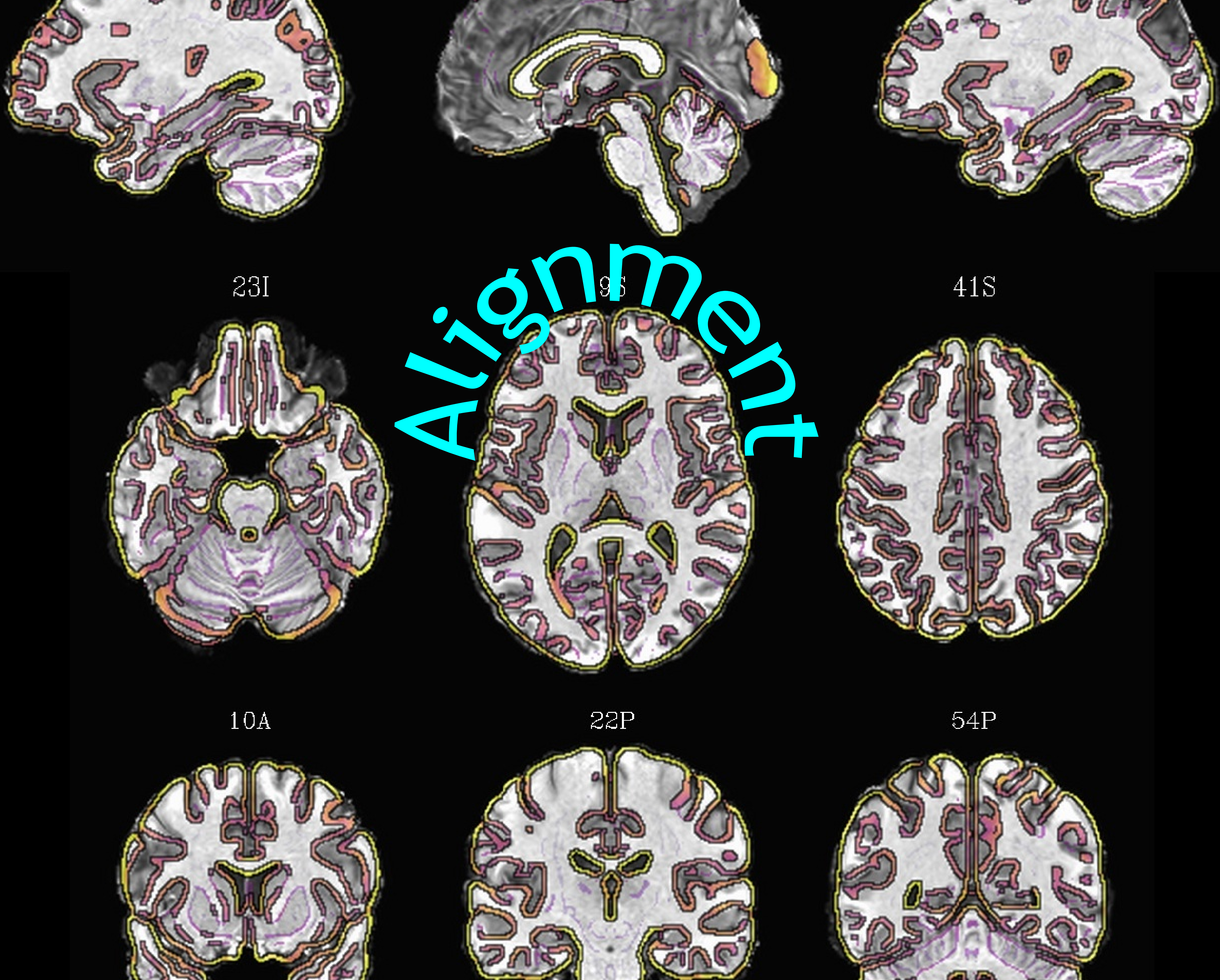
Alignment



Abbrevs used here

abbrev = abbreviation
AKA = also known as
anat = anatomical
diff = difference
dset = dataset
e.g. = exempli gratia (= “for example”)
EPI = echo planar image
Ex = example
FOV = field of view
i.e. = id est (= “that is”)
ijk = coordinate indices (integer)
NB = nota bene (= “note well”)
phys = physics or physical
ref = reference
subj = subject
vol = volume
vox = voxel(s)
xyz = physical coordinates (units of mm)

Alignment and its purpose(s)

- **Alignment (AKA registration):** bringing separate objects into the same space so that each location (e.g., voxel) within one object corresponds to the same location in the other
  - note: other software might refer to this process as “normalization,” but not in AFNI; too many things can be “normalized” (e.g., a vol’s brightness), so it is not descriptive enough on its own.

Alignment and its purpose(s)

- **Alignment (AKA registration):** bringing separate objects into the same space so that each location (e.g., voxel) within one object corresponds to the same location in the other
- <u>Common types of alignment</u>
  - ✧ **EPI-EPI:** align a subject's EPI vols across time (to a selected ref EPI)
    - for motion "correction" (= "reduction" or "mitigation")
  - ✧ **EPI-anat:** align a subject's EPI with anatomical vol
    - to assign a location with a functional result
  - ✧ **anat-template:** register a subject's anatomical to a standard template
    - for group level alignment, also compare/use locations from lit.
    - can then utilize associated atlases (= maps of regions)
  - ✧ and more:
    - distortion correction (such as EPI distortion)
    - "axialize": rotate brain to standardize slice viewing
    - quality control: check for left-right flipping
    - ….

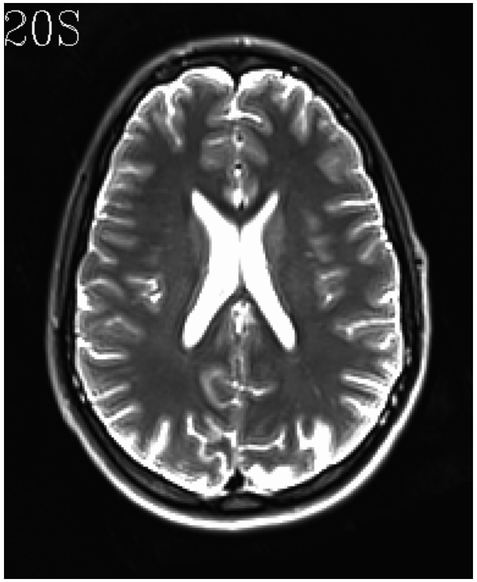
Mechanics of alignment

- Several different tools exist for alignment
  - choose based on properties of the images, what type of distortion/diff is present, and the **type** of desired alignment
- Some important questions about each alignment:
  - what is the **purpose** (e.g., undo distortion, or just overlay good volumes)?
  - are we aligning data from the same or different **subjects**?
  - do the data sets have similar or different **appearance** (both have bright CSF? or opposite tissue contrast?)
  - do both data sets have whole brain coverage? do both data sets have skull on or off?
- important concepts for selecting parameters include: contrast, smoothness, detail, resolution, field of view (FOV), concatenation, source/base dsets, grid
- NB: **afni_proc.py** will take care of ***many*** of these details for FMRI processing (so use it!), but it is still good to have some familiarity with them!

*We highlight **some** alignment/registration programs in this presentation.*
See the complete list on the AFNI website:
https://afni.nimh.nih.gov/pub/dist/doc/htmldoc/educational/classified_progs.html#align-register-warp-axialize-spatially

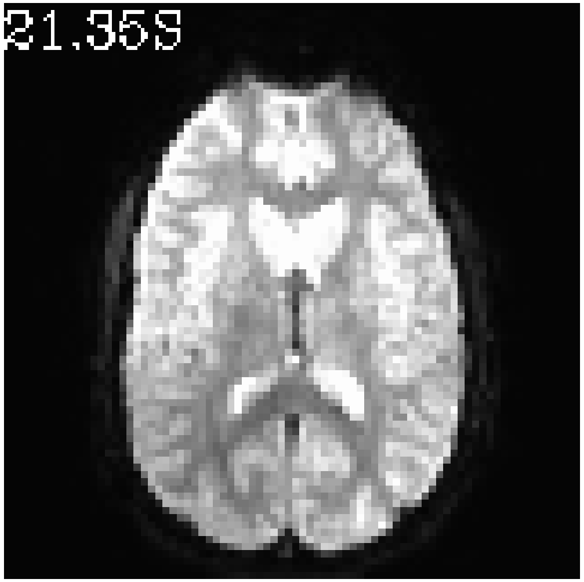
Alignment concepts: tissue contrast

- Do the vols have similar or different tissue contrast?



A) T2w

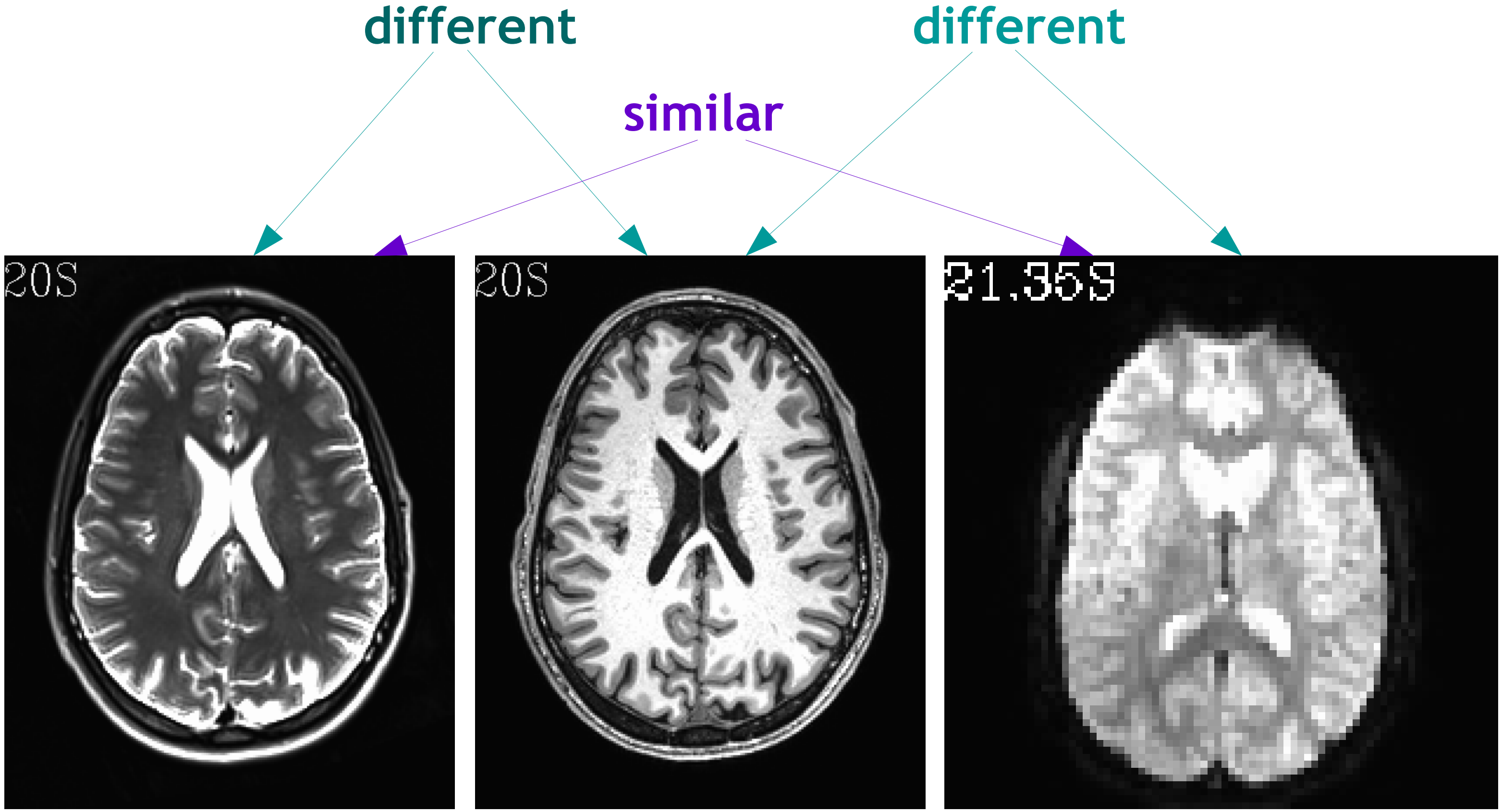
B) T1w



C) EPI

Alignment concepts: tissue contrast

- Do the vols have similar or different tissue contrast?
  → *determines what cost function we use (see slides 25-27).*



different

different

similar

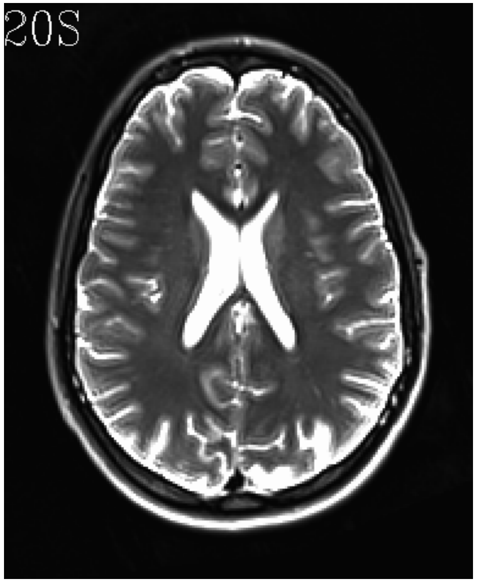
A) T2w

B) T1w

C) EPI

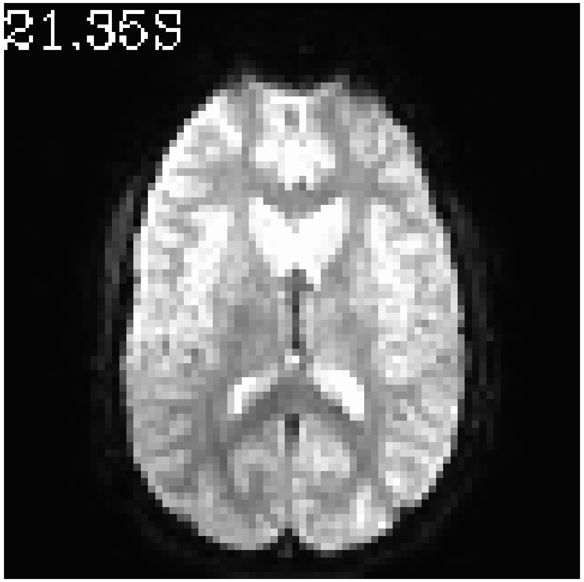
Alignment concepts: spatial resolution

- What kind of resolution do vols have (high/low)?



A) T2w

B) T1w



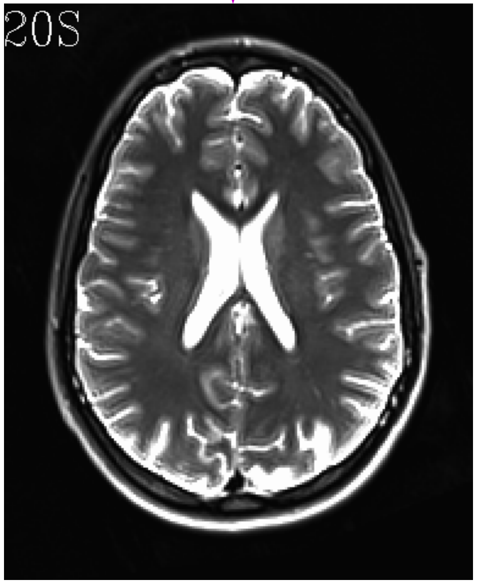
C) EPI

Alignment concepts: spatial resolution

- What kind of resolution do vols have (high/low)?

  → *affects how we choose which vol is "source" (to-be-warped) and which is "base" (=target) and level of warping*

high res

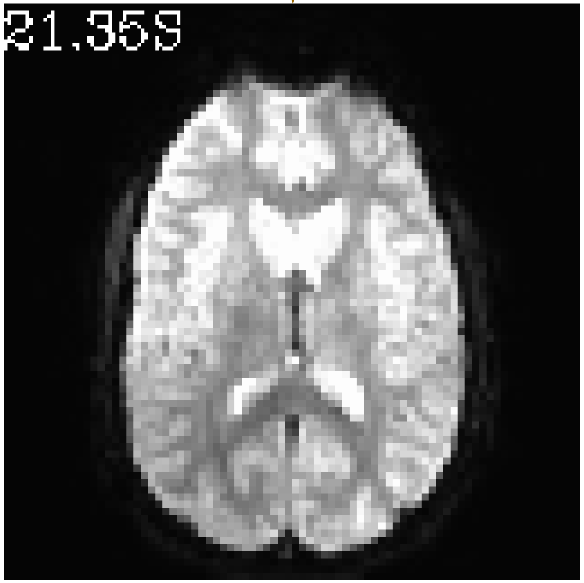


A) T2w

high res

B) T1w

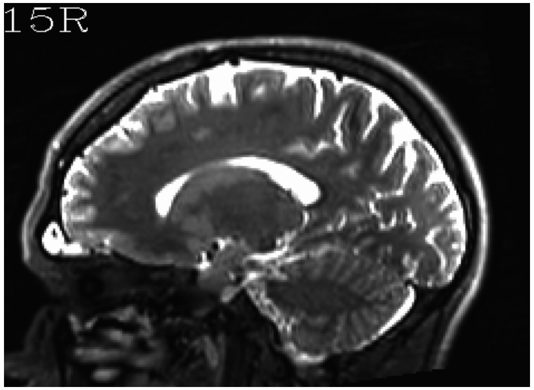
low res



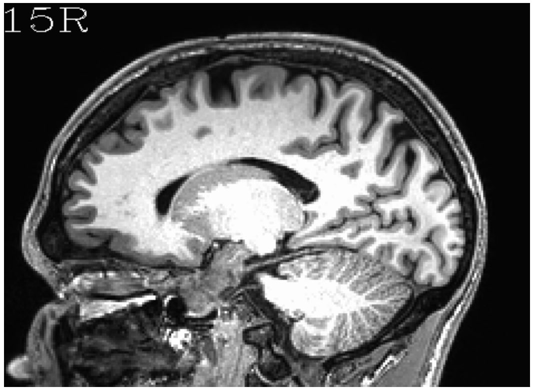
C) EPI

Alignment concepts: field of view (FOV) coverage

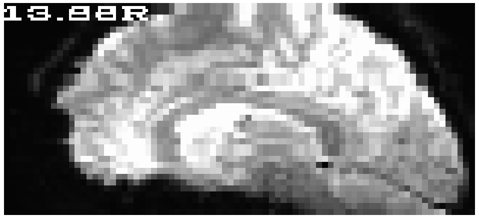
- What kind of coverage does the FOV have? Does it include the whole brain?



A) T2w



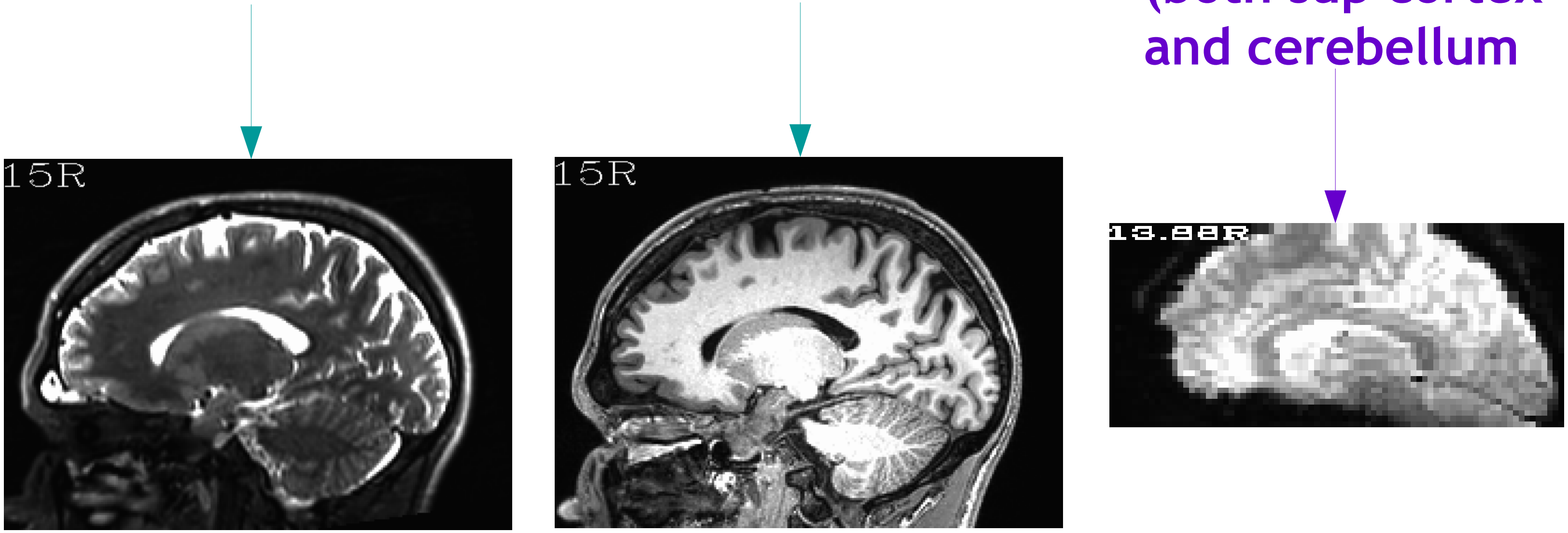
B) T1w



C) EPI

Alignment concepts: field of view (FOV) coverage

- What kind of coverage does the FOV have? Does it include the whole brain?
  → *might need extra options, affects how we view quality of alignment, and we might just expect some alignment problems*



whole brain FOV

whole brain FOV

**cut-off FOV
(both sup cortex
and cerebellum**

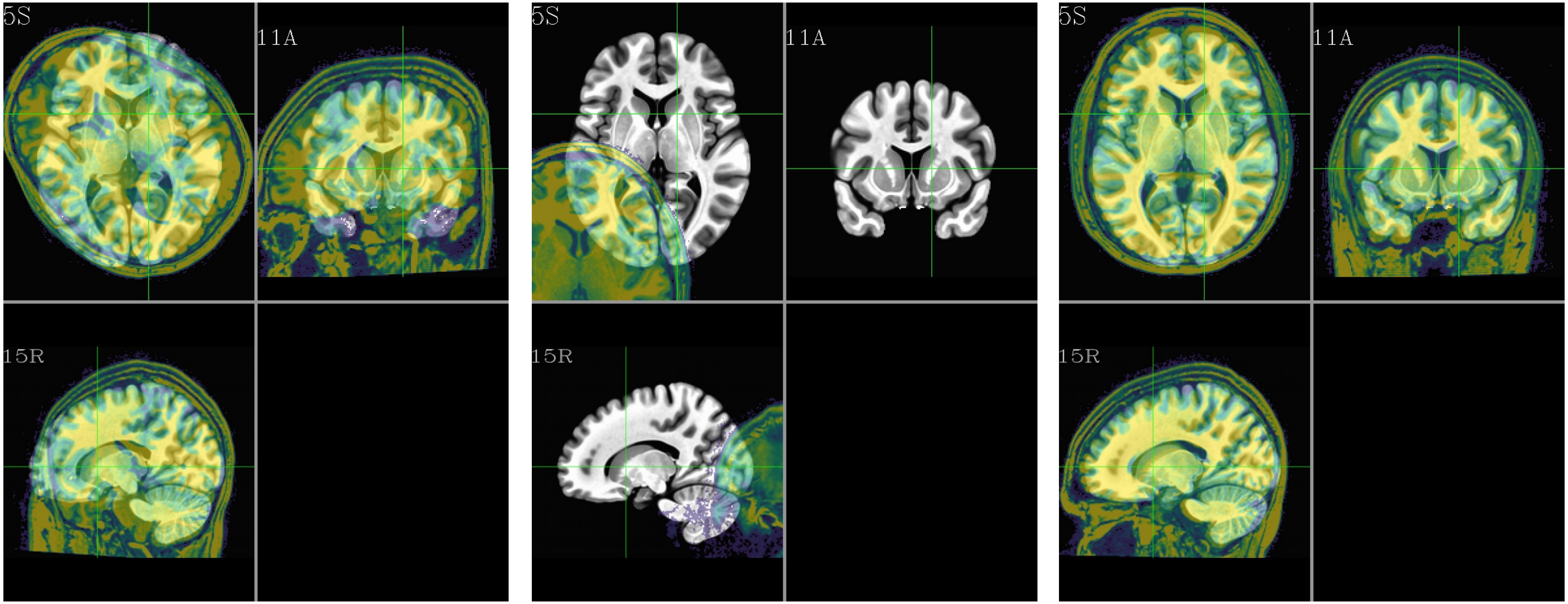
A) T2w

B) T1w

C) EPI

Alignment concepts: dset overlap

- How far apart are the “source” and “base” dsets? Do they overlap in the GUI?

