

# 第1部(4) recon-all の実際 / freeview

筑波大学医学医療系  
精神医学

根本 清貴

# 本勉強会のルール

- ターミナルでタイプするものは、青色で表示
  - 例: `freeview -v bert/mri/T1.mgz`
- 入力する必要のないコマンドは、紫色で表示
- スクリプトに記載してある内容は緑色で表示
- ショートカットキーの組み合わせは、`ctrl+c` のように水色で色づけ
- GUIでの動作は、`View → Show Control Panel` のように紫で色づけ
- # 以降は、解説でありタイプする必要はない
- 「フォルダ」=「ディレクトリ」

# 本セクションの目標

- **recon-all** の実際を理解する
  - recon-all を連続して実行するスクリプト
  - recon-all のログの確認
  - 海馬の subfield 解析
  - 脳幹の構造解析
- **freeview** の使い方に慣れる
  - GUI
  - volume 画像、surface 画像の表示
  - 海馬 subfield, 脳幹構造の表示

# 本セクションのチートシート

- nisg-202001/docs にある ex4.html をダブルクリック
- コマンドの意味も記載されている

# SUBJECTS\_DIRの設定

- Lin4Neuroの場合

`nisg=/media/sf_share/nisg-202001`

- MacOSの場合

`nisg=~/git/nisg-202001`

`SUBJECTS_DIR=$nisg/subjects`

# recon-all の具体例

- recon-all は、`-i` で入力ファイル、`-s` で各 subject の id (fsid) を指定する
- ここでは、画像ファイル名の先頭部分を id として指定する
- 入力ファイル名が `subj1.nii.gz`, fsid が `foo` としたい場合、以下のようになる

```
recon-all -i subj1.nii.gz -s foo \
-all -qcache
```

# **recon-all -all -qcache**

- **recon-all -i <nifti> -s <fsid> -all** で全てではない
- グループ解析のためには、さらに次のコマンドをタイプする  
**recon-all -s <fsid> -qcache**
- 各個人の様々な画像を fsaverage にあわせこみ、かつ、平滑化を 5, 10, 15, 20, 25mmで行う
  - SPMでのnormalise, smoothingと同じことをしている
- この処理を経てはじめてグループ解析ができる
- 所要時間は1例あたりおおよそ5-10分程度
- 以前は、**recon-all** をしてから、もう一度 **recon-all -qcache** をしていたが、**recon-all -all -qcache** を全例にした方が効率よいと気づき、今は、最初から全例に行っている

# recon-all 関連スクリプト

- `recon-all` を連続してやるためにスクリプトを準備してある
- `~/git/fs-scripts` に入っている `fs_autorecon.sh` をテキストエディタで見てみる

- Lin4Neuro

```
gedit ~/git/fs-scripts/fs_autorecon.sh
```

- macOS

```
open -e ~/git/fs-scripts/fs_autorecon.sh
```

# fs\_autorecon.sh の開発動機

- MRIをDICOM → niftiに変換する際に、ファイル名にIDを入れて整理している
- MRIのファイル名を fsid に流用したらいいのではないか？
- MRIのファイル名を指定することで自動で fsid も指定できるので、繰り返しが簡単になる

# fs\_autorecon.sh のコア

```
for f in "$@"
do
    fsid=${f%.ini*}
    recon-all -i $f -s $fsid -all -qcache
done
```

# **fs\_autorecon.sh** の解説

**for f in "\$@"** 引数をひとつずつ変数fに代入

**do** doからdoneの中を繰り返し

**fsid=\${f%.nii\*}**

変数fから拡張子を取り除いたものを変数fsidとしてセット

例: 画像ファイル名がv\_ID01.niiならば fsid は v\_ID01 となる

**recon-all -i \$f -s \$fsid -all -qcache**

ファイル名が v\_ID01.nii ならば、以下を実行

**recon-all -i v\_ID01.nii -s v\_ID01 -all -qcache**

**done**

# $\${\color{red}f\%.nii*}$ をもう少し

- 変数  $f$  に `subj01.nii.gz` を代入したとし、そこから `subj01` の部分だけ取り出したいとする
- $\${\color{red}f\%.nii*}$  は、「文字列の最後から見て、ドットがあって `nii` という文字列が続くというパターンにマッチする文字列を取り除く」という意味
- つまり、今の場合は、`.nii.gz` が取り除かれ `subj01` となる

# fs\_autoqcache.sh

- `recon-all -s <fsid> -qcache` を繰り返し行うスクリプト
- `recon-all` を一度行った subject に対して、`-qcache` を追加で処理したい時に使います
- `fs_autorecon.sh` と同じくforループを使っている

# 仮想マシン環境での FreeSurfer解析時のTIPS

- ノートパソコンで仮想マシンを走らせて `recon-all` を実行する場合、Windows/macOSは非情にも勝手にスリープします…
  - `recon-all` をやり直しするはめになりました
- 解決法は以下のとおりです
  - Windows: 電源設定で、「電源がつながっている時は、スリープもモニターもオフにしない」という設定にする
  - macOS: "caffeinate" コマンドを使用する
    - `$ caffeinate -i` で、Ctrl+Cを押すまでスリープが抑制される

# recon-all のログの確認

- recon-all の詳細なログは、  
\$SUBJECTS\_DIR/<fsid>/scripts/recon-all.log におさめ  
られている
- 各プロセスがうまくいったかどうかの情報は、\$SUBJECTS\_DIR/  
<fsid>/scripts/recon-all-status.log におさめられている
- ernie のログを見てみる

```
cd $SUBJECTS_DIR/ernie/scripts  
less recon-all-status.log
```

# recon-all-status.log

status file for recon-all

Mon Dec 30 20:03:29 JST 2019

#@# MotionCor Mon Dec 30 20:03:36 JST 2019

recon-all -s ernie finished **without error** at Mon Dec 30 20:06:17 JST  
2019

Motion correctionがだいたい3分で終了

New invocation of recon-all

status file for recon-all

Mon Dec 30 20:06:17 JST 2019

#@# Talairach Mon Dec 30 20:06:17 JST 2019

recon-all -s ernie finished without error at Mon Dec 30 20:07:56 JST  
2019

recon-all の各プロセスを理解するため、ひとつずつ directive を実行したため、新たにログが生成されている

※lessを終了するときは、q をタイプする

# 海馬のsubfield解析

- FreeSurfer 6.0から、海馬のsubfield解析が可能になった
- Matlabのランタイムのインストールが必要
- 今回配布したLin4Neuroには、既にインストールしてある



# 海馬のsubfield解析

## Matlabランタイムインストール

- Matlabランタイムのインストールのためのスクリプトを準備してある
- Matlabランタイムのダウンロードからインストールまで自動で行う
- LinuxにもmacOSにも対応しているため、macOSの方で海馬のsubfield解析を自分でしたい方は以下  
(Lin4Neuroは既にセットアップ済)

```
cd ~/git/fs-scripts  
./fs_download_matlabruntime.sh
```

# 海馬のsubfield解析

- T1画像のみある場合
  - まだ `recon-all -all` をしていない場合

```
recon-all -i <nifti> -s <fsid> -all \
-hippocampal-subfields-T1
```
  - すでに `recon-all -all` が済んでいる場合

```
recon-all -s <fsid> -hippocampal-subfields-T1
```
- 1例あたり約30分かかる

# 海馬のsubfield解析一括処理

- 海馬のsubfield解析を一括できたらというリクエストに応えて、スクリプトを準備してある
- `fs-scripts`の中にある `fs_autohipposf.sh`
- 使い方  
`fs_autohipposf.sh <fsid>`
- もし、`OAS2_*`を一括して処理したければ  
`fs_autohipposf.sh OAS2_*`  
でsubject名に `OAS2_` が入っているファイルが一括で処理される

# 海馬のsubfield解析結果

- T1画像のみある場合は6つのファイルが生成される
  - [lr]h.hippoSfLabels-<T1>-<analysisID>.v10.mgz
    - もともとのT1画像と同じ空間で、高解像度（ボクセルの1辺が0.333mm）のsegmentationの結果
  - [lr]h.hippoSfLabels-<T1>-<analysisID>.v10.FsvoxelSpace.mgz
    - 低解像度での結果。解像度が悪くなる
  - [lr]h.hippoSfVolumes-<T1>-<analysisID>.v10.txt
    - 海馬のsubfieldのボリュームと海馬全体の容積が記されたテキスト
- 結果の見方は後に扱う

# 脳幹の構造解析

- 海馬と同じように、脳幹の構造解析も行える
  - recon-allを調べる中で普通にできることに気づいた
  - 海馬と同じMatlab Runtimeが必要
  - すでに `recon-all -all` が済んでいる場合`recon-all -s <fsid> \-brainstem-structures`
- 1例あたり約30分かかる

# 脳幹の構造解析一括処理

- 海馬と同様に一括処理のスクリプトを準備してある
- `fs-scripts`の中にある `fs_autombrainstem.sh`

- 使い方

`fs_autombrainstem.sh <fsid>`

- もし、OAS2\_\*を一括して処理したければ

`fs_autombrainstem.sh OAS2_*`

でsubject名に OAS2\_ が入っているファイルが一括で  
処理される

# 脳幹の構造解析結果

- 3つのファイルが `mri/` に生成される
  - `brainstemSslabels.v10.mgz`
    - もともとのT1画像と同じ空間で、高解像度（ボクセルの1辺が 0.333mm）のsegmentationの結果
  - `brainstemSslabels.v10.FsvoxelSpace.mgz`
    - 低解像度での結果。解像度が悪くなる
  - `brainstemSsVolumes.v10.txt`
    - 脳幹の構造の容積が記されたテキスト
- 結果の見方は後に扱う

# freeview の理解

- freeviewのGUIやコマンドを理解する
- ernie の画像を表示する

```
cd $SUBJECTS_DIR/ernie/mri  
freeview -v T1.mgz wm.mgz brainmask.mgz \  
aseg.mgz:colormap=lut:opacity=0.2 \  
-layout 2 -viewport sagittal
```

