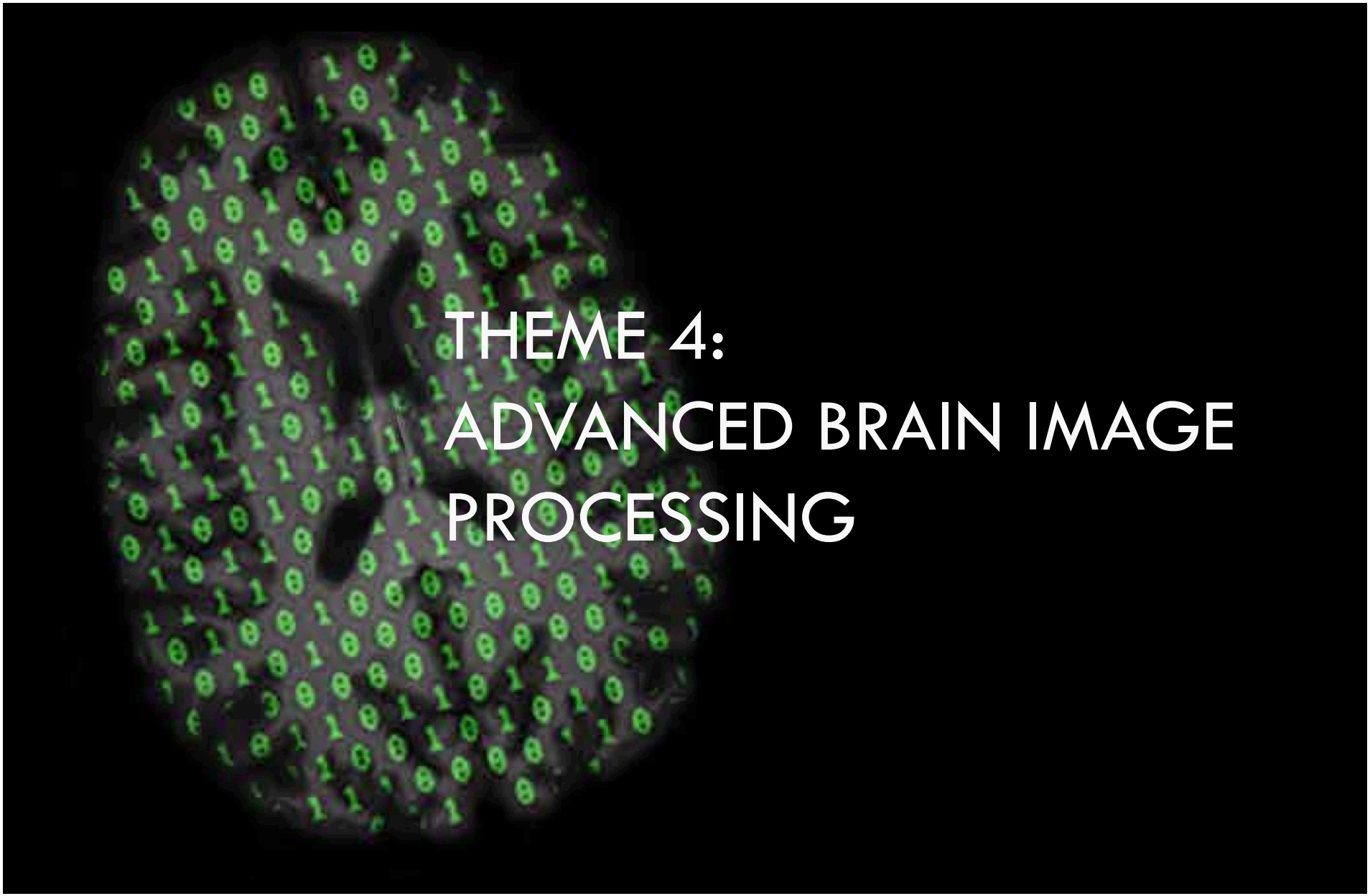
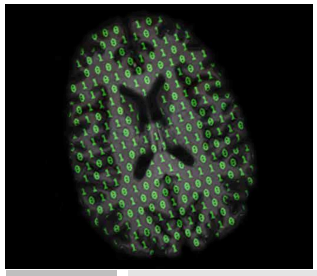
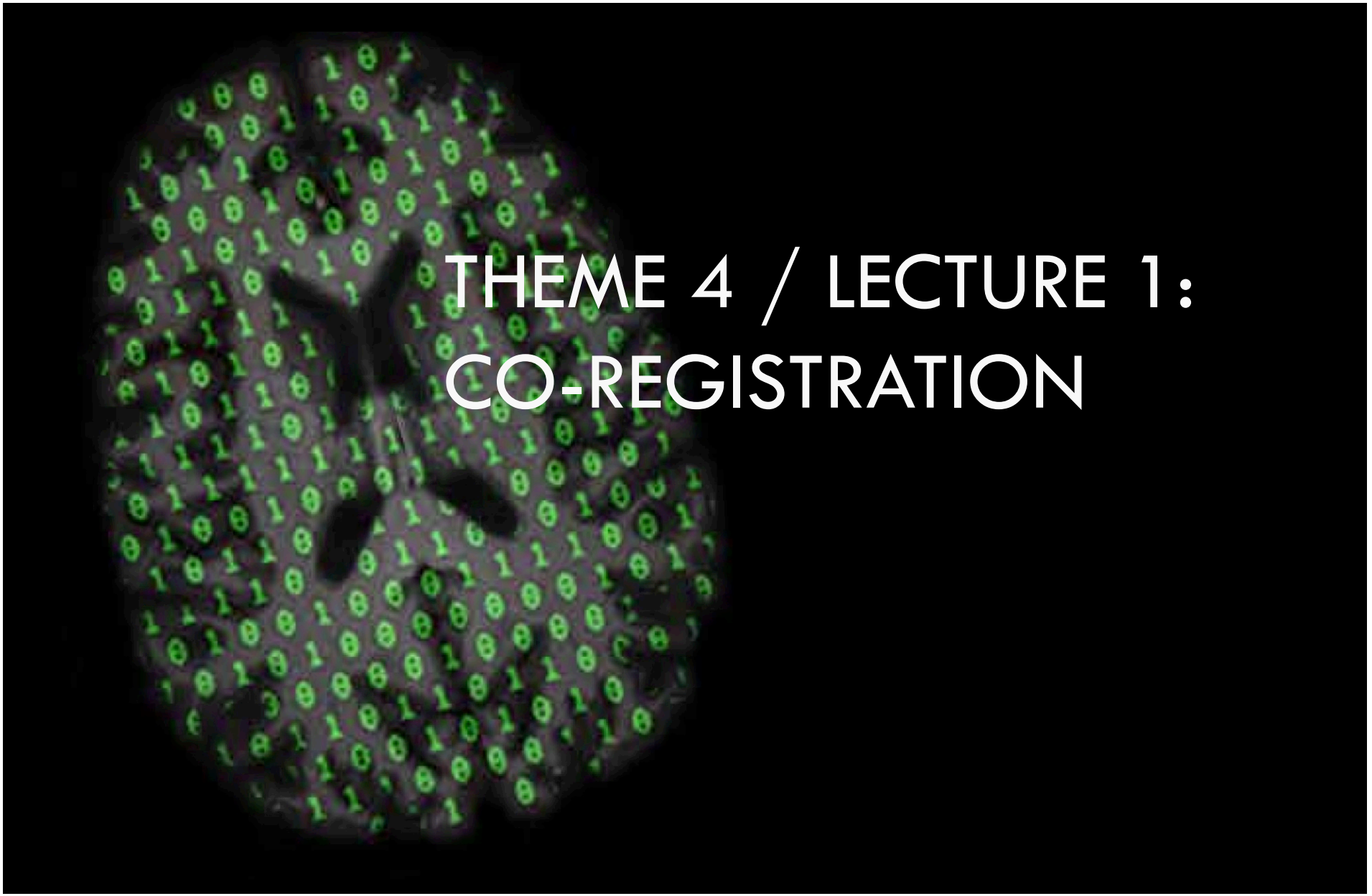


THEME 4: ADVANCED BRAIN IMAGE PROCESSING

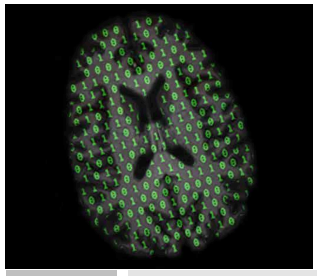


Advanced Brain Image Processing

- Within-subject registration
- ROI localization
- Segmentation

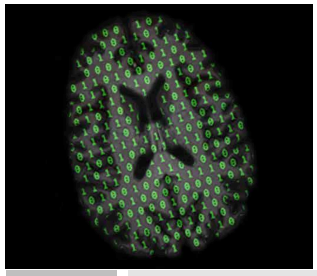


THEME 4 / LECTURE 1: CO-REGISTRATION



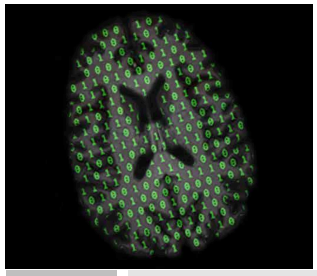
Co-Registration

- Definitions
- Basic components
- Pipeline tools



Types of Registration

- Complexity
  - **rigid (6df)**
  - affine (12df)
  - nonlinear (>12df)
- Co-registration (within the same person)
  - **Cross-sectional between-modalities**
  - **Longitudinal within-modality**
  - **Longitudinal between-modalities**
- Registration to a template
  - A template image is necessary
    - MNI template stored in …/data/Template/MNI152_T1_1mm_brain.nii.gz
    - Eve template stored in …/data/Template/JHU_MNI_SS_T1.nii.gz
  - There are many different templates
- One subject to another



Linear Registration: Rigid

- Rigid registration has 6 degrees of freedom and consists of a translation and a rotation.

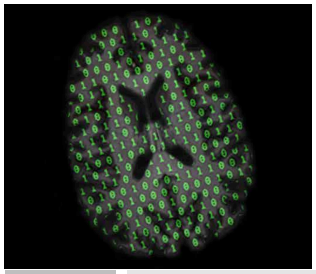
$$T_{\text{rigid}}(v) = Rv + t$$

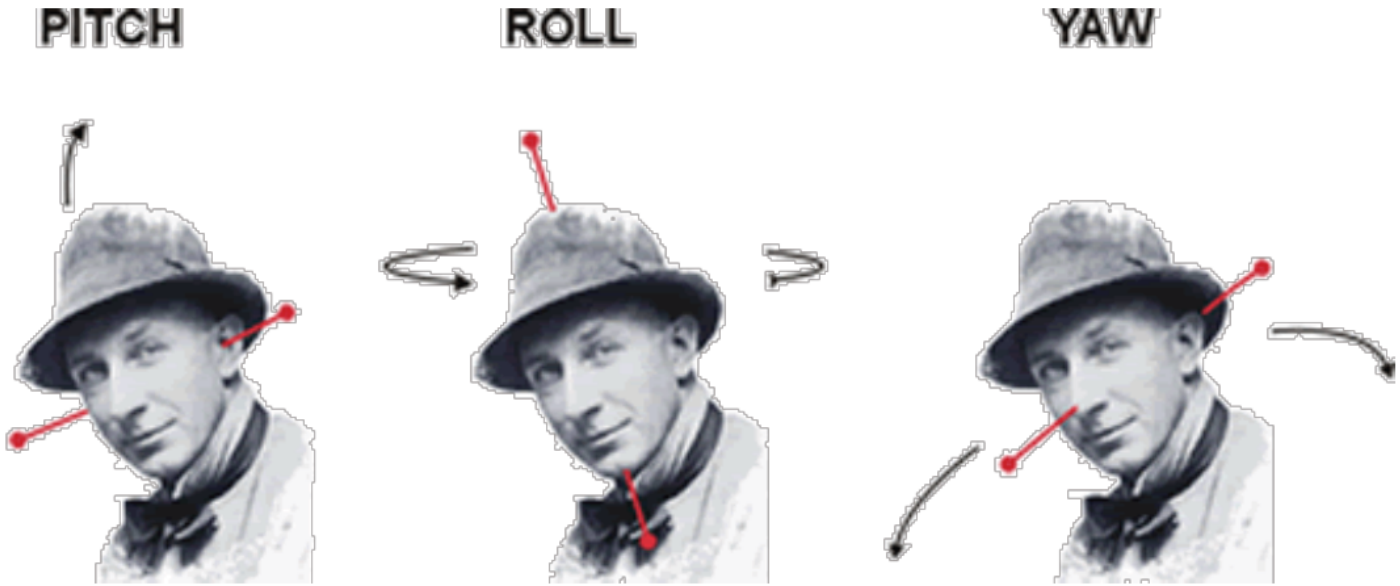
- Rotation Matrix

$$R = \begin{bmatrix} \cos\beta\cos\gamma & \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \sin\alpha\sin\gamma - \cos\alpha\sin\beta\cos\gamma \\ -\cos\beta\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & \sin\alpha\cos\gamma + \cos\alpha\sin\beta\sin\gamma \\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}$$

- Translation vector

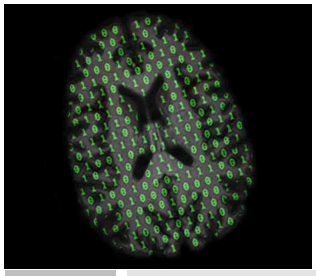
$$t = (t_x, t_y, t_z)$$



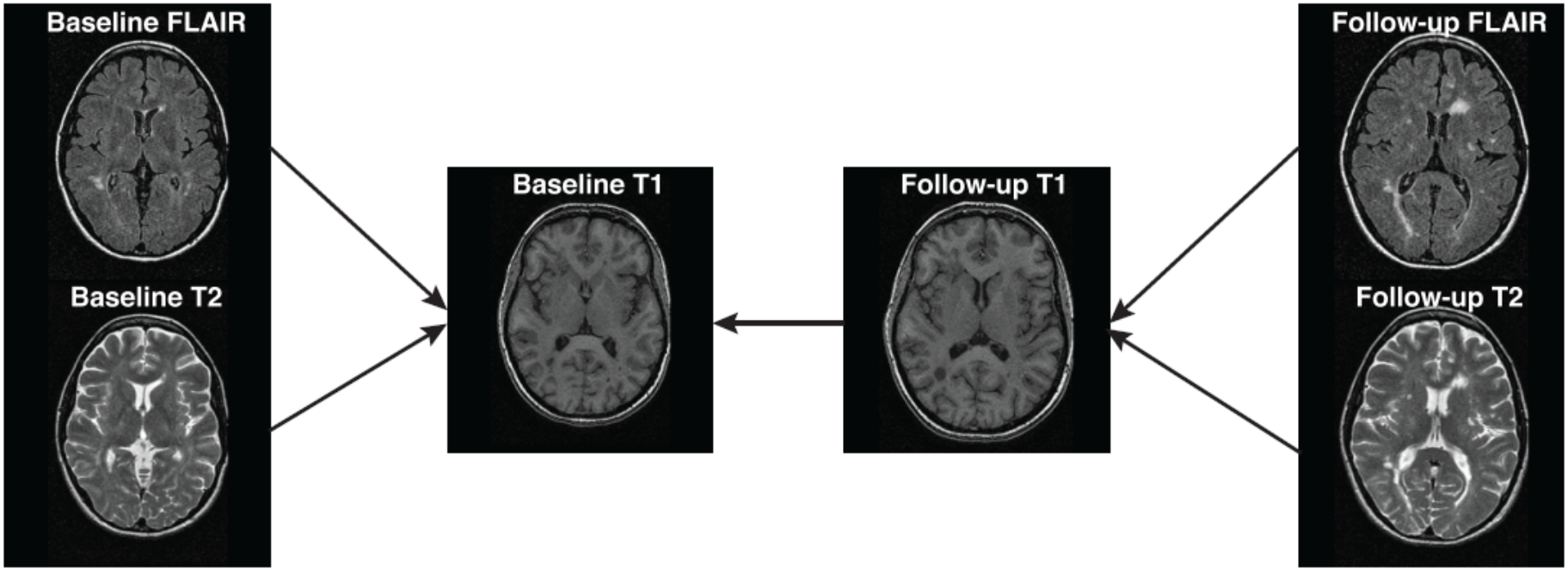


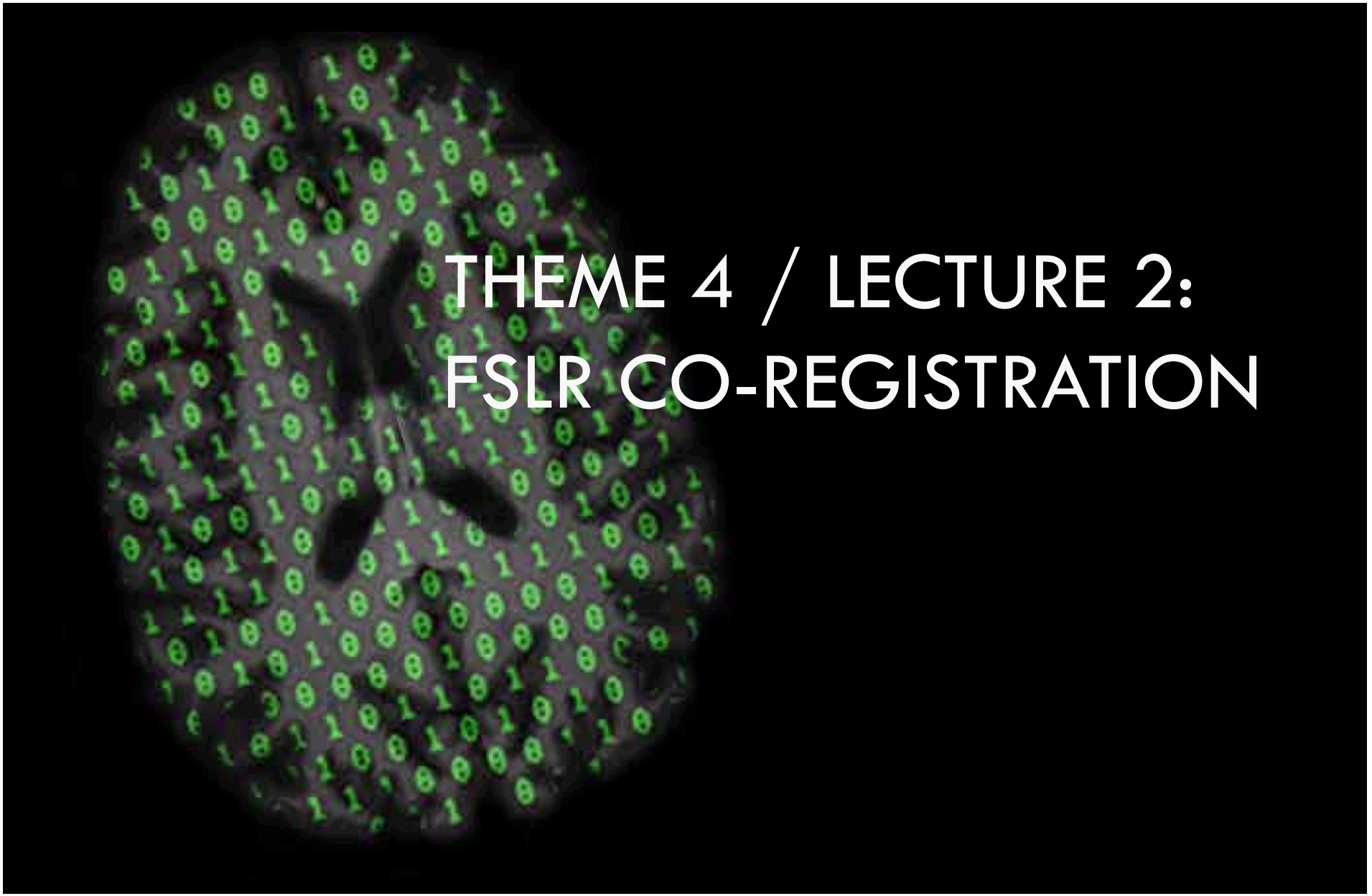
- Pitch: think of nodding “yes”
- Roll: think of shaking head “no” (not in Bulgaria or India!)
- Yaw: think of shoulder shrugging (I don’t know)