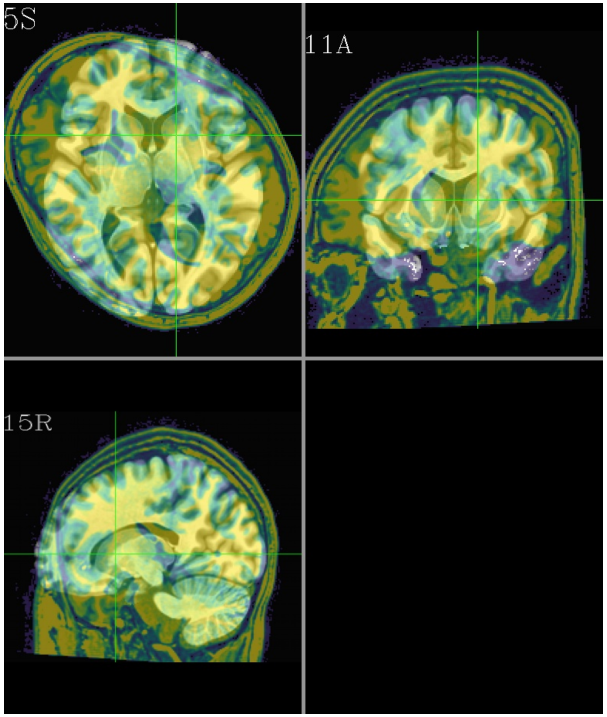


Alignment concepts: dset overlap

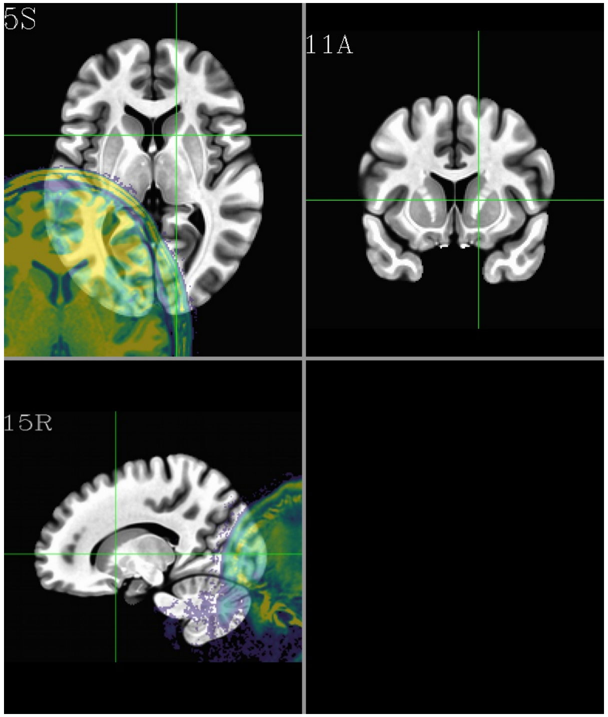
- How far apart are the “source” and “base” dsets? Do they overlap in the GUI?

  → *large differences in location, rotation, size, etc. can make alignment tricky.* ***The greater the overlap/similarity of the dsets, the better the alignment prospects.***

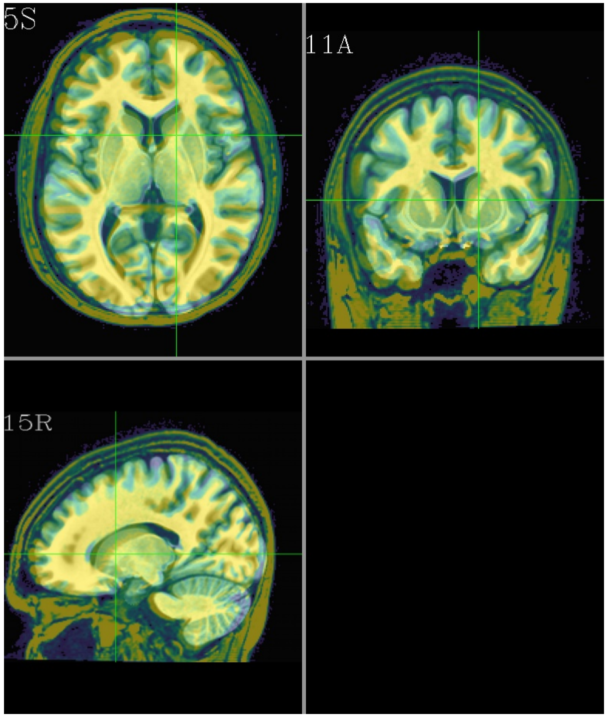
large rotation!



large translation!



pretty darn close



- To deal with large relative differences, one might need to use additional preprocessing steps or options for the alignment programs. [*See S1 at end of slides for image script.*]

How is alignment transformation calculated?

- The fundamental aspects are the same across the tools:

  Take two images, a base (=reference) and source (=to be adjusted)

  1) quantify "how similar" they are,

  2) if they are "similar enough," then stop;
     otherwise, tweak/perturb the source image,

  3) quantify "how similar" the new source and base are, … [repeat]

How is alignment transformation calculated?

- The fundamental aspects are the same across the tools:

  Take two images, a base (=reference) and source (=to be adjusted)

  1) quantify "how similar" they are,

  2) if they are "similar enough," then stop;

  otherwise, tweak/perturb the source image,

  3) quantify "how similar" the new source and base are, … [repeat]
- To *quantify how similar dsets are*, we need a **cost function**
  - the **cost function** is how we take values from both dsets and make an overall assessment of a desired property; different functions yield different alignments
  - For example, could:
    - reward similarity or when peak values line up (e.g., for matching contrasts)
    - reward when *anti*-peaks line up (e.g., for opposite contrasts)

Cost functions in AFNI

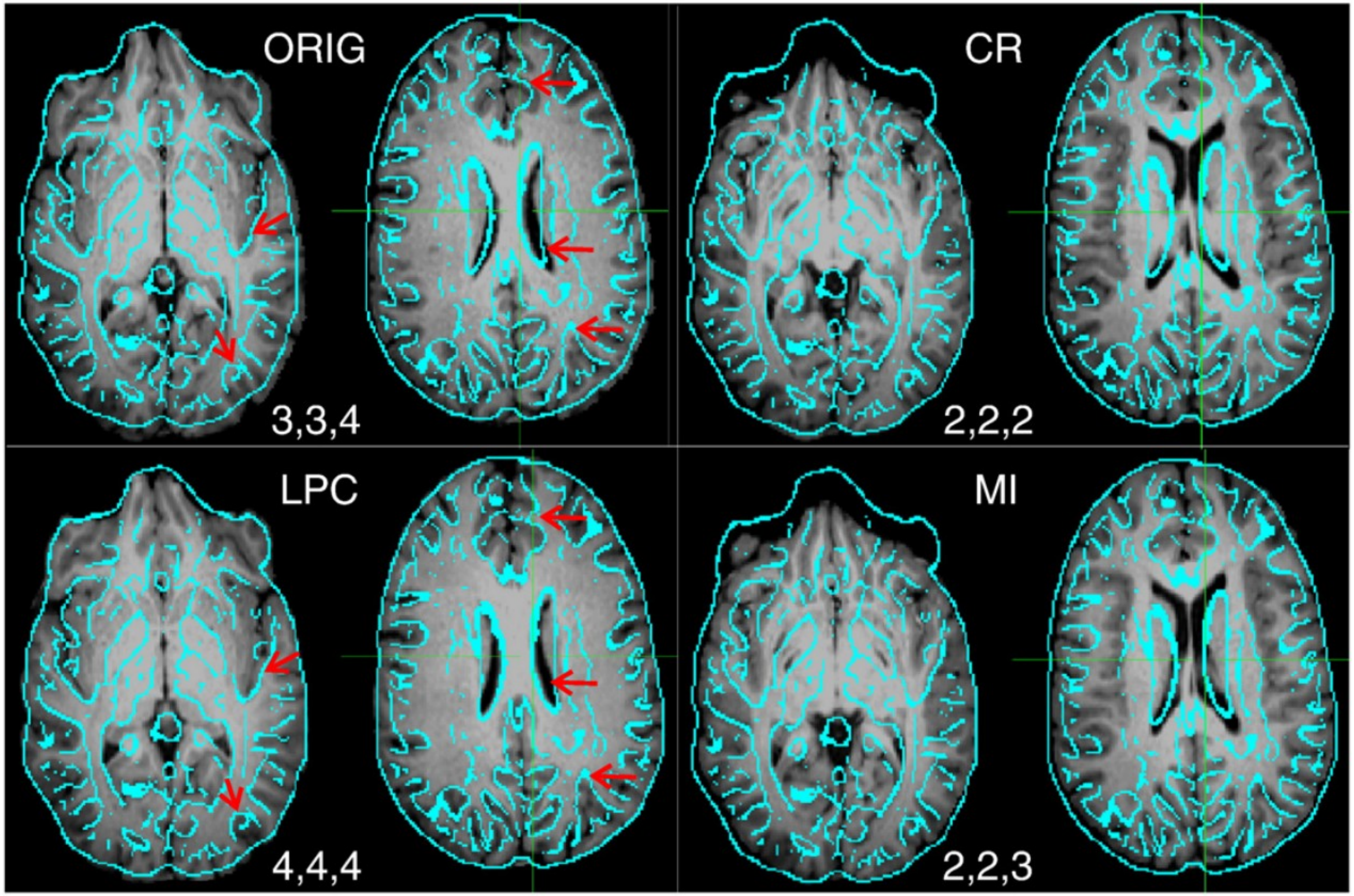
- There are *many* cost functions in AFNI, from historical development/evolution but there are just a few we now recommend (from 3dAllineate's help):

```
ls  :: 1 - abs(Pearson correlation coefficient)
sp  :: 1 - abs(Spearman correlation coefficient)
mi  :: - Mutual Information = H(base,source)-H(base)-H(source)
crM :: 1 - abs[ CR(base,source) * CR(source,base) ]
nmi :: 1/Normalized MI = H(base,source)/[H(base)+H(source)]
je  :: H(base,source) = joint entropy of image pair
hel :: - Hellinger distance(base,source)
crA :: 1 - abs[ CR(base,source) + CR(source,base) ]
crU :: CR(source,base) = Var(source|base) / Var(source)
lss :: Pearson correlation coefficient between image pair
lpc :: nonlinear average of Pearson cc over local neighborhoods
lpa :: 1 - abs(lpc)
lpc+:: lpc + hel + mi + nmi + crA + overlap
ncd :: mutual compressibility (via zlib) -- doesn't work yet
```

- "**lpa**" is basic choice for vols with *similar* contrast
- "**lpc**" is basic choice for vols with *differing/opposite* contrast
  - and "**lpc+ZZ**" might be even better for awkward cases: robuster, but slower
- ("**ls**" is okay for simple EPI-EPI alignment in 3dvolreg)

Cost functions in AFNI

- **“lpc”** in action, EPI-to-anat (olay: edges from same EPI slice; ulay: matched anat):



ZS Saad, DR Glen, G Chen, MS Beauchamp, R Desai, RW Cox (2009). A new method for improving functional-to-structural MRI alignment using local Pearson correlation. NeuroImage 44: 839-848.

How is alignment transformation calculated?

- The fundamental aspects are the same across the tools:

  Take two images, a base (=reference) and source (=to be adjusted)

  1) quantify “how similar” they are,

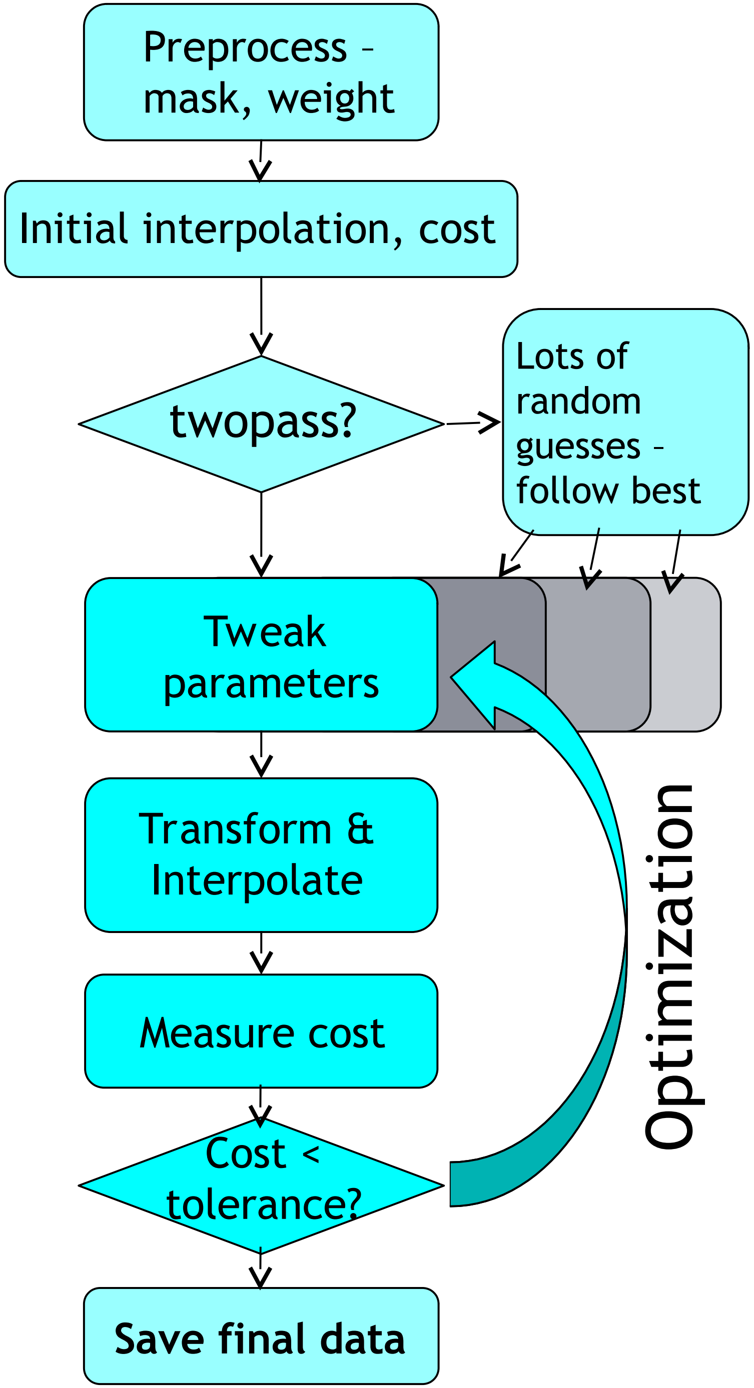
  2) if they are “similar enough,” then stop;

  otherwise, tweak/perturb the source image,

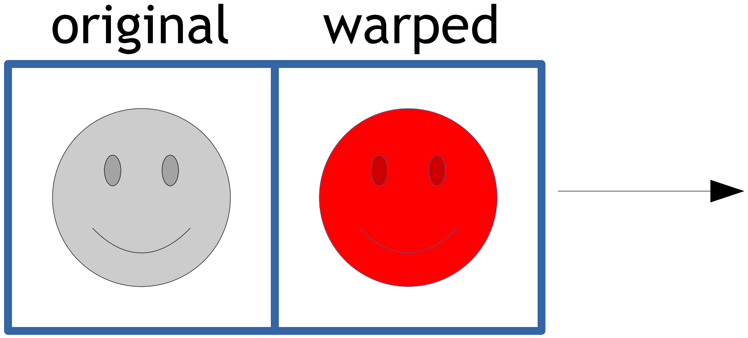
  3) quantify “how similar” the new source and base are, … [repeat]
- To *quantify how similar dsets are*, we need a **cost function**
  - the **cost function** is how we take values from both dsets and make an overall assessment of a desired property; different functions yield different alignments
  - For example, could:
    - reward similarity or when peak values line up (e.g., for matching contrasts)
    - reward when *anti*-peaks line up (e.g., for opposite contrasts)
- To *tweak/perturb the source*, we need to know what kind of parameters to change
  - rotations, translations, shear, scaling, higher order (determined by **degrees of freedom)**
  - and whether we are matching entire FOV or smaller pieces (family: **linear/affine** or **nonlinear**)

View inside *alignment* process

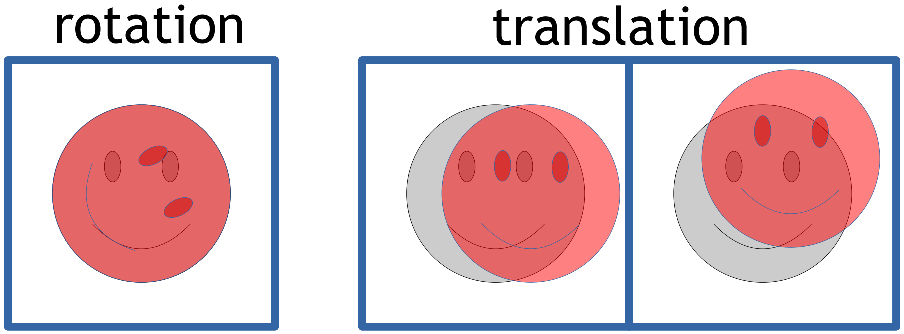
- **Preprocess** – mask data, weight data
- If far off, take some random guesses (-twopass)
- Optimize parameters on initial or best sets (6,12,39,1000's)
  - Use new parameters to transform input
    - Interpolate onto base data's grid
  - Measure alignment error with cost functional
    - Less than minimum error - finished
    - Better - keep adjusting with same direction
    - Worse – try other parameters
- Create final output by interpolating onto output grid
  - save datasets, transform parameters



Types of warps: Rigid body (average FOV fit)



Parameters (applied globally)

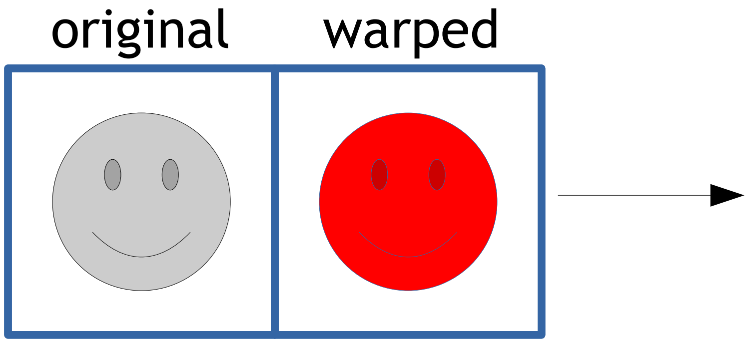


There are two parameters in the **rigid body** family of warps:
rotation *translation*

In 3D alignment, each parameter has three **degrees of freedom (DOFs)**:

\+ **<u>rigid body (AKA "solid body")</u>**
6 DOF = 3 transl + 3 rot
Ex: *3dvolreg*

Types of warps: Linear affine (average FOV fit)



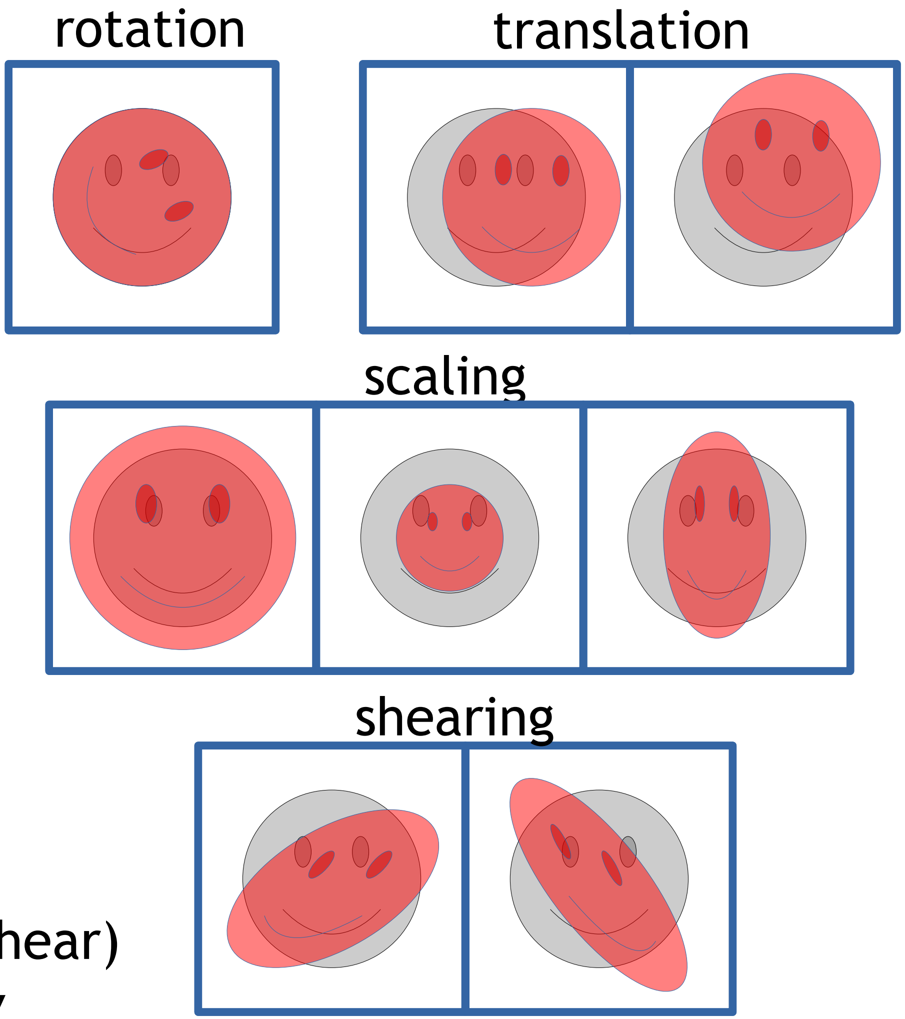
There are four parameters in the **linear affine** family of warps:

rotation *translation*
shearing *scaling*

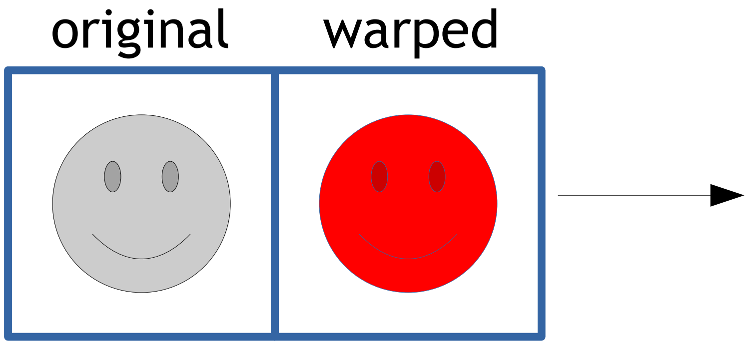
In 3D alignment, each parameter has three **degrees of freedom (DOFs)**:

- \+ rigid body (AKA “solid body”)
  6 DOF = 3 transl + 3 rot
  Ex: *3dvolreg*
- \+ **linear affine**
  12 DOF = 3*(transl + rot + scale + shear)
  Ex: *3dAllineate*, *align_epi_anat.py*

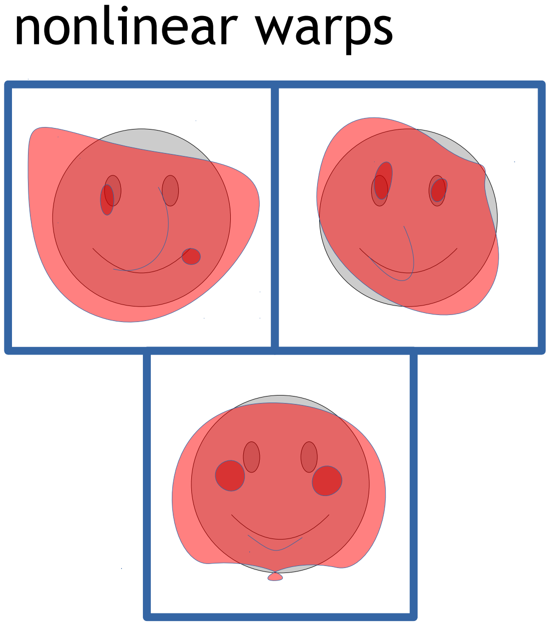
Parameters (applied globally)



Types of warps: nonlinear (patch refinement)



Parameters (applied *locally*)



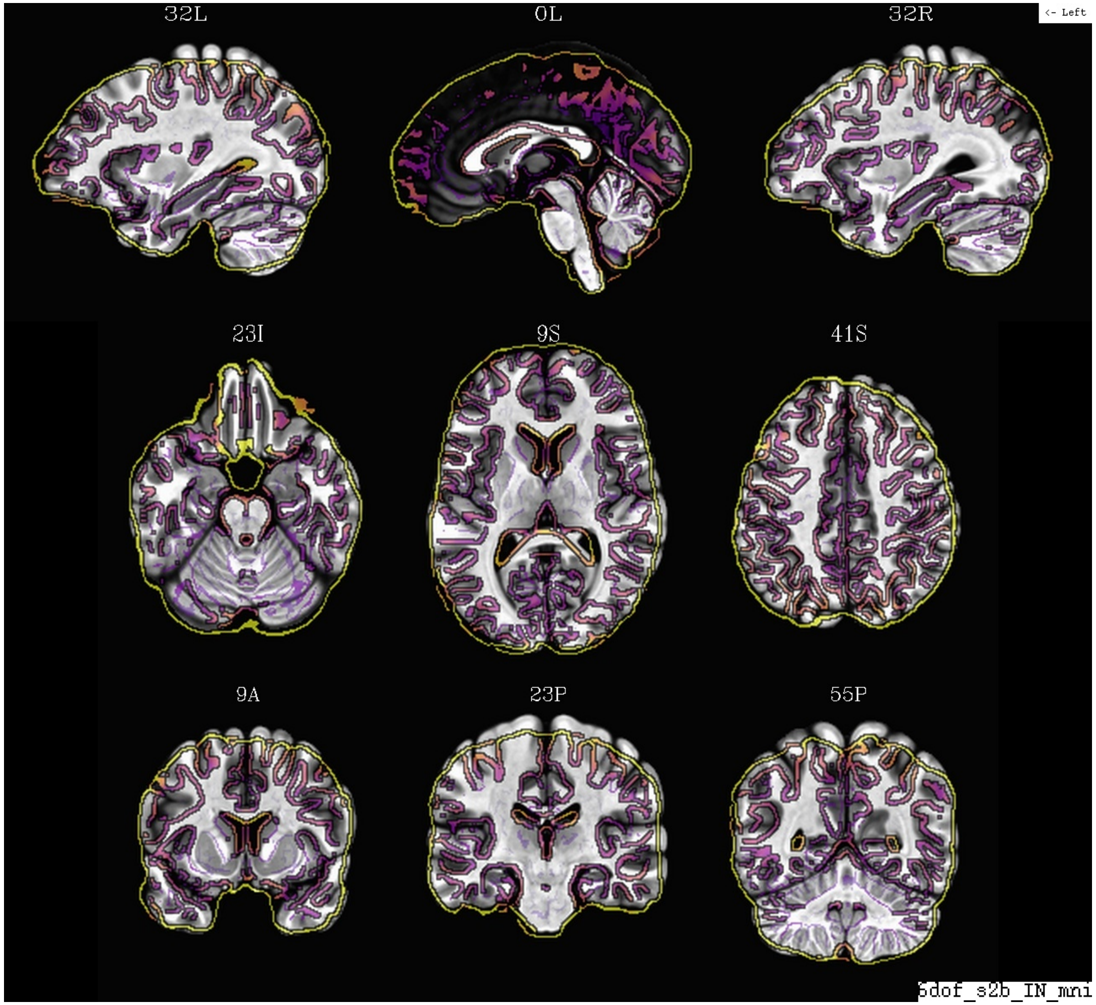
In nonlinear warping, a first pass is done with simpler *global* alignment, and then the matching is refined *locally* in “patches” down to a certain length scale.

There are many more DOFs used here, and one can match more features-- but details also matter more (e.g., skullstripping)

+ **<u>nonlinear warping</u>**
  12 DOF initially + hundreds (or thousands) more DOFs in refinement steps
  Ex: *3dQwarp*, *sswarper2*, *@SSwarper*, *@animal_warper*, *auto_warp.py*

Types of warps: example case “rigid”

ulay: base template
olay: warped dset (6DOF, linear, “rigid body”)



PLOT: **`@snapshot_volreg TEMPLATE_VOL WARPED_VOL`**

Compare alignment with overlaying edges of warped source dset on base template

“Base” volume in this case is MNI template; “source” is typical subject anat vol.

Overall fit: OK, some major parts align, but brain shapes are (and remain) different.

Types of warps: example case “linear affine”

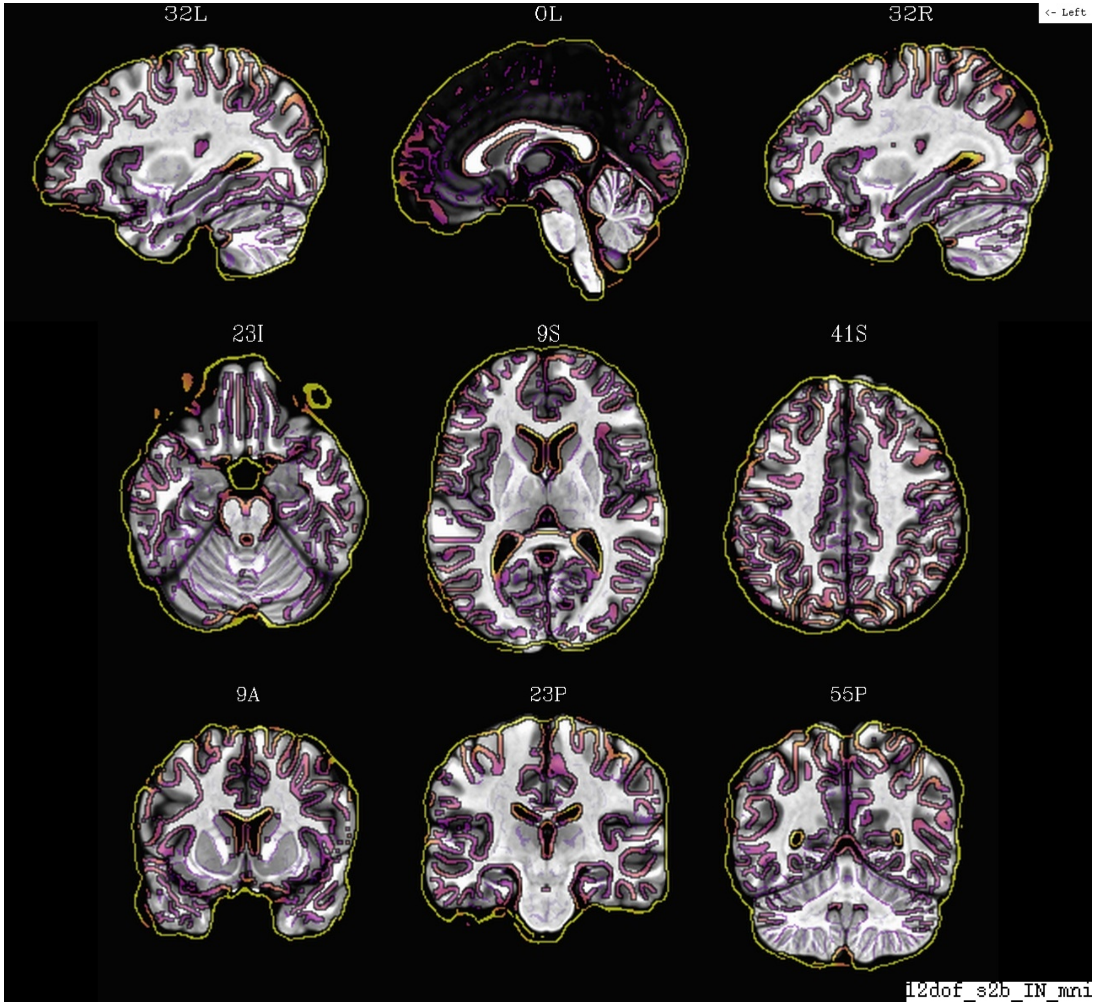
Compare alignment with overlaying edges of warped source dset on base template

“Base” volume in this case is MNI template; “source” is typical subject anat vol.

Overall fit: Good, general shape and many sulci/gyri match, but not all and many are approximate.

Bonus Type -
rigid equivalent
6-parameters extracted.
"Axializing" subjects to template of different size

ulay: base template
olay: warped dset (12DOF, affine)



PLOT: **@snapshot_volreg TEMPLATE_VOL WARPED_VOL**

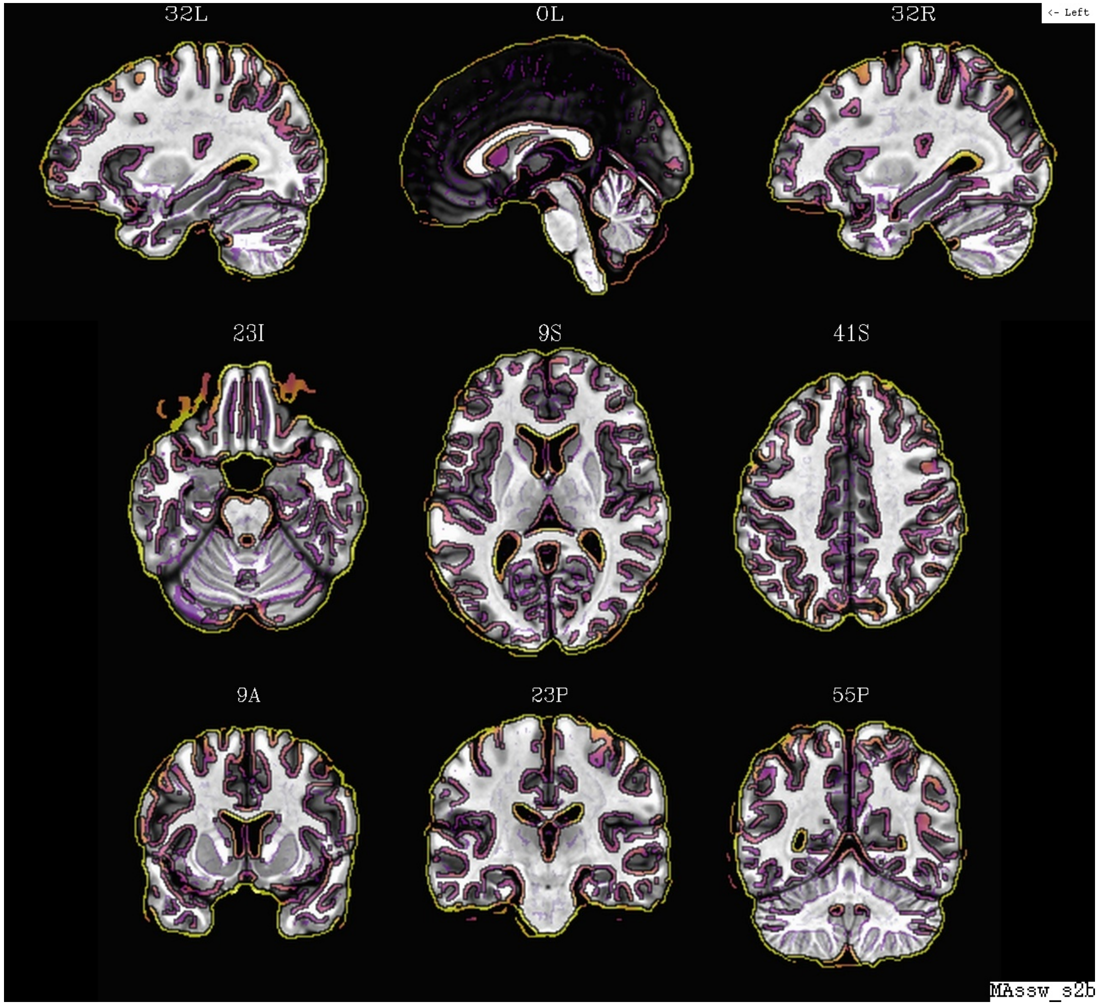
Types of warps: example case “nonlinear”

Compare alignment with overlaying edges of warped source dset on base template

“Base” volume in this case is MNI template; “source” is typical subject anat vol.

Overall fit: Very good, nearly all sulci/gyri match closely throughout.

ulay: base template
olay: warped dset (nonlinear, sswarper2/@SSwarper)



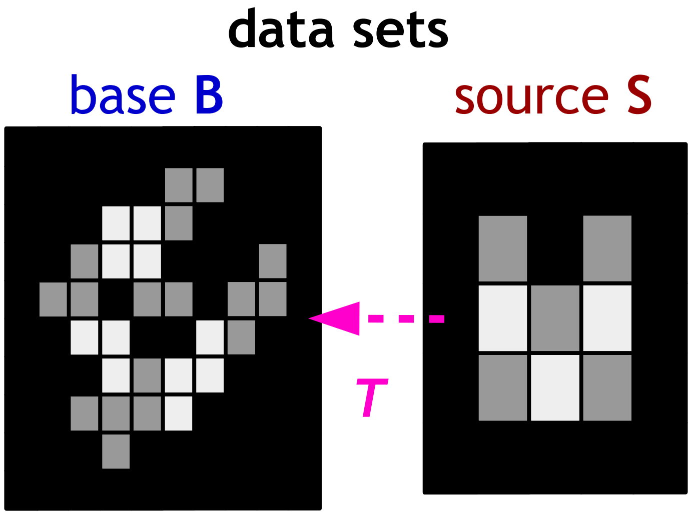
`PLOT: @snapshot_volreg TEMPLATE_VOL WARPED_VOL`

Transformation and mapping

- What is a transformation (or warp)?

→ *It is a rule for mapping information from a source dset **S** onto a base **B**'s grid.*

*It says where a given spot (x_B, y_B, z_B) in **B** pulls information from in **S**.*

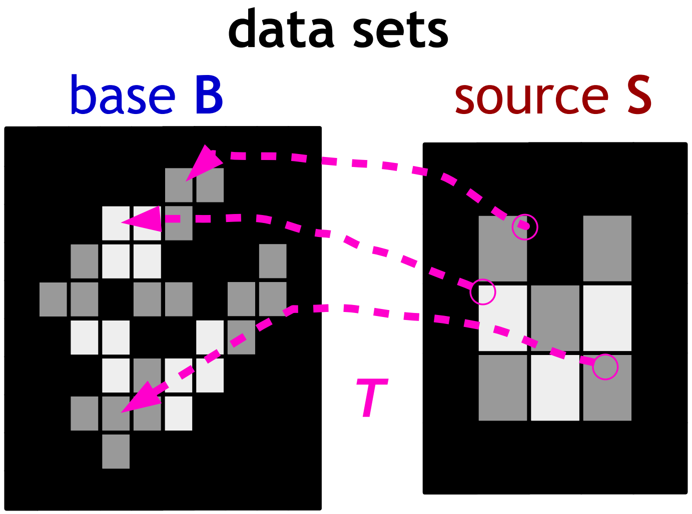


For alignment, we want a transform ***T***, such that ***B' = T(S)*** is similar to ***B***.

Transformation and mapping

- What is a transformation (or warp)?

→ *It is a rule for mapping information from a source dset **S** onto a base **B**'s grid. It says where a given spot (x_B, y_B, z_B) in **B** pulls information from in **S**.*



For alignment, we want a transform ***T***, such that ***B' = T(S)*** is similar to ***B***.

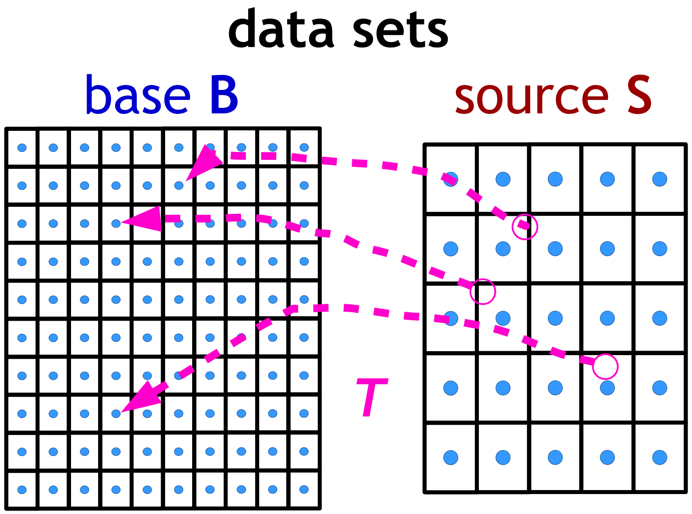
We choose how much warping ***T*** is allowed to do: match FOV average or refine over smaller patches; and how much "freedom" of movement.

Goal: maximize structure matching, and minimize smoothing.

Transformation and mapping

- What is a transformation (or warp)?

→ *It is a rule for mapping information from a source dset **S** onto a base **B**'s grid.*
*It says where a given spot (x_B, y_B, z_B) in **B** pulls information from in **S**.*

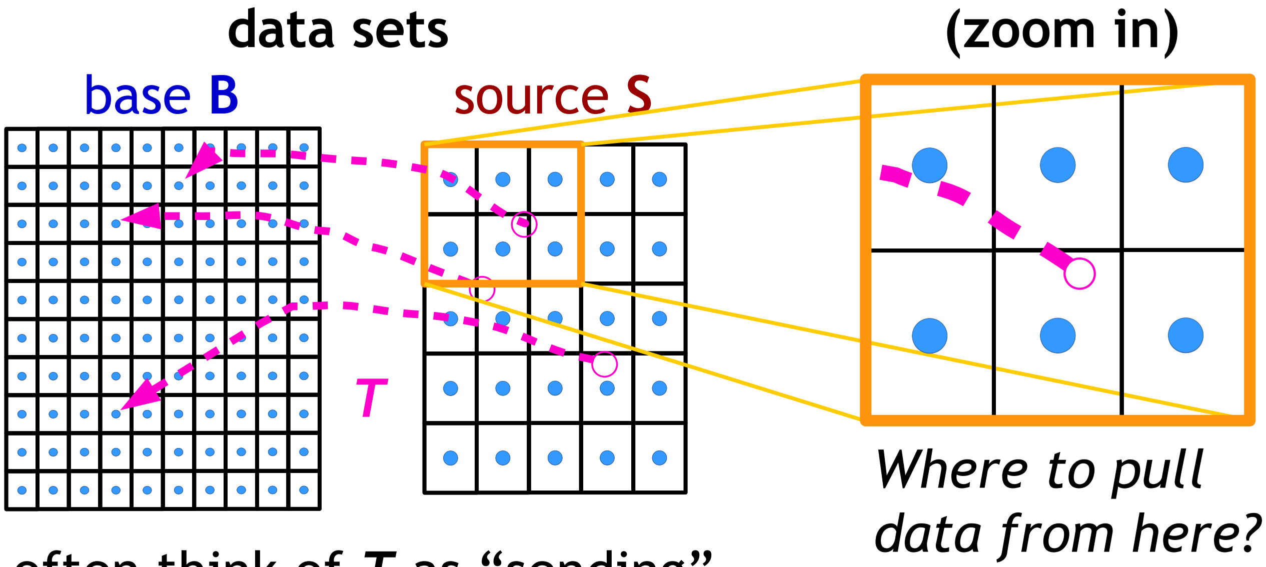


We often think of ***T*** as "sending" values from ***S*** to ***B***, but actually we start from a point in ***B*** and ask, "Where do I get my value from in ***S***?"

Transformation and mapping

- What is a transformation (or warp)?