# freeview -v



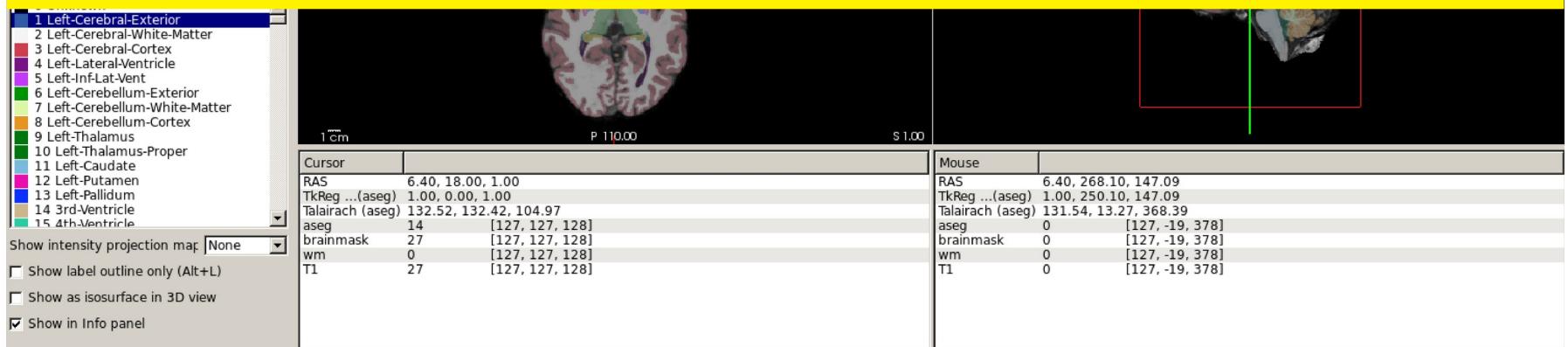
freeview -v \ # -vはvolumeの意味

T1.mgz \ #T1, wm, brainmask, aseg

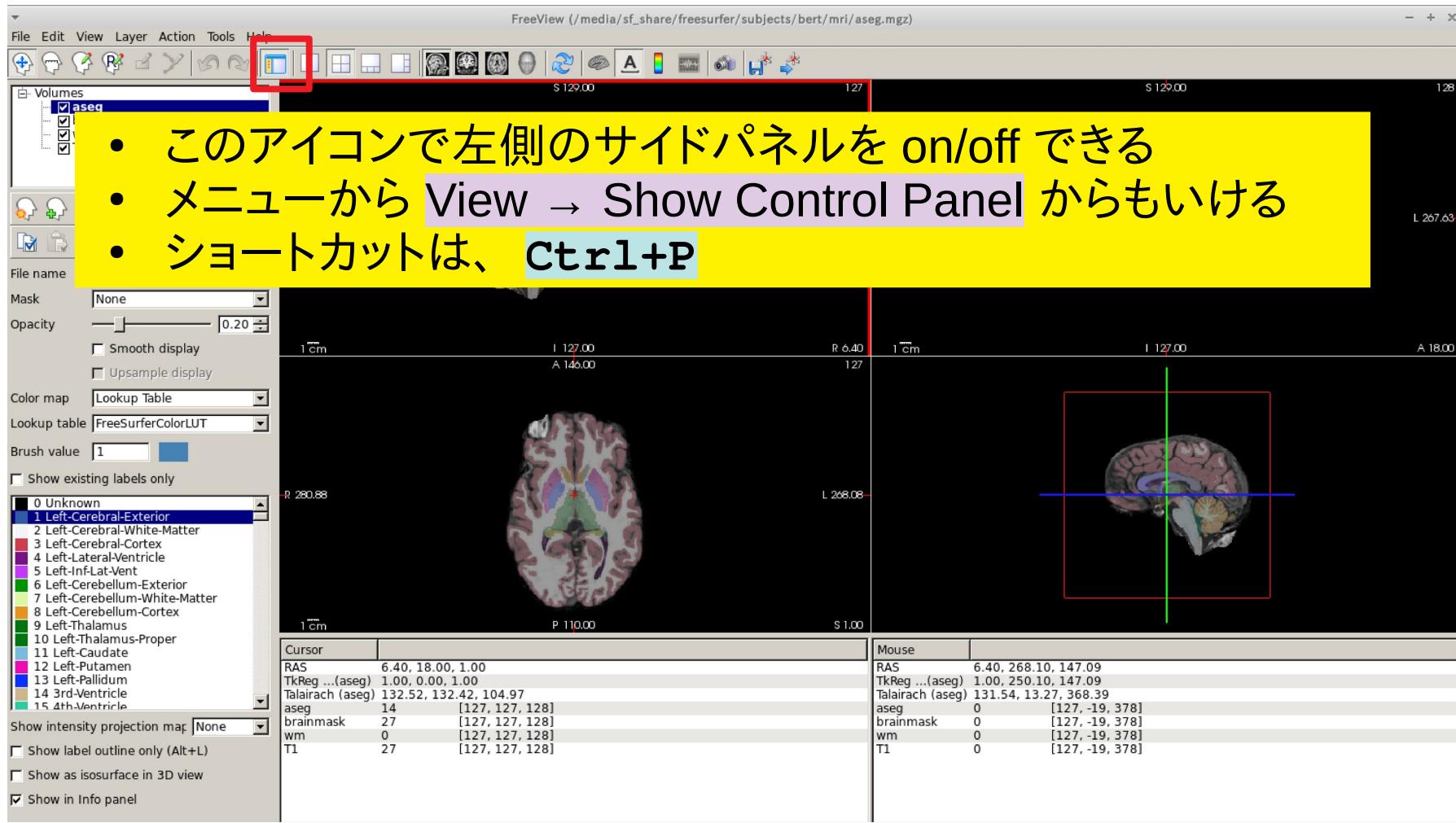
wm.mgz \ #ファイルを順番に読み込み

brainmask.mgz \ #後のファイルが上のレイヤーに来る

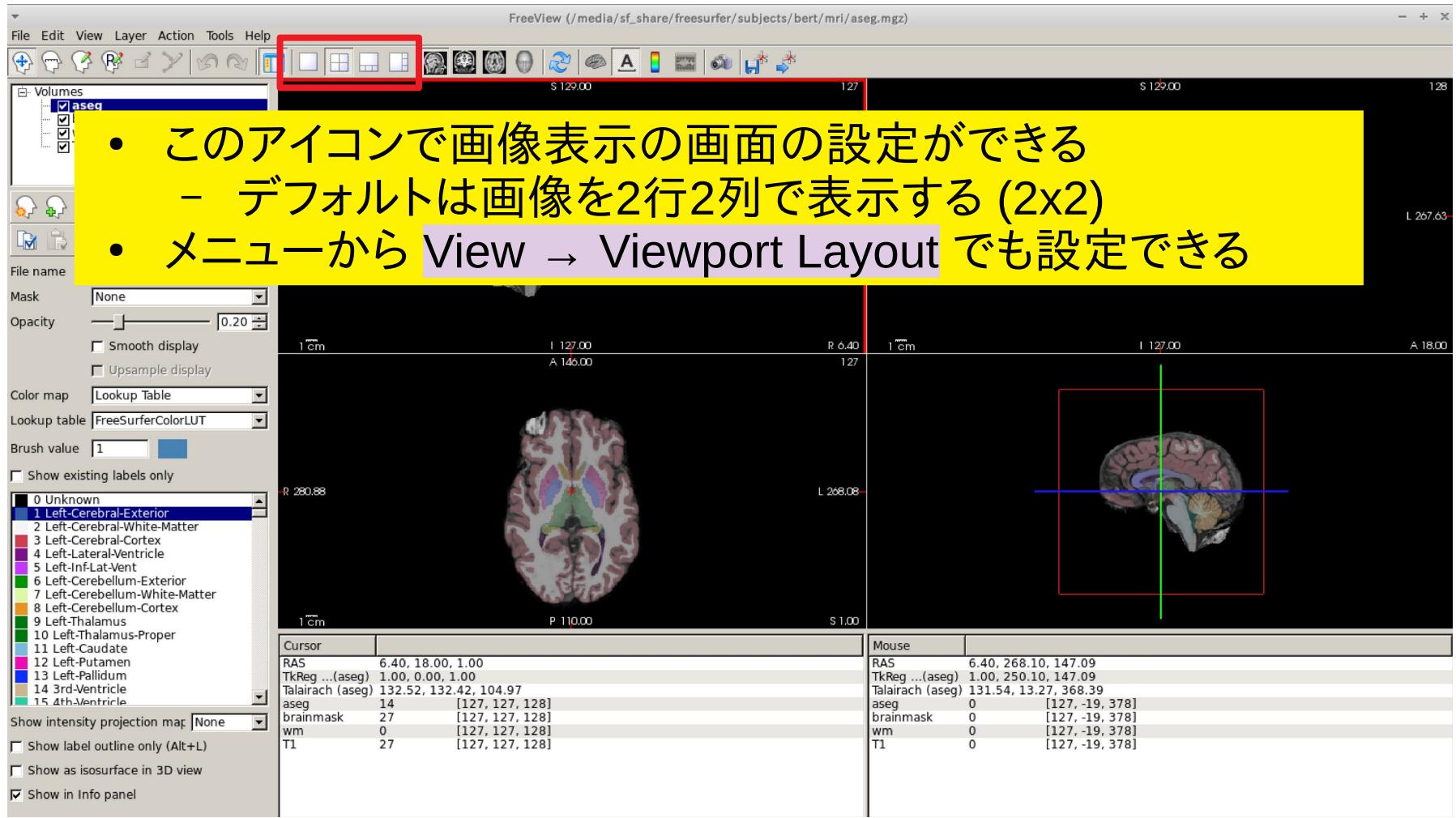
aseg.mgz #今はaseg.mgzが一番上



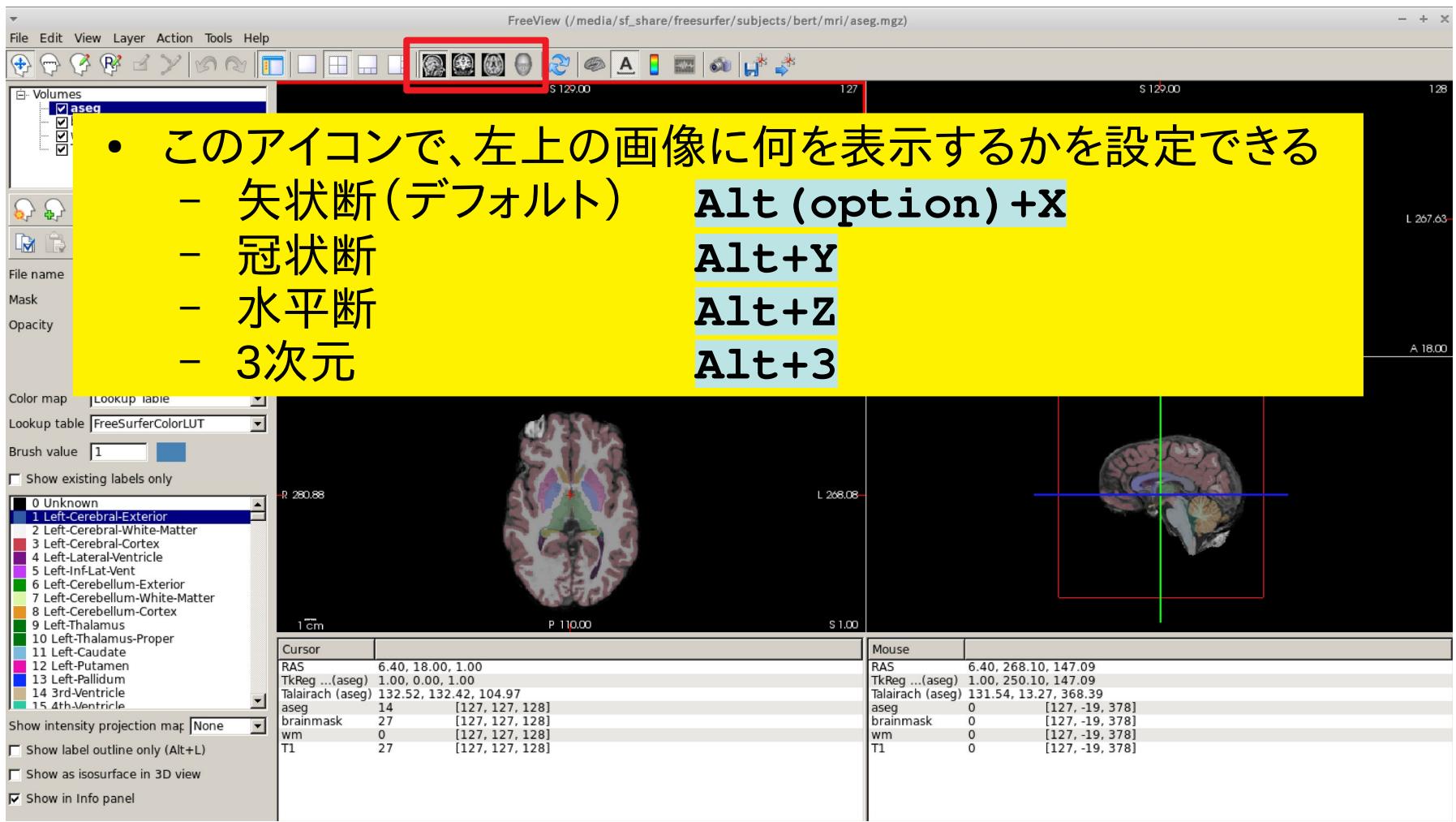
# 画面表示: コントロールパネル



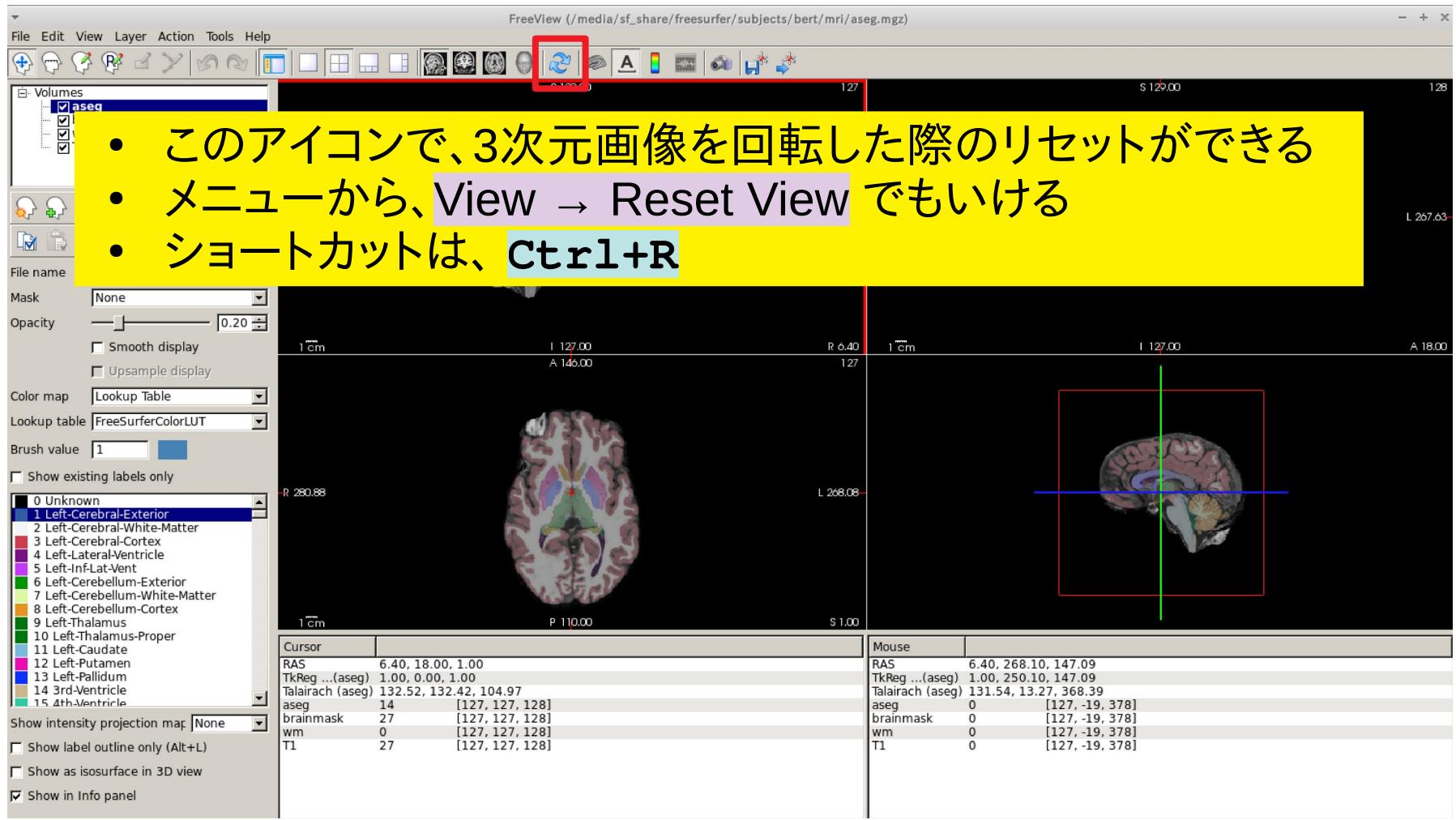