Image from http://cnl.web.arizona.edu/imageprops.htm

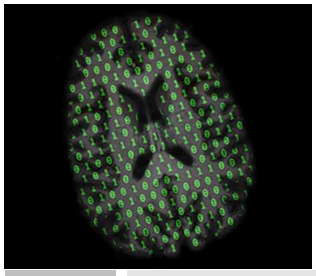


Overall Framework

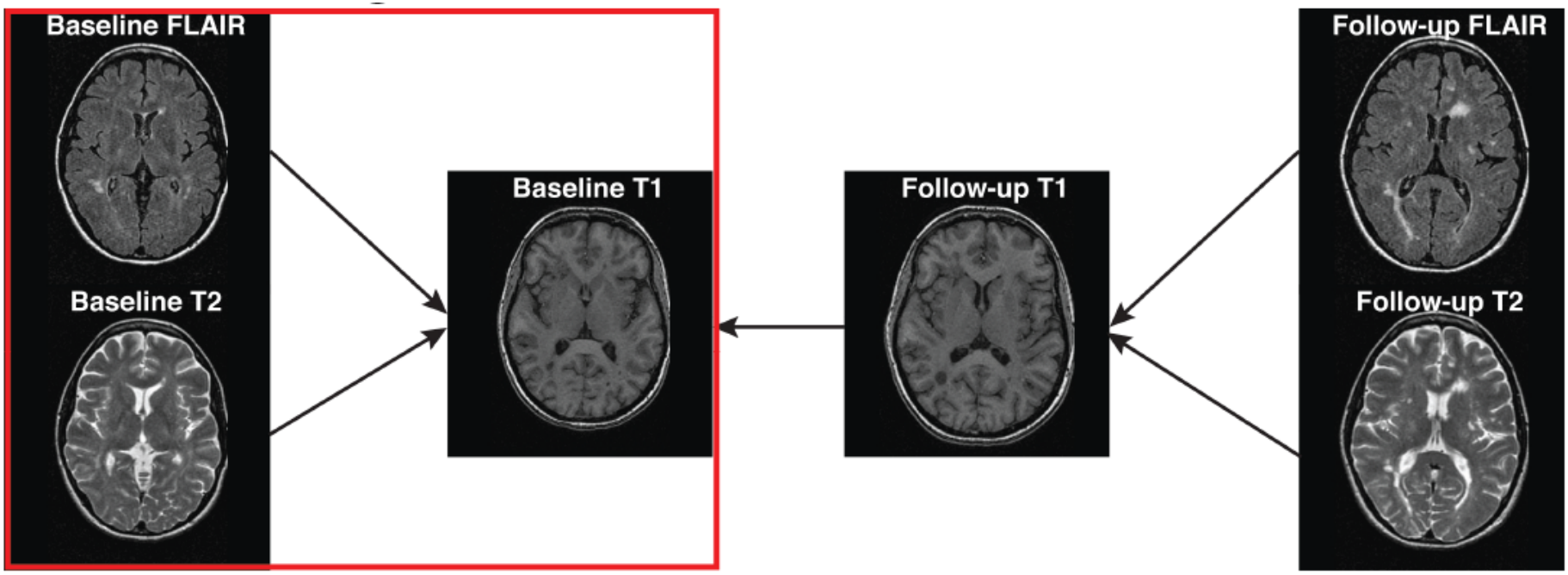


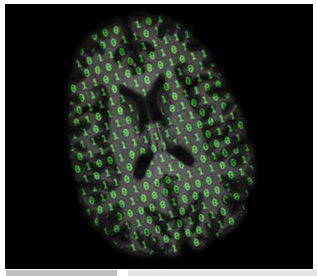


THEME 4 / LECTURE 2: FSLR CO-REGISTRATION



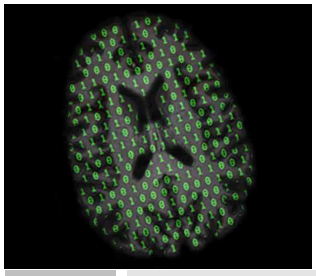
fslr: Co-Registration





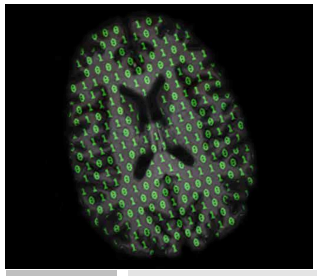
Co-Registration: Registration Within the Same Subject

- Requires fewer degrees of freedom
  - sequences from the same individual/brain are more alike than images from different subjects
- Example analyses that do not require a reference template
  - Identify location-specific longitudinal changes within an individual
  - Tissue class or structural segmentation
  - Analysis of individual-subject change in intensities



Reading the T1 Scan from Visit 1

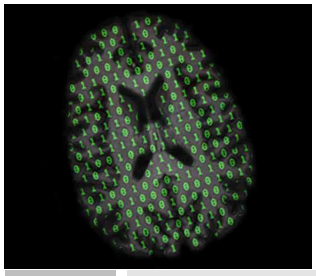
```
library(oro.nifti)
library(extrantsr)
library(fslr)
kirbydir <- "/home/fsluser/Desktop/MOOC-2015/kirby21"
mridir=file.path(kirbydir, "visit_1", "113")
T1_file=file.path(mridir, "113-01-MPRAGE.nii.gz")
T1=readNIfTI(T1_file,reorient=FALSE)
```



FLIRT: FSL's Linear Registration Tool

- From FSL: "FLIRT (FMRIB's Linear Image Registration Tool) is an automated and robust tool for linear (affine) intra- and inter-modal brain image registration"

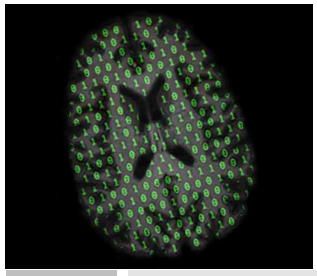
- Here we will register the scan with the skull on



FLIRT: Kirby21 Co-Registration of T2w to T1

Use the `fslr` function `flirt` to register the T2w (infile) to the T1 (reffile), which calls the `FSL` function `flirt`

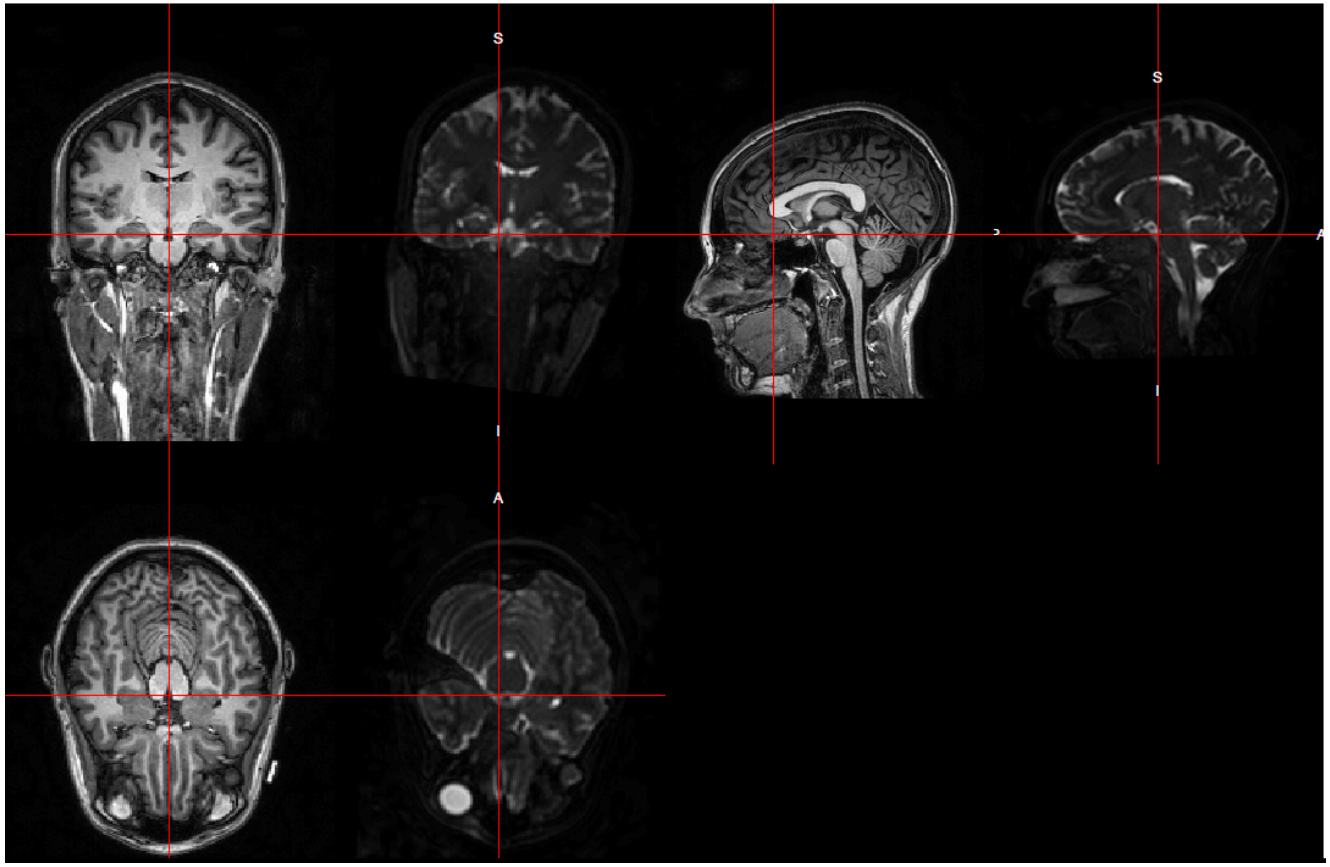
```
T2_file=file.path(mridir, "113-01-T2w.nii.gz")
T2w=readNIfTI(T2_file)
flirt_reg_t2_img = flirt(infile = T2_file, reffile =
T1,dof = 6,verbose = FALSE)
```

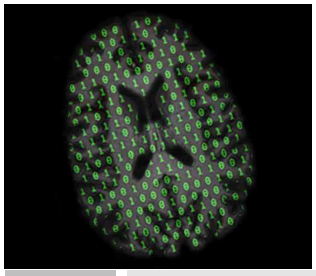


Results

T2w image is tilted and the eye in the axial slice (bottom left) is out of sync

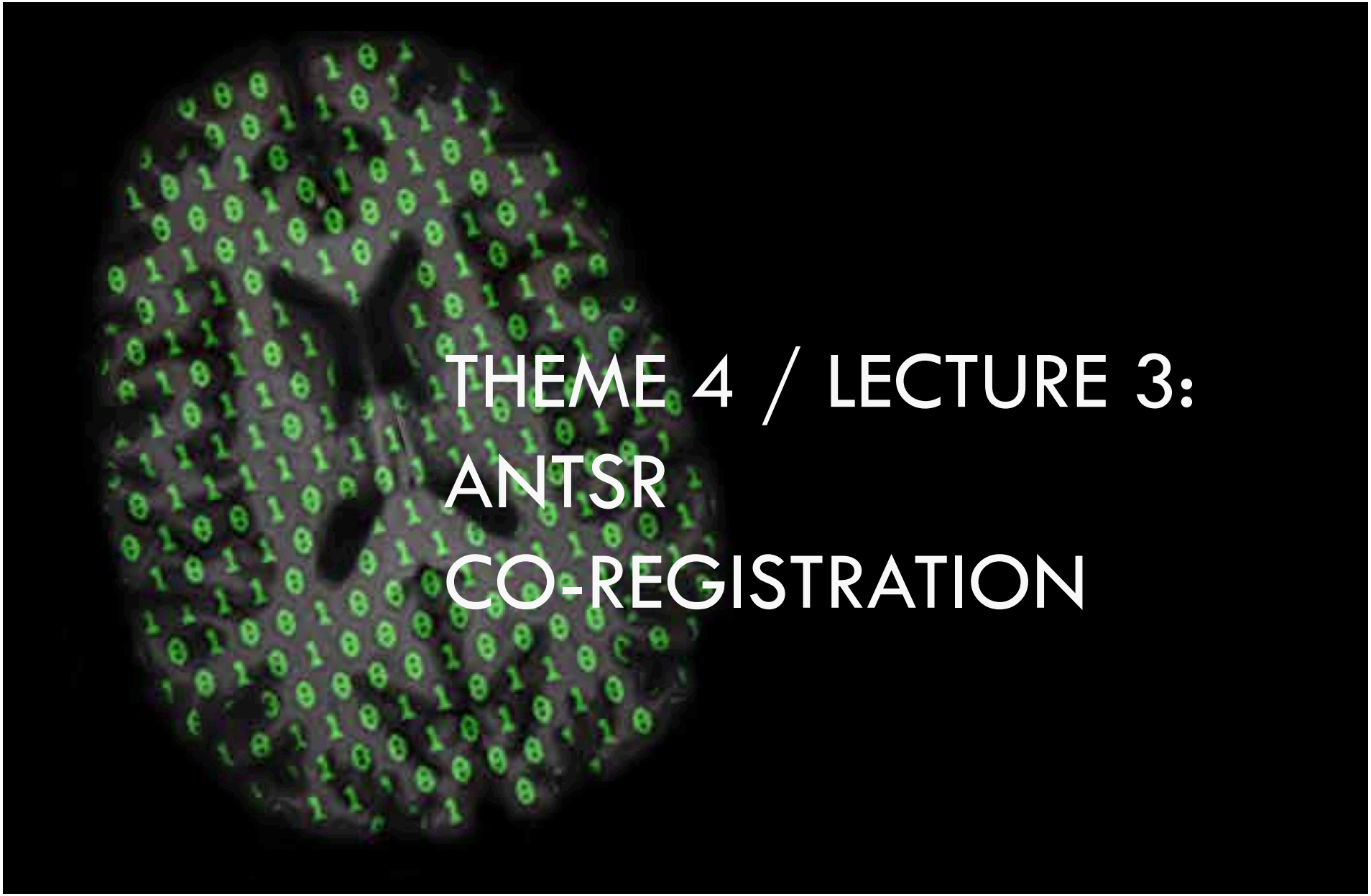
```
double_ortho(T1, flirt_reg_t2_img)
```



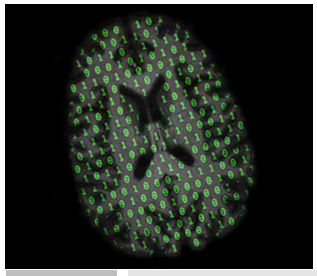


Dimensions of Images

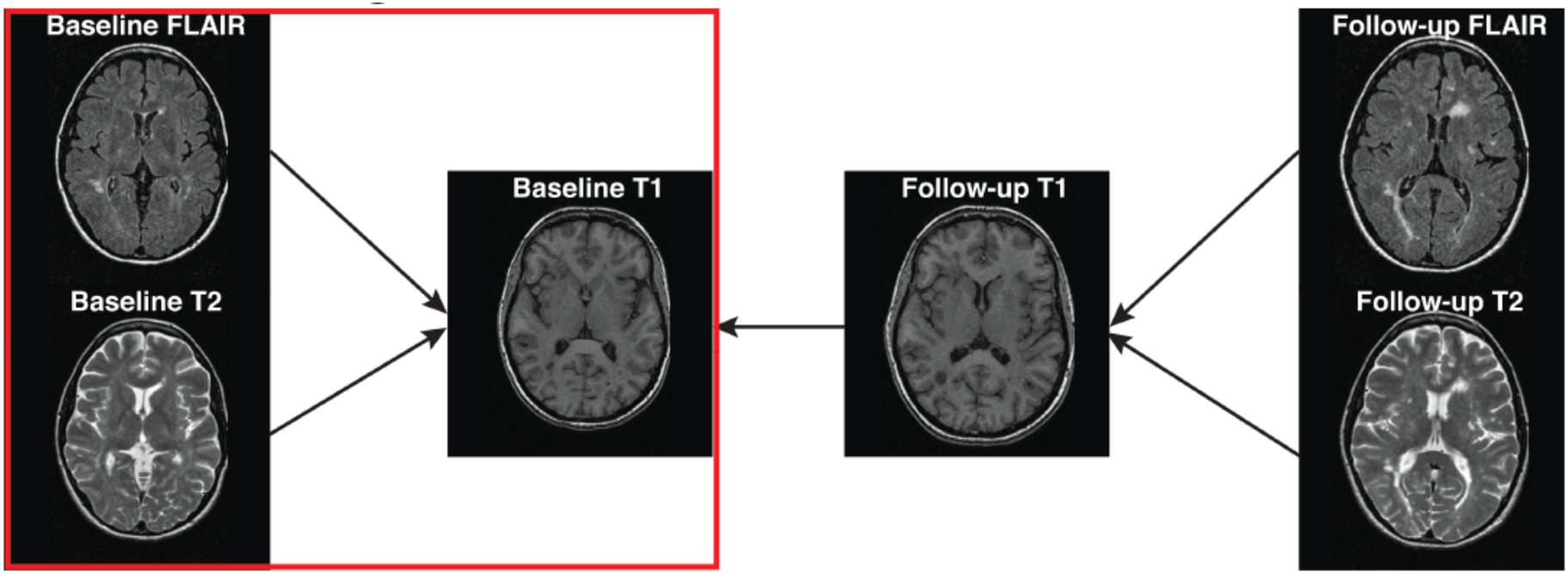
```
dim(T1)
[1] 170 256 256

dim(flirt_reg_t2_img)
[1] 170 256 256

dim(T2w)
[1] 180 256 256
```

