSurfer가 없어도 되며, 잘 작업된 DB를 복사해서 사용하셔도 됩니다. (공유해 드린 drive에 오늘까지 진행한 DB가 들어 있습니다) 특히 여러분들이 MNI space에서 작업하시는 경우엔 이 모든 과정을 딱 한번만 하시면 됩니다.

2. 제가 자른 flatten surface에 살짝 왜곡이 있습니다.

저는 visualize 할 때, flatten map을 사용하지 않아 수정하지 않았습니다. 여러분들에게 공유해 드린 것은 제가 처음에 Pycortex를 연습 할 때 만들어본 surface입니다. Temporal lobe 부분을 제대로 자르지 않아서 살짝 왜곡이 있습니다. 만약 flatten map을 여러분들의 실제 분석에서 사용하길 원하시면 다시 정밀하게 잘라 보시길 추천 드립니다.

3. 정밀한 transform matrix를 만드는데 시간을 많이 들이길 바랍니다.

Manual alignment matrix를 수정 할 때 훨씬 더 값을 작게 조정하며 여러 각도에서 확인하시길 바랍니다.

4. 원하는 기능이 있다면 직접 만들 수 있습니다.

Pycortex는 다른 visualization tool과 다르게 연구자의 입맛에 맞게 얼마든지 튜닝을 할 수 있습니다. 예를 들어 자주 사용하시는 figure layout이 있는 경우엔 코드 한줄로 해당 layout으로 figure를 곧바로 export하는 기능을 쉽게 만들 수 있습니다. 이 부분이 Pycortex의 가장 큰 장점이라고 생각 합니다.