# 画面表示: 画像のレイアウト



# 画面表示: 画像の表示



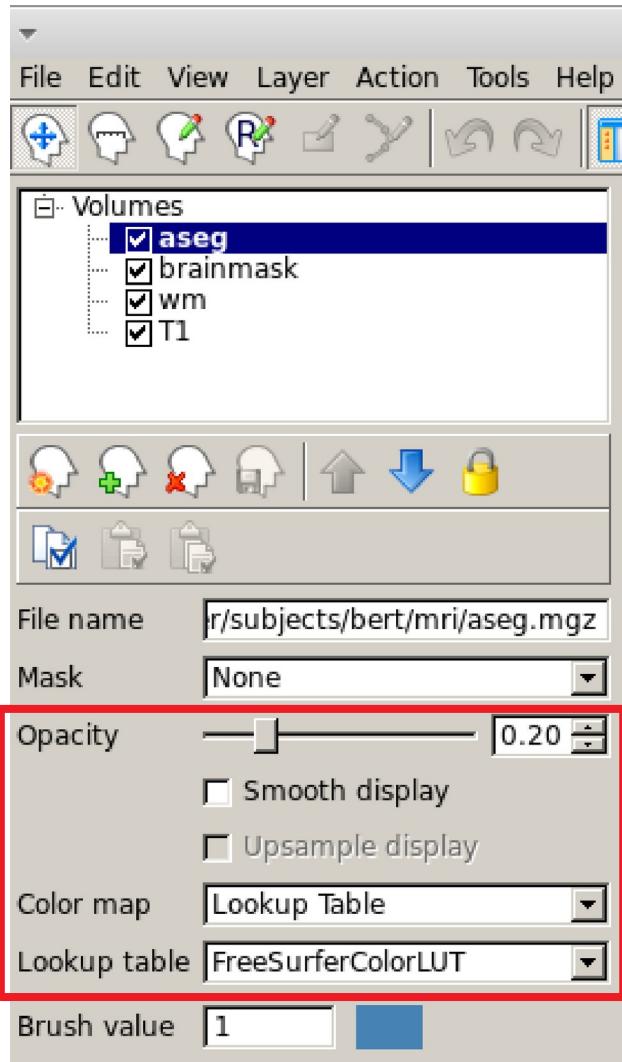
# 画面表示: 3次元画像のリセット



# 画像のズーム・パン / スライスの移動

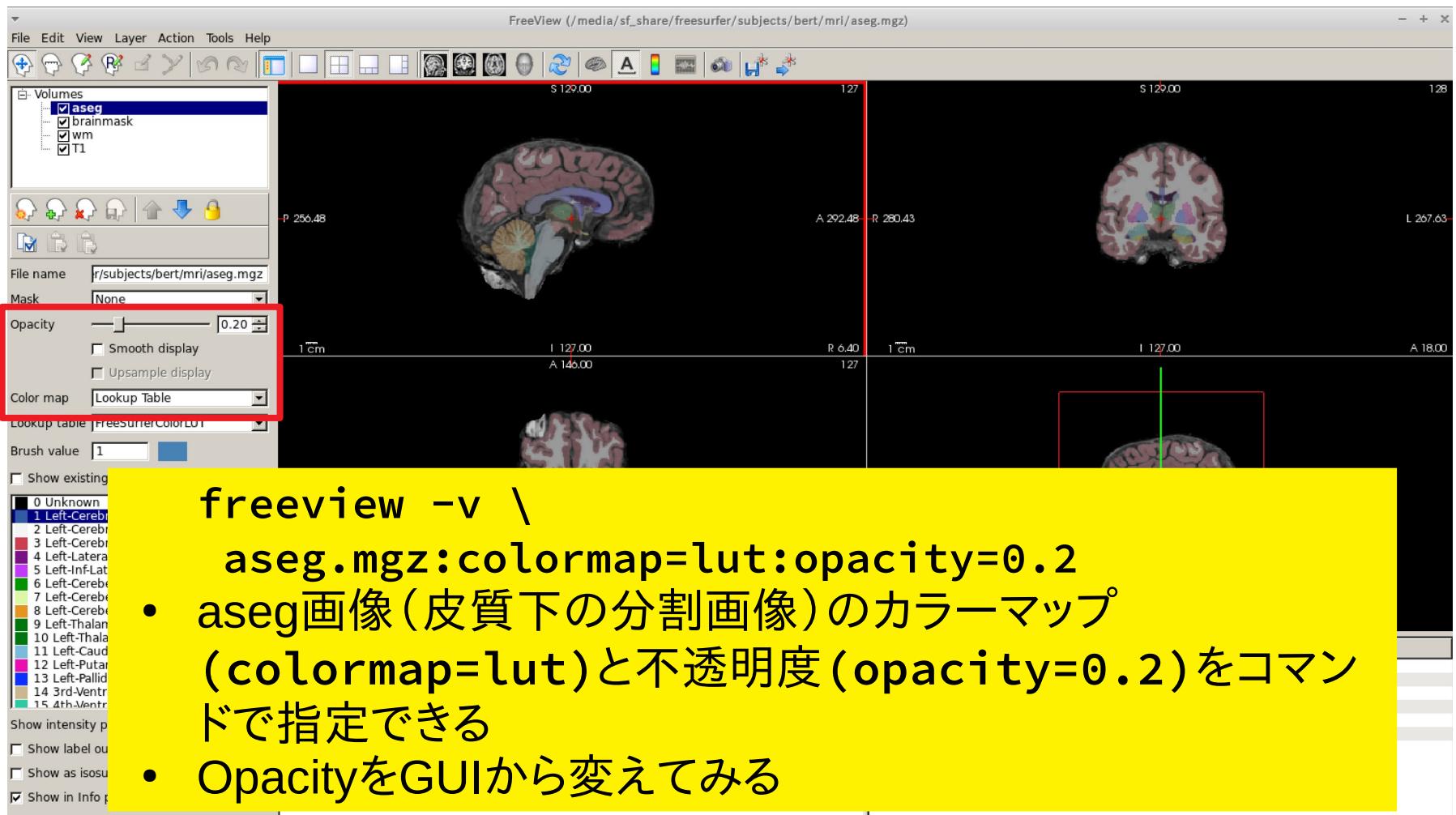
- ズーム
  - ズームイン(拡大)はマウスの右ボタンを押しながら上に移動
  - ズームアウト(縮小)はマウスの右ボタンを押しながら下に移動
  - マウスではスクロールがズーム
- パン(画像 자체を移動)
  - Lin4Neuro: 左Ctrl + 矢印 で移動
  - Mac: command + 矢印 で移動
  - マウスではスクロールボタンを押しながらマウスを移動
  - パンはズームした後に役立つ
- スライスを移動
  -   キーを使って移動