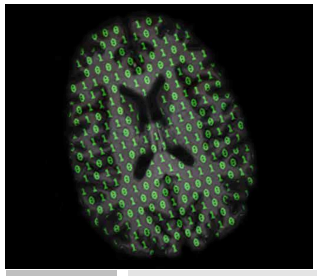


THEME 4 / LECTURE 3:
ANTSR
CO-REGISTRATION



ANTsR Co-Registration

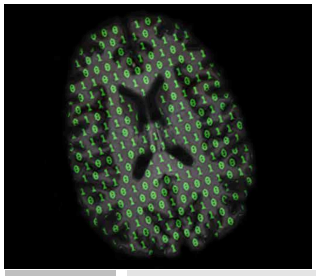




ANTsR: Kirby21 Co-Registration of T2w to T1

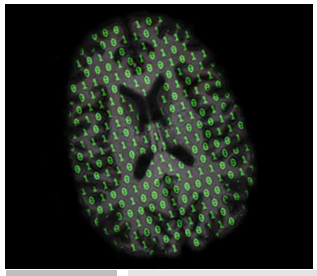
- We will use the `extrantsr` function `ants_regwrite` to register the T2 (filename) to the T1 (template.file) using `ANTsR::antsRegistration`
- Skull on registration

```
T2_file=file.path(mridir, "113-01-T2w.nii.gz")

reg_t2_img = ants_regwrite(filename = T2_file,
template.file=T1,typeofTransform="Rigid",verbose= FALSE)
```



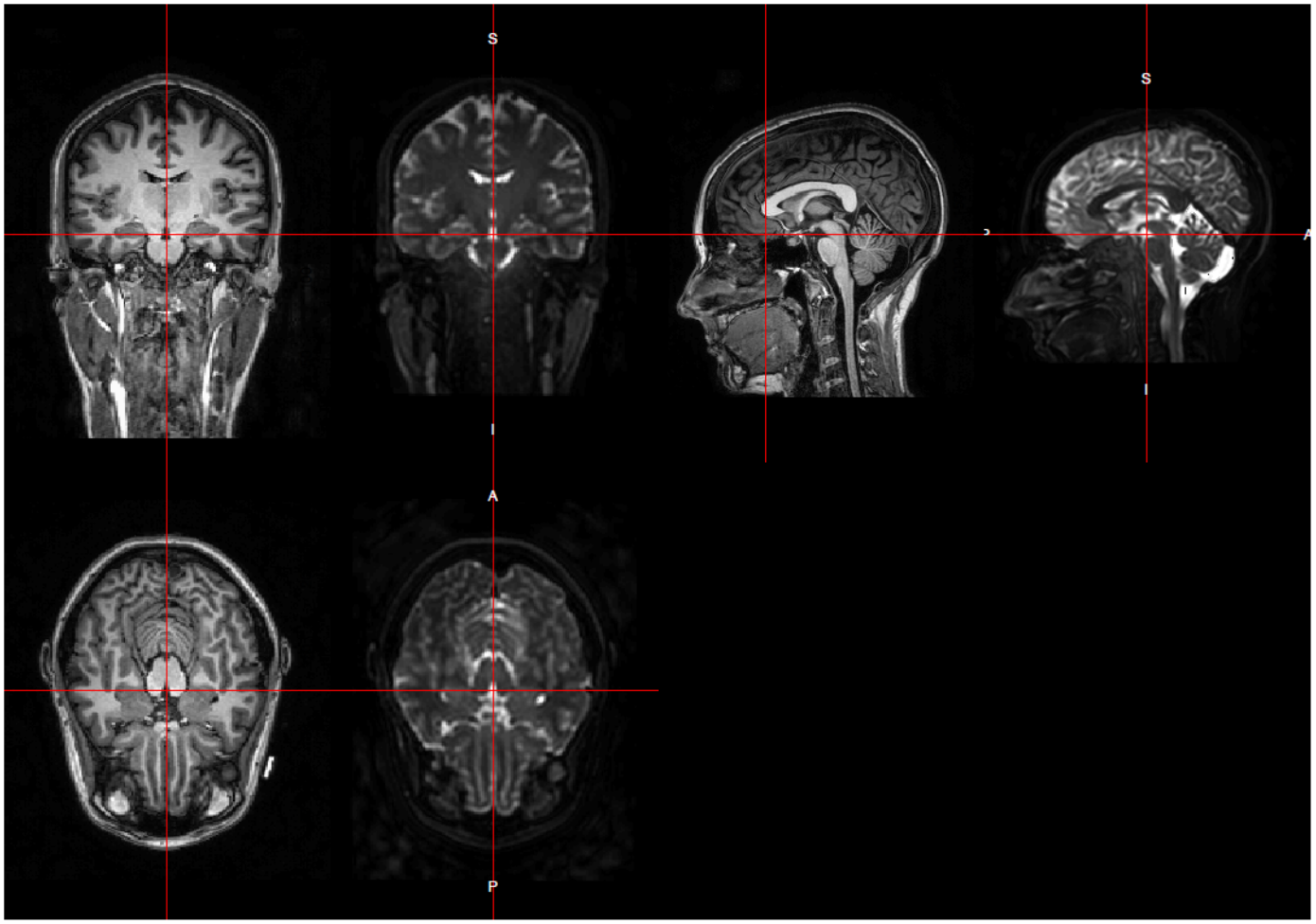
ANTsR: Kirby21 Co-Registration of FLAIR to T1

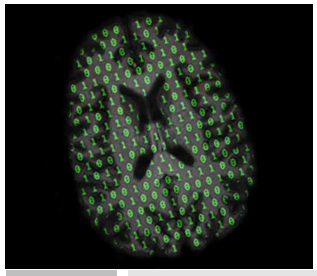
```
flair_file=file.path(mridir, "113-01-FLAIR.nii.gz")

reg_flair_img = ants_regwrite(filename = flair_file,
template.file=T1,typeofTransform="Rigid",verbose= FALSE)
```



T2 Registration Results: ANTsR

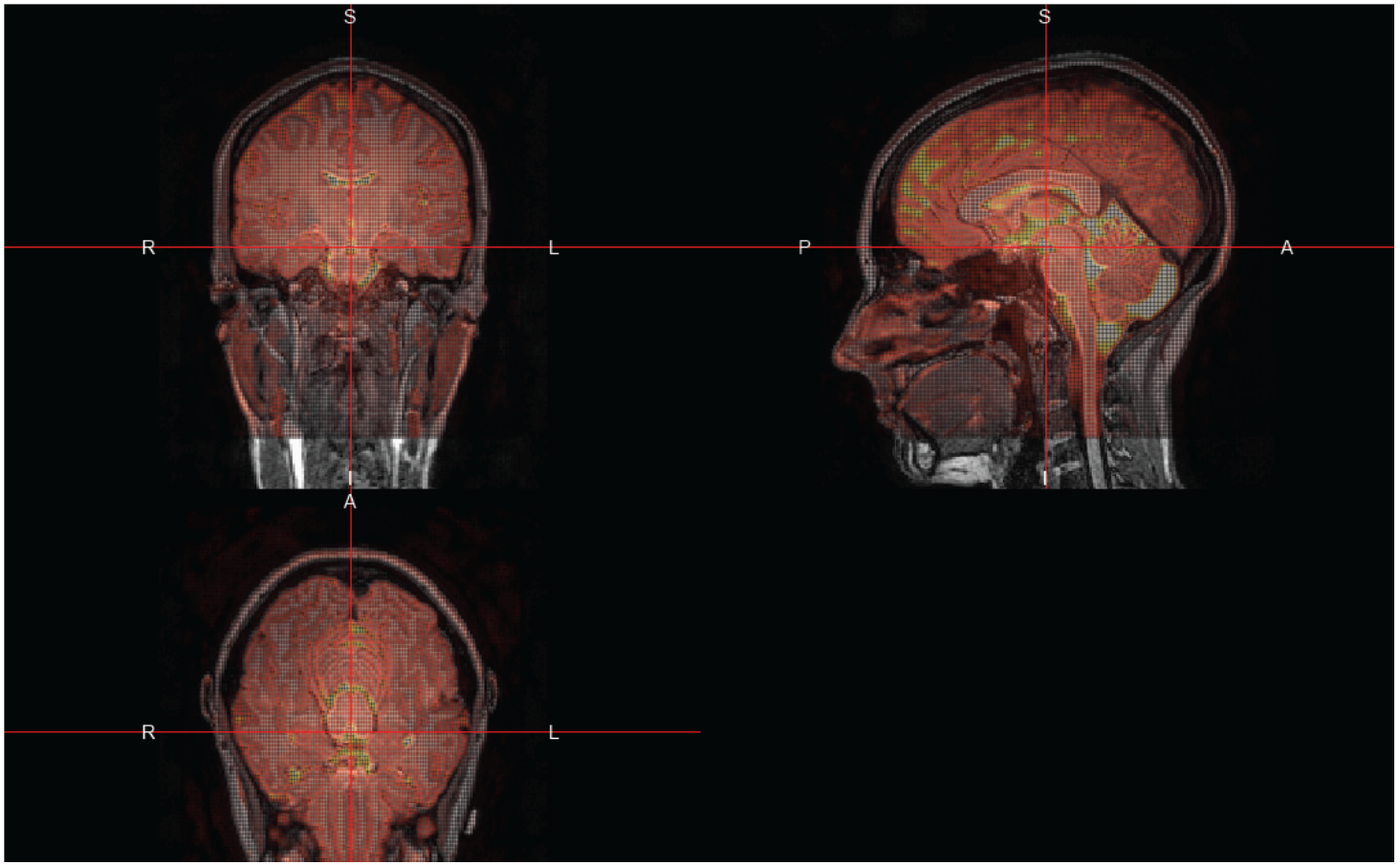
```
double_ortho(T1, reg_t2_img)
```

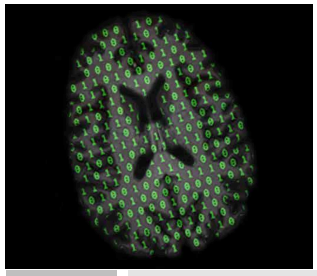




T2 Registration Results: ANTsR, Overlay

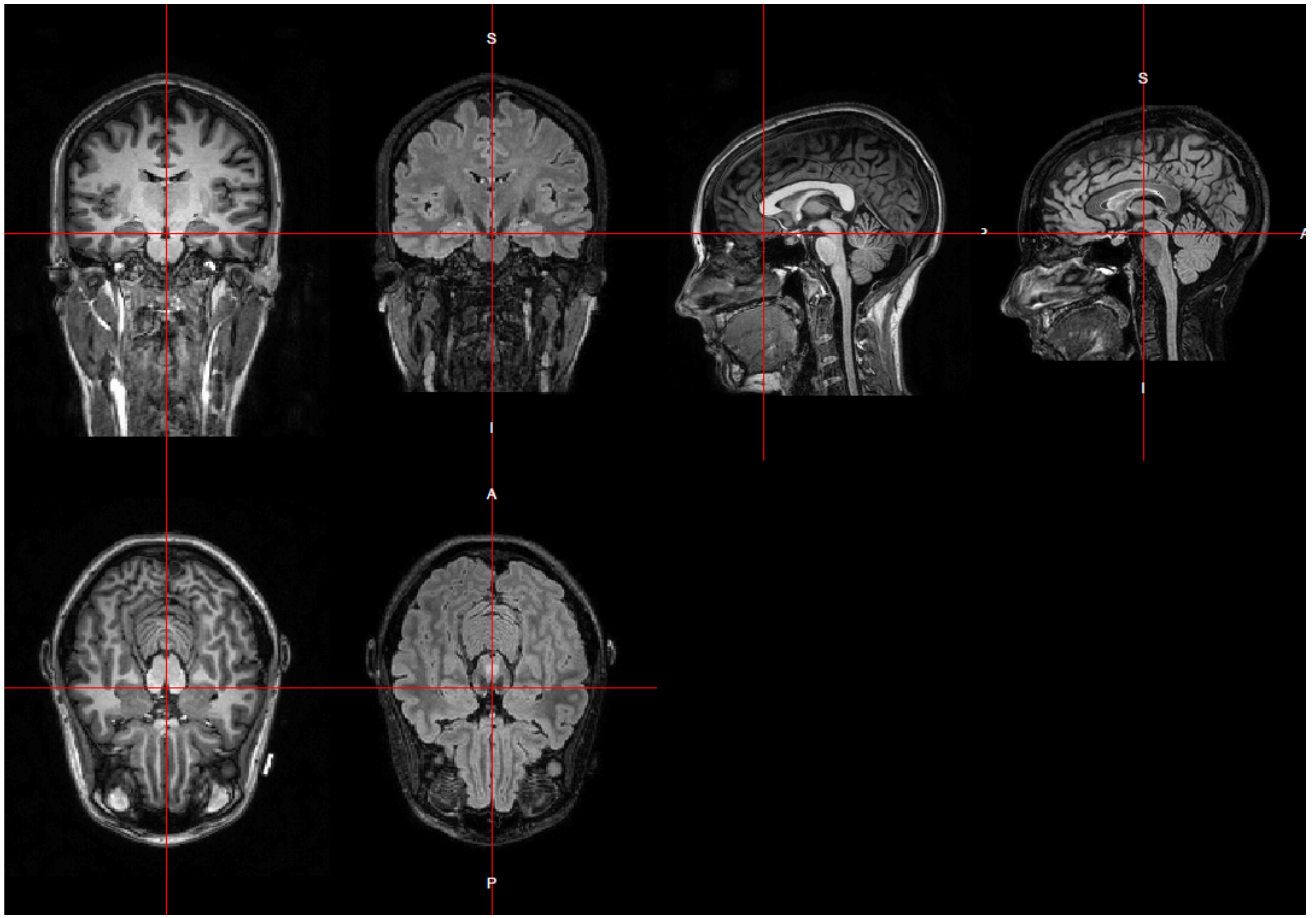
```
library(scales)
ortho2(T1, reg_t2_img, col.y = alpha(hotmetal(), 0.25))
```

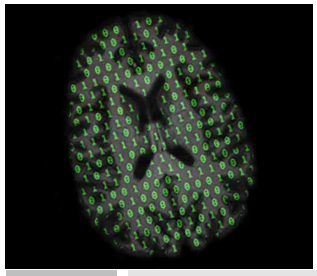




FLAIR Registration Results: ANTsR

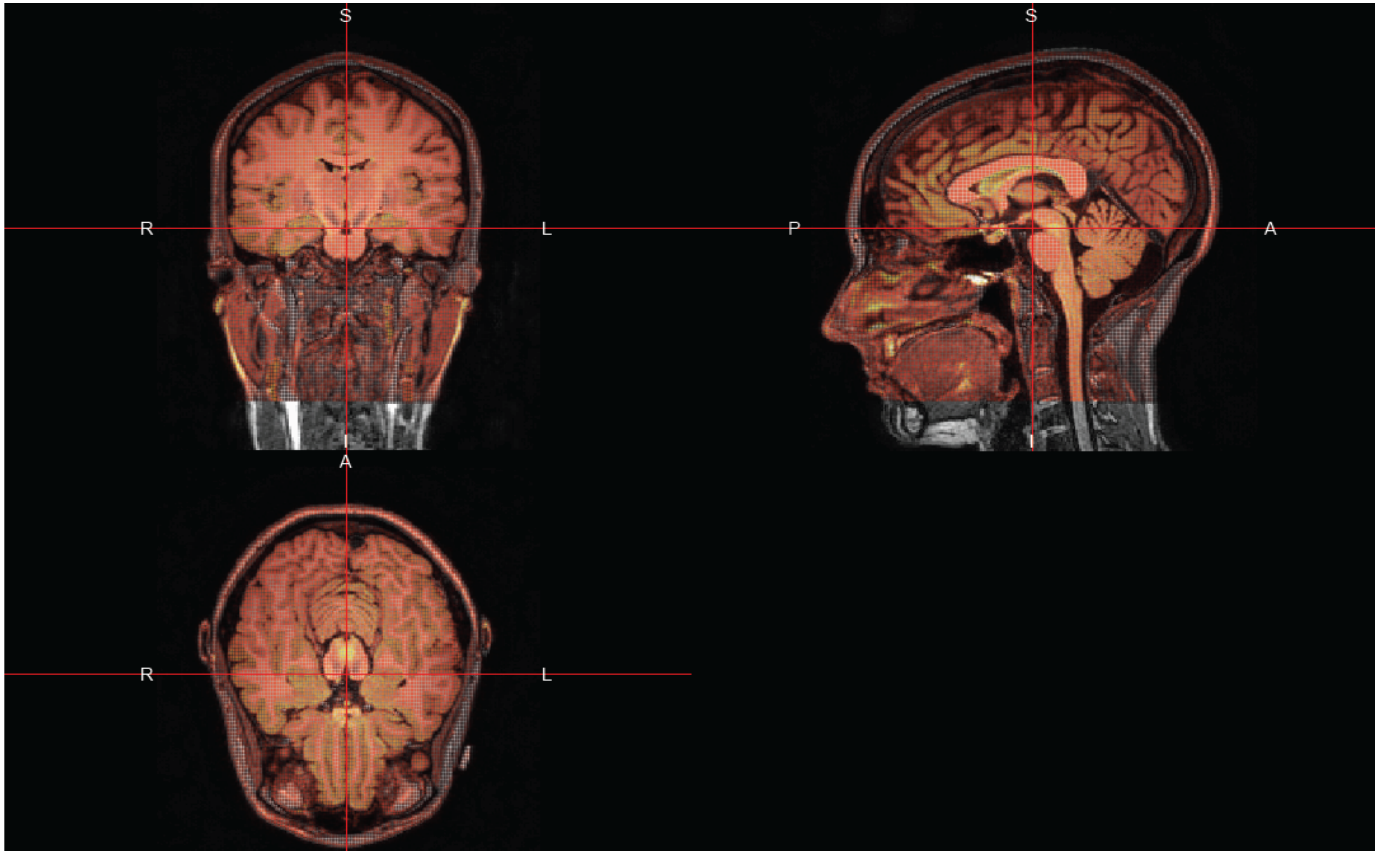
```
double_ortho(T1, reg_flair_img)
```

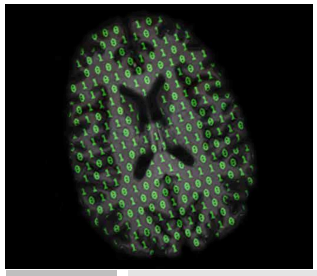




FLAIR Registration Results: ANTsR, Overlay

```
library(scales)
ortho2(T1, reg_flair_img, col.y = alpha(hotmetal(), 0.25))
```