  → *It is a rule for mapping information from a source dset **S** onto a base **B**'s grid.*
  *It says where a given spot (x_B, y_B, z_B) in **B** pulls information from in **S**.*

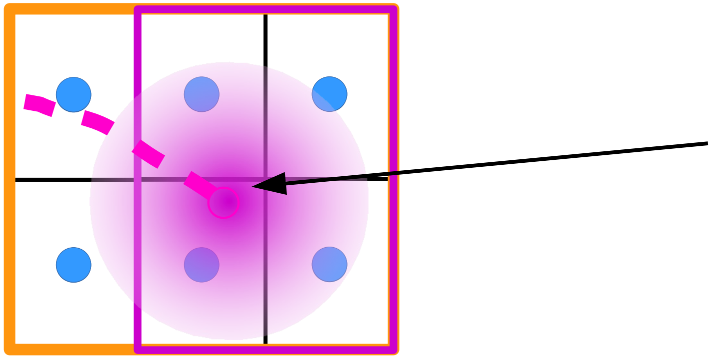


Where to pull data from here?

We often think of ***T*** as "sending" values from ***S*** to ***B***, but actually we start from a point in ***B*** and ask, "Where do I get my value from in ***S***?"

Mapping and Interpolation

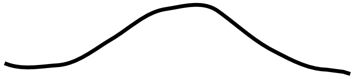
Where exactly to pull data from?
Choose with interpolation (or resampling) mode



<u>Basic case:</u>
the location to pull data "from" is not directly on a centroid, so we interpolate with a weighted average-- here a cubic spline (in 3D)

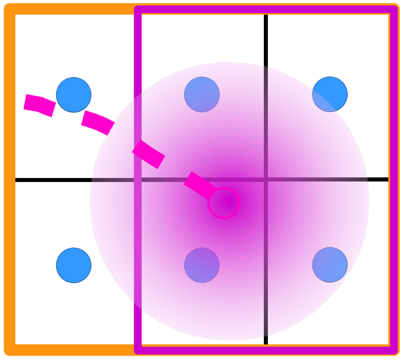
Cu = "cubic Lagrange polynomial"
→ decent interp (bit of smoothing)

1D vers:

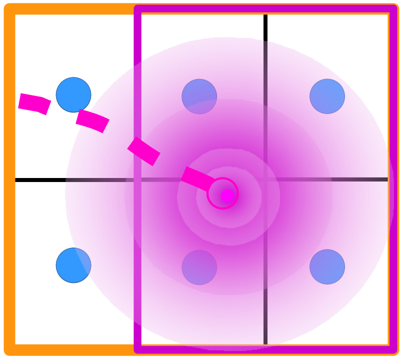


Mapping and Interpolation

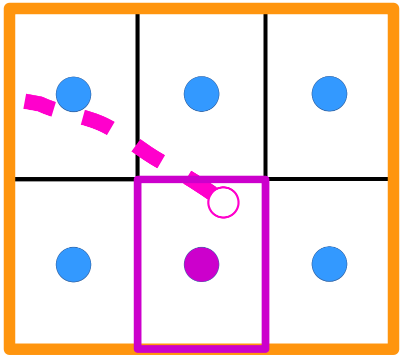
Where exactly to pull data from?
Choose with interpolation (or resampling) mode



Cu = “cubic Lagrange polynomial”
→ decent interp (bit of smoothing)

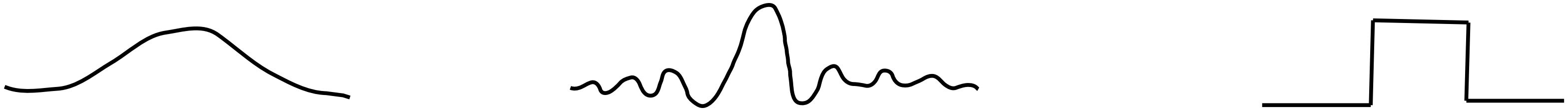


wsinc5 = “sinc function”
→ preserve edges and sharpness (some ringing)



NN = “nearest neighbor”
→ preserve int values (atlases!)

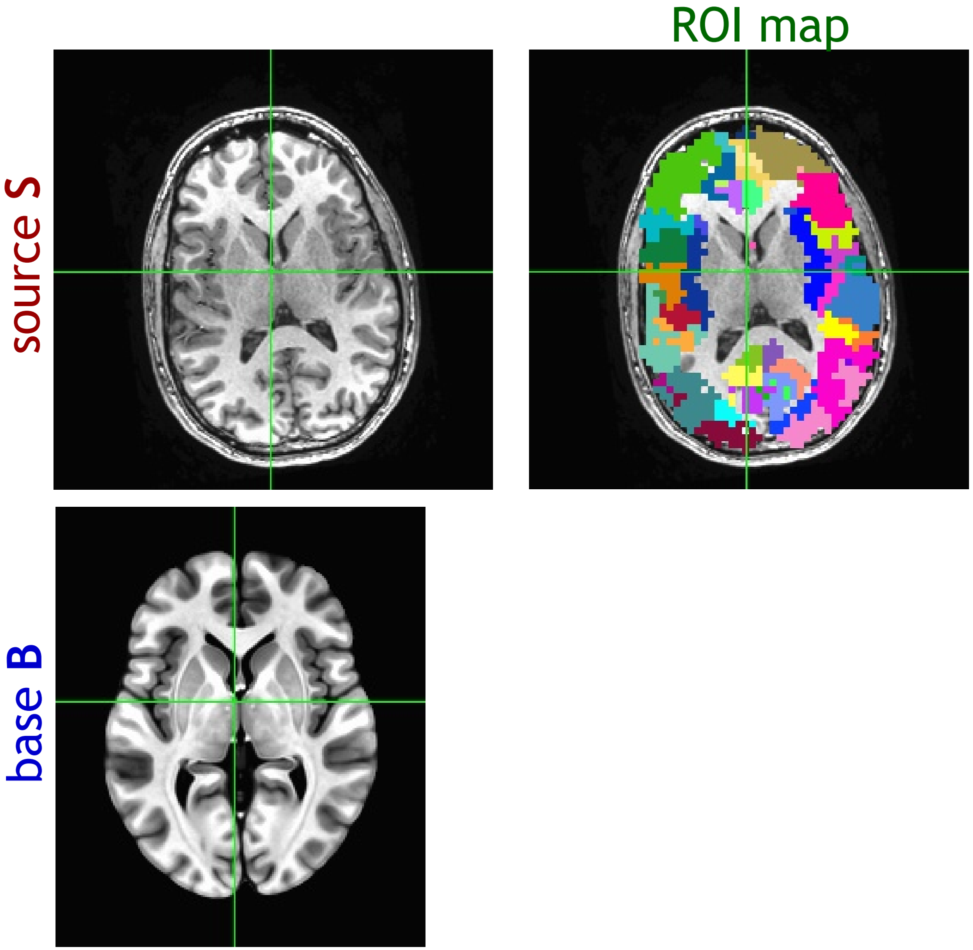
1D vers:



*(There are other modes, such as linear **Li**, etc.; see program help files.)*

Applying transformations: moving ROIs

- How can we move ROIs between spaces?

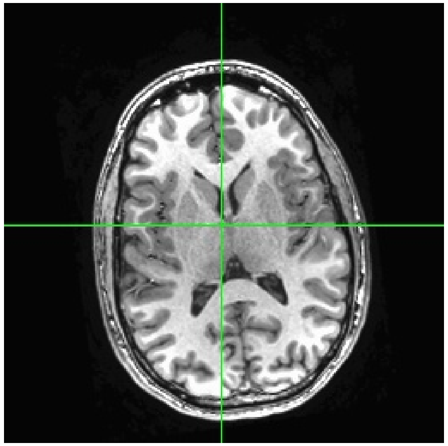


Applying transformations: moving ROIs

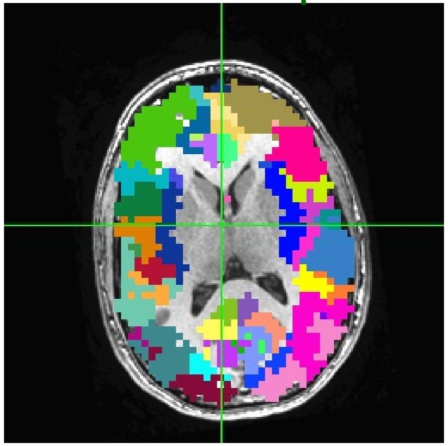
- How can we move ROIs between spaces?

  → *Once we have the mapping* ***T*** *from* ***S*** *to* ***B****, we can* ***apply*** *it to other dsets-- like an atlas, a map of ROIs, and more.*

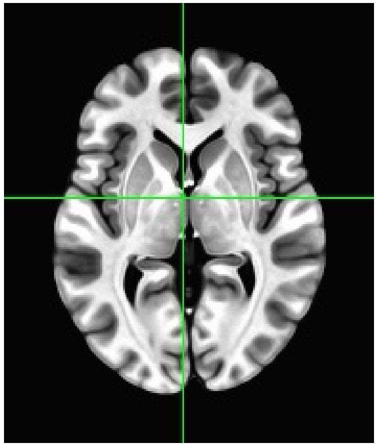
source S



ROI map



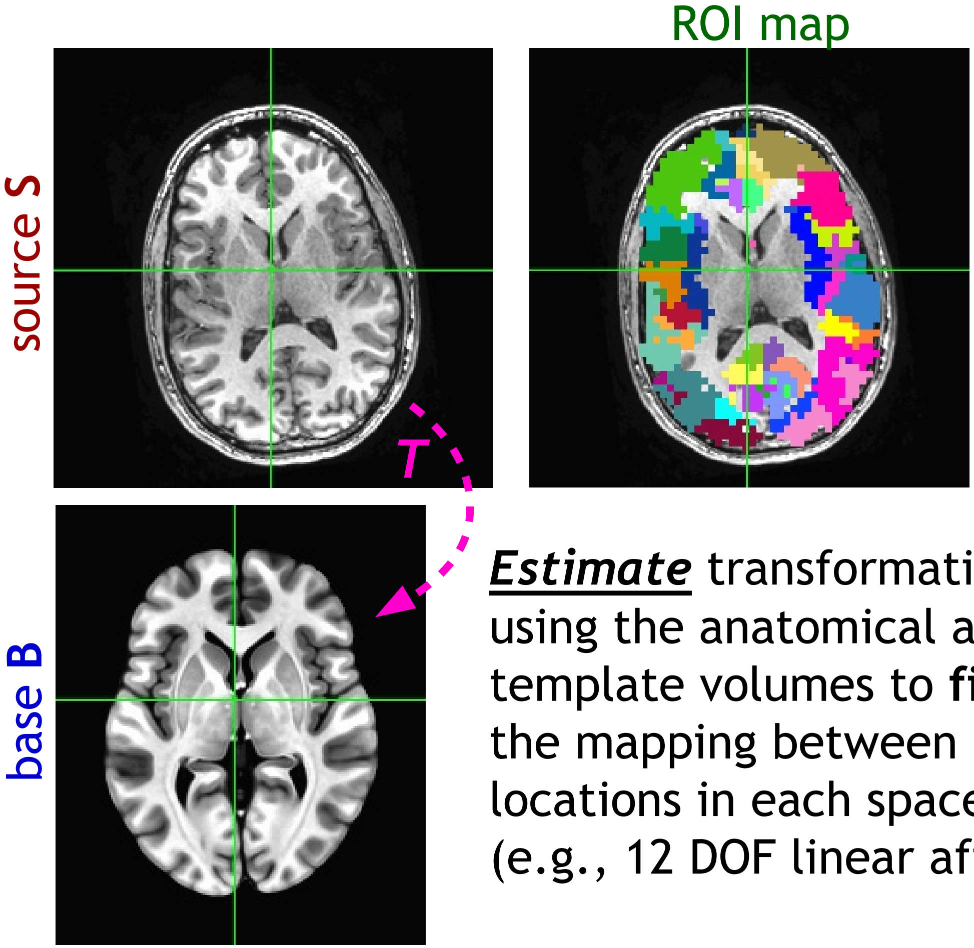
base B



Applying transformations: moving ROIs

- How can we move ROIs between spaces?

  → *Once we have the mapping **T** from **S** to **B**, we can **apply** it to other dsets--like an atlas, a map of ROIs, and more.*

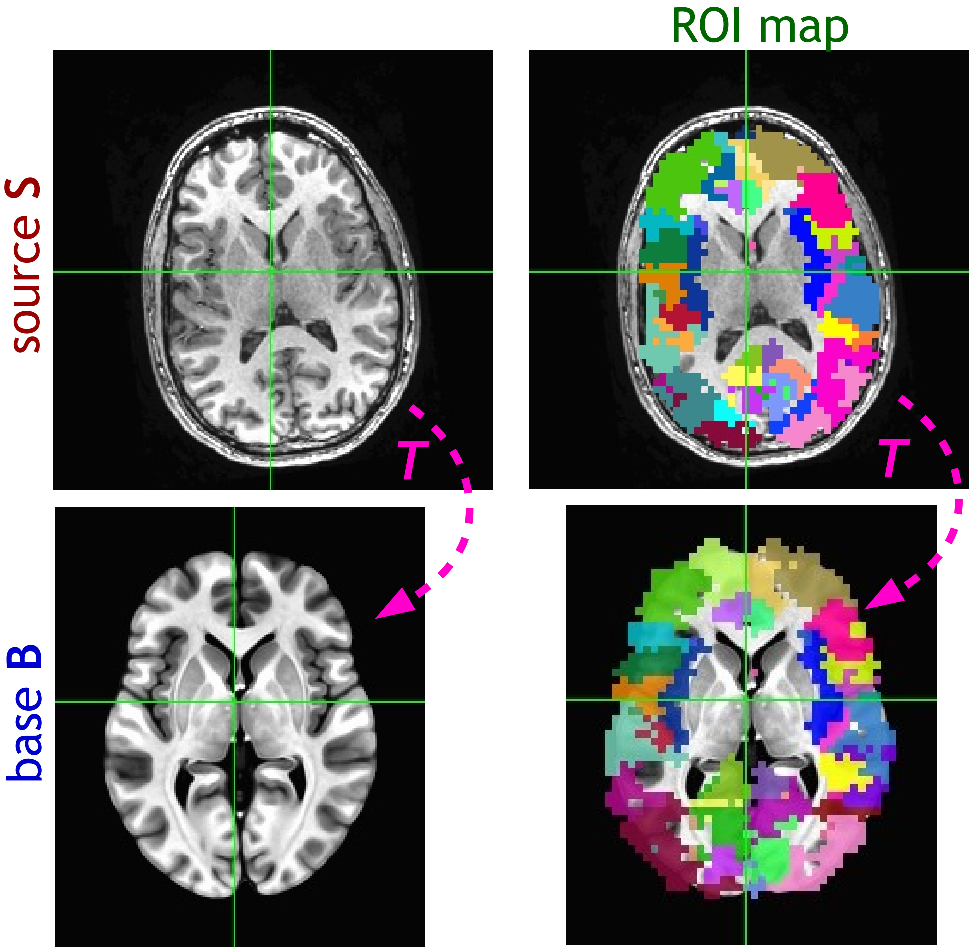


<u>Estimate</u> transformation *T*, using the anatomical and template volumes to **find** the mapping between locations in each space (e.g., 12 DOF linear affine).

Applying transformations: moving ROIs

- How can we move ROIs between spaces?

  → *Once we have the mapping **T** from **S** to **B**, we can **apply** it to other dsets--like an atlas, a map of ROIs, and more.*



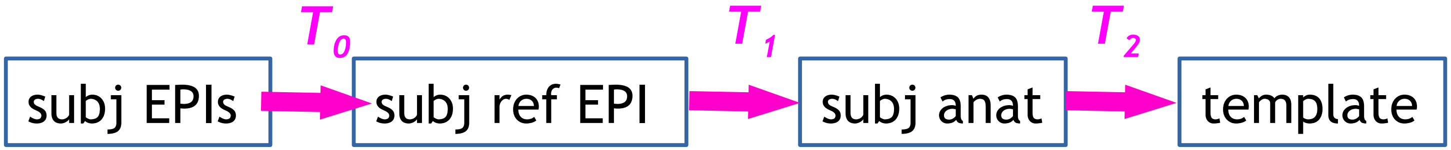
<u>Apply</u> transformation *T* to the ROI map.
We also include an option to make the interpolation be "NN" (nearest neighbor) in order to avoid smoothing of ROIs: ints stay ints, so the ROIs keep their identity.

Concatenating transforms

- Note: as observed earlier, any alignment/warping/interpolation/resampling leads to ***smoothing*** of data.
- Therefore, when mapping data through several spaces (EPI → anat → template), we **DON'T** want to regrid at each step; we ***concatenate*** *the transforms into a single one, and then have to only regrid once!*

Concatenating transforms

- Note: as observed earlier, any alignment/warping/interpolation/resampling leads to ***smoothing*** of data.
- Therefore, when mapping data through several spaces (EPI → anat → template), we **DON'T** want to regrid at each step; we ***concatenate*** *the transforms into a single one, and then have to only regrid once!*
- So, calculate individual transforms, such as within EPI (e.g., 3dvolreg), EPI-anat (e.g., align_epi_anat.py) and anat-template (e.g. sswarper2):



subj EPIs

T_0

subj ref EPI

T_1

subj anat

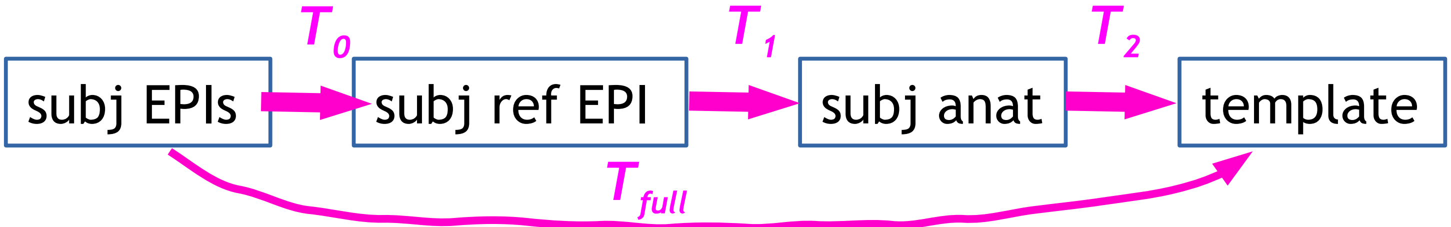
T_2

template

- but ***DON'T*** apply T_0 to make a new dset, and then apply T_1 to that, etc.

Concatenating transforms

- Note: as observed earlier, any alignment/warping/interpolation/resampling leads to ***smoothing*** of data.
- Therefore, when mapping data through several spaces (EPI → anat → template), we **DON'T** want to regrid at each step; we ***concatenate*** *the transforms into a single one, and then have to only regrid once!*
- So, calculate individual transforms, such as within EPI (e.g., 3dvolreg), EPI-anat (e.g., align_epi_anat.py) and anat-template (e.g. sswarper2):



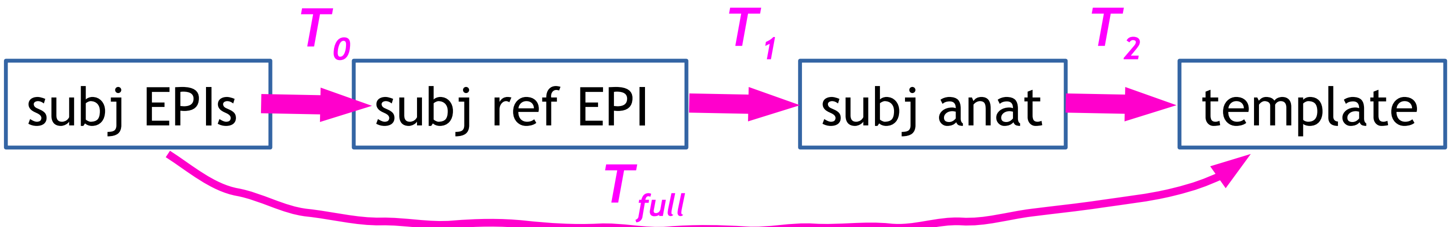
subj EPIs $\xrightarrow{T_0}$ subj ref EPI $\xrightarrow{T_1}$ subj anat $\xrightarrow{T_2}$ template

T_{full}

- but ***DON'T*** apply T_0 to make a new dset, and then apply T_1 to that, etc.
- ***DO*** concatenate T_0, T_1, and T_2 to make T_{full}, and then apply T_{full} to the EPIs to send the data to the template space in a single step (i.e., only one regridding)

Concatenating transforms

- Note: as observed earlier, any alignment/warping/interpolation/resampling leads to ***smoothing*** of data.
- Therefore, when mapping data through several spaces (EPI → anat → template), we **DON'T** want to regrid at each step; we ***concatenate** the transforms into a single one, and then have to only regrid once!*
- So, calculate individual transforms, such as within EPI (e.g., 3dvolreg), EPI-anat (e.g., align_epi_anat.py) and anat-template (e.g. sswarper2):



T_0 T_1 T_2

subj EPIs

subj ref EPI

subj anat

template

T_{full}

- but ***DON'T*** apply T_0 to make a new dset, and then apply T_1 to that, etc.
- ***DO*** concatenate T_0, T_1, and T_2 to make T_{full}, and then apply T_{full} to the EPIs to send the data to the template space in a single step (i.e., only one regridding)
- sidenote: probably don't upsample the EPI data a lot, e.g., from 3.5mm to 2mm iso
  - cannot "create" higher resolution information, no matter how it *looks*
  - just makes larger dsets, more processing time, more disk space, etc.