# コントロールパネル

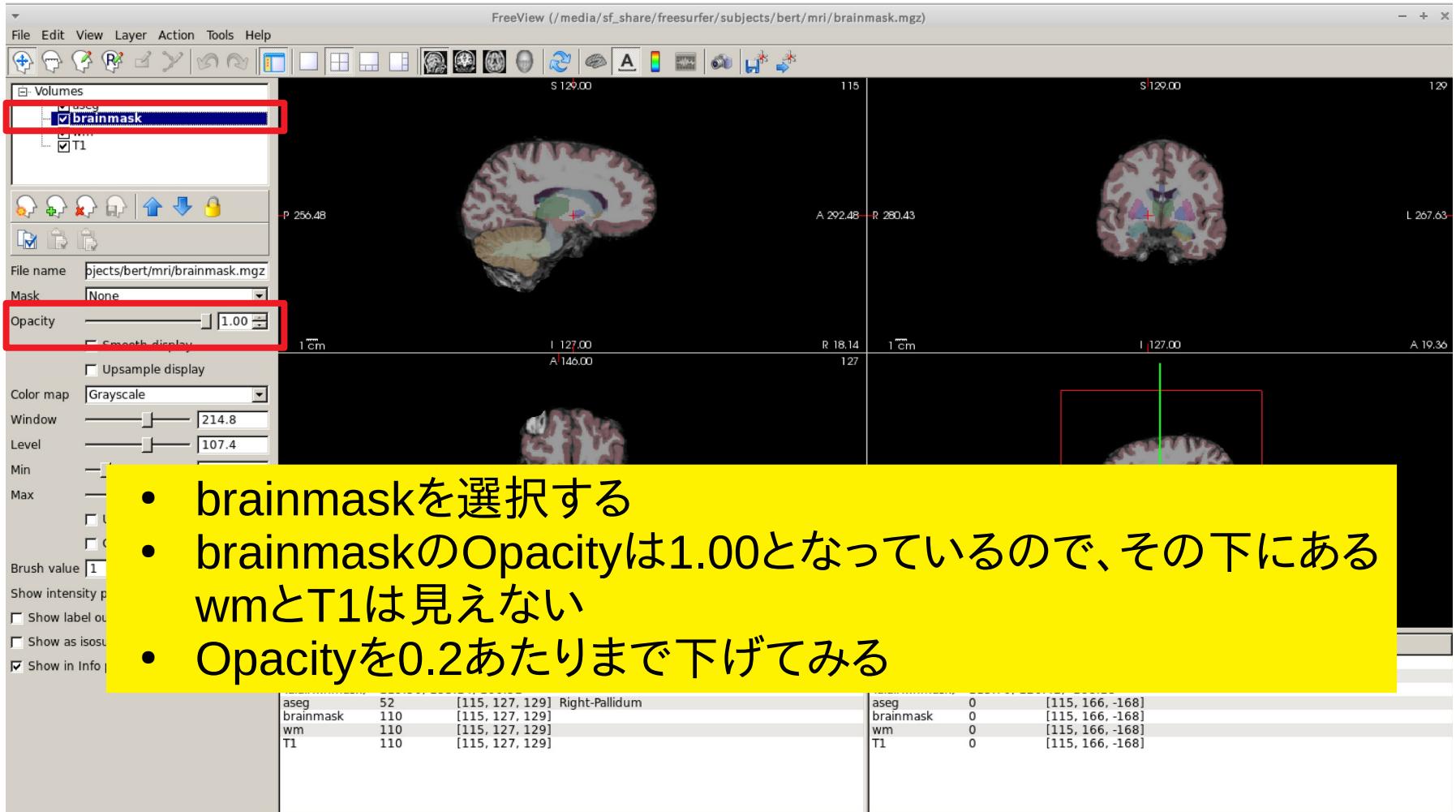


- 画面左側のコントロールパネルを用いてパラメータを変更することができる
- パラメータの一例として、次のスライドで不透明度(opacity)を変更する

# Opacity と Colormap



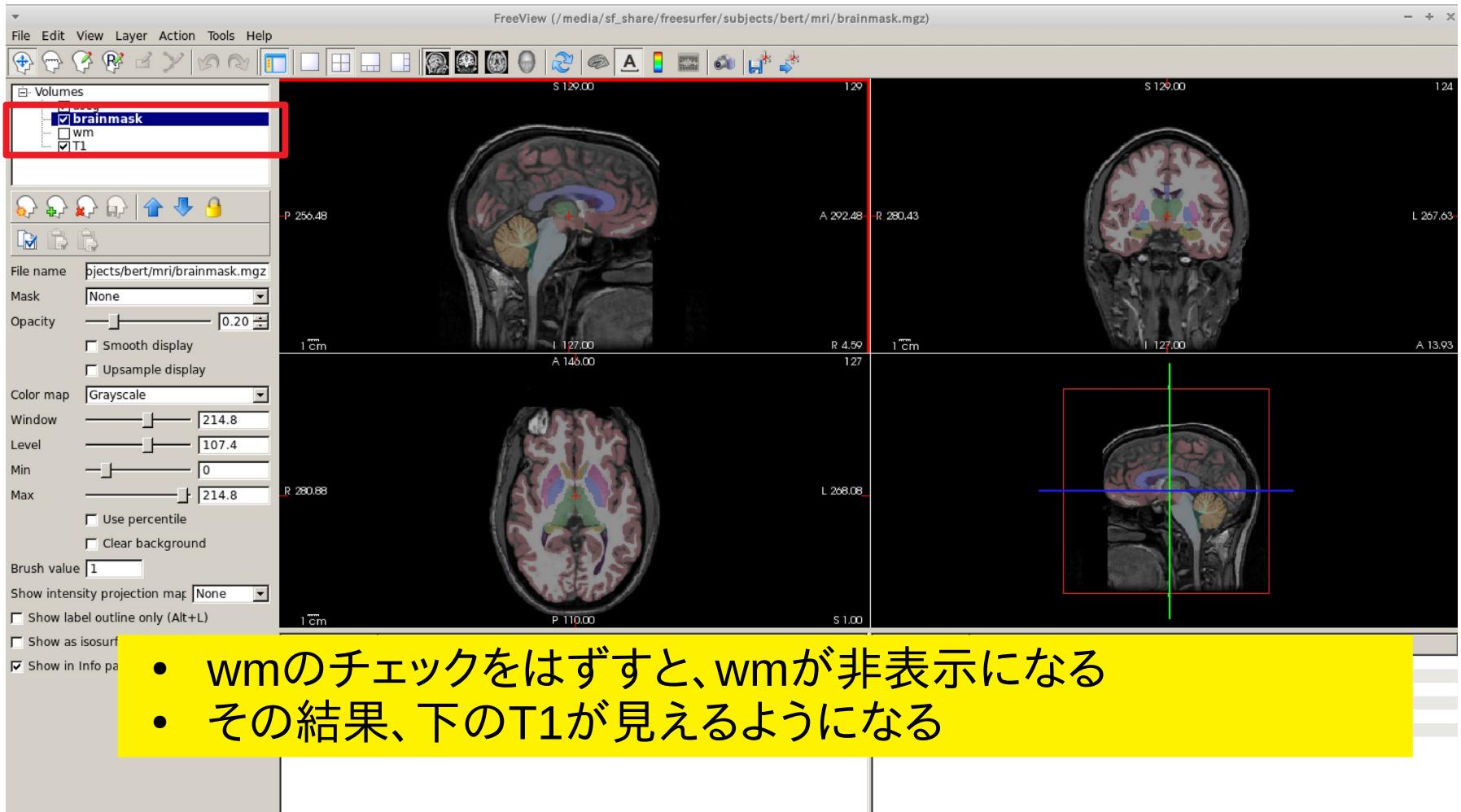
# レイヤー



# レイヤーのOpacity



# レイヤーの非表示

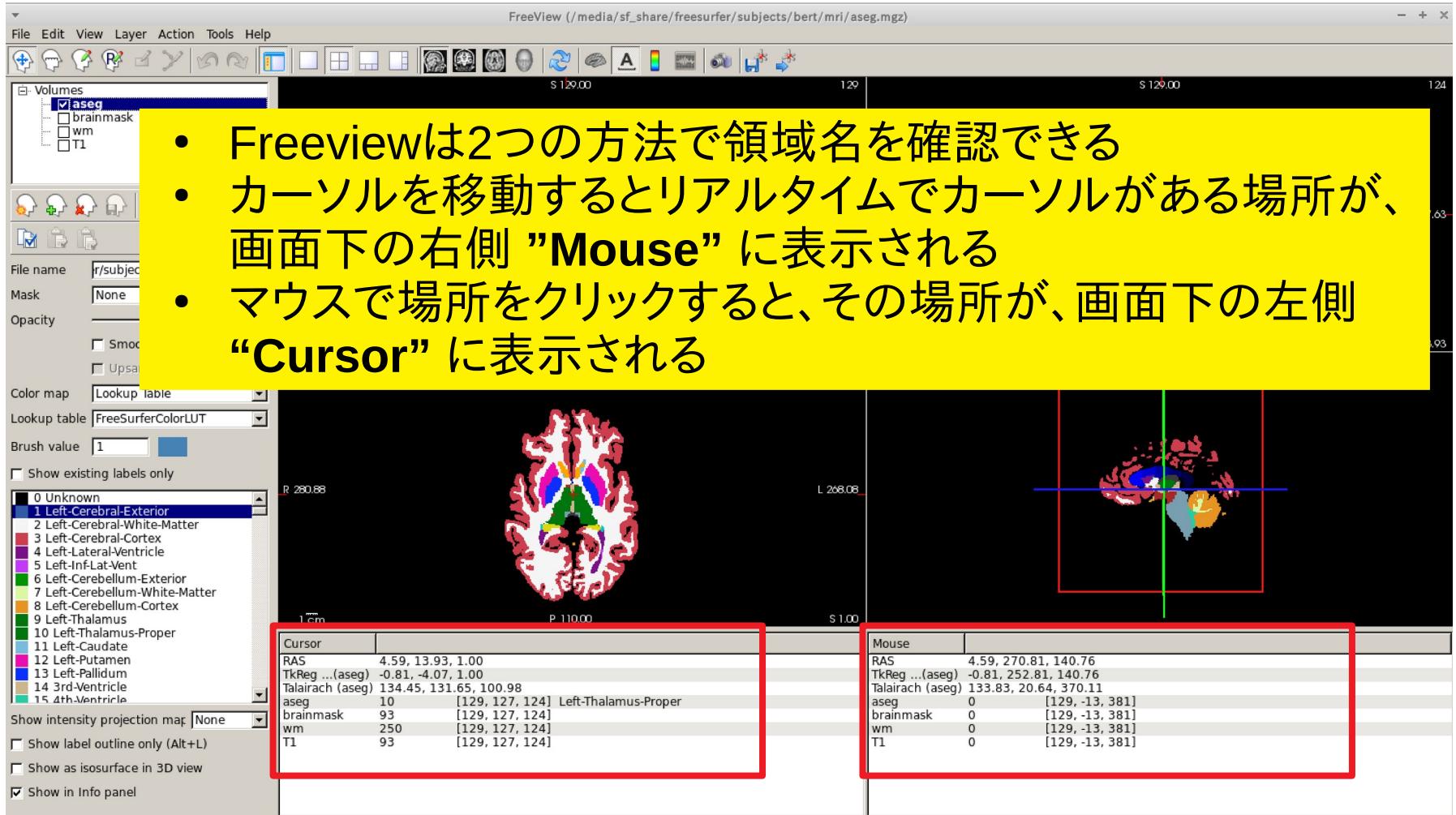


# Linux/MacにおけるFreeviewのキーボードショートカット

- MacとLinuxでは、Freeviewのキーボード・ショートカットが異なる

Function	機能（根本訳）	Linux	Mac
Cycle through layer list	レイヤーリストを循環	<b>Alt+C</b>	<b>Fn+Option+C</b>
Toggle volume visibility in layer list	レイヤーリスト内のvolumeデータ表示のon/off	<b>Alt+V</b>	<b>Option+V</b>
Toggle surface visibility in layer list	レイヤーリスト内のsurfaceデータ表示のon/off	<b>Alt+F</b>	<b>Command+F</b>
Toggle slices on/off when in 3D View	3D Viewにおいて断面表示のon/off	<b>Ctrl+Shift+S</b>	<b>Shift+Command+S</b>

# 領域名の確認



# **fs\_volume.sh**

- Freeviewを起動するために毎回コマンドを叩くのは大変なため、スクリプトを書いて、それを使っている
- Freeviewを閉じ、ターミナルから以下をタイプ  