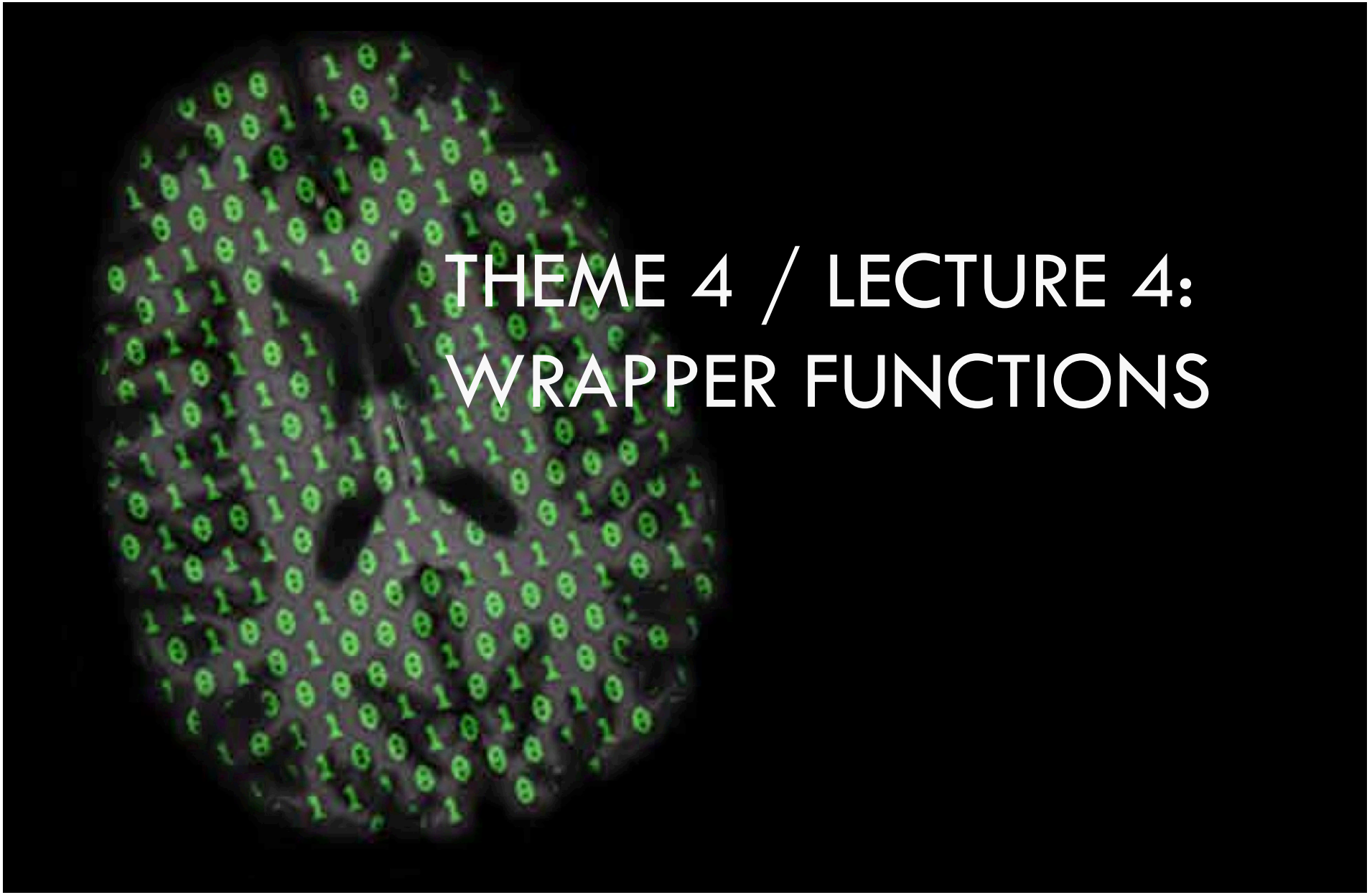




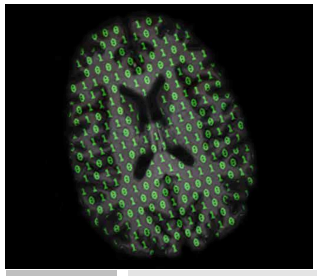
Co-Registration Results

- Overall, there seems to be good overlap after registration with ANTsR
  - Somewhat surprising flirt did not perform well
  - May be due to non-brain tissue

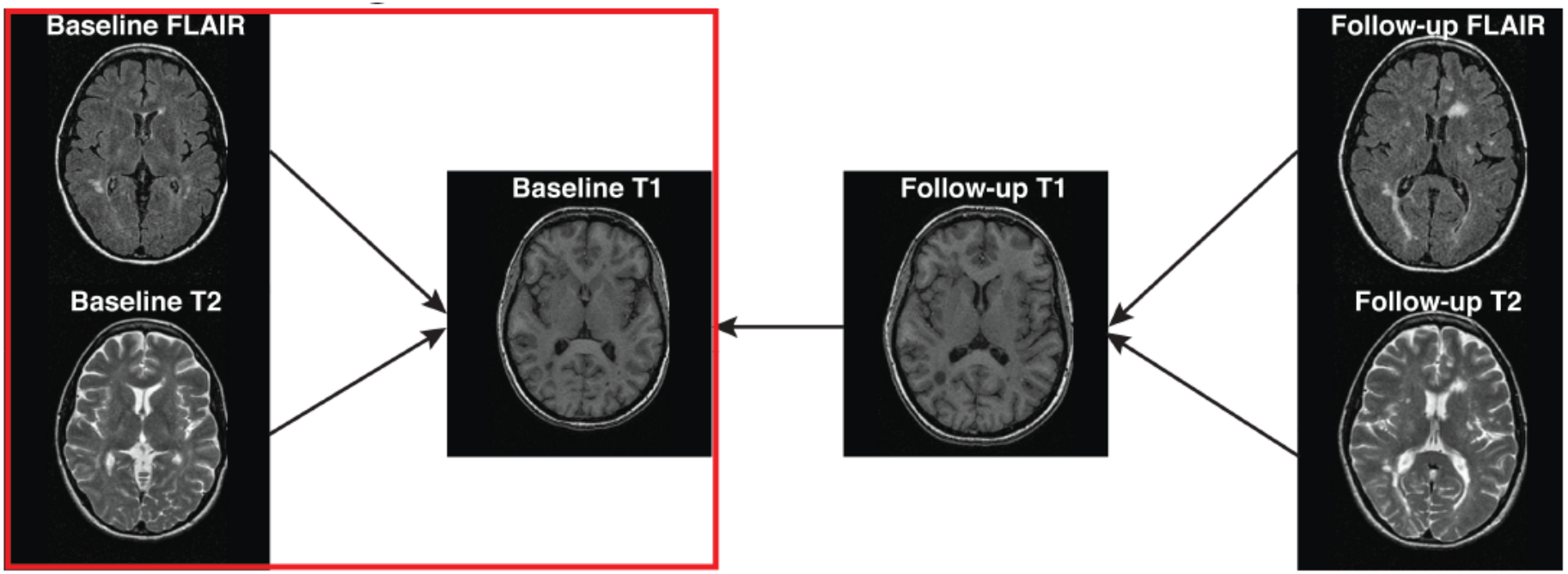
- Registration on the raw data
  - Inhomogeneity correction before registration may be necessary

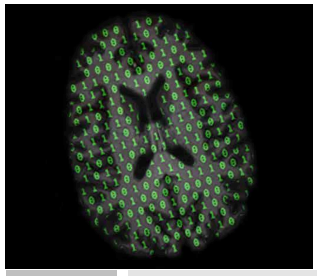


THEME 4 / LECTURE 4: WRAPPER FUNCTIONS



Wrapper Functions

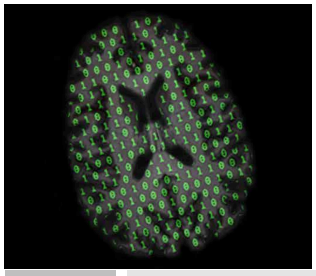




Wrapper Functions

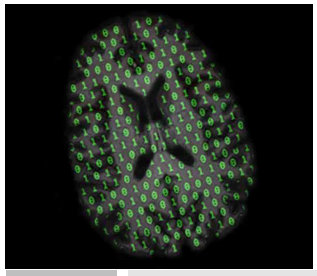
The `extrantsr` function `preprocess_mri_within` will do the following steps:

- Inhomogeneity correction
- Registration of the files to the first filename



Within Visit 1 N4 Correction and Co-Registration

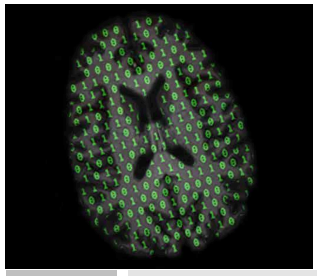
```
files = c("113-01-MPRAGE.nii.gz",
          "113-01-T2w.nii.gz",
          "113-01-FLAIR.nii.gz")

files = file.path(mridir, files)
outfiles = c("113-01-MPRAGE_processed.nii.gz",
             "113-01-T2w_processed.nii.gz",
             "113-01-FLAIR_processed.nii.gz")

outfiles = file.path(mridir, outfiles)

preprocess_mri_within(files = files, retimg = FALSE,
outfiles = outfiles,correction = "N4",skull_strip = FALSE)
```



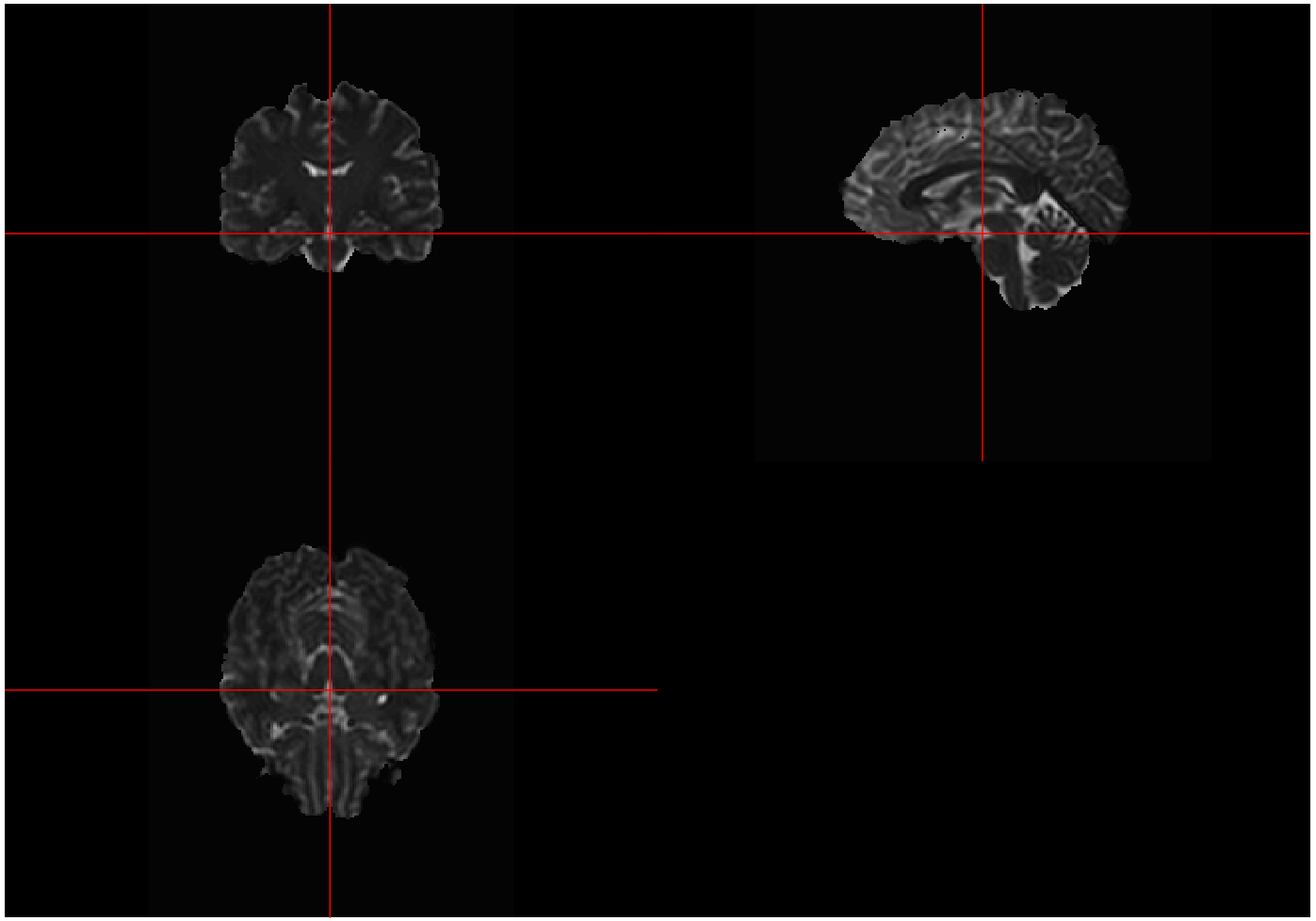
Applying a Brain Mask to All Registered Images

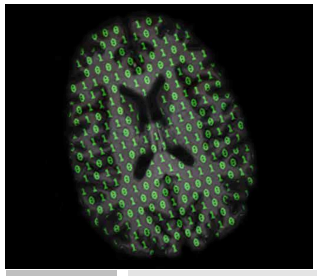
- Images from visit 1 are all in the same space as T1
- If we skull strip the T1 image then the mask can be applied to the other images to extract brain tissue

```
brain = fslbet_robust(img = outfiles[1],
                      correct = FALSE, verbose = FALSE)
mask = brain > 0

masked_imgs = lapply(outfiles, fslmask,
                     mask = mask,verbose = FALSE)

orthographic(masked_imgs[[2]])
```

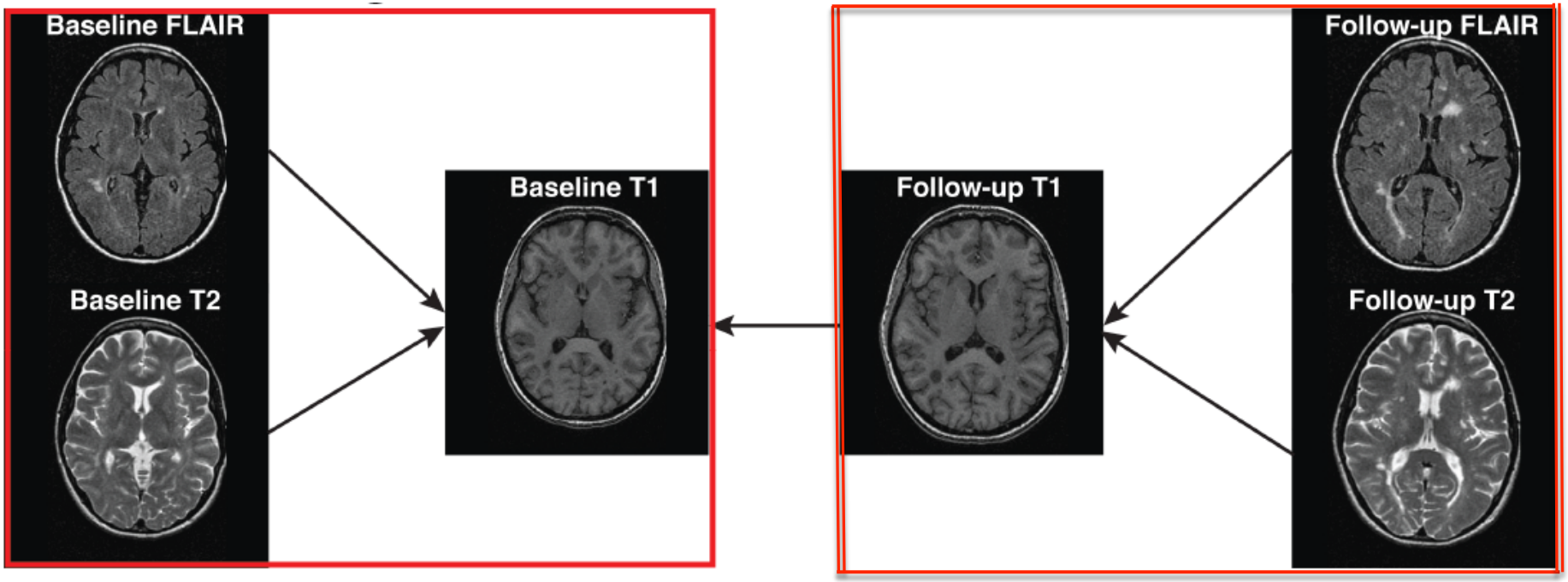


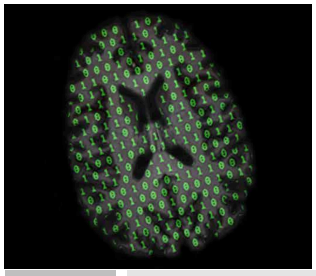
Results for Baseline Masking





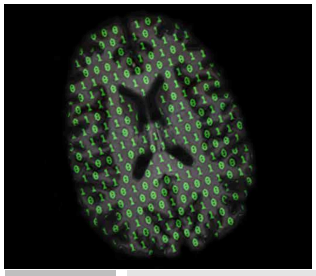
Multi-Sequence Within-visit Co-Registration





Within Visit 2 N4 Correction and Co-Registration

```
mridir2=file.path(kirbydir, "visit_2", "113")

files2 = c("113-02-MPRAGE.nii.gz",
           "113-02-T2w.nii.gz",
           "113-02-FLAIR.nii.gz")

files2 = file.path(mridir2, files2)
outfiles2 = c("113-02-MPRAGE_processed.nii.gz",
              "113-02-T2w_processed.nii.gz",
              "113-02-FLAIR_processed.nii.gz")

outfiles2 = file.path(mridir2, outfiles2)

preprocess_mri_within(files = files2, retimg = FALSE,
outfiles = outfiles2,correction = "N4",skull_strip = FALSE)
```