Some warping guidelines

Warping guidelines

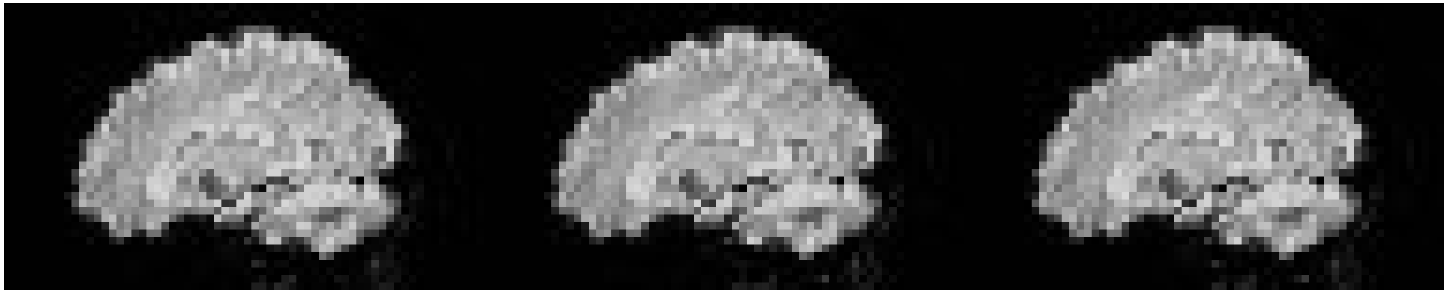
- For most EPI-EPI alignment, **ls** cost function (via 3dvolreg) is fine
  - for blip up-down EPI, there are nonlinear warping opts (**lpa** cost)
  - always check subj motion profiles and censoring
- For all other warping, look to **lpa** (similar contrast) and **lpc** or **lpc+ZZ** (differing contrasts)
- Use nonlinear warping when aligning anat data between different subjects (including subject to reference)
  - consider **sswarper2** for combined nonlinear warping + skullstripping
  - can input results directly into afni_proc.py
  - lots of templates are fine, but use one with good detail as base
- Calculate all warps in a chain individually, then *concatenate* them into one before applying, to reduce smoothing (less regridding steps)
- Use afni_proc.py when processing FMRI data
  - concatenation and other tiny details are managed *automatically*

Considerations for:
within EPI (EPI-EPI) alignment

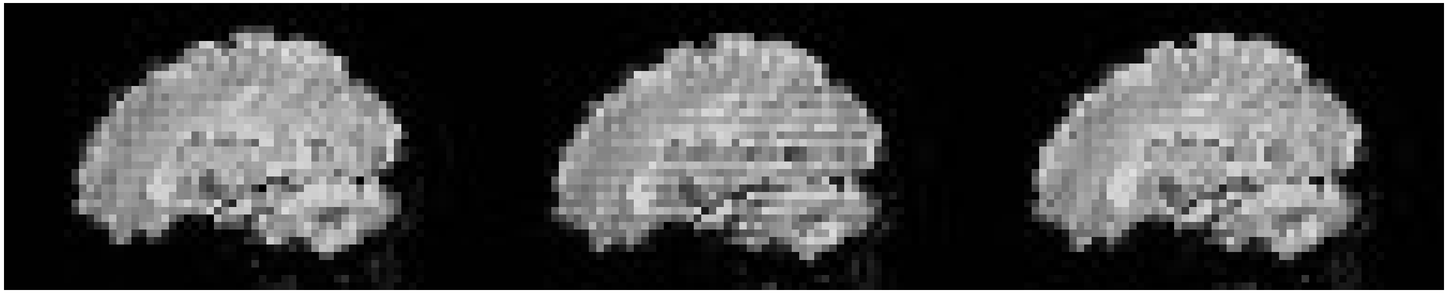
EPI time series: motion

EPI vols: standard, interleave acquisition

without motion

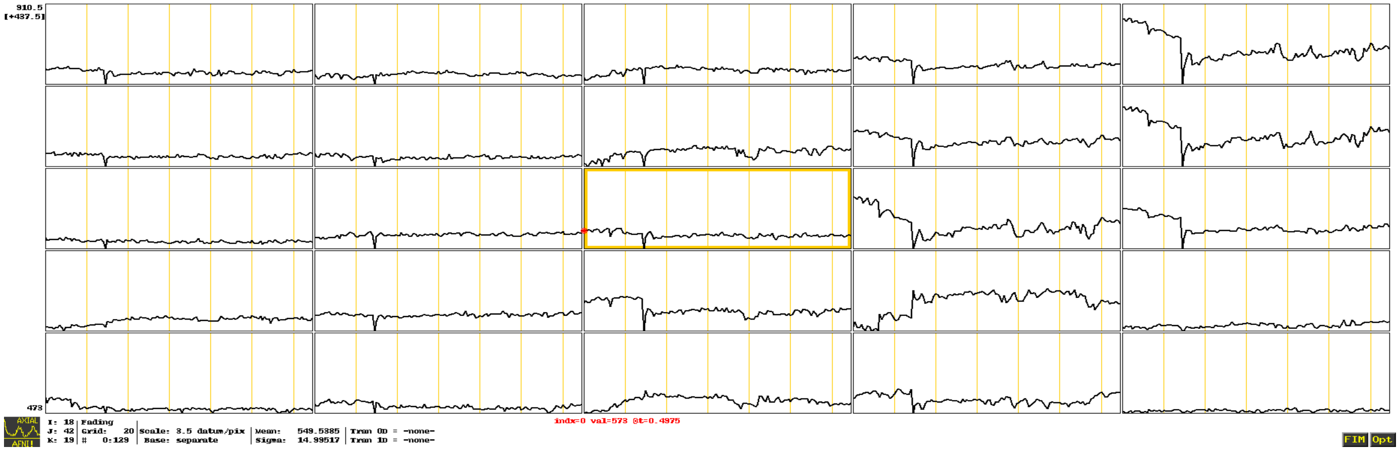


with motion



EPI time series

+ see the motion effect(s)?
+ see differences between voxels?



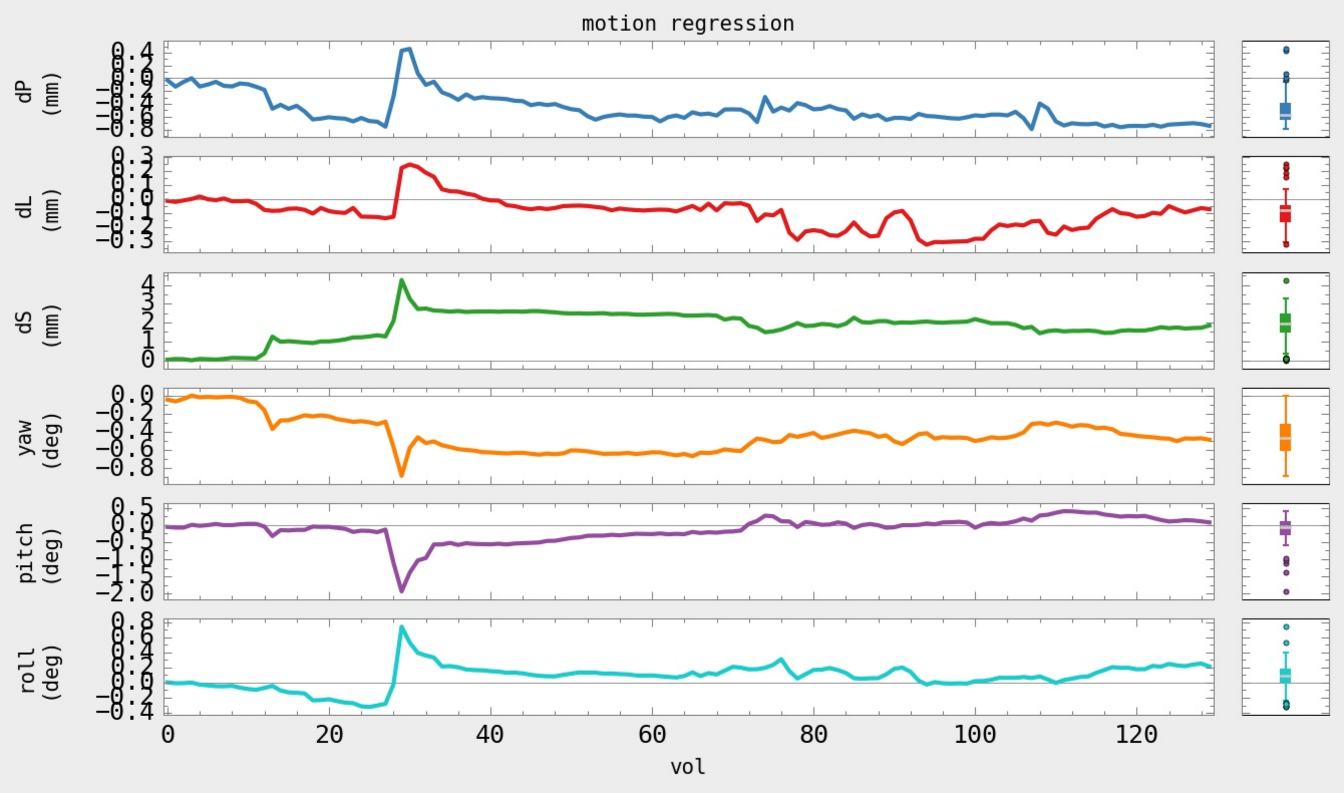
I: 18 | Fading | Scale: 3.5 datum/pix | Mean: 549.5385 | Tran 0D = -none-
J: 42 | Grid: 20 | Base: separate | Sigma: 14.99517 | Tran 1D = -none-
K: 19 | # 0:129

indx=0 val=573 @t=0.4975

EPI time series: motion

Main tool: **3dvolreg** (often via **afni_proc.py**). Use EPI-EPI alignment to:

1) estimate motion "time series" (rigid body params: rot + transl) and
2) use these "time series" as regressors and as censoring criteria



"with motion" subject from above

```
PLOT: 1dplot.py -sepscl -boxplot_on -reverse_order -ylabels VOLREG -xlabel vol \
        -title "motion regression" -prefix OUTPUT_NAME -infiles MOT_EST_VR.1D
```

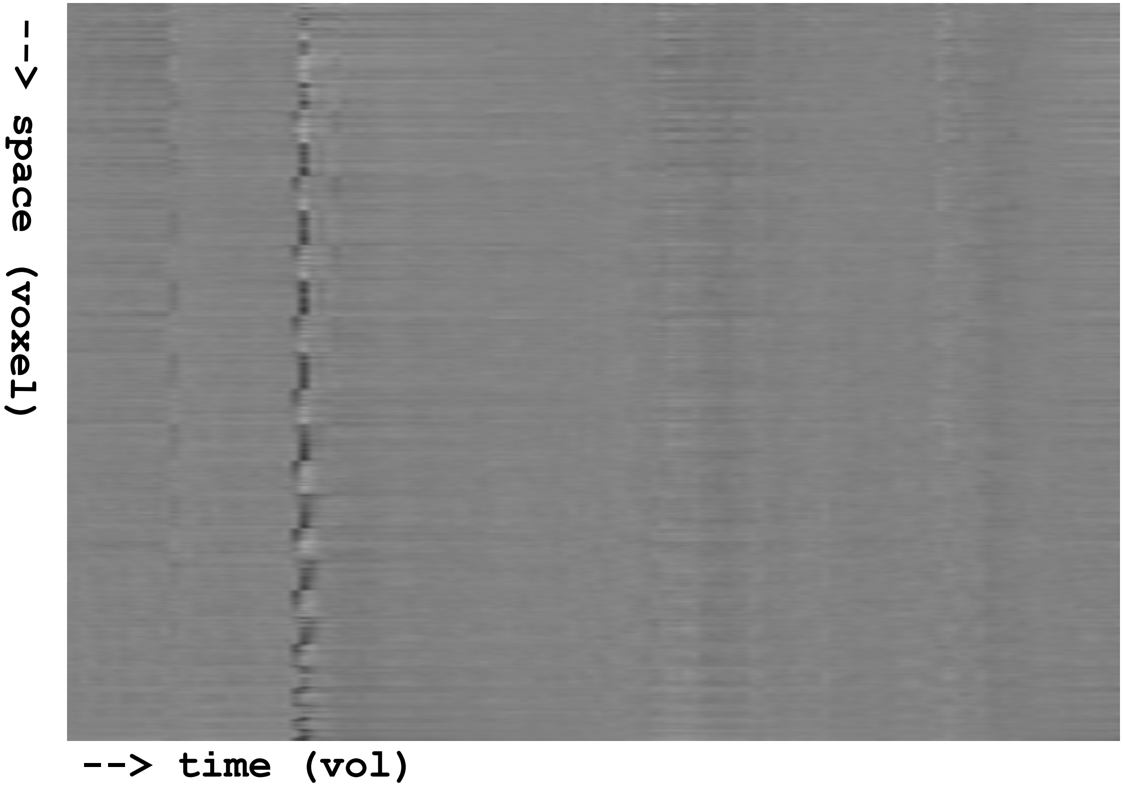
EPI time series: motion

Can visualize motion’s effects in time series as a (normalized) “**grayplot**”:

+ each row is one time series (values translated to grayscale)

+ each col is one instant/volume in time

Probably large outlier fractions where motion occurs (→ *censoring*)



“with motion” subject from above →

```
PLOT: 3dGrayplot -mask mask+orig. -input FILE_TIMESERIES -prefix OUTPUT_NAME
```

EPI time series: censoring

Typically two measures to detect and censor motion

- **enorm** (Euclidean norm): based on 3dvolreg's motion parameter estimates
  - suprathreshold enorm leads to censoring multiple vols

For each [i]th volume, use the derivatives of motion profiles to calc enorm:

$$\mathrm{enorm}_i = \left(\Delta dP_i^2 + \Delta dL_i^2 + \Delta dS_i^2 + \Delta yaw_i^2 + \Delta pitch_i^2 + \Delta roll_i^2\right)^{0.5}$$

… which is to say, where there are bigger changes in motion due to either translation, rotation or both, the **enorm** value is larger.

- Enorm has units of ~mm (for human head radius, approx. 1deg ~1mm).

  Can threshold with a preset value, e.g., 0.2 (~mm); since enorm is based on a derivative (difference), ***two*** volumes get censored for suprathreshold values.

  Ex. calculation from output of 3dvolreg:

```
1d_tool.py                                          \
    -infile VOLREG_OUTPUT.1D -set_nruns 1           \
    -derivative  -collapse_cols euclidean_norm      \
    -write ENORM_OUTPUT
```

EPI time series: censoring

Typically two measures to detect and censor motion

- **enorm** (Euclidean norm): based on 3dvolreg's motion parameter estimates
  - suprathreshold enorm leads to censoring multiple vols
- **outliers**: based on outliers from time series trends (total fraction in volume)

For each [i]th volume, count how many voxels have an outlier; record fraction:

$\texttt{outlier\_frac}_i = (\texttt{\# vox with outlier})_i \,/\, (\texttt{\# vox in automask})$

- Outlier frac has no units; just has a range [0, 1].

  Can threshold with a preset value, e.g., 0.05. Only one volume gets censored for supratheshold values.

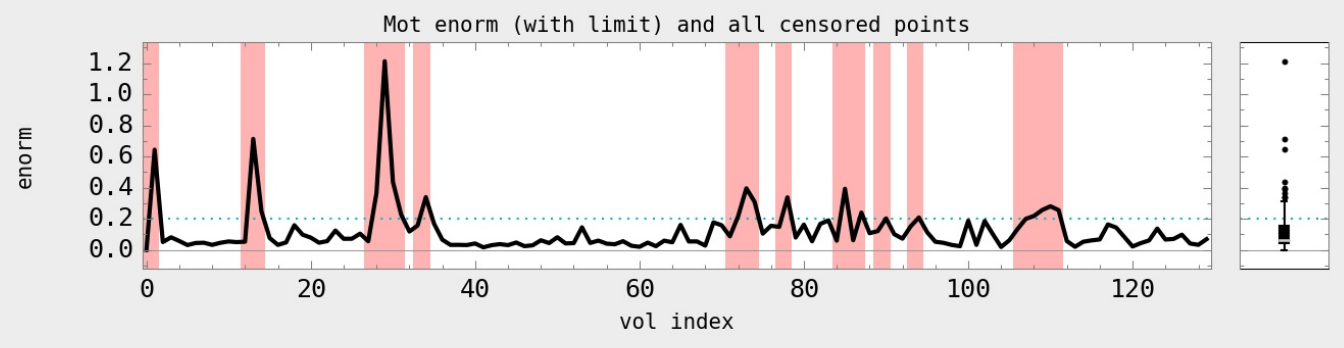
*Ex. calculation from **unprocessed** data set:*

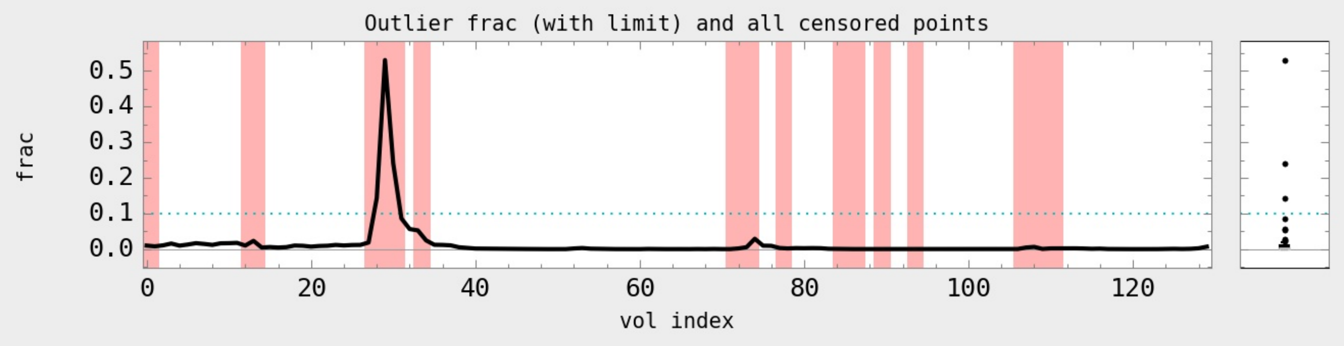
```
3dToutcount                                        \
    -automask -fraction -polort 3 -legendre        \
    DSET > OUTLIER_FRAC.1D
```

EPI time series: censoring

Typically two measures to detect and censor motion

- **enorm** (Euclidean norm): based on 3dvolreg's motion parameter estimates
  - suprathreshold enorm leads to censoring multiple vols
- **outliers**: based on outliers from time series trends (total fraction in volume)





```
PLOT (for motion-enorm, and similar for outlier frac):
  1dplot.py -boxplot_on -reverse_order -infiles MOT_ENORM.1D -censor_hline VAL    \
      -censor_files CEN_FILE.1D  -xlabel "vol index" -ylabels "enorm"             \
      -title "Mot enorm (with limit) and all censored points" -prefix OUT_MOT.jpg
```

Example: 3dvolreg

Can run the following in "AFNI_data6/afni/":

```
3dvolreg -base 3 -cubic -zpad 1          \
         -1Dfile dfile_vr.1D             \
         -1Dmatrix_save mat_vr.aff12.1D  \
         -prefix epi_r1_vrt              \
         epi_r1+orig
```

- epi_r1+orig: (last entry) input time series to align
- -base 3: select brick [3] of input epi_r1+orig dset as reference vol

  (NB: in afni_proc.py, one can select the ref vol to be the one with fewest outliers, with keyword: MIN_OUTLIERS)
- -cubic: use cubic polynomial interpolation
- -zpad 1: put (intermed) layer of zeros around base vol, helps if large rotations exist
- -1Dfile ...: save motion estimates as columns of numbers in text (1D) file
- -1Dmatrix_save ...: save motion params (12 DOF in matrix form) to text (1D) file
- -prefix ...: save motion-corrected vol to new 4D volumetric dset

... and a quick view of motion estimates:

```
1dplot -volreg dfile_vr.1D &
```

Example: 3dvolreg (cont’d)

... and a quick calculation of ‘enorm’ (= Euclidean norm) and possible censoring (see “1d_tool.py -help” for description of opts here):

```
1d_tool.py -infile dfile_vr.1D      \
           -set_nruns 1             \
           -show_censor_count       \
           -censor_prev_TR          \
           -censor_motion 0.3 SUBJ
```

... with a quick view of *those* results (with censorline fanciness):

```
1dplot -one SUBJ_enorm.1D "1D: 152@0.3" &
```

and to see the (command line-usable) string of indices to be censored:

```
cat SUBJ_CENSORTR.txt
```

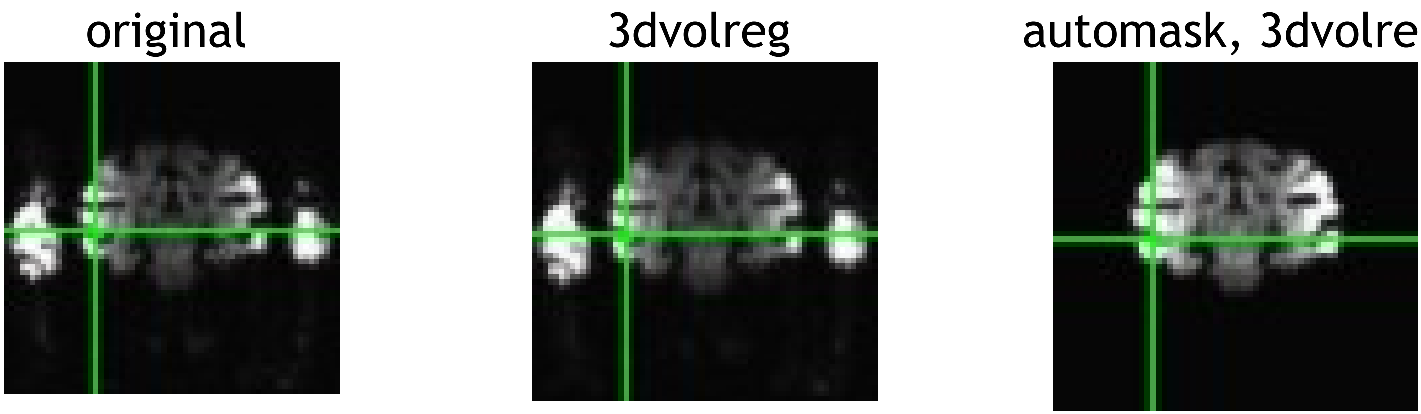
or pro-level fanciness combining the previous two:

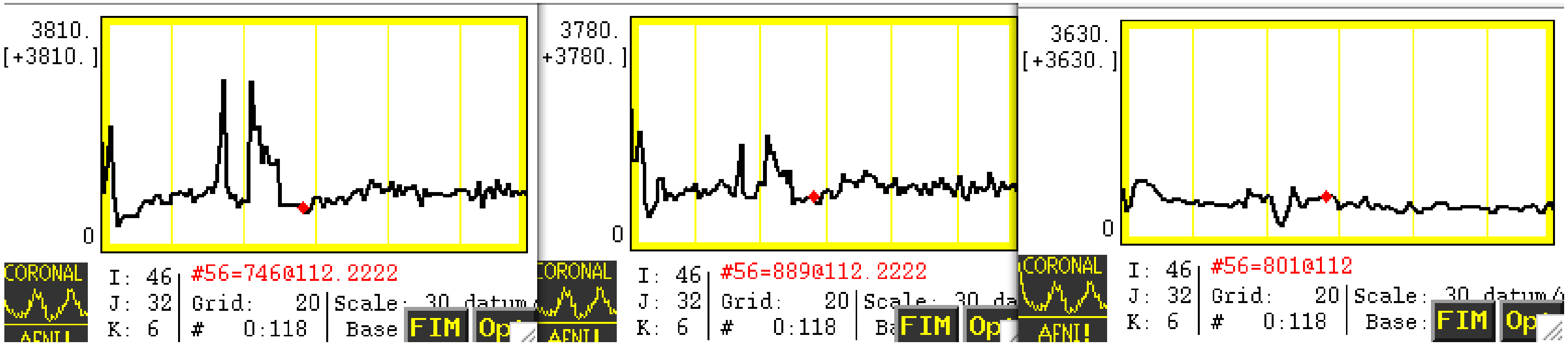
```
1dplot -one `cat SUBJ_CENSORTR.txt` SUBJ_enorm.1D "1D: 152@0.3" &
```

Motion correction: caveats

- Alignment can***not*** fix motion; use motion pars as regressors (helps more, not total).
- Motion correction → volume alignment → regridding → smoothing.
- Check in the AFNI GUI to be sure the “corrected” data is not “bouncing” around
- **Example:** Monkey sips juice at stimulus time, and large jaw muscles move. If the muscles are not masked, then motion correction may track muscles, not brain.