**fs\_volume.sh ernie**
- 先ほどと同じような結果になる
- 違いは、GMとWMの境界を描くよう正在していること

# Surface画像の表示

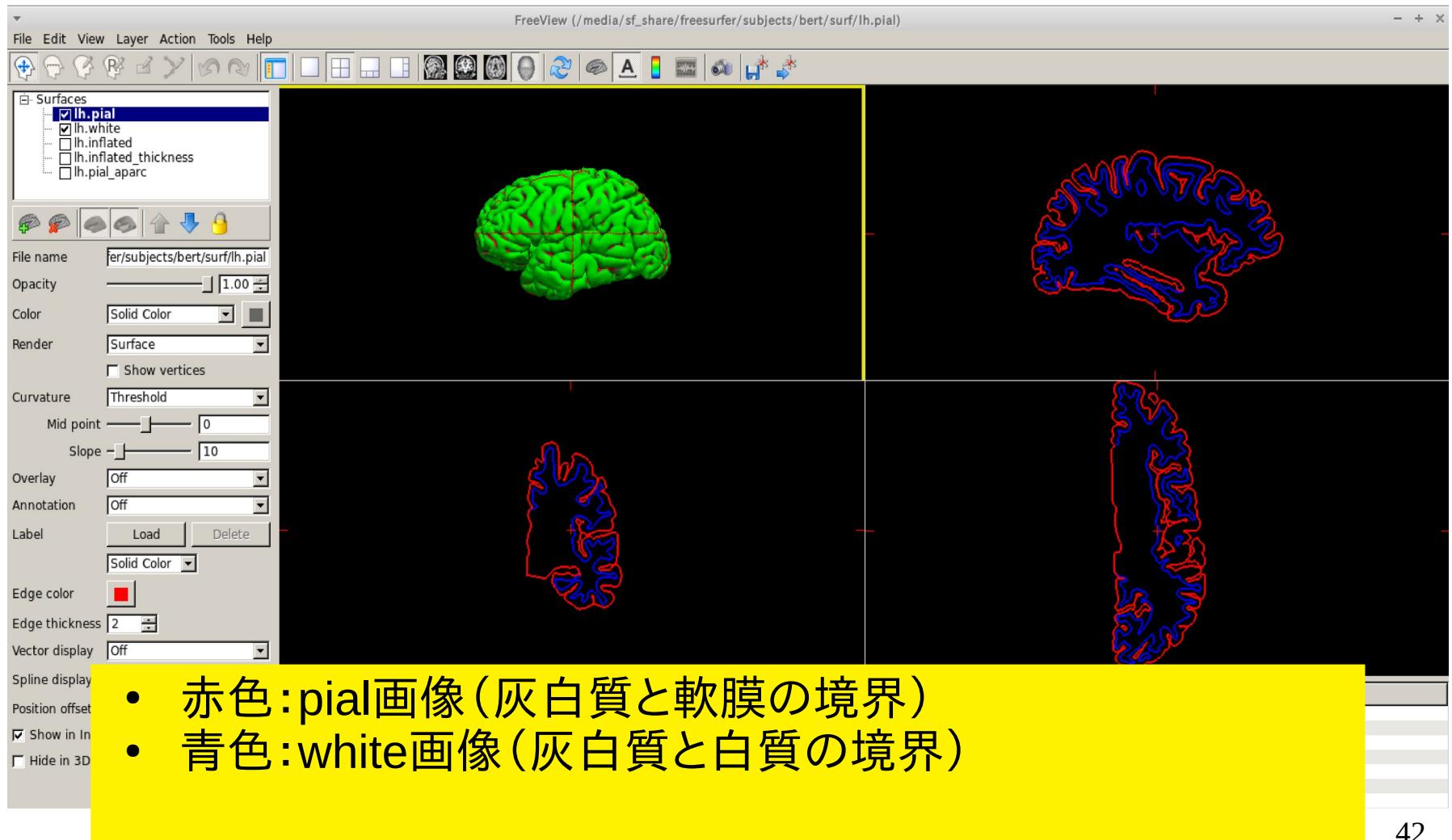
- FreeviewでSurface画像を確認

```
cd $SUBJECTS_DIR/ernie/surf  
freeview -f \  
lh.pial:annot=aparc.annot:name=lh.pial_aparc:visible=0 \  
lh.inflated:overlay=lh.thickness:overlay_threshold=0.1,3  
:name=lh.inflated_thickness:visible=0 \  
lh.inflated:visible=0 \  
lh.white:edgecolor=blue \  
lh.pial:edgecolor=red \  
-viewport 3d
```

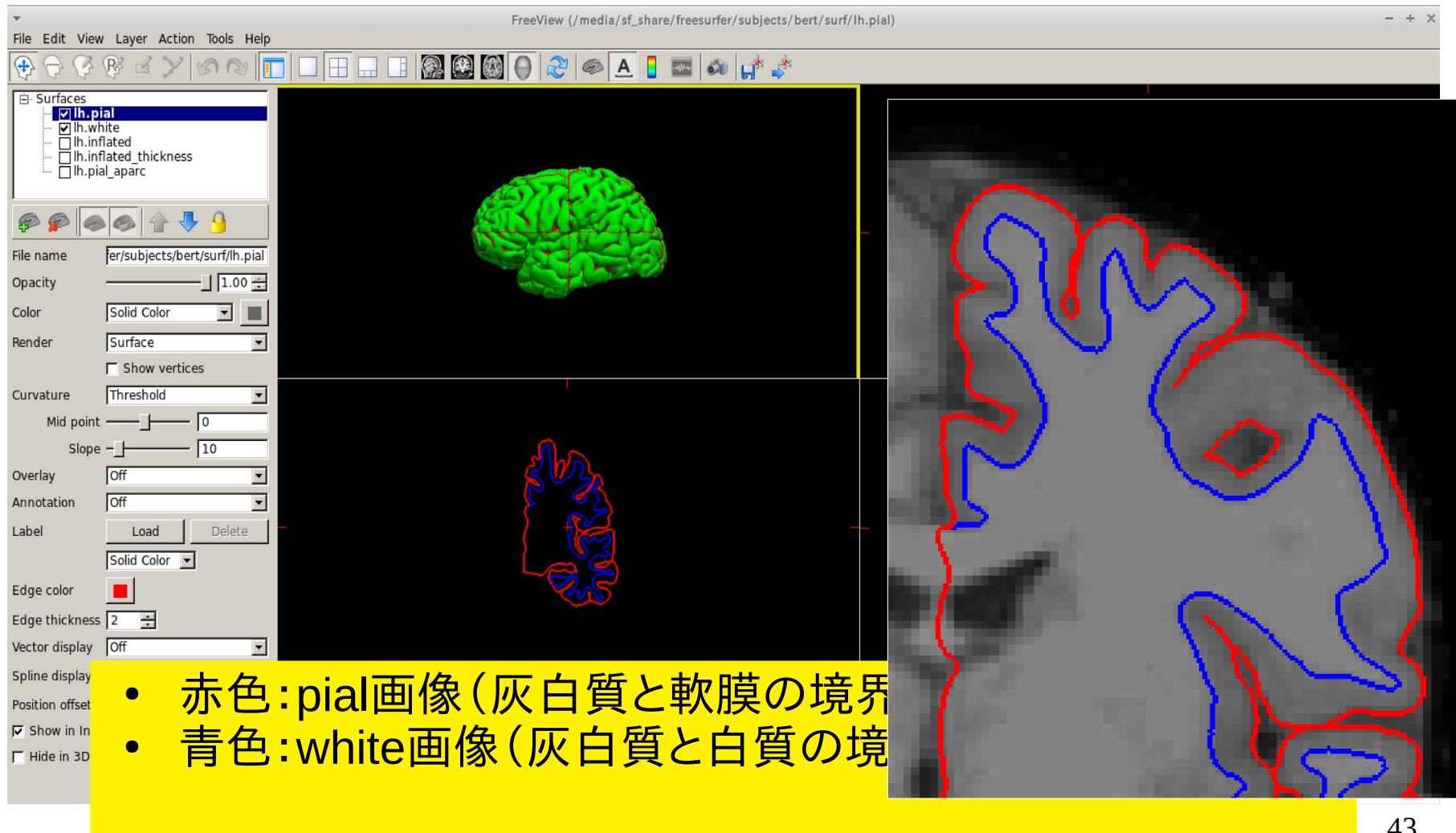
# Freeviewの様々なオプション

オプション	意味
<code>annot=aparc.annot</code>	<code>label/?h.aparc.annot</code> を読み込む
<code>name=lh.pial_aparc</code>	FreeViewに表示される名前
<code>visible=0</code>	表示するチェックボックスをoffにする
<code>overlay=lh.thickness</code>	<code>lh.thickness</code> を重ねあわせる
<code>overlay_threshold=0.1,3</code>	<code>lh.thickness</code> の表示範囲
<code>edgecolor=blue</code>	境界を青色で示す
<code>-viewport 3d</code>	3次元表示の画面を左上に持つくる