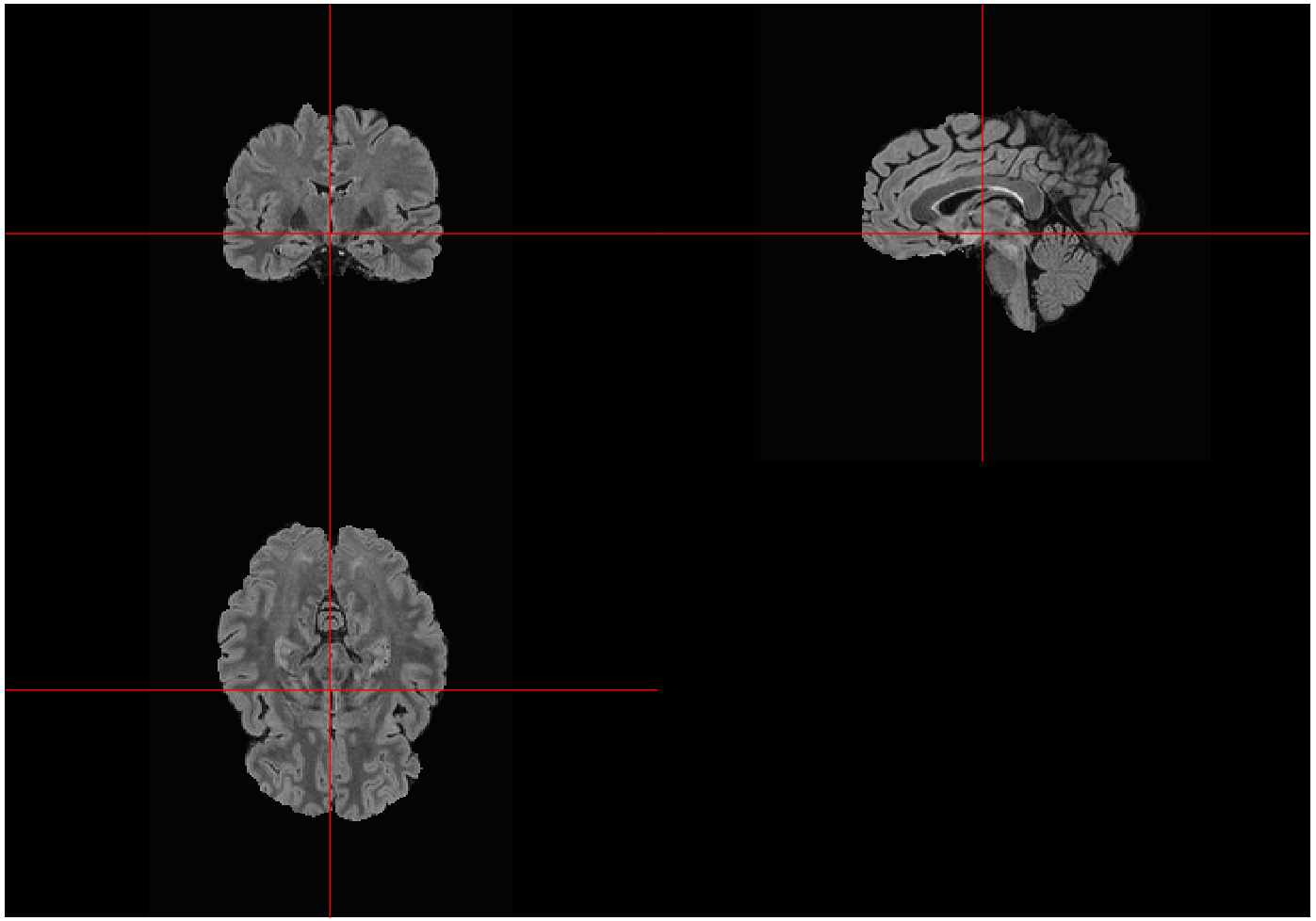


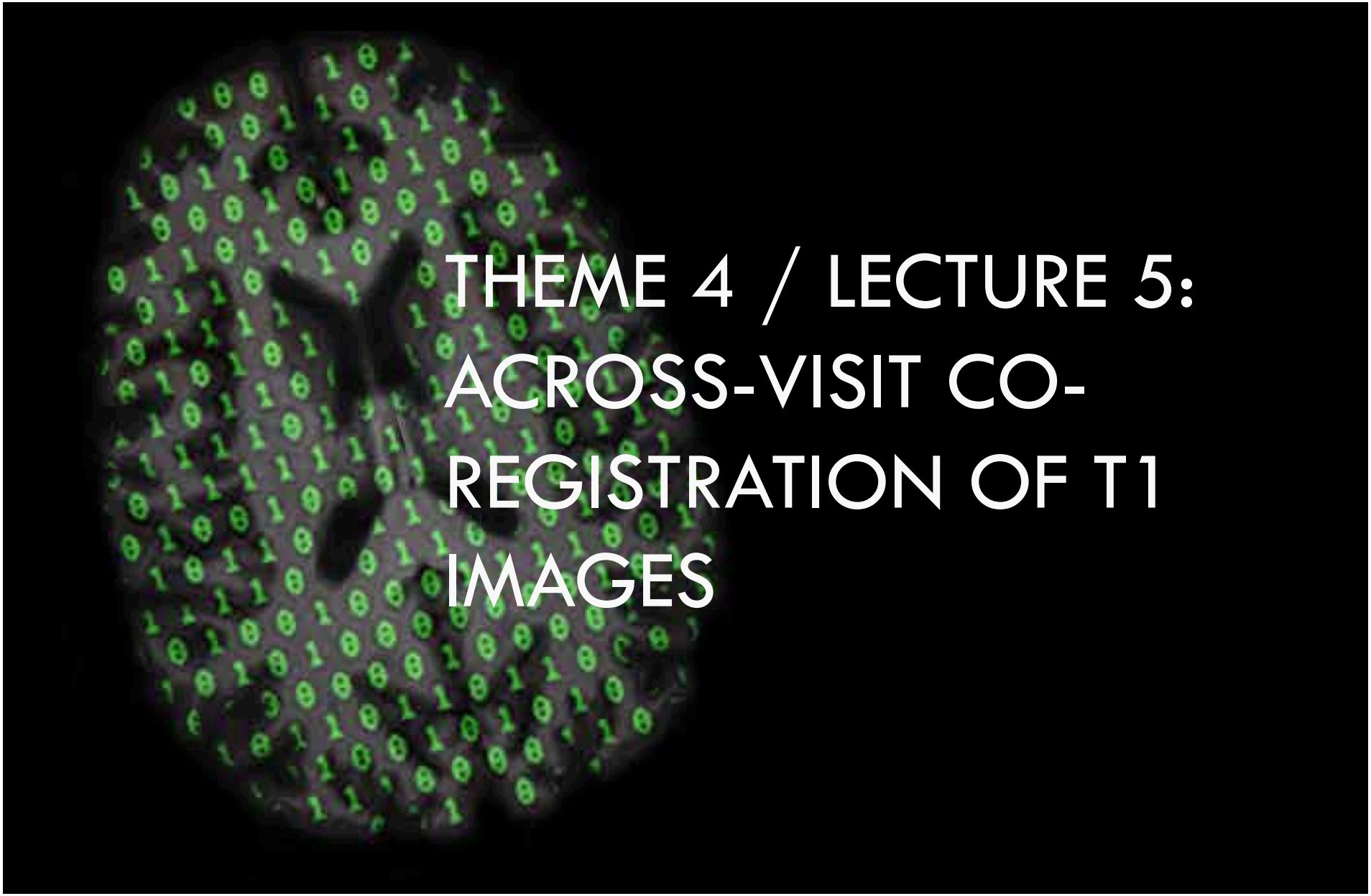
Applying a Brain Mask to All Registered Images

- Images from visit 1 are all in the same space as T1
- If we skull strip the T1 image then the mask can be applied to the other images to extract brain tissue

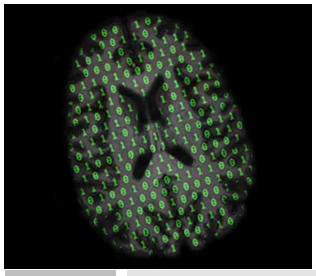
```
brain2 = fslbet_robust(img = outfiles2[1],
                       correct = FALSE, verbose = FALSE)
mask2 = brain2 > 0

masked_imgs2 = lapply(outfiles2, fslmask,
                      mask = mask2,verbose = FALSE)

orthographic(masked_imgs2[[3]])
```

Results for Follow-up Masking

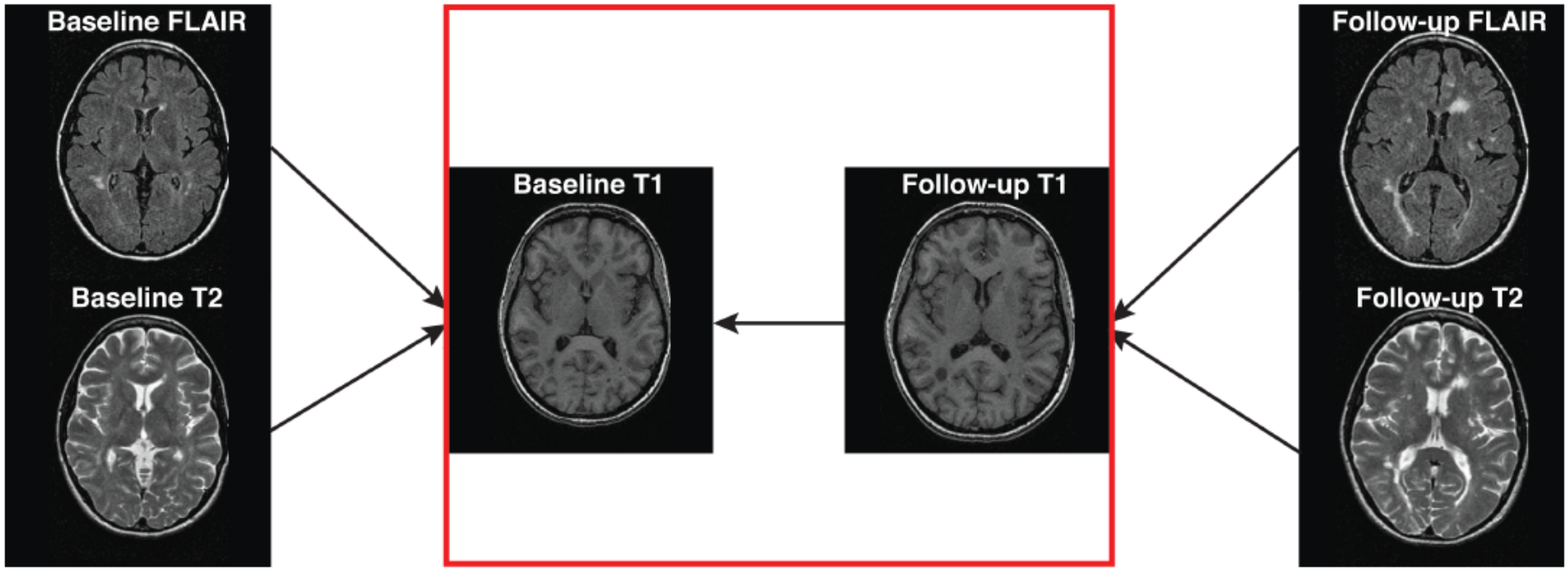


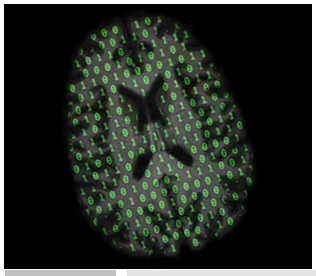


THEME 4 / LECTURE 5: ACROSS-VISIT CO-REGISTRATION OF T1 IMAGES



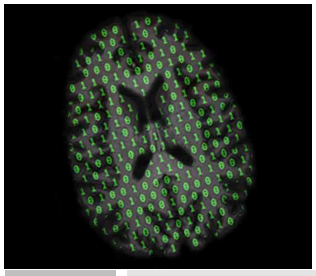
Across-Visit Co-Registration of T1 Images





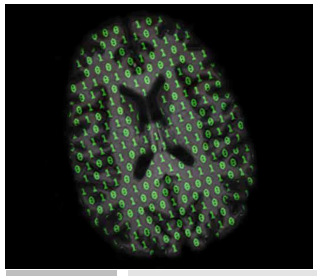
Across-Visit Co-Registration of T1 Images

- Visit 1 files are in the visit 1 T1 space and visit 2 files are in the visit 2 T1 space
- We will register the follow-up/visit 2 T1 scan to the visit 1/baseline scan
- Next slide:
  - take the skull-stripped T1 from visit 2, register it to the skull-stripped T1 from visit 1
  - use this transformation to put the skull-stripped T2 and skull-stripped FLAIR from visit 2 into the T1 space from visit 1



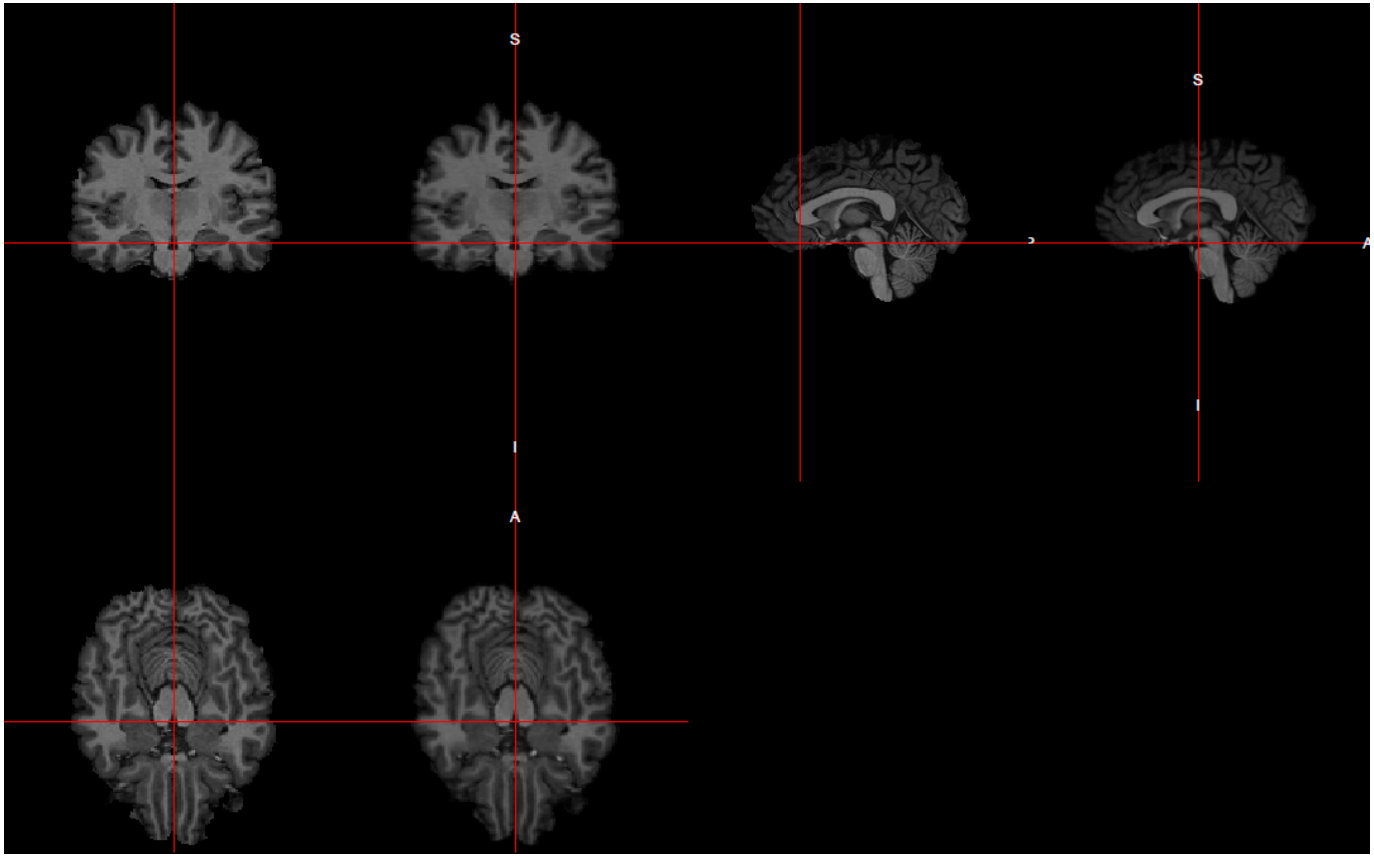
Across-Visit Co-Registration Skull-Off Images

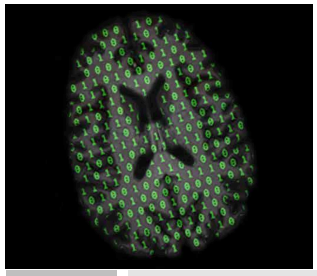
```
visit2_files = file.path(mridir2,
                  c("113-02-MPRAGE_processed.nii.gz",
                    "113-02-T2w_processed.nii.gz",
                    "113-02-FLAIR_processed.nii.gz"))
outfiles2 = sub(".nii.gz", "_reg.nii.gz", visit2_files)

ants_regwrite(filename = masked_imgs2[[1]],
              retimg = FALSE, outfile = outfiles2[1],
              template.file = masked_imgs[[1]],
              other.files = masked_imgs2[2:3],
              other.outfiles = outfiles2[2:3],
              typeofTransform = "Rigid", verbose = FALSE)
```



Plotting Registered T1 Images Side-By-Side

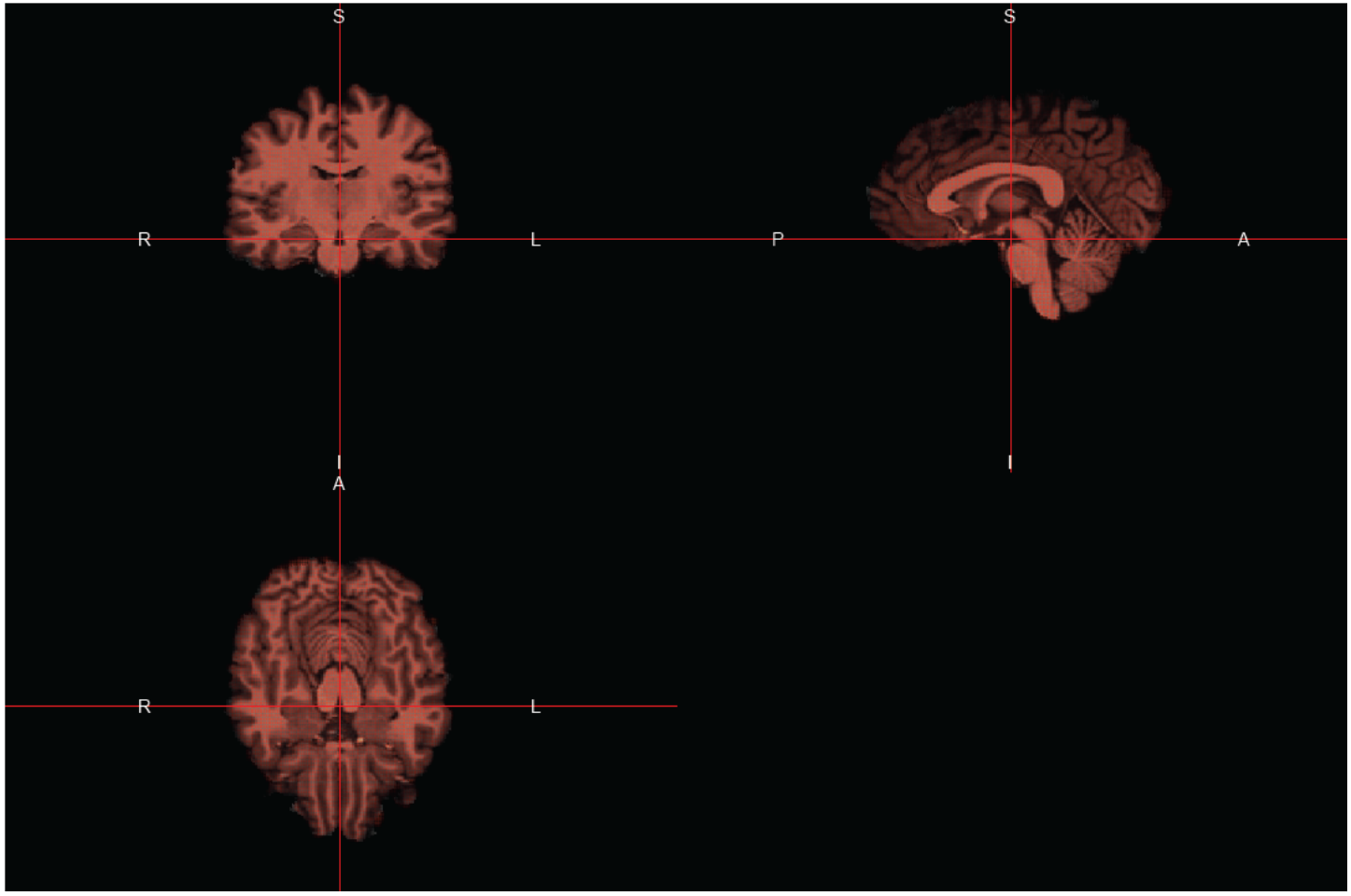
```
ss_t1=masked_imgs[[1]]
visit_2_t1=readNIfTI(outfiles2[1],reorient=FALSE)
double_ortho(ss_t1,visit_2_t1)
```