[movies→]




3810.
[+3810.]
0
CORONAL
I: 46 #56=746@112.2222
J: 32 Grid: 20 Scale: 30 datum
K: 6 # 0:118 Base

3780.
+3780.]
0
CORONAL
I: 46 #56=889@112.2222
J: 32 Grid: 20 Scale: 30 da
K: 6 # 0:118 B

3630.
[+3630.]
0
CORONAL
I: 46 #56=801@112
J: 32 Grid: 20 Scale: 30 datum
K: 6 # 0:118 Base:

“Fixing” Motion

- Motion occurs over slices and not volumes and moves data off original grid
- “Regressing out” – motion parameters, derivatives, displacement, euclidean norm of derivatives (summary parameters)
- Censoring (“scrubbing”, “sweeping under the mat”,...)
- Experimental design – kids, monkeys, juice, talking, waving hands wildly....
- Interpreting results – differences in motion between groups or something physiological
- Be on the lookout – activation following high contrast borders, bright/dark patterns in sagittal views ("blinds" effects)

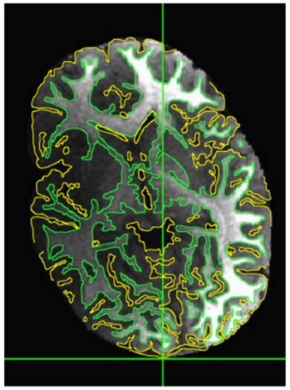
EPI distortion correction in EPI data (1/2)

- Data acquired with an EPI sequence has distortions due to B0 inhomogeneity
- One way to correct: acquire data with opposite phase encoding (PE) to help “undo” the warping (AKA, “blip up, blip down” acquisitions)
- 1) Align opposite PE dsets nonlinearly. 2) Apply “half” warp to est. undistorted dset.

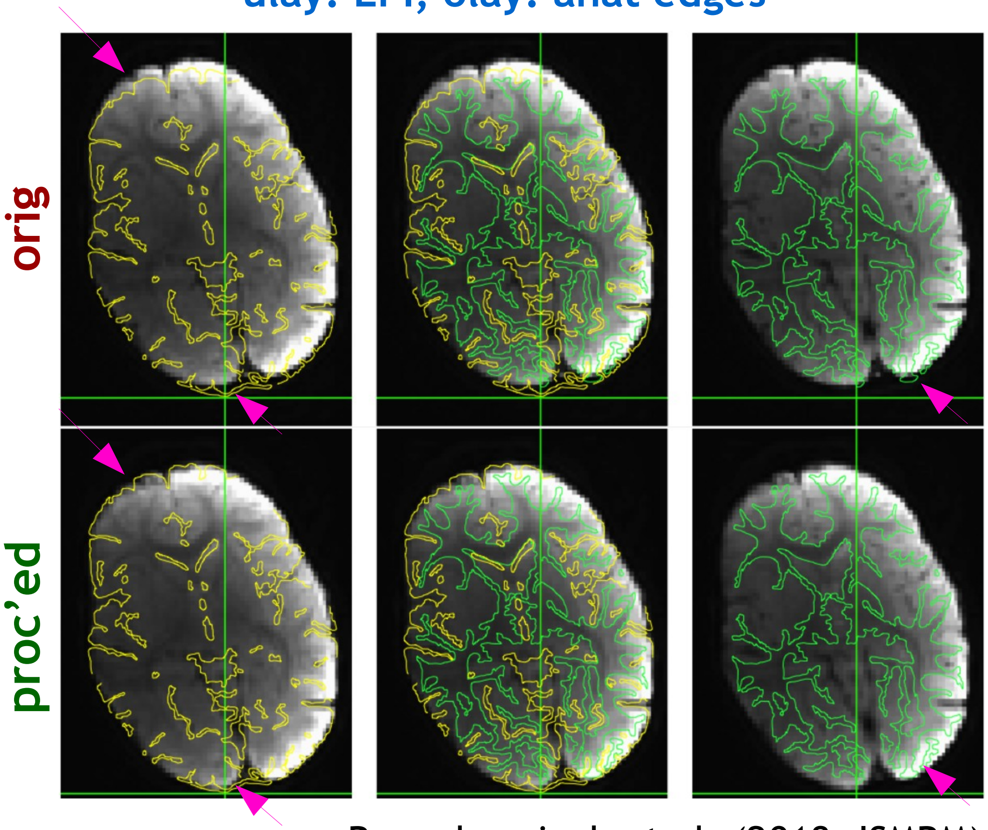
Example

+ EPI dset from toddler study
+ Unwarping calculated with *3dQwarp+3dNwarpApply* in *afni_proc.py*.
+ [Watch this space for more scripts/tools]

anat



ulay: EPI, olay: anat edges



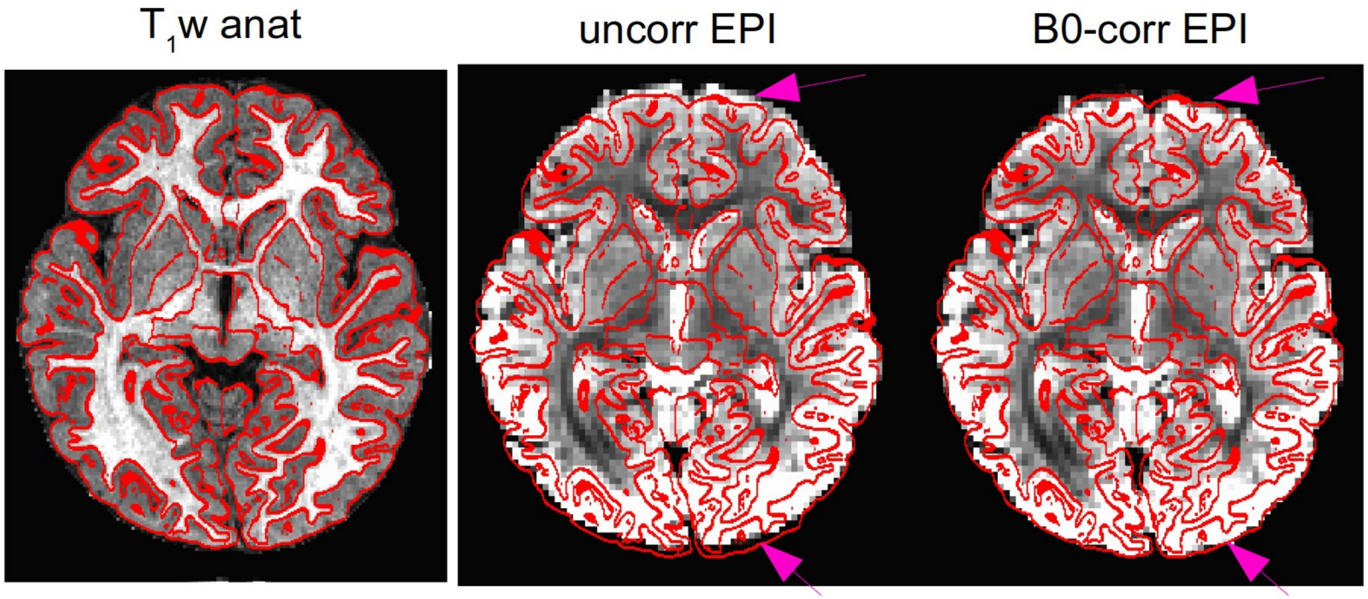
Roopchansingh et al. (2018, ISMRM)

EPI distortion correction in EPI data (2/2)

- New tool in AFNI for B0 (phase dset) correction: `epi_b0_correct.py`
  - Also put together by Vinai Roopchansingh, uses 3dNwarpApply
  - can be integrated in afni_proc.py

Example

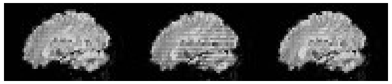
+ EPI dset from toddler study (Roopchansingh et al., in prep)
+ compare outlines of anatomical tissues (red)
  with EPI structures before and after correction

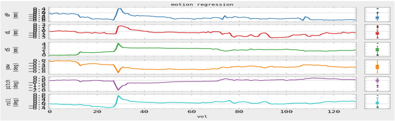


EPI alignment: general comments

Comments on motion “correction” (= reduction) in EPI

- One can***not*** fully “correct” effects of motion
  - intravol motion *can’t* be corrected, but bad cases will likely be **censored** (e.g., via outlier count, enorm vals)
  - motion parameter estimation+regression is useful, but won’t totally account for motion variance





- Motion is probably a bigger problem in resting state (and naturalistic) FMRI than in task-based, because there aren’t beta weights to focus on
- In general, global signal regression is ***not*** recommended (Saad et al., 2012; Gotts et al. 2013; Murphy et al. 2009)
- **GCOR** parameter might be one way to include motion information on group level, esp. for rs-fMRI (Saad et al. 2013; Gotts et al. 3013)
- Dealing with motion requires planning ***before*** scanning (make subjects comfortable, practice scanner, study design)
- Work must always be done to show that motion differences between groups is not driving effects

Considerations for:
EPI-anatomical alignment

EPI-anatomical alignment

Main tool: **align_epi_anat.py** (often via **afni_proc.py**), and **3dAllineate**.
+ align EPI to anatomical of same subject
+ use **linear affine** alignment (assume no major/nonlinear distortions)

align_epi_anat.py
+ Default “lpc” or “lpc+ZZ” cost function, to align vols with opposite contrast (e.g., T1w and EPI)
+ Can combine alignment with other transformations (as a single transform): deobliquing, motion correction, alignment and talairach transforms
+ Can also perform slice timing correction and apply transformations to “child” datasets.
+ Can perform checks on left-right consistency between datasets (e.g. check for bad header info---surprisingly common problem!)

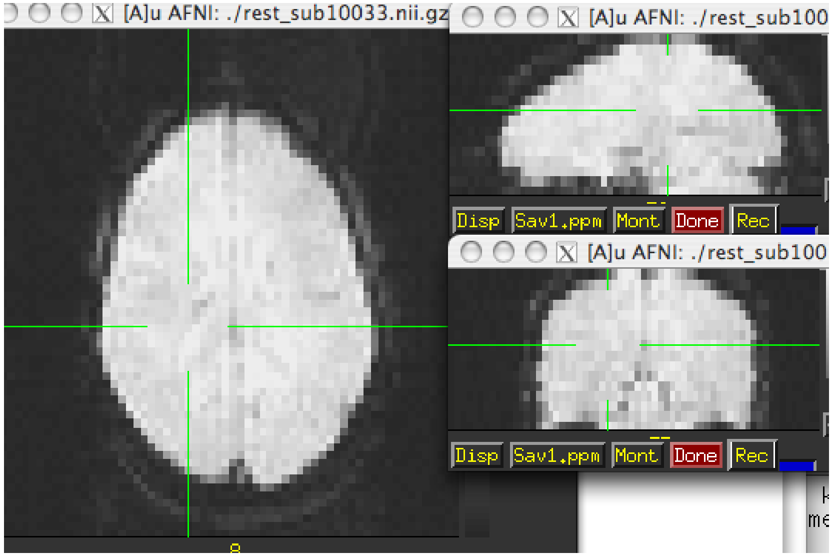
Alignment strategies with align_epi_anat.py

Defaults work for a large fraction of cases (>90% - FCON1000)

For problematic data → some solutions to try:

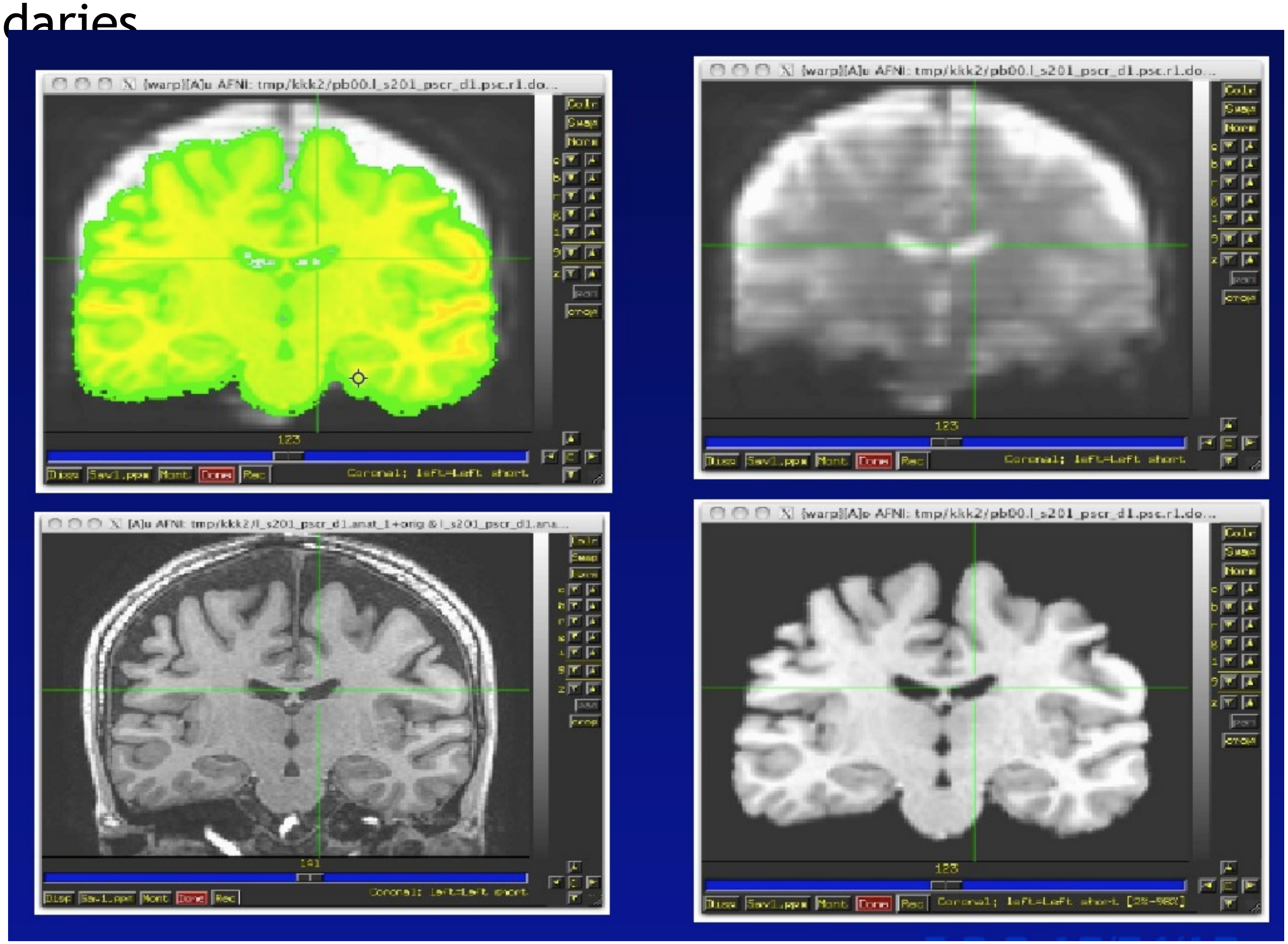
- Far off start → “-big_move”, “-giant_move”,”-ginormous_move”
- Poor contrast → “-cost lpa”, “-cost nmi”, “-cost lpc+ZZ”
- Poor non-uniformity → “-edge”, “-cost lpa”

For non-standard dsets: stroke/MS lesions, tumors, monkeys, rats, multi-modality (CT/PET/DTI/...), something else? → see us, post on Message Board

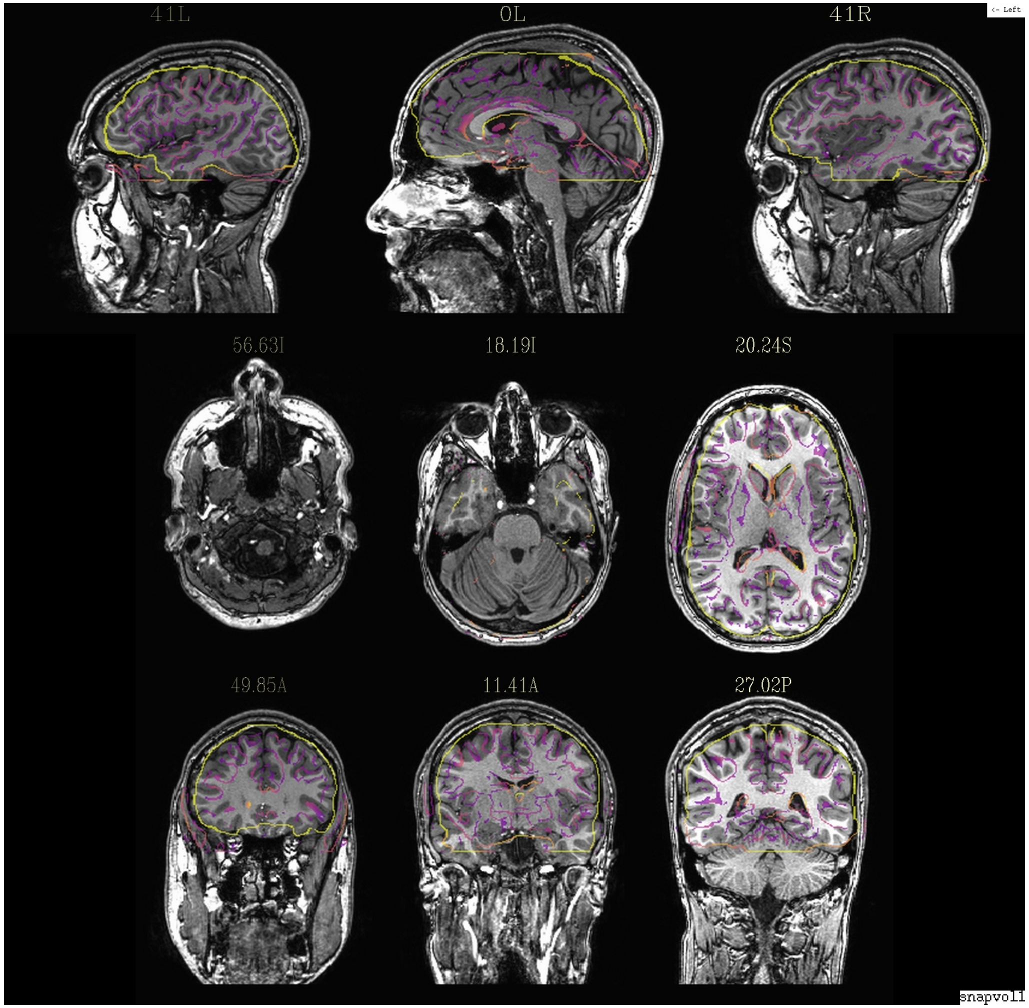


Checking alignment: what to look for

It’s what’s on the ***inside*** that counts!
→ match sulcal gyral patterns, ventricle location, tissue boundaries



Checking alignment: edge-y view



```
CMD : @snapshot_volreg VOL_ULAY VOL_OLAY output.jpg
view: aiv output.jpg
```

Alignment visualization in AFNI

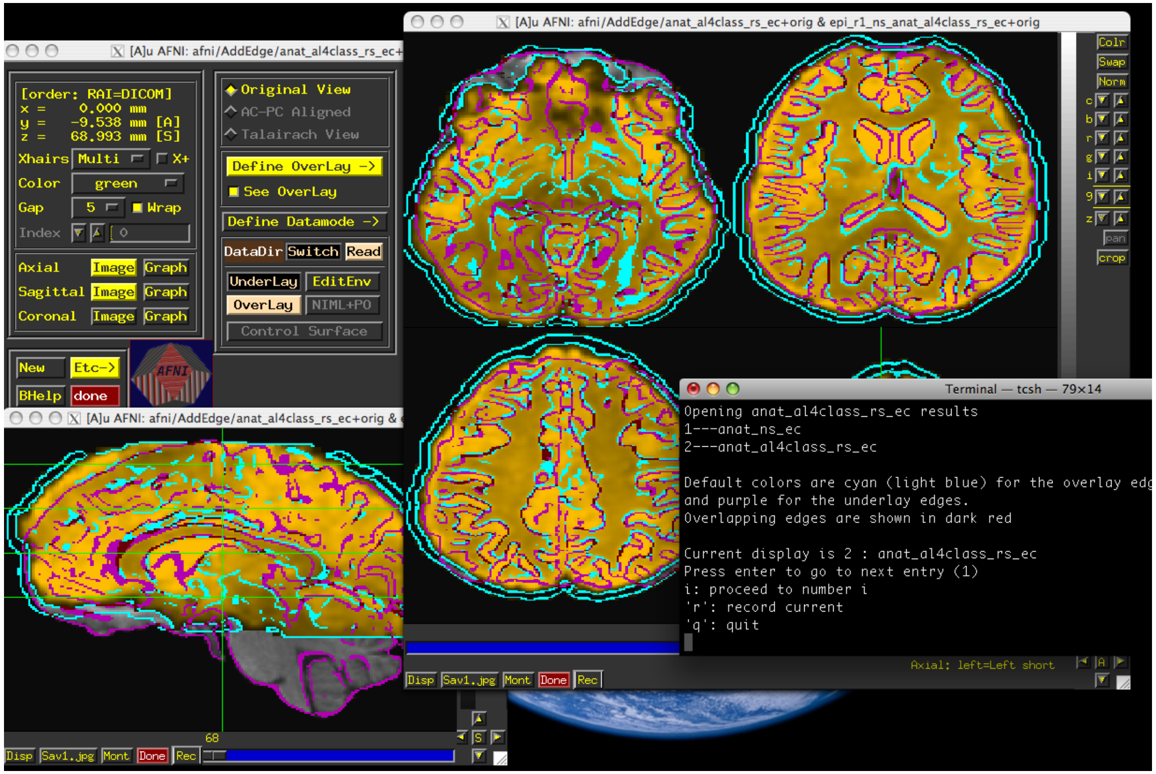
- **Graph and image** – travel through time for motion correction or for a thousand datasets in a row.
- **Multiple controllers and crosshairs** – up to ten datasets at a time, quick and rough.
- **Overlay display** – opacity control, thresholding. A single pair – good for different or similar datasets.
- **Overlay toggle, Underlay toggle** – wiggle, good but a little tricky ('o' and 'u' keys in image viewer)
- **Sliding Overlay** ('4'/'5' keys), **fade-in overlay** ('6' key), c**heckerboard underlay** – ('#' key) two similar datasets in underlay but must be virtually identical. Good for comparing two processing methods
- **Edge display for underlay** – effective pairwise comparison for quick fine structure display and comparison with overlay dataset with opacity. One dataset should have reliable structure and contrast. Now with 'e' toggle.
- **@AddEdge** – single or dual edges with good contrast for pairwise comparison.