# freeview -f



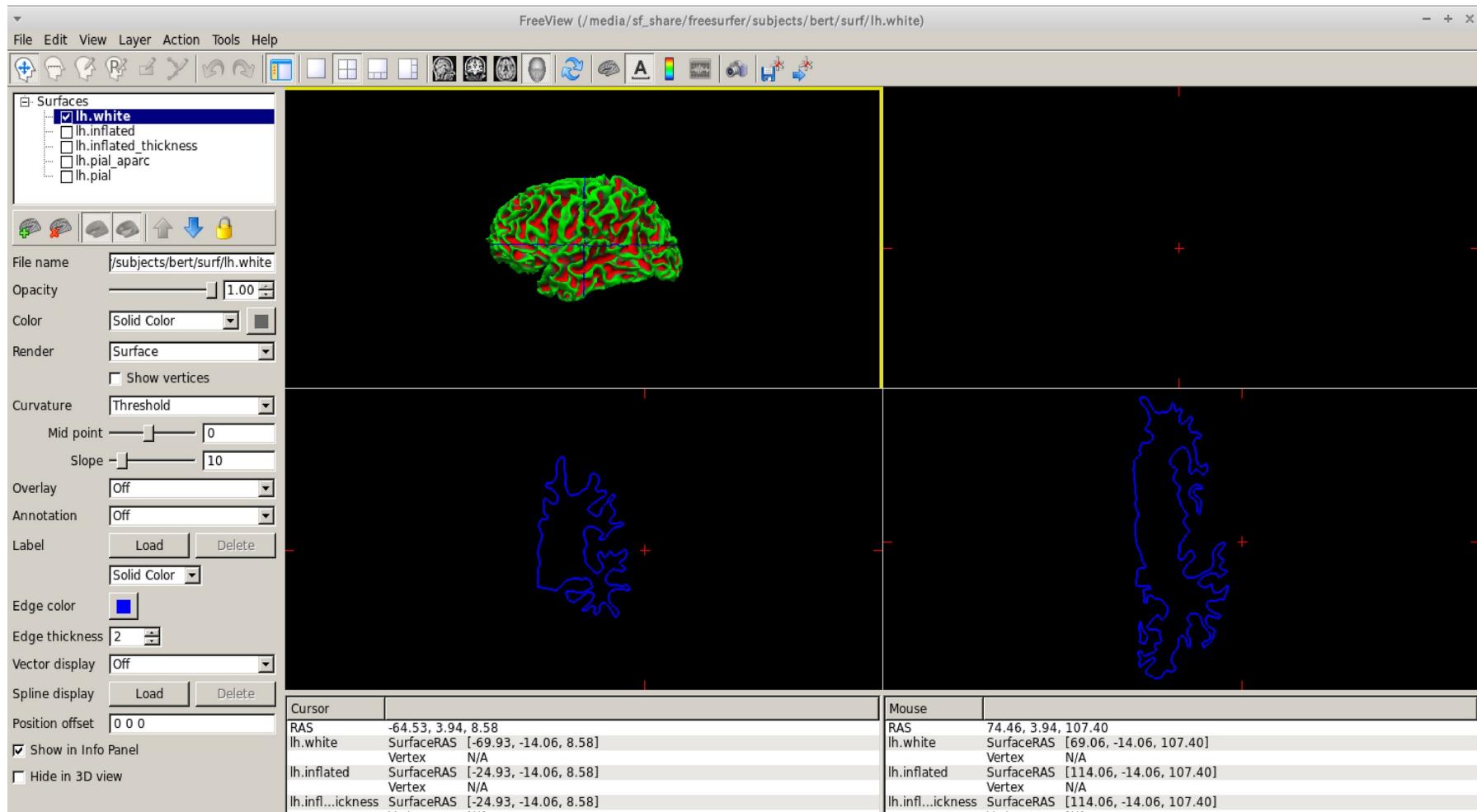
# freeview -f



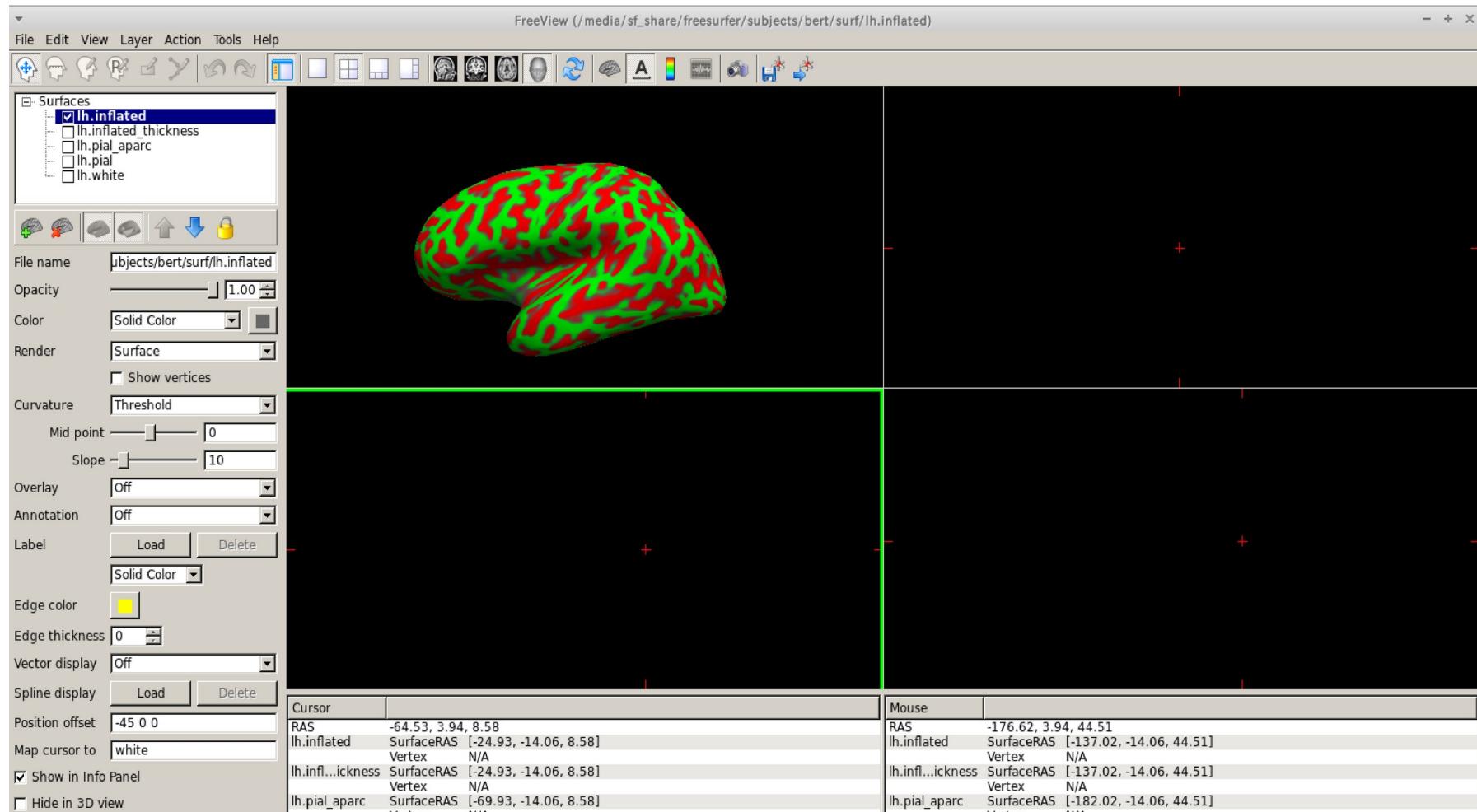
# さまざまなsurface画像の確認

- Lin4Neuroは、**Alt (Option) +C** を、macOSは、**Fn+Option+C** を押しながら、さまざまなsurface画像を確認
  - pial (今見ている画像)
  - white
  - inflated
  - inflated に thickness を重ねあわせたもの
  - aparc