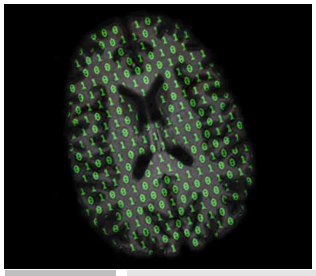




Plotting Registered T1 Image Overlay

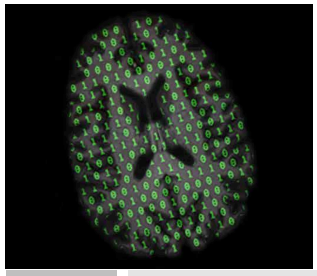
```
ortho2(ss_t1,visit_2_t1,col.y=alpha(hotmetal(),0.25))
```





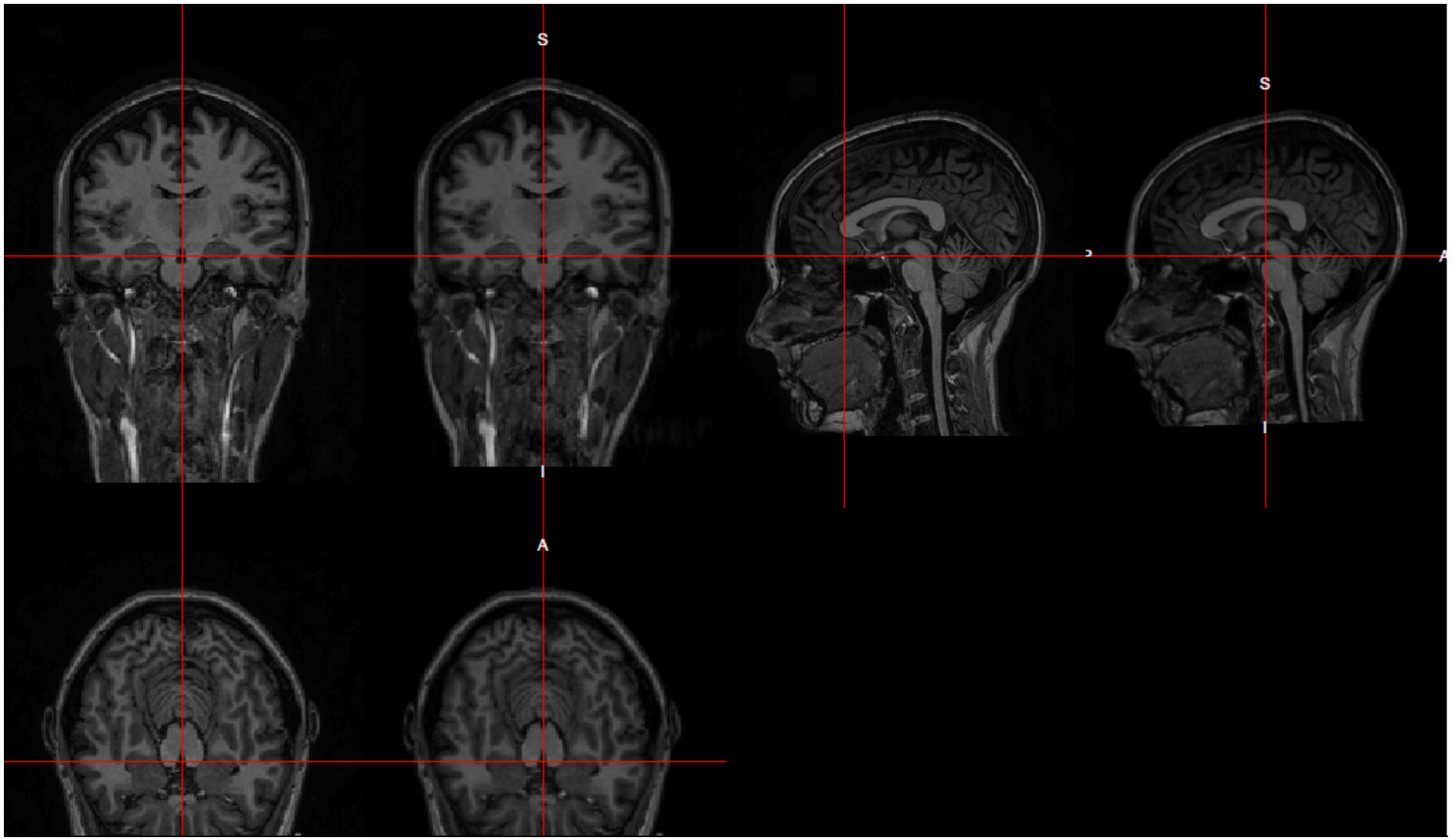
Across-Visit Co-Registration Skull-On Images

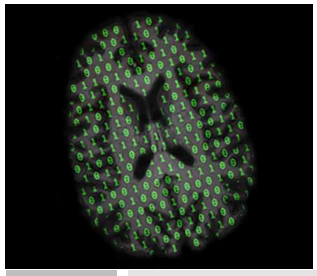
```
visit2_files_skull = file.path(mridir2,
                        c("113-02-MPRAGE_processed.nii.gz",
                        "113-02-T2w_processed.nii.gz",
                        "113-02-FLAIR_processed.nii.gz"))
outfiles2_skull = sub(".nii.gz", "_reg.nii.gz",
visit2_files_skull)

ants_regwrite(filename = visit2_files_skull[[1]],
              retimg = FALSE, outfile = outfiles2_skull[1],
              template.file = outfiles[[1]],
              other.files = visit2_files_skull[2:3],
              other.outfiles = outfiles2_skull[2:3],
              typeofTransform = "Rigid", verbose = FALSE)
```



T1 Skull-On Images Side-By-Side

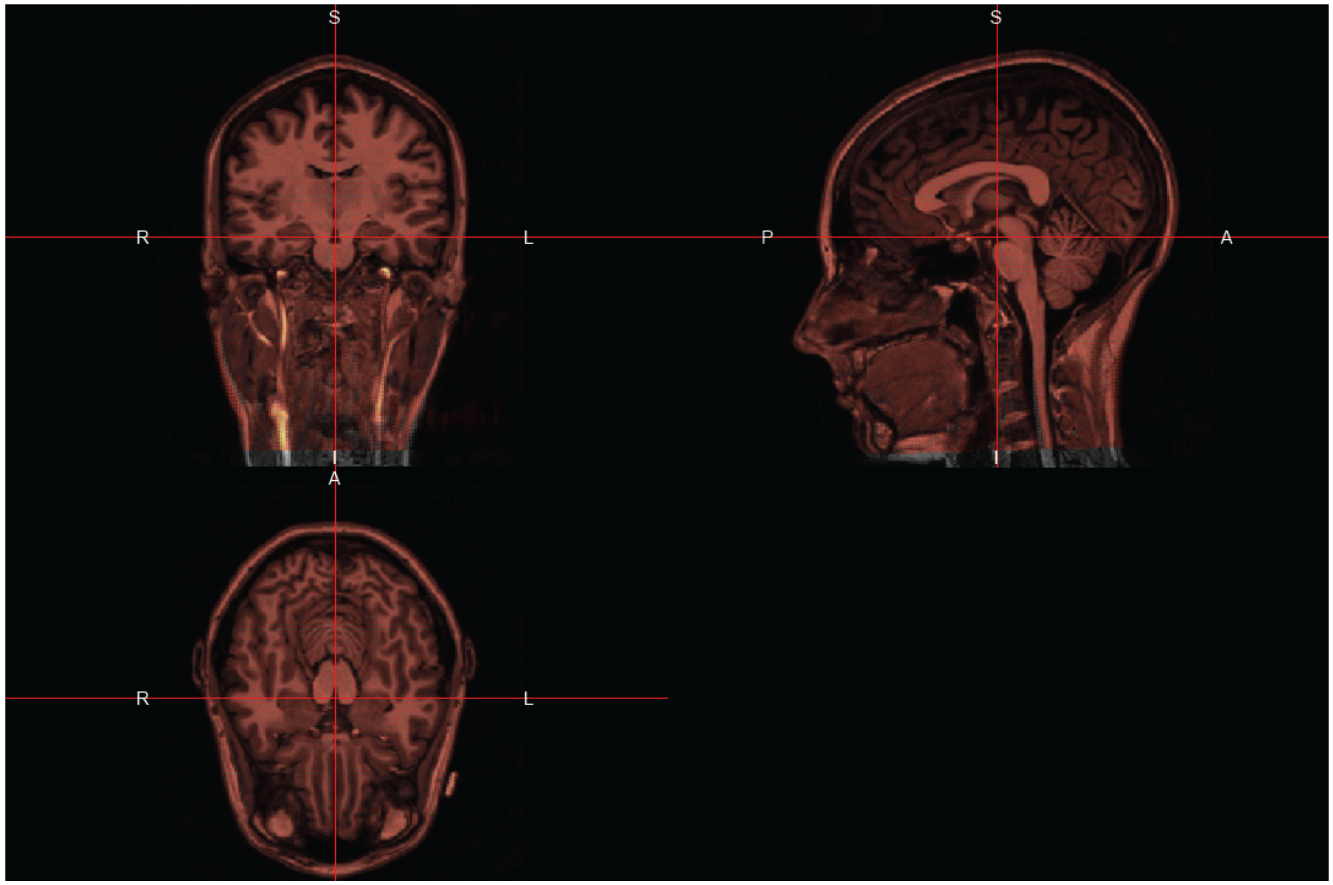
```
t1_skull=readNIfTI(outfiles[1],reorient=FALSE)
v2_t1_skull=readNIfTI(outfiles2_skull[1],reorient=FALSE)
double_ortho(t1_skull, v2_t1_skull)
```

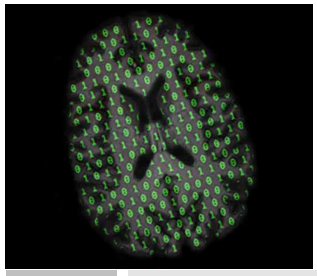




T1 Skull-On Images, Overlay

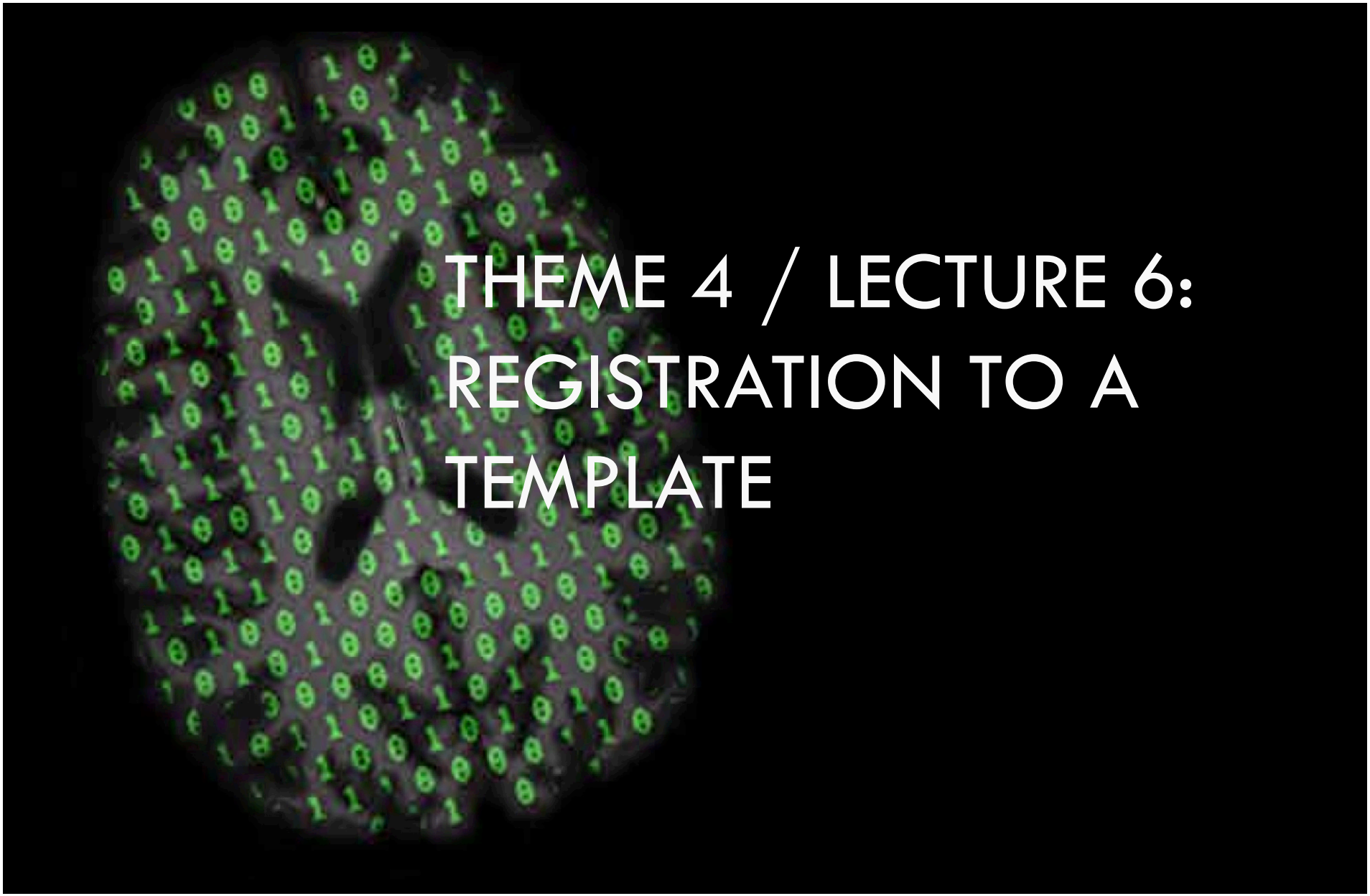
```
ortho2(t1_skull, v2_t1_skull,col.y=alpha(hotmetal(),0.25))
```



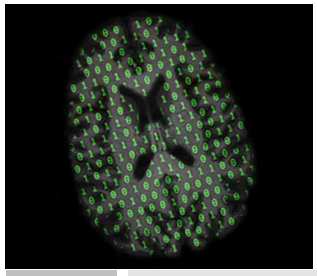


Overview

- Registration within a subject can be done in R
  - `ants_regwrite` wraps around the reading/writing of images and applying transformations
  - `double_ortho` and `ortho2` can provide some basic visual checks to assess registration quality
- Once images are registered in the same space, operations can be applied to all the images:
  - Masking with a brain mask
  - Transforming images to new spaces with one modality