Affine alignment: auto-QC

Can run **align_epi_anat.py** with assistance for QCing alignment

+ use “-AddEdge” option to view edge-highlights of dsets in AFNI GUI

AFNI GUI with “@AddEdge” display settings



```
Opening anat_al4class_rs_ec results
1---anat_ns_ec
2---anat_al4class_rs_ec

Default colors are cyan (light blue) for the overlay edg
and purple for the underlay edges.
Overlapping edges are shown in dark red

Current display is 2 : anat_al4class_rs_ec
Press enter to go to next entry (1)
i: proceed to number i
'r': record current
'q': quit
```

Affine alignment: auto-QC

Can run **align_epi_anat.py** with assistance for QCing alignment
+ use “-AddEdge” option to view edge-highlights of dsets in AFNI GUI

Example to run in AFNI_data6/afni/ (NB: takes a few mins in total):

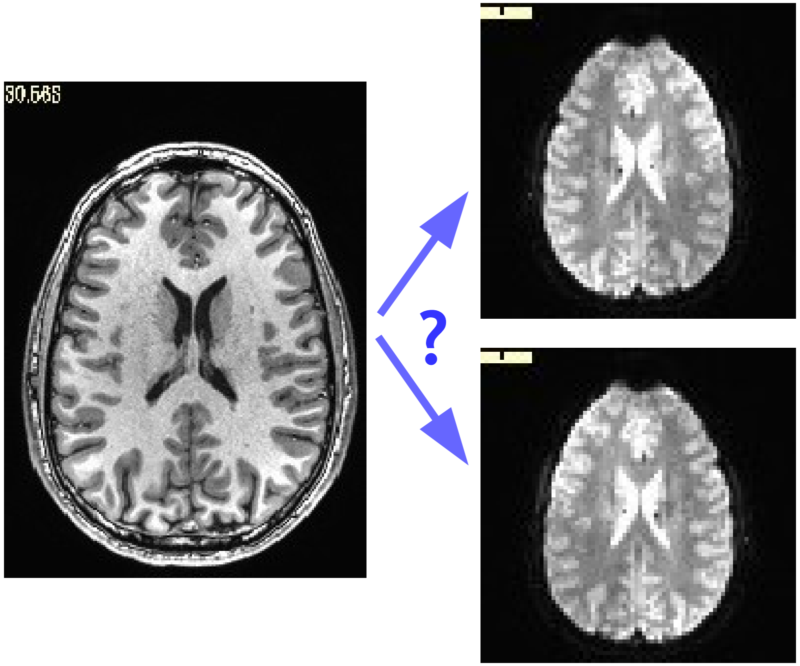
```
align_epi_anat.py                 \
   -anat anat+orig                \
   -epi  epi_r1+orig              \
   -epi_base 0                    \
   -suffix _adde                  \
   -AddEdge

cd AddEdge
@AddEdge
```

Using alignment to check for left-right flipping

Check for dset header problems (esp. conversion software issues):

```
$ align_epi_anat.py -check_flip ...
$ afni_proc.py -check_flip ...
```

→ compare EPI<->anatomical alignment cost results for both flipped and unflipped dsets.

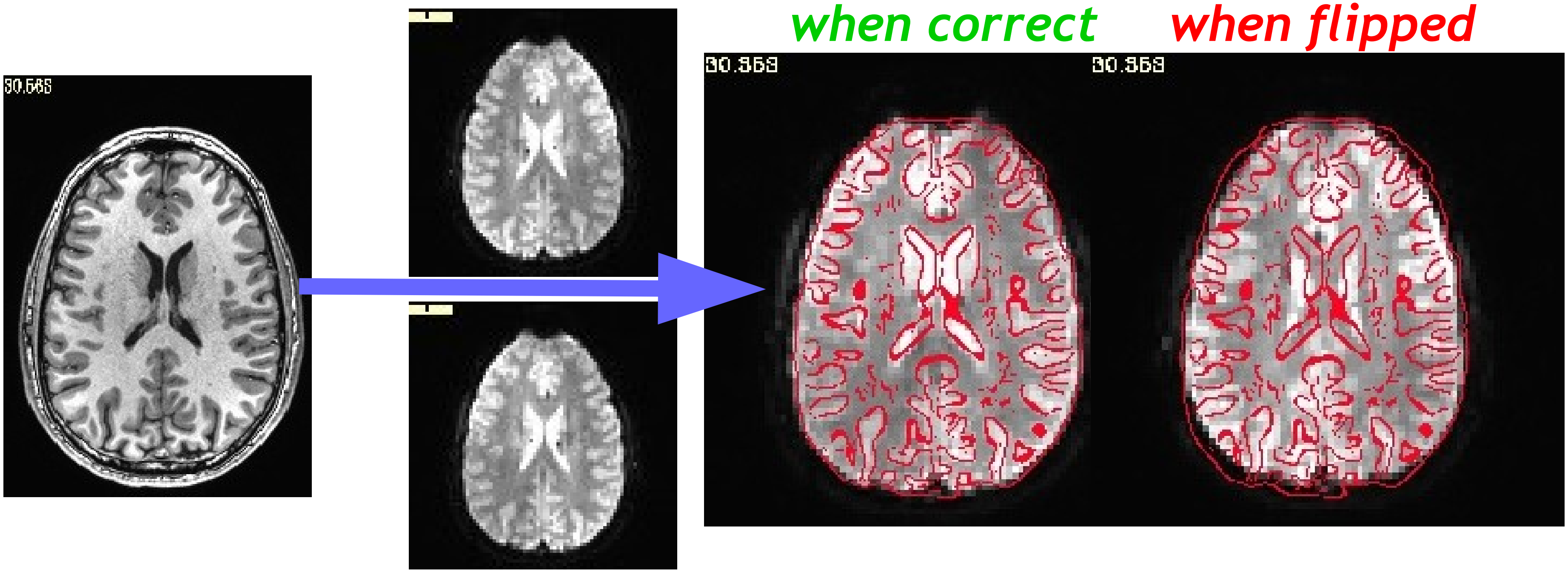


Using alignment to check for left-right flipping

Check for dset header problems (esp. conversion software issues):

```
$ align_epi_anat.py -check_flip ...
$ afni_proc.py -check_flip ...
```

→ compare EPI<->anatomical alignment cost results for both flipped and unflipped dsets.



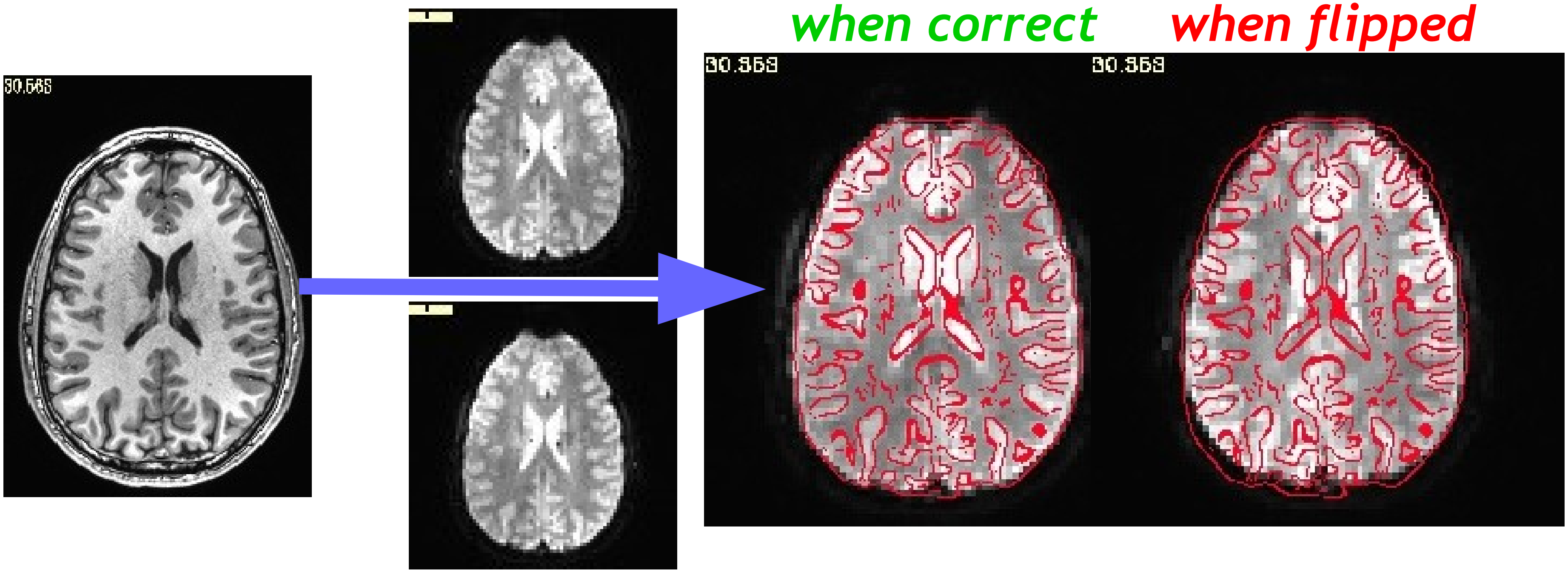
Glen et al. (2018, OHBM).

Using alignment to check for left-right flipping

Check for dset header problems (esp. conversion software issues):

```
$ align_epi_anat.py -check_flip ...
$ afni_proc.py -check_flip ...
```

→ compare EPI<->anatomical alignment cost results for both flipped and unflipped dsets.



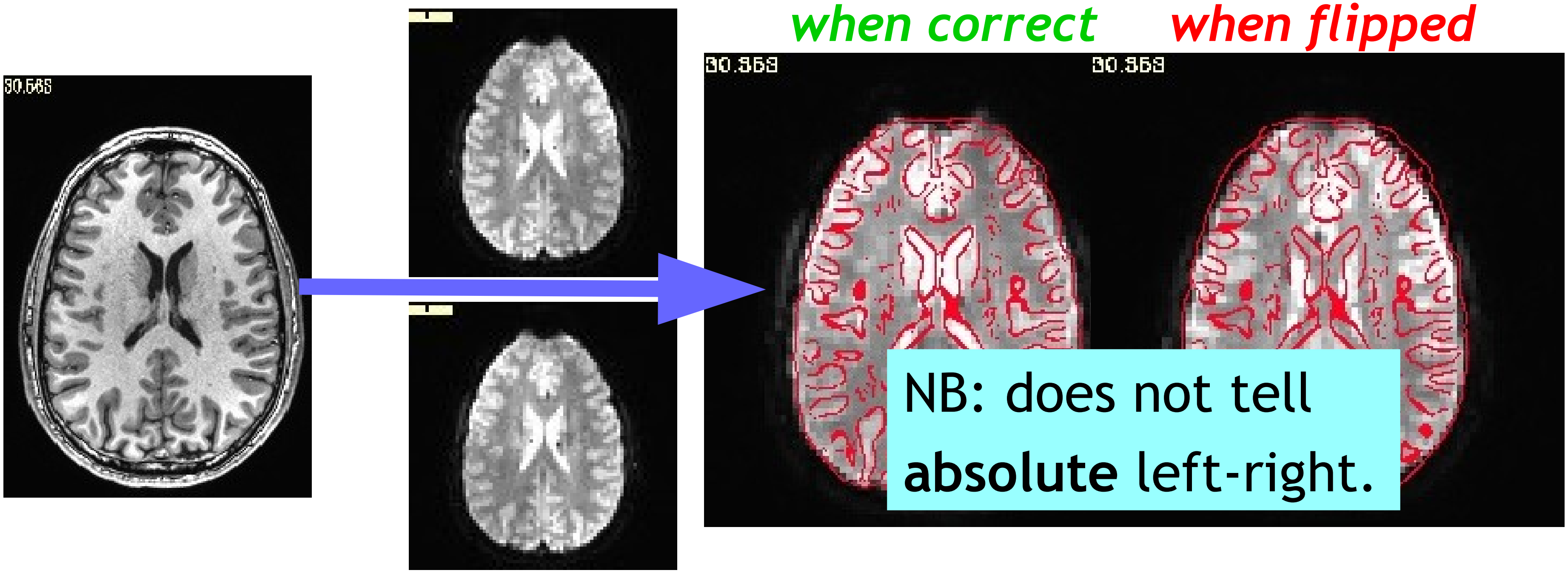
(Has found systematic LR-flips in public FCP and OpenFMRI data sets.)

Using alignment to check for left-right flipping

Check for dset header problems (esp. conversion software issues):

```
$ align_epi_anat.py -check_flip ...
$ afni_proc.py -check_flip ...
```

→ compare EPI<->anatomical alignment cost results for both flipped and unflipped dsets.



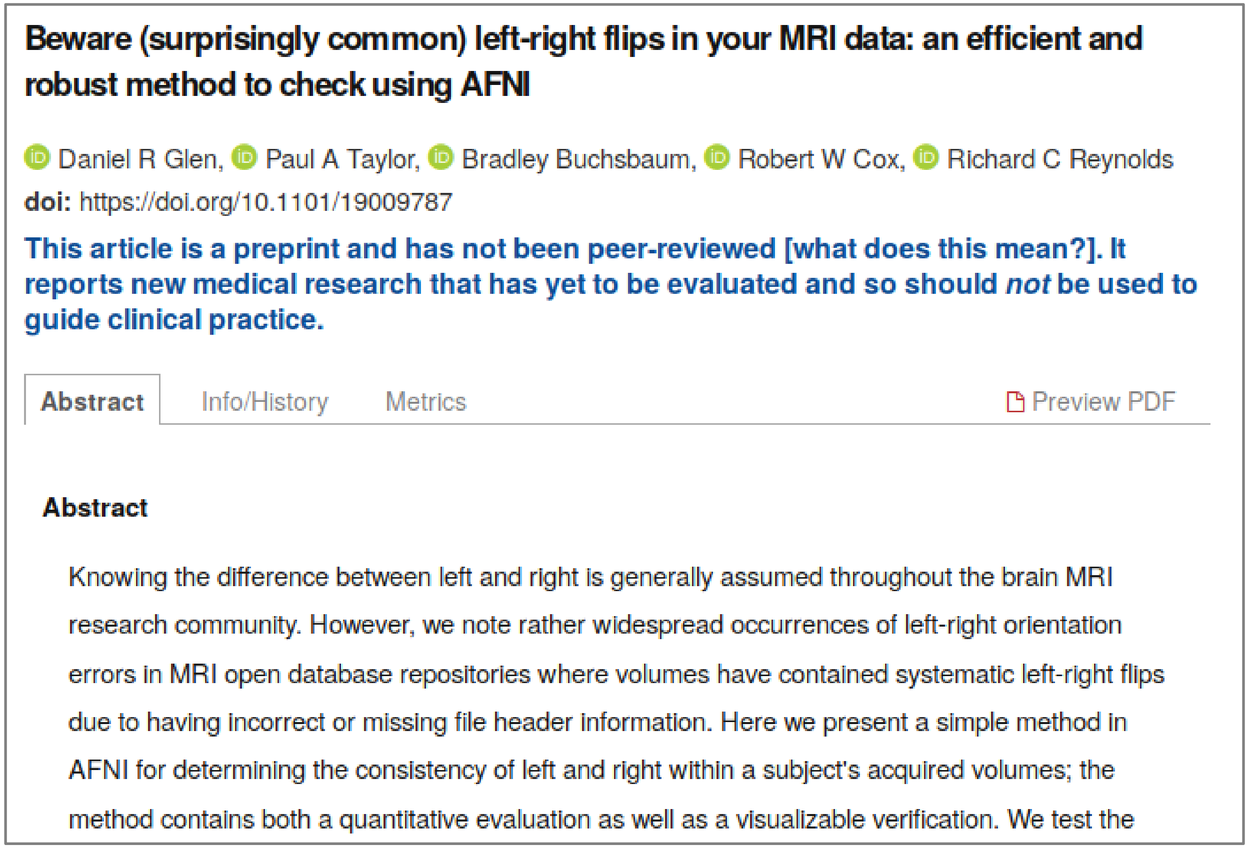
when correct *when flipped*

NB: does not tell **absolute** left-right.

(Has found systematic LR-flips in public FCP and OpenFMRI data sets.)

Using alignment to check for left-right flipping

Read more about it here:
https://www.medrxiv.org/content/10.1101/19009787v1



Beware (surprisingly common) left-right flips in your MRI data: an efficient and robust method to check using AFNI

Daniel R Glen, Paul A Taylor, Bradley Buchsbaum, Robert W Cox, Richard C Reynolds

doi: https://doi.org/10.1101/19009787

This article is a preprint and has not been peer-reviewed [what does this mean?]. It reports new medical research that has yet to be evaluated and so should *not* be used to guide clinical practice.

Abstract

Knowing the difference between left and right is generally assumed throughout the brain MRI research community. However, we note rather widespread occurrences of left-right orientation errors in MRI open database repositories where volumes have contained systematic left-right flips due to having incorrect or missing file header information. Here we present a simple method in AFNI for determining the consistency of left and right within a subject's acquired volumes; the method contains both a quantitative evaluation as well as a visualizable verification. We test the

Considerations for:
anatomical-template alignment

Registration to a template: nonlinear warps

- Main tools: **3dQwarp**, **sswarper2** (or **@SSwarper)** (skullstrip & align), **auto_warp.py**
  - for animal dsets: **@animal_warper** (skullstrip & align, too)
- In general, one should use a nonlinear warp for all registration between different subjects (if the resolution/detail is high enough)

- Nonlinear warping is slooow, but parallelized for speedup on multi-core machines
  - set an environment variable for specifying number of CPUs:
    - # for tcsh script or ~/.cshrc file:
      set OMP_NUM_THREADS = 7
    - # for bash script or ~/.bashrc file:
      export OMP_NUM_THREADS=7
  - check setting (yours or default) in terminal with “3dQwarp -hview”

Registration to a template: nonlinear warps

- Choose a template similar/relevant to your subject group
  - e.g., use a pediatric template for a study of children

- With nonlinear warping,
  - one wants a detailed template to latch onto
  - details of the source dset (like skullstripping) matter
    - e.g., small bits of skull left on can lead to odd warping

- (and see Bootcamp presentation on Templates and Atlases)

sswarper2 / @SSwarper: skull-strip and nonlinear warp

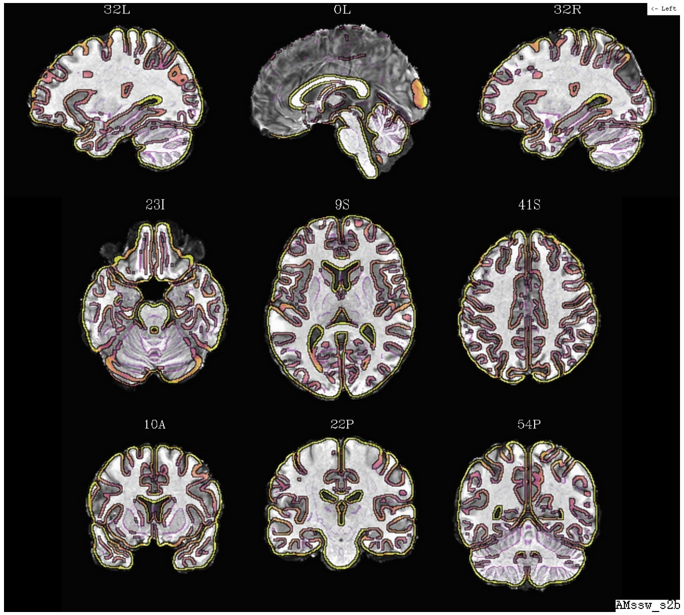
- sswarper2 (@SSwarper) does two jobs for the prices of one!

  1) Remove the skull of the subject’s anatomical in native space,

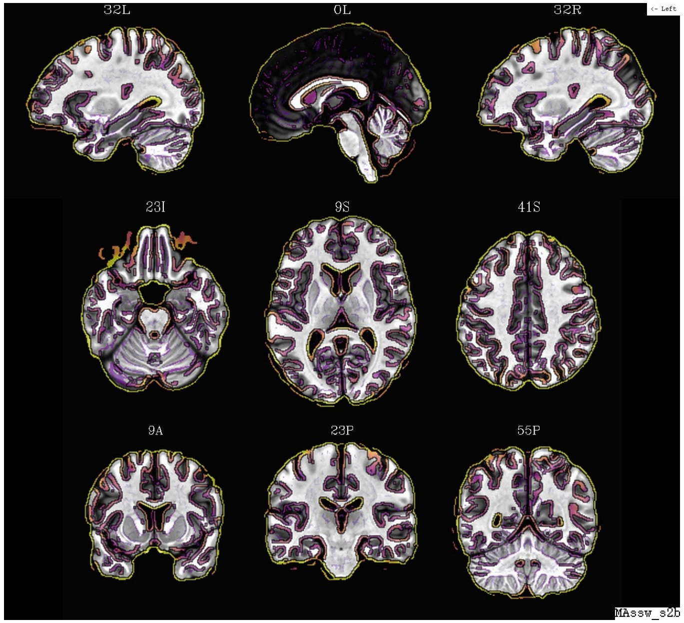
  2) Estimate the nonlinear warp to a standard space template

  Call now and receive a bonus set of QC images of the alignment:

ulay: anat, olay: template edges



ulay: template, olay: anat edges



- Presently, **sswarper2** is the recommended tool for skullstripping (and maybe/probably for nonlinear warping, as well)
- Results can be passed directly to afni_proc.py (see “sswarper2 -help”).

Measuring quality of alignment

Can compare 3dQwarp with other available nonlinear alignment tools

+ For a group of subjects, estimate warp from anat to template
+ Apply warp to labeled ROIs, and measure % overlap in results.