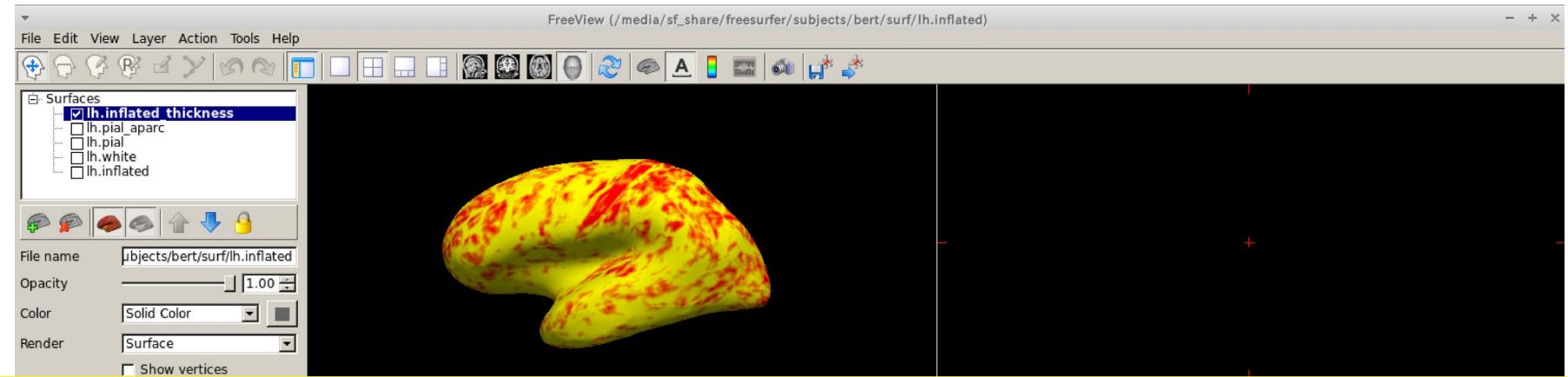
# lh.white



# lh.inflated



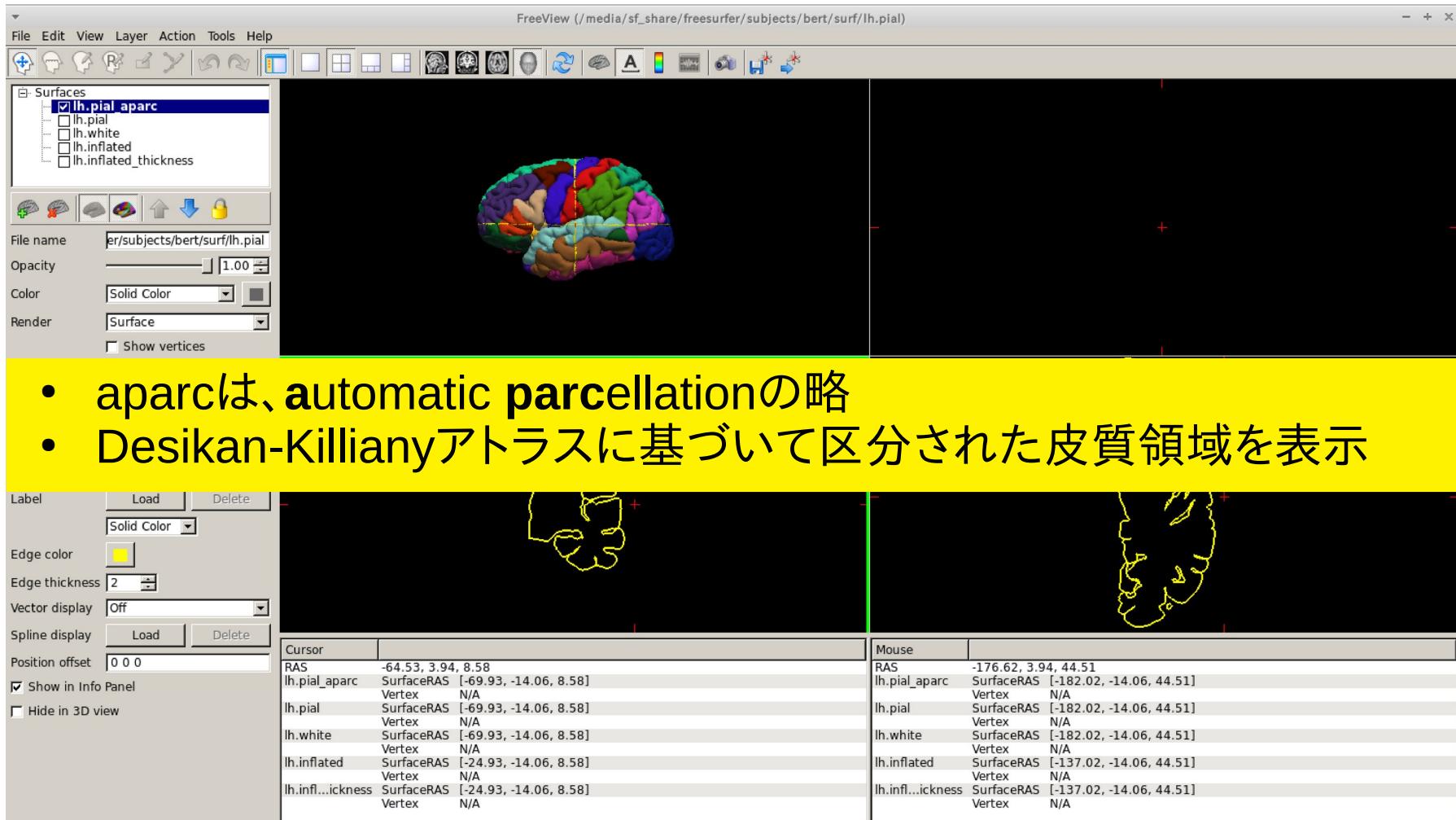
# lh.inflated.thickness



```
freeview -f \ surf/lh.inflated #lh.inflatedを表示
:overlay=lh.thickness          #lh.thicknessを重ねあわせ
:overlay_threshold=0.1,3        #閾値は0.1～3を表示
:name=lh.inflated_thickness    #コンパネに表示する名前
```

Position offset	-45 0 0	RAS	-64.53, 3.94, 8.58	RAS	-176.62, 3.94, 44.51
Map cursor to	white	lh.infl...ickness	SurfaceRAS [-24.93, -14.06, 8.58]	lh.infl...ickness	SurfaceRAS [-137.02, -14.06, 44.51]
<input checked="" type="checkbox"/> Show in Info Panel		Vertex	N/A	Vertex	N/A
<input type="checkbox"/> Hide in 3D view		lh.pial_aparc	SurfaceRAS [-69.93, -14.06, 8.58]	lh.pial_aparc	SurfaceRAS [-182.02, -14.06, 44.51]
		Vertex	N/A	Vertex	N/A
		lh.pial	SurfaceRAS [-69.93, -14.06, 8.58]	lh.pial	SurfaceRAS [-182.02, -14.06, 44.51]
		Vertex	N/A	Vertex	N/A
		lh.white	SurfaceRAS [-69.93, -14.06, 8.58]	lh.white	SurfaceRAS [-182.02, -14.06, 44.51]
		Vertex	N/A	Vertex	N/A
		lh.inflated	SurfaceRAS [-24.93, -14.06, 8.58]	lh.inflated	SurfaceRAS [-137.02, -14.06, 44.51]
		Vertex	N/A	Vertex	N/A

# lh.pial.aparc



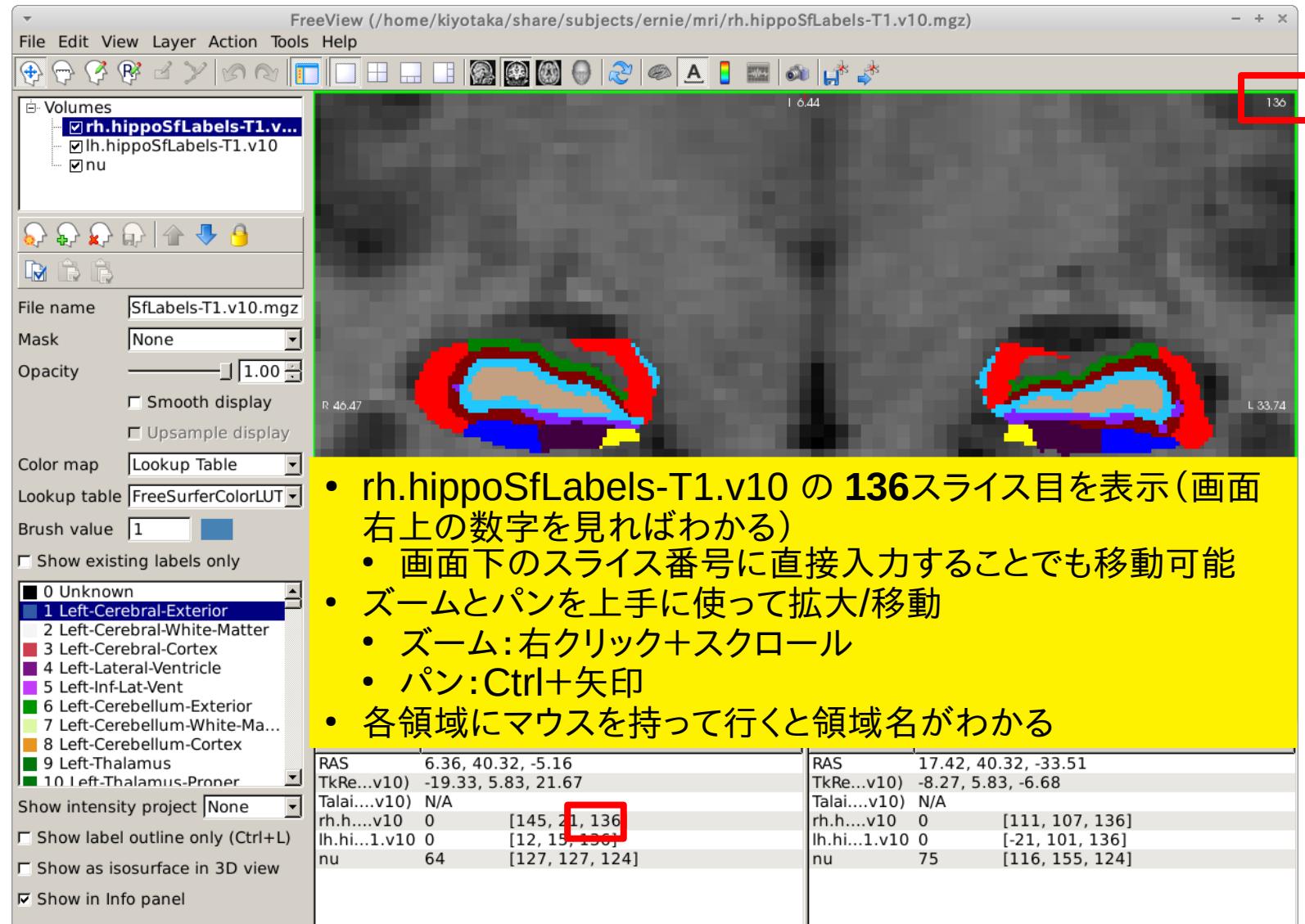
- aparcは、automatic parcellationの略
- Desikan-Killianyアトラスに基づいて区分された皮質領域を表示

# 海馬のsubfield解析: 高解像度

- ernieに解析済みデータあり
- 元画像空間に重ねあわせるには、mri ディレクトリにある nu.mgz に [lr]h.hippoSfLabels-T1.v10.mgz を重ねあわせる
- 高解像度(ボクセル1辺が0.333mm)

```
cd $SUBJECTS_DIR/ernie/mri  
freeview -v nu.mgz \  
lh.hippoSfLabels-T1.v10.mgz:colormap=lut \  
rh.hippoSfLabels-T1.v10.mgz:colormap=lut \  
-layout 1 -viewport coronal -zoom 5
```

# 海馬のsubfield: 高解像度

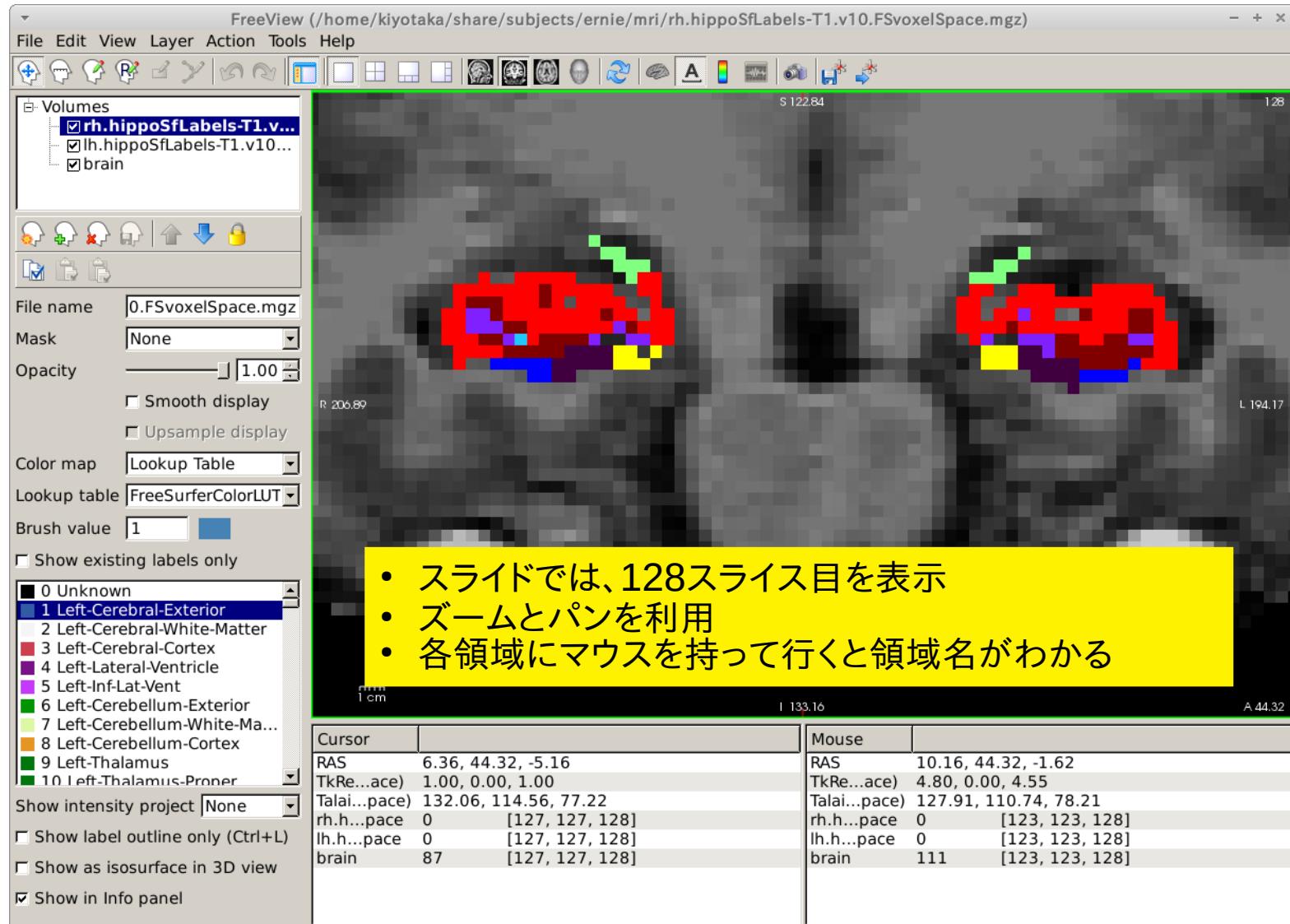


# 海馬のsubfield解析:低解像度

- FreeSurferで標準的に使われている空間 (conformed space) に重ね合わせるには、[lr]h.hippoSfLabels-T1.v10.mgz を重ねあわせる

```
freeview -v brainmask.mgz \
lh.hippoSfLabels-T1.v10.FSvoxelSpace.mgz:colormap=lut \
rh.hippoSfLabels-T1.v10.FSvoxelSpace.mgz:colormap=lut \
-layout 1 -viewport coronal -zoom 5
```

# 海馬のsubfield: 低解像度

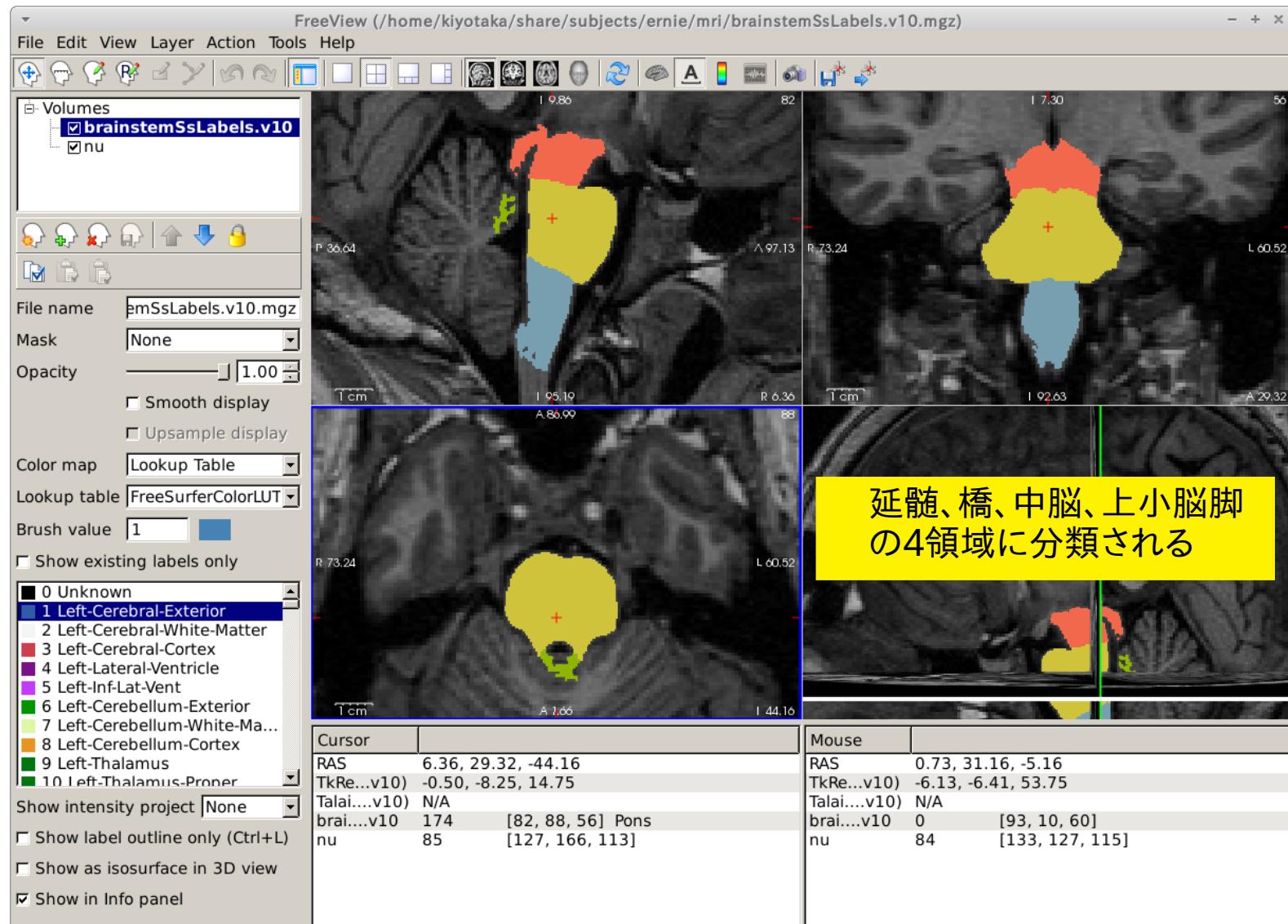


# 脳幹の構造解析: 高解像度

- ernieに解析済みデータあり
- 元画像空間に重ねあわせるには、mri ディレクトリにある  
`nu.mgz` に `brainstemSsLabels.v10.mgz` を重ねあわせる
- 高解像度(ボクセル1辺が0.333mm)

```
freeview -v nu.mgz \
brainstemSsLabels.v10.mgz:colormap=lut \
-layout 2 -viewport sagittal -zoom 3
```

# 脳幹の構造: 高解像度

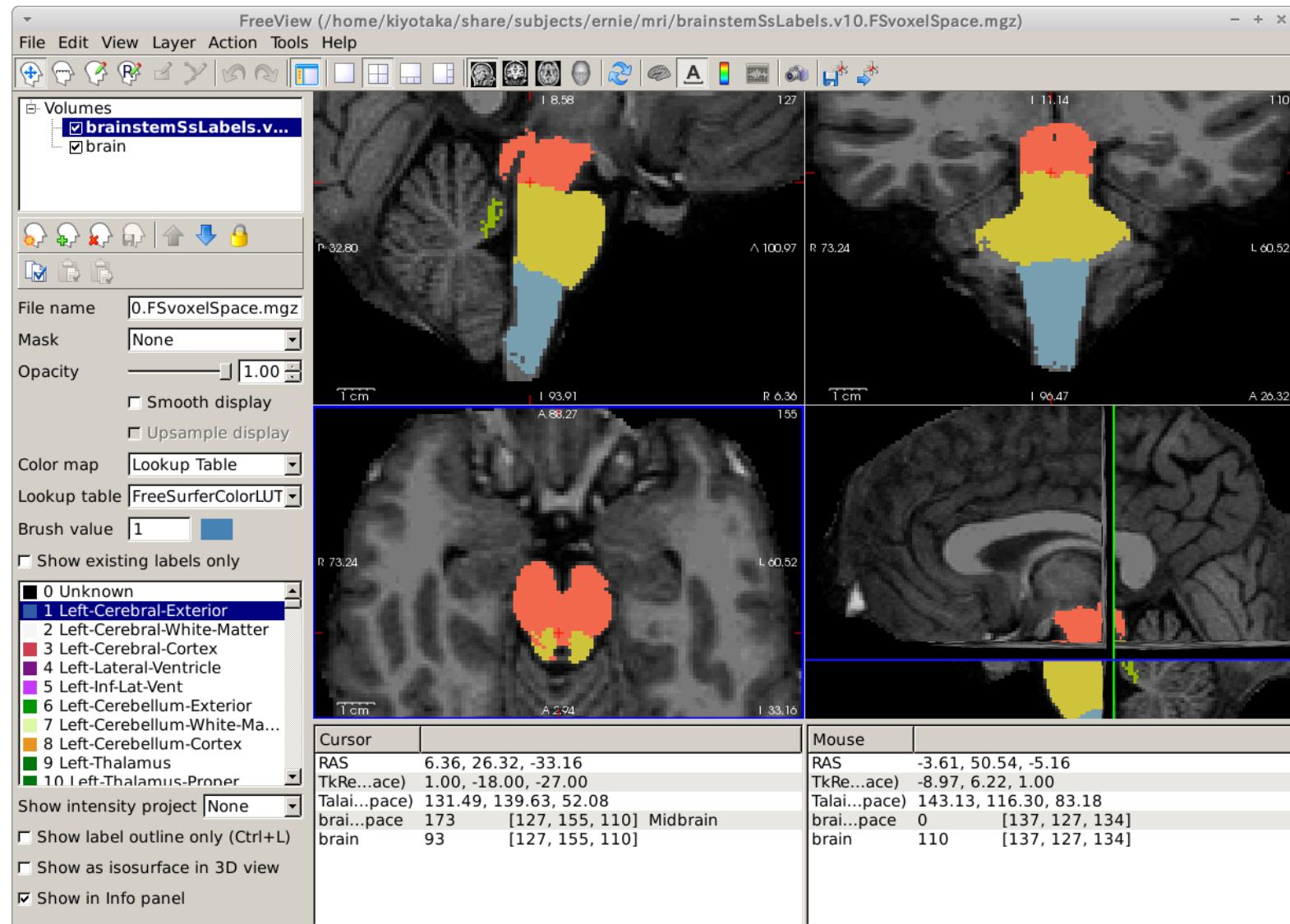


# 脳幹の構造解析: 低解像度

- FreeSurferで標準的に使われている空間 (conformed space) に重ね合わせるには、**brainstemSsLabels.v10.mgz** を重ねあわせる

```
freeview -v brain.mgz \
brainstemSsLabels.v10.FSvoxelSpace.mgz:colormap=lut -zoom 3
```

# 脳幹の構造: 低解像度



# Questions?