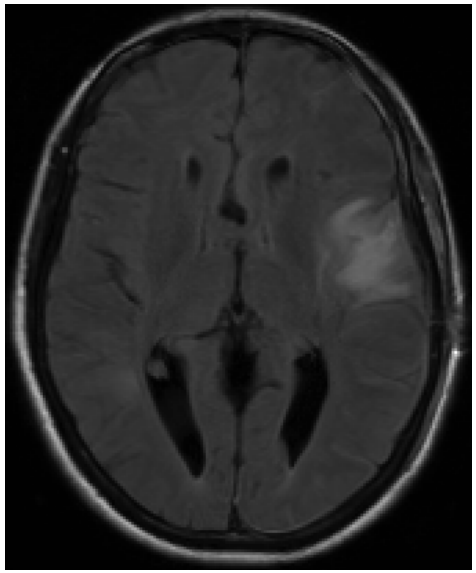


THEME 4 / LECTURE 6: REGISTRATION TO A TEMPLATE



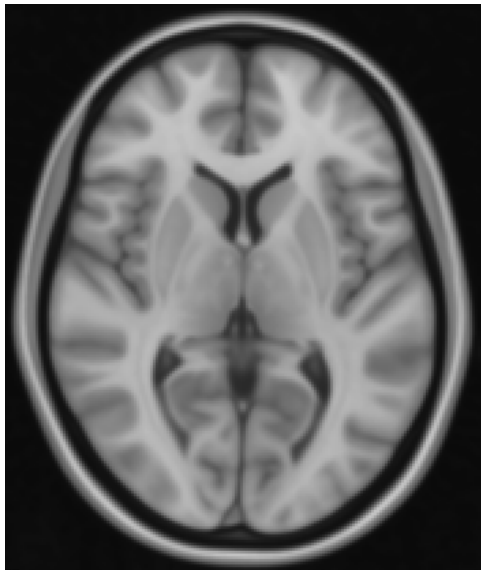
Registration to a Template

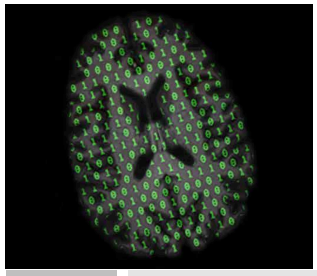
Warped FLAIR image



Warped ROI image

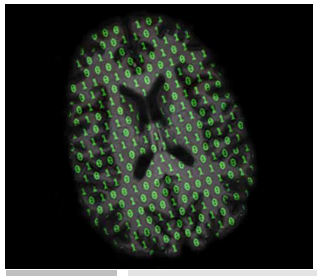
Eve/MNI T1 Template



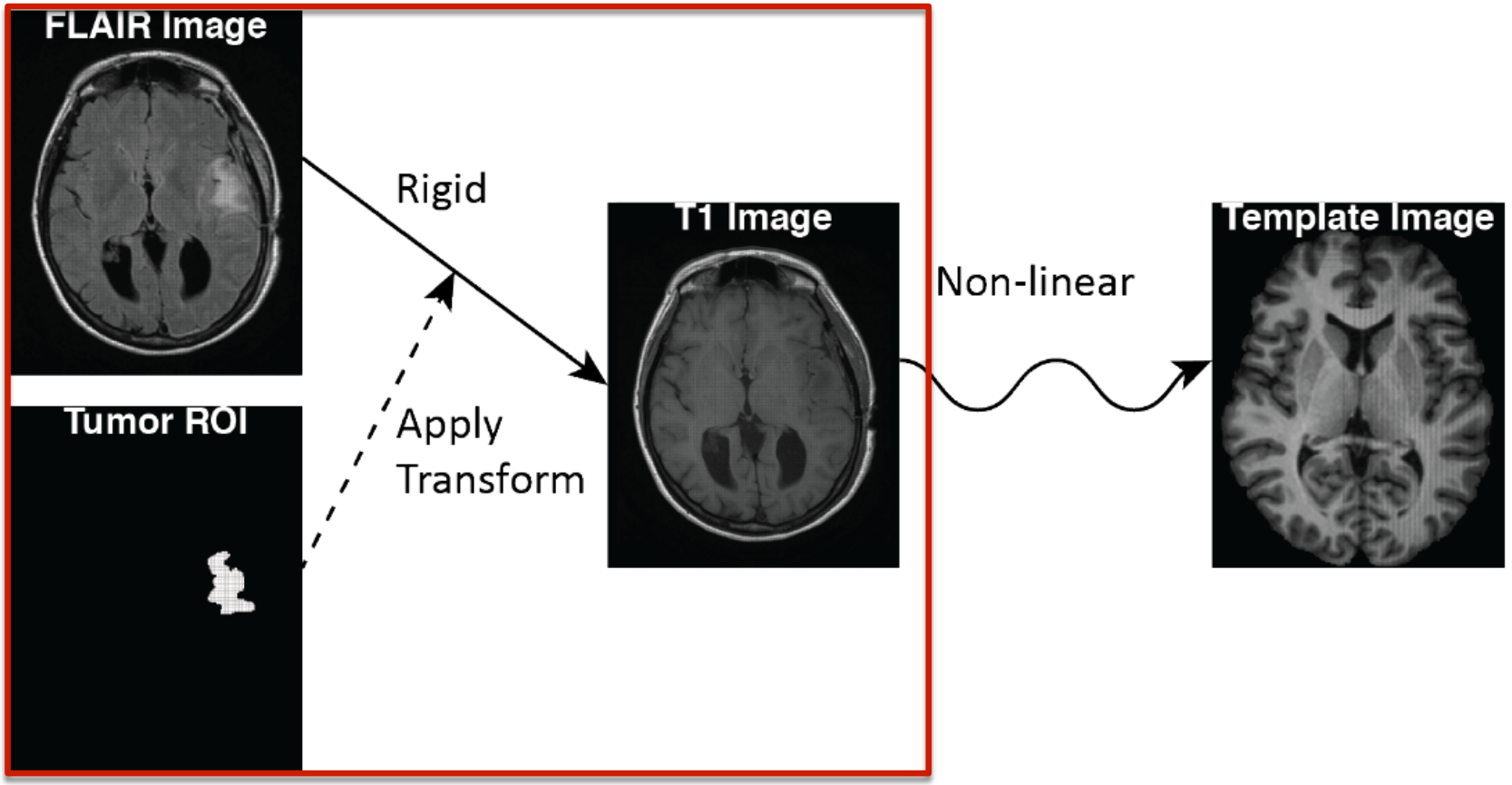


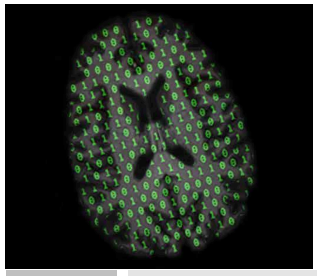
Types of Registration

- Complexity
  - rigid (6df)
  - affine (12df)
  - **nonlinear (>12df)**
- Co-registration (within the same person)
  - Cross-sectional between-modalities
  - Longitudinal within-modality
  - Longitudinal between-modalities
- **Registration to a template**
  - **A template image is necessary**
    - **MNI template stored in …/data/Template/MNI152_T1_1mm_brain.nii.gz**
    - **Eve template stored in …/data/Template/JHU_MNI_SS_T1.nii.gz**
  - **There are many different templates**
- One subject to another



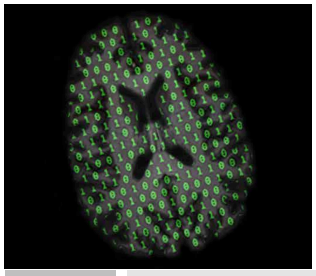
Overall Framework





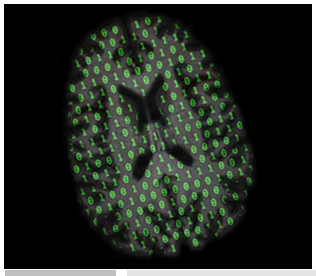
Registration to a Template

- Commonly requires an affine transformation
  - brain may be different sizes and in different spaces
  - translate/rotate as in rigid-body registration
  - scale up or down in size and shearing
- Affine transformation then non-linear transformation
- Usually achieves better local agreement
- Can change the volume of structures differentially



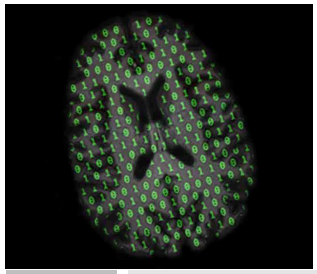
Reading in the T1 Scan from BRAINIX

```
library(oro.nifti)
library(extrantsr)
library(fslr)
library(scales)
neurodir <- "/home/fsluser/Desktop/MOOC-2015"
mridir = file.path(neurodir, "BRAINIX", "NIfTI")
t1 = file.path(mridir, "T1.nii.gz")
t1 = readNIfTI(t1, reorient = FALSE)
```

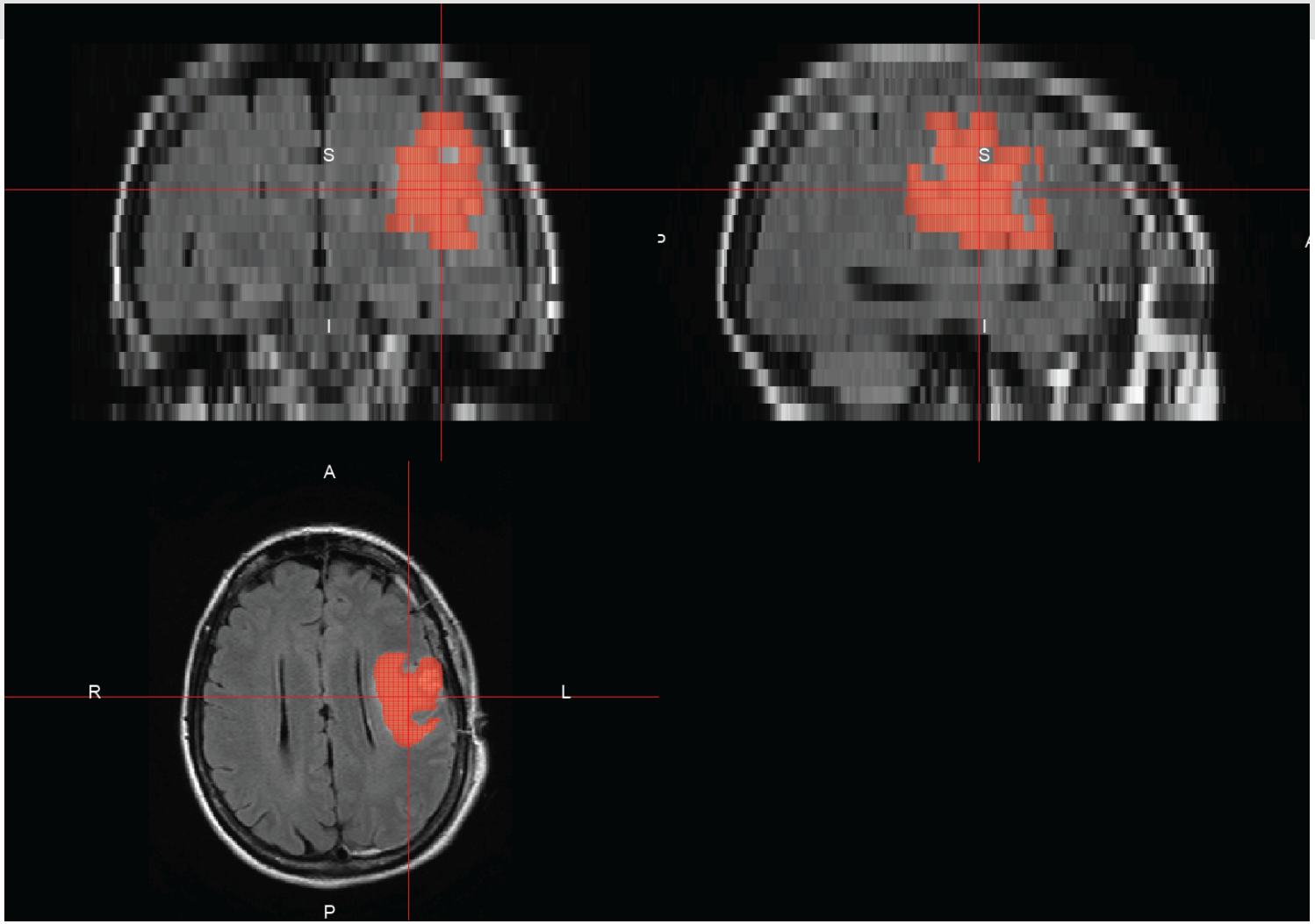


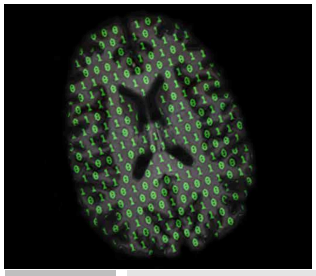
A Region of Interest from a Tumor on FLAIR

```
flair = file.path(mridir, "FLAIR.nii.gz")
roi = file.path(mridir, "ROI.nii.gz")

flair_file=readNIfTI(flair, reorient = FALSE)

roi_file=readNIfTI(roi, reorient = FALSE)
is_tumor<- (roi_file>0)
roi_file[!is_tumor]=NA

orthographic(flair_file,roi_file, xyz=c(200,155,12),
             col.y=alpha("red",0.2),
             text="Image overlaid with mask",
             text.cex = 1.5)
```



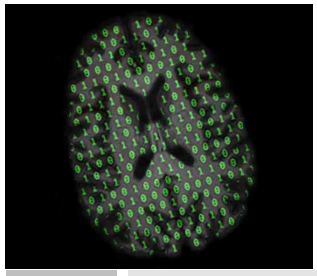
A Region of Interest from a Tumor on FLAIR





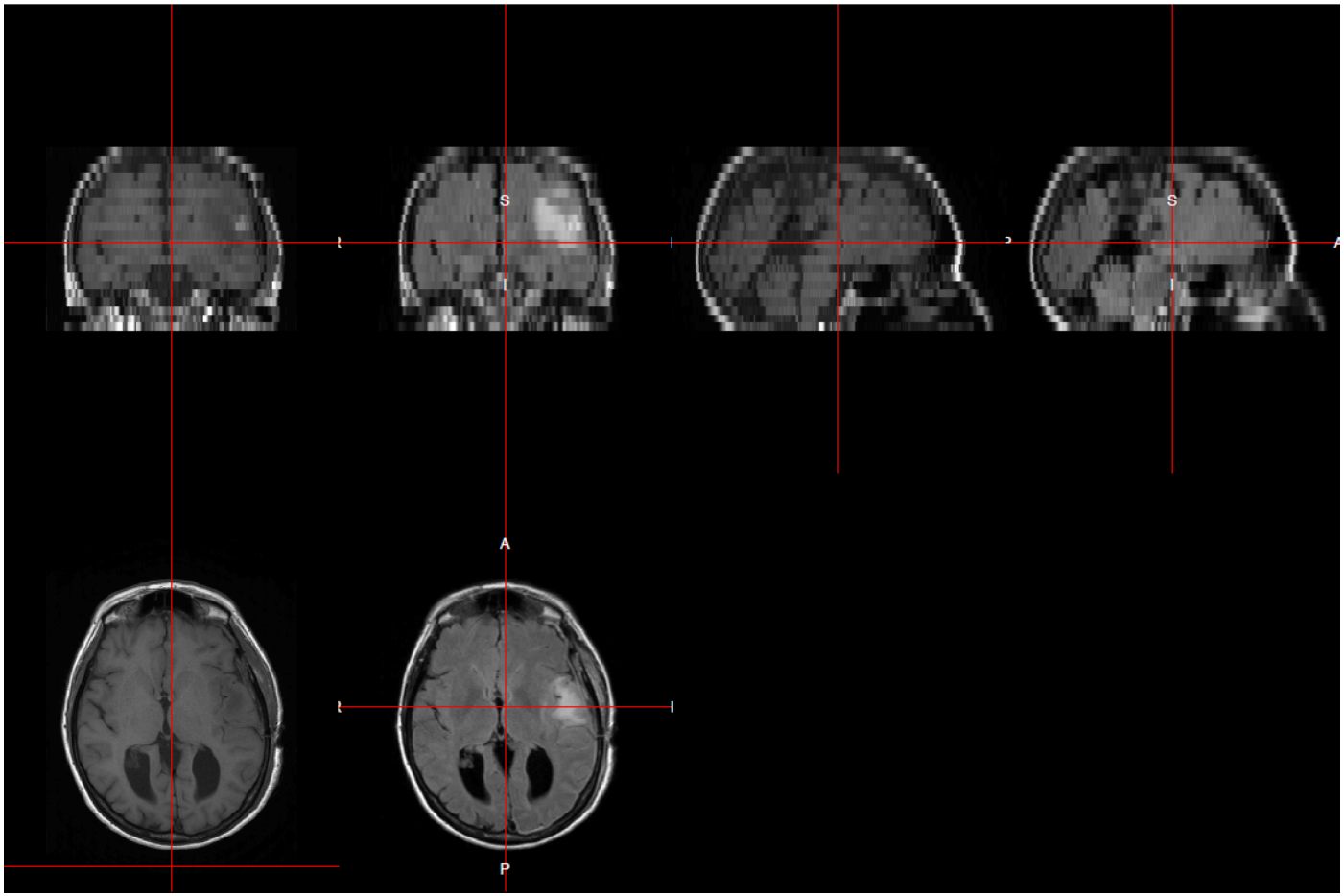
Rigid FLAIR to T1, Apply Transform to ROI

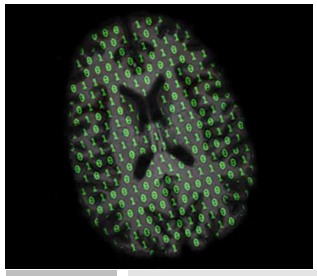
```
reg_flair = file.path(mridir,"FLAIR_regToT1.nii.gz")
reg_roi = file.path(mridir,"ROI_regToT1.nii.gz")

reg_flair_img = ants_regwrite(filename = flair,
                              template.file = t1,
                              outfile = reg_flair,
                              typeofTransform = "Rigid",
                              other.files = roi,
                              other.outfiles = reg_roi,
                              verbose = FALSE)

reg_roi_img = readNIfTI(reg_roi, reorient = FALSE)
```



FLAIR to T1 Registration Results

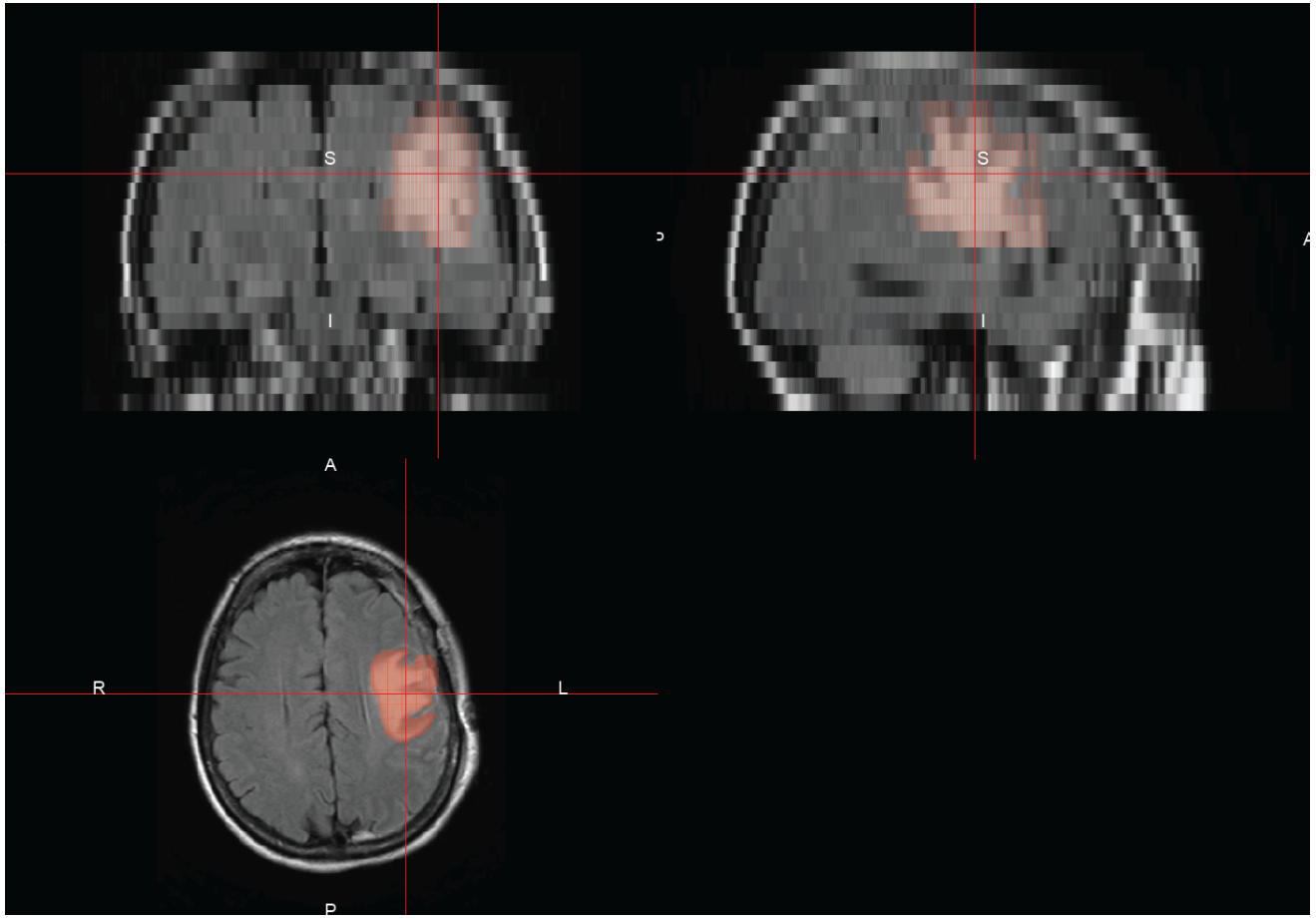
```
double_ortho(t1,reg_flair_img)
```

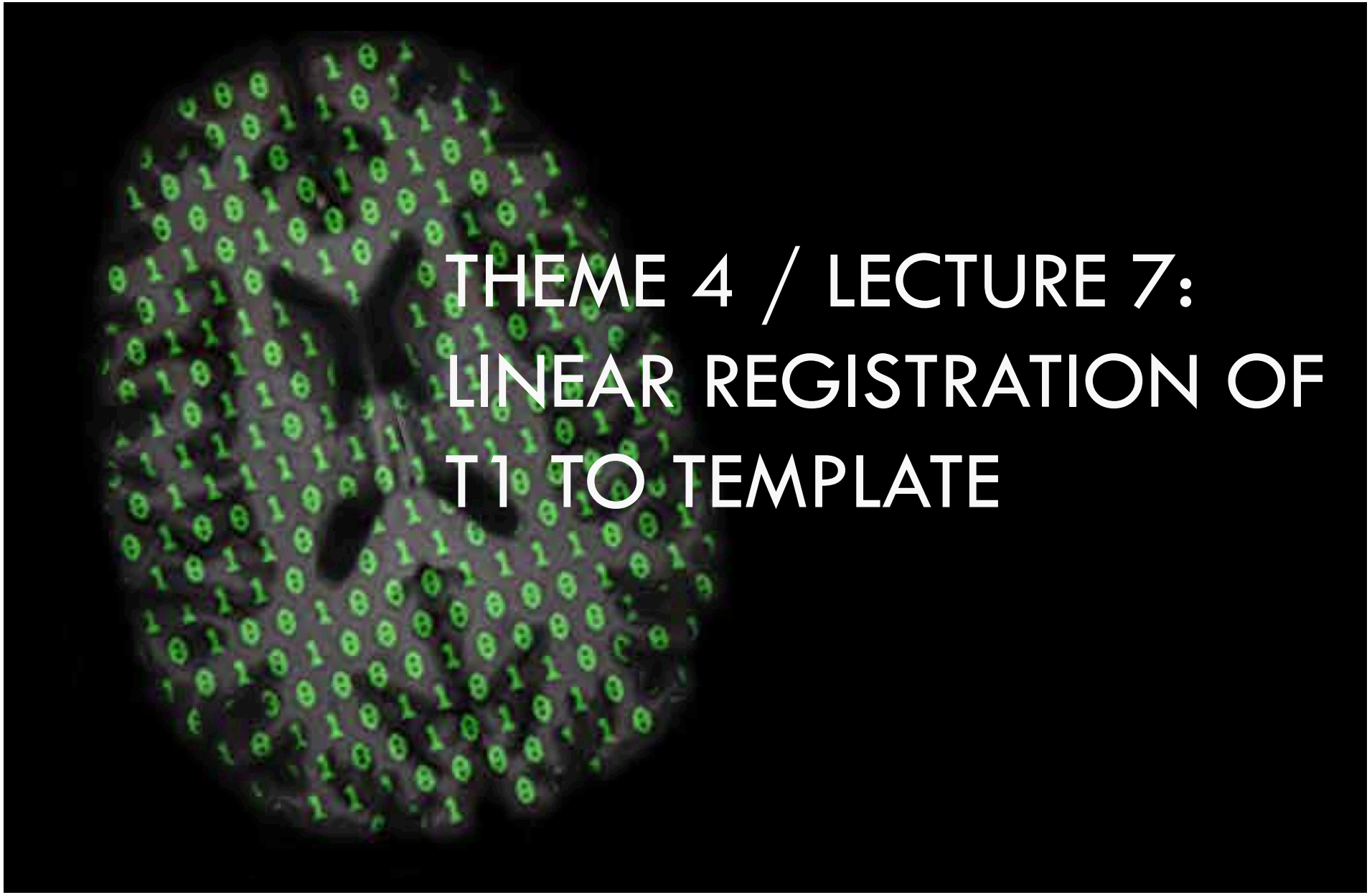




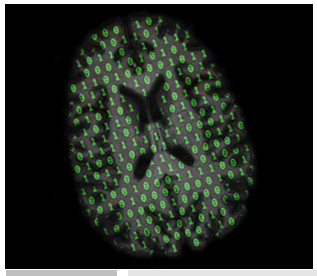
FLAIR to T1 Registration: ROI Overlay

```
ortho2(reg_flair_img,reg_roi_img,col.y=alpha("red",0.2))
```

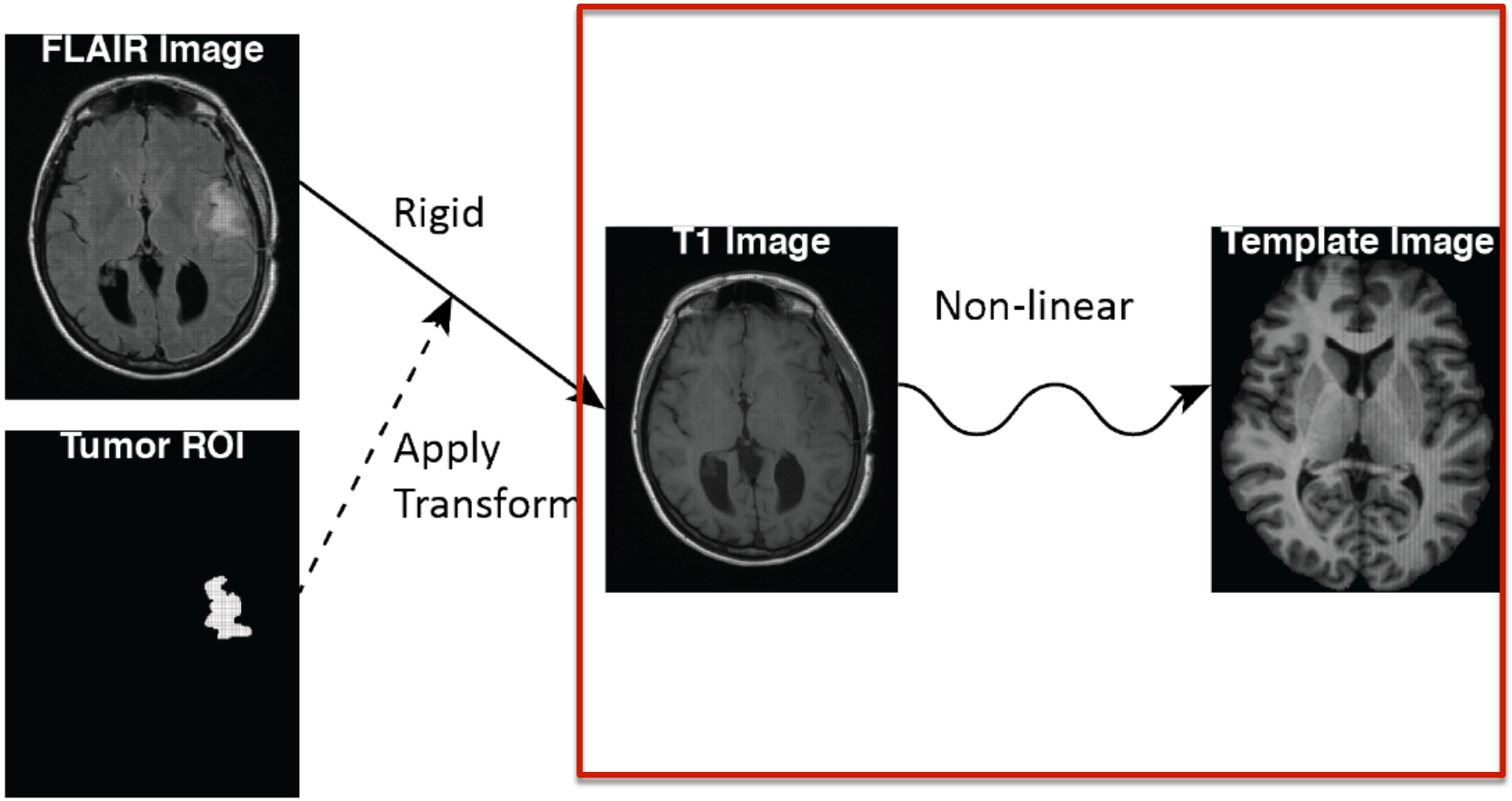


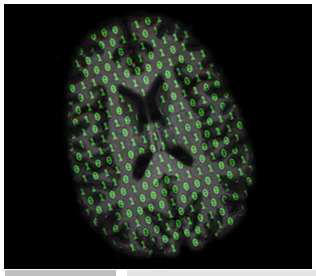


THEME 4 / LECTURE 7: LINEAR REGISTRATION OF T1 TO TEMPLATE



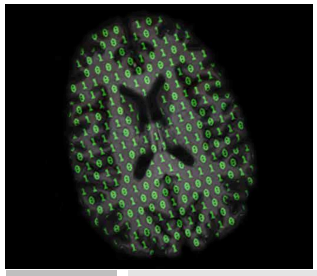
Linear Registration of T1 to Template





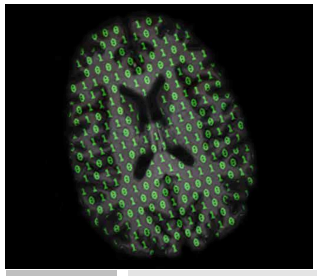
Skull Stripping and Inhomogeneity Correction

```
n4_t1 = bias_correct(t1, correction = "N4")

brain = fslbet_robust(img = n4_t1,
                      correct = FALSE, verbose = FALSE)
```



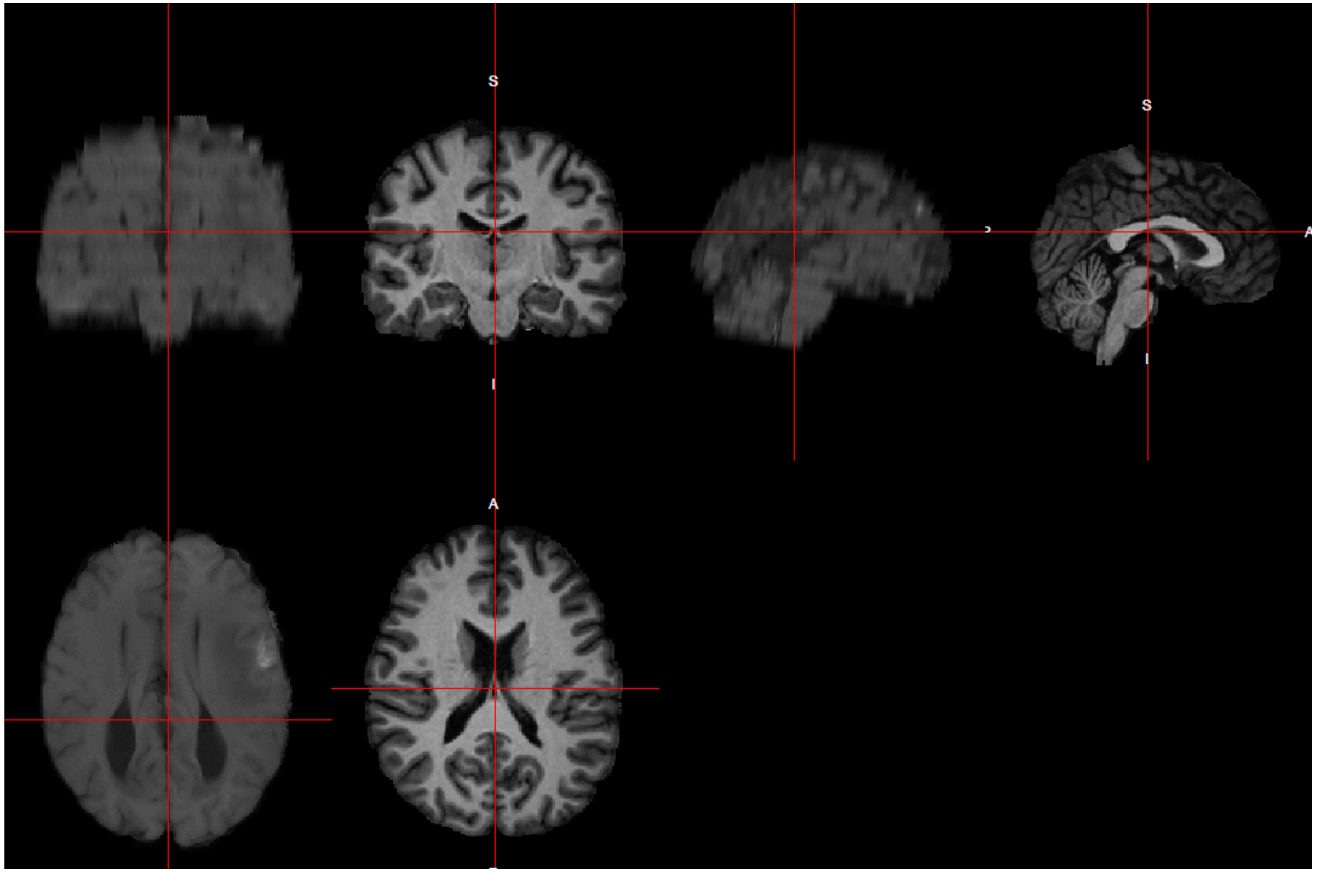
Affine Registration of T1 to the Eve Atlas

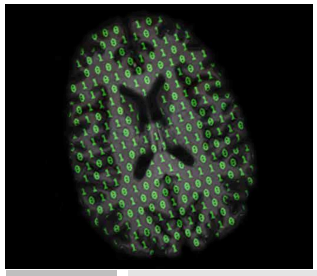
```
template.file = file.path(neurodir,
"Template","JHU_MNI_SS_T1_brain.nii.gz")
aff_t1_outfile = file.path(mridir,"T1_AffinetoEve.nii.gz")
aff_roi_outfile = file.path(mridir,
"ROI_regToT1_AffinetoEve.nii.gz")

aff_brain = ants_regwrite(filename = brain,
                          outfile = aff_t1_outfile,
                          other.files = reg_roi,
                          other.outfiles = aff_roi_outfile,
                          template.file = template.file,
                          typeofTransform = "Affine",
                          verbose = FALSE)

aff_roi = readNIfTI(aff_roi_outfile, reorient = FALSE)
```



Affine T1 Registration to Template Results

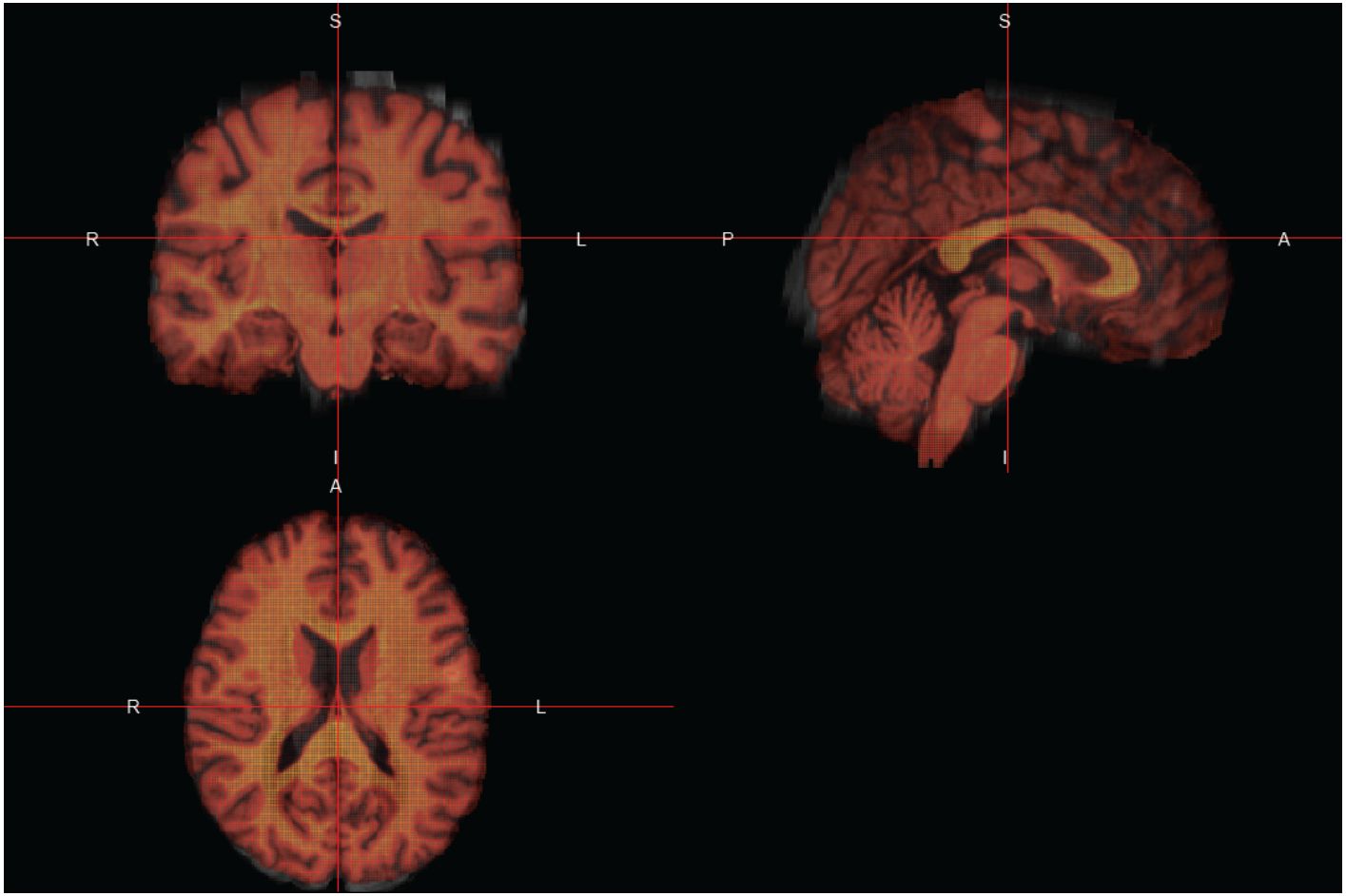
```
template = readNIfTI(template.file, reorient= FALSE)
double_ortho(aff_brain, template)
```

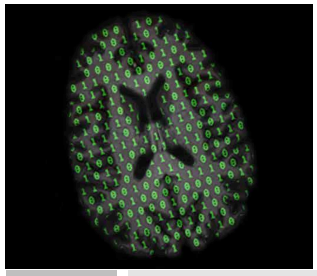




Affine T1 Registration to Template Results: Overlay