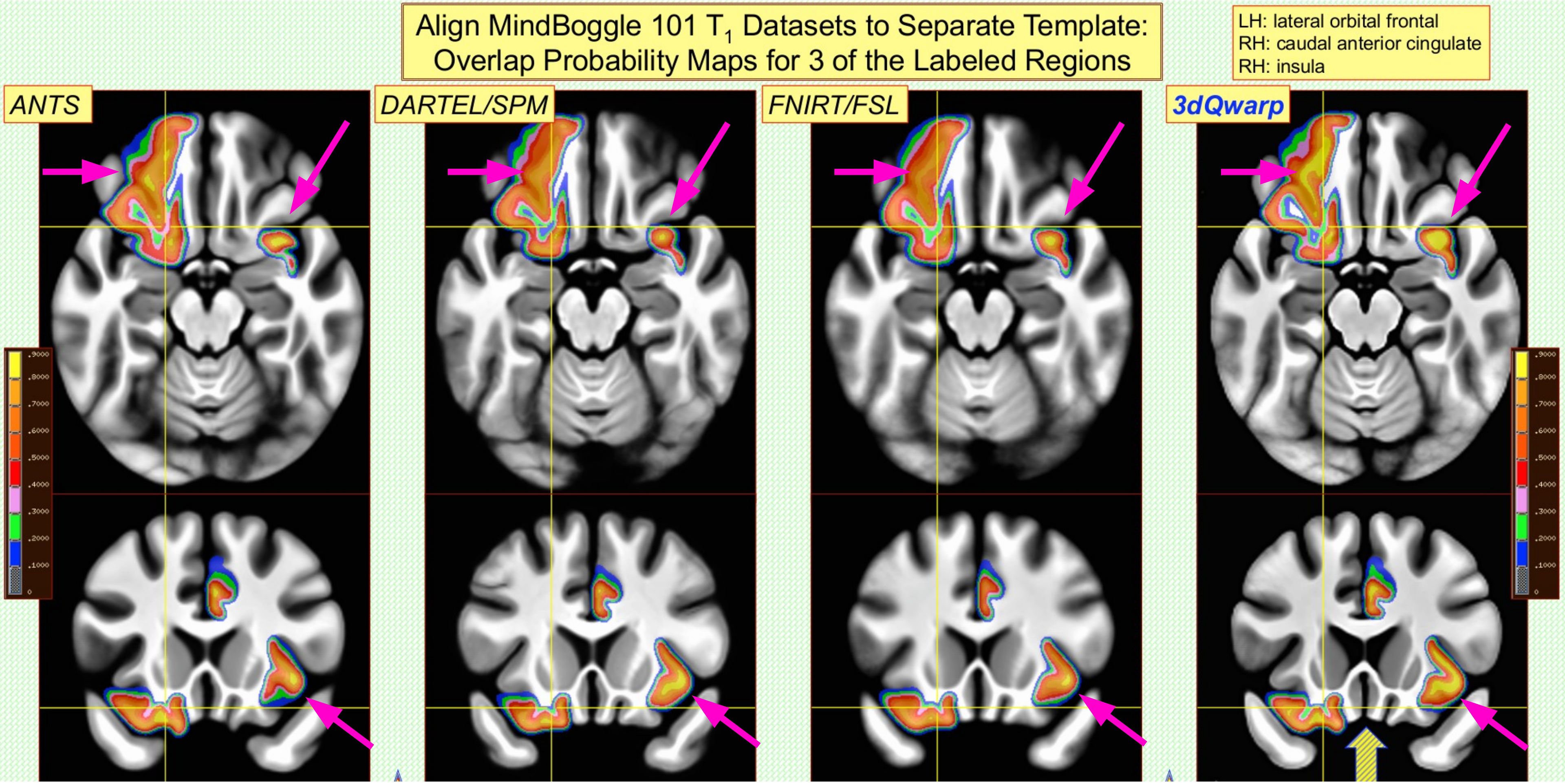
ANTS, DARTEL and FNIRT run with default settings

(Cox & Glen, 2013, OHBM)

Measuring quality of alignment

Can compare 3dQwarp with other available nonlinear alignment tools

+ For a group of subjects, estimate warp from anat to template
+ Apply warp to labeled ROIs, and measure % overlap in results. (Yellow: >90% overlap)



Align MindBoggle 101 T_1 Datasets to Separate Template:
Overlap Probability Maps for 3 of the Labeled Regions

LH: lateral orbital frontal
RH: caudal anterior cingulate
RH: insula

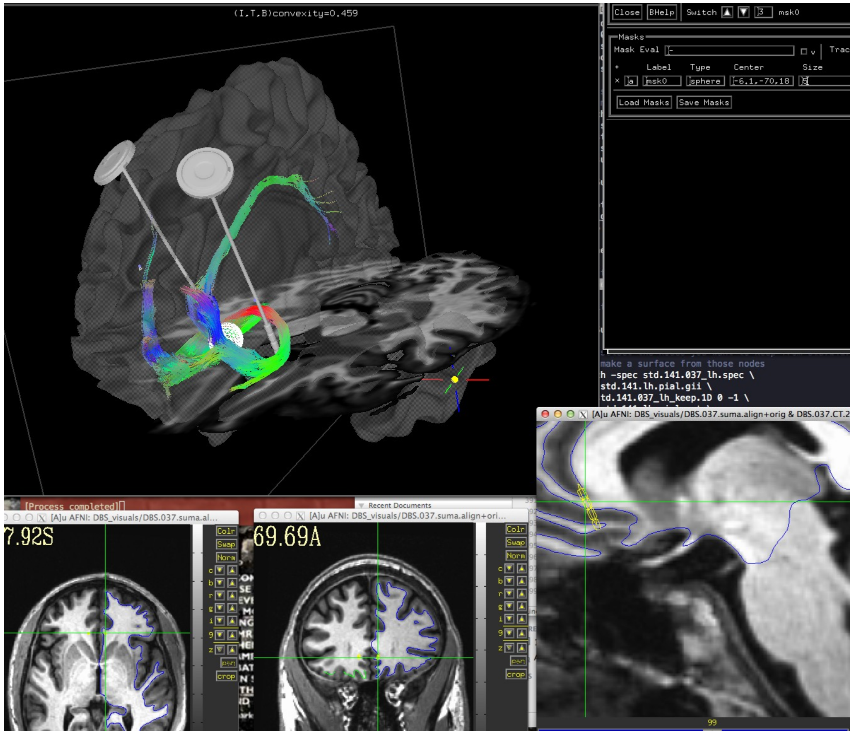
ANTS, DARTEL and FNIRT run with default settings

(Cox & Glen, 2013, OHBM)

other types of alignments
that are possible in AFNI

Alignment with non-MRI data

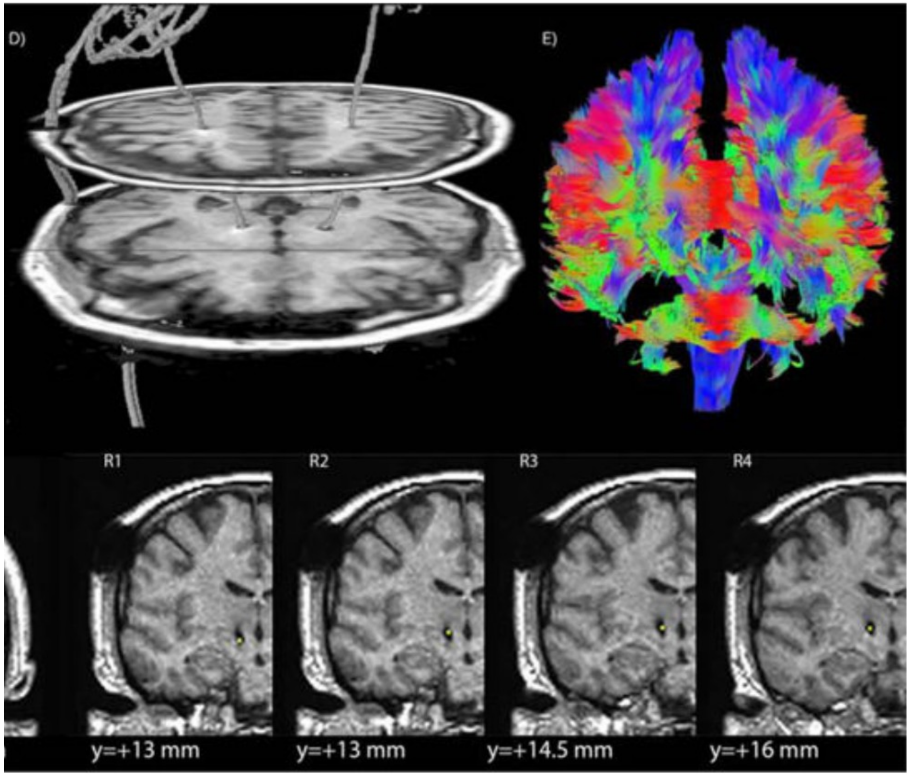
DBS – align CT with electrodes to pre-surgical MRI, PET (and sometimes DWI)



Masks
Mask Eval: -
Label: msk0 | Type: sphere | Center: -6,1,-70,18 | Size: 5

```
make a surface from those nodes
h -spec std.141.037_lh.spec \
std.141.lh.pial.gii \
td.141.037_lh_keep.1D 0 -1 \
```

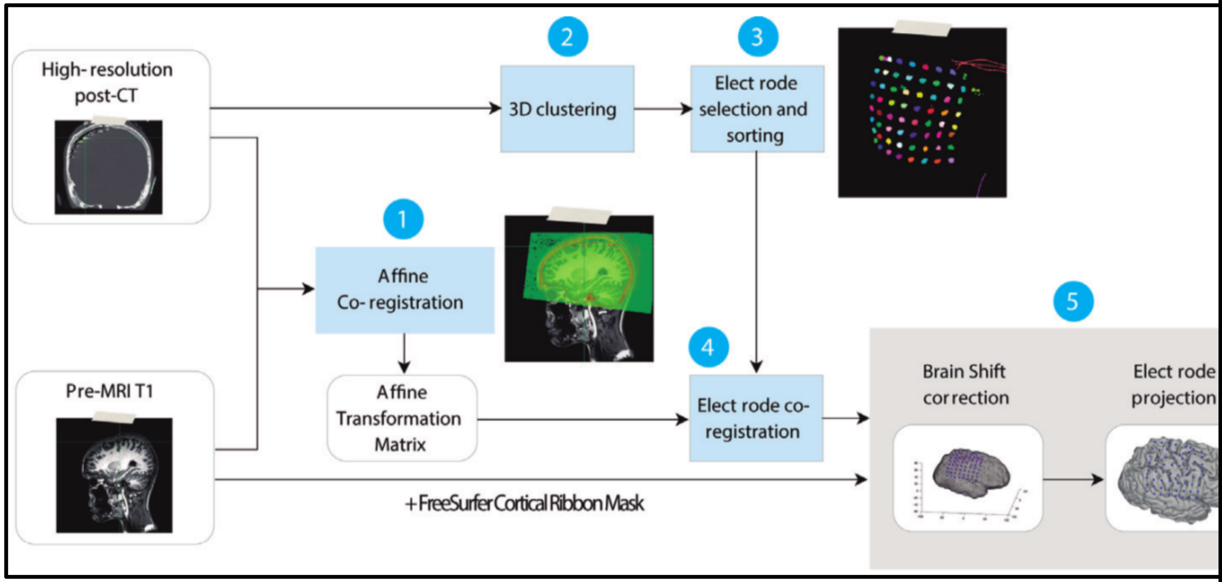
Dset courtesy of Justin Rajendra (now in AFNI group!) and Helen Mayberg.

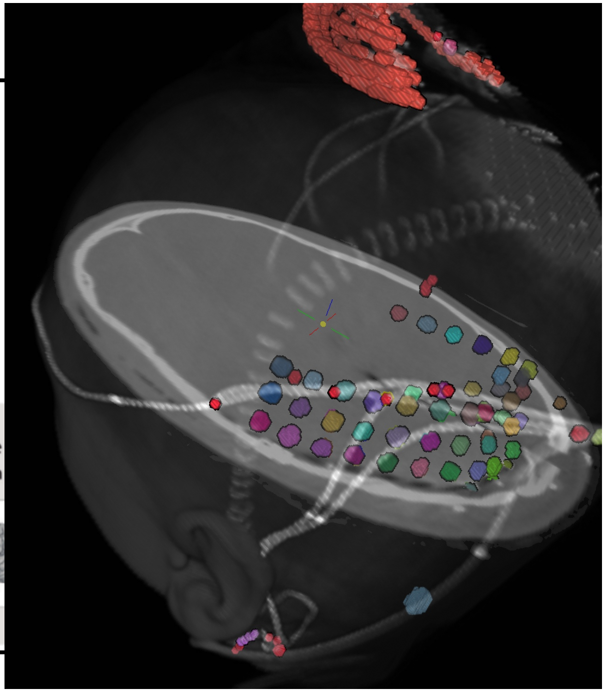


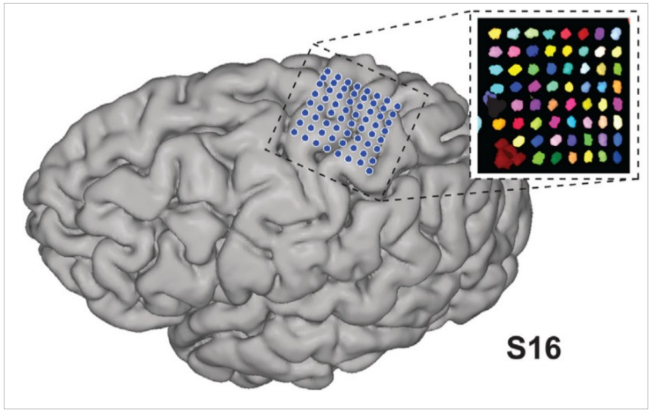
Lauro et al. (2016). And check out *@Install_DBSproc* for demo on DBS processing with CT and DTI processing in AFNI (with Silvina Horovitz).

Alignment with non-MRI data

ECOG – align post-op CT with electrodes to pre-surgical MRI, view grids in 2D (AFNI slices) and 3D (SUMA surfaces).







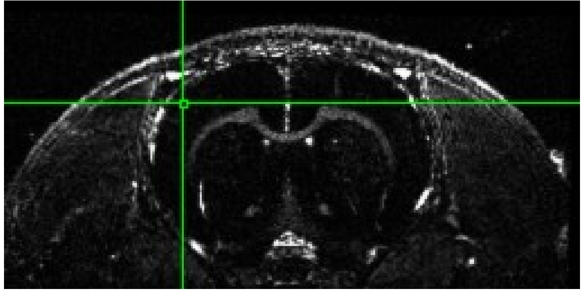
Branco et al. (2018): *ALICE toolbox using AFNI+SUMA.*

Alignment with non-human data: rats

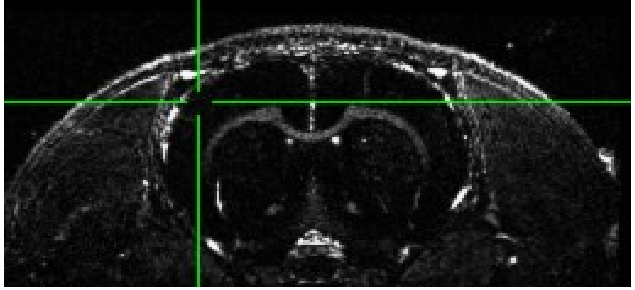
Alignment of 12 hour Manganese enhanced MRI scan (MEMRI) to start

[movies→]

before



after

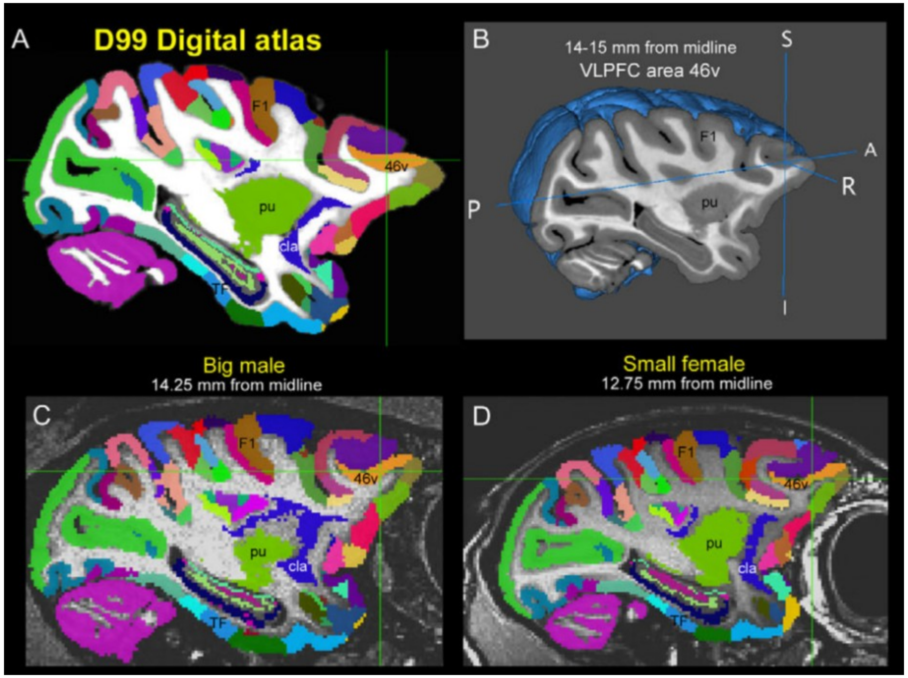


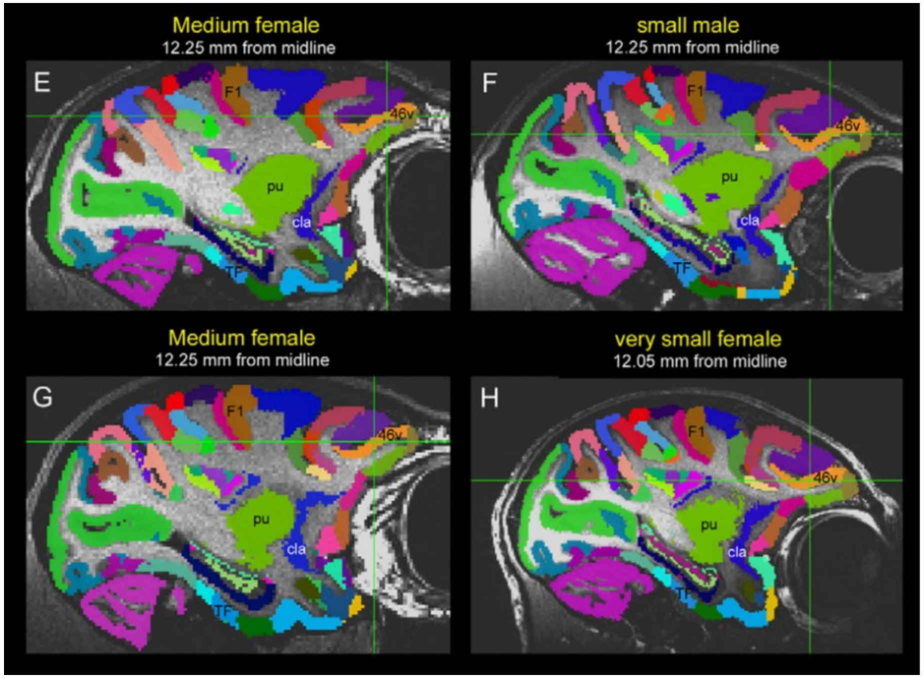
```
#!/bin/tcsh
# align_times.csh
set basedset = 14_pre+orig
foreach timedset ( 14_*hr+orig.HEAD)
  align_epi_anat.py -prep_off -anat $timedset -epi $basedset          \
      -epi_base 0 -anat_has_skull no -epi_strip None -suffix _edge2prep \
      -cost lpa -overwrite -edge -rat_align
end
3dTcat -prefix 14_timealigned_edge 14_pre+orig. 14*edge2prep+orig.HEAD
```

Data from Der-Yow Chen (NINDS)

Alignment with non-human data: macaques

Align new D99 template (Reveley et al. 2017) to individual subjects





See here for alignment scripts and more:

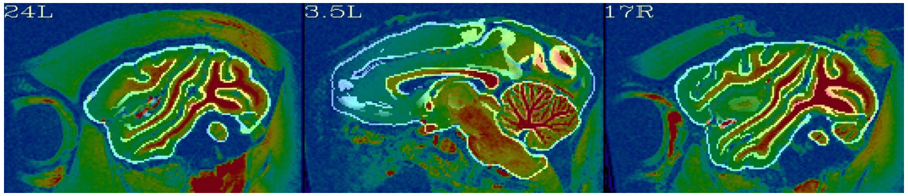
+ Demo with scripts (e.g., *macaque_align.csh*): **@Install_D99_macaque**
+ https://github.com/jms290/NMT (e.g., *NMT_subject_align.csh*)

For questions on non-human alignment, contact D. Glen in AFNI group (email, Message Board).

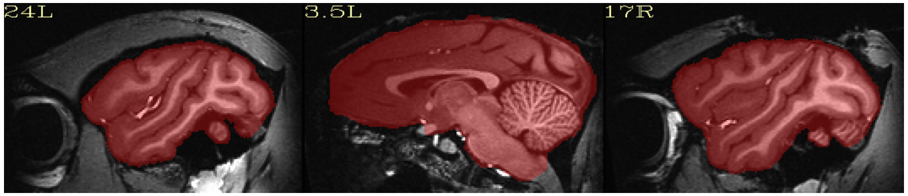
Alignment with non-human data: macaques

New macaque FMRI demo with afni_proc.py (`@Install_MACAQUE_DEMO`)
1) Uses new `@animal_warper` program in AFNI for nonlinear alignment
2) `afni_proc.py` generates pipeline with stats modeling + automatic QC

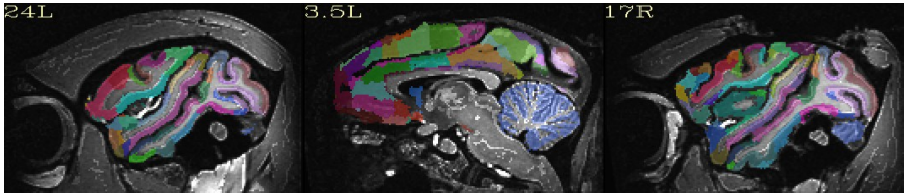
align: template (white lines) to orig anatomical (color)



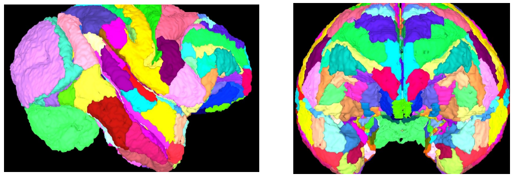
skullstrip: focus on brain (red) from tissue



map atlas: map D99 atlas locations to anatomy



Bonus: SUMA surfaces



For questions on non-human alignment, contact D. Glen in AFNI group (email, Message Board).

Conclusions

Lots of details and concepts to keep in mind:
+ what are the volume contrasts? resolutions? FOV? overlaps? ...?

Always visually check results of alignments
+ some scripts/commands make images automatically
+ the AFNI GUI has many useful features to check ulay/olay
+ you can build your own image-making commands (@chauffeur_afni)

For FMRI: ***afni_proc.py is your new best friend***
+ many features of alignment are taken care of for you, for free!
+ you can also learn more alignment tips by reading the AP script

Supplements

extra stuff

S1: script for @chauffeur_afni + imcat

```
#!/bin/tcsh
# Image-generating script for "Alignment concepts: dset overlap" slide; might be useful.
set odir = .                                # output dir: here
set lcol = ( 150 150 150 )                  # grey gaps in imcat cmd
set refv = MNI152_2009_template_SSW.nii.gz  # refvol, ulay

foreach ff ( `\ls mni_match*gz` )
    # base name of vol, and make a list of all prefixes for later
    set ibase = `3dinfo -prefix_noext "${ff}"`
    set cpref = img0_${ibase}

    # Make a montage of the zeroth brick of each image
    @chauffeur_afni                                               \
        -ulay ${refv]                                             \
        -olay $ff                                                 \
        -thr_olay 14                                              \
        -set_subbricks 0 -1 -1                                    \
        -prefix     $odir/${cpref}                                \
        -pbar_posonly                                             \
        -opacity 5                                                \
        -func_range 110                                           \
        -cbar Viridis                                             \
        -montx 1 -monty 1                                         \
        -set_dicom_xyz   -15 -11 5                                \
        -set_xhairs     MULTI                                     \
        -label_mode 1 -label_size 3                               \
        -do_clean

    # Glue together in 2x2 square, with empty matrix element zero-filled
    imcat                                                         \
        -echo_edu                                                 \
        -gap 5                                                    \
        -gap_col $lcol                                            \
        -nx 2                                                     \
        -ny 2                                                     \
        -zero_wrap                                                \
        -prefix $odir/ALL_subj_${ibase}.jpg                       \
        $odir/${cpref}*
end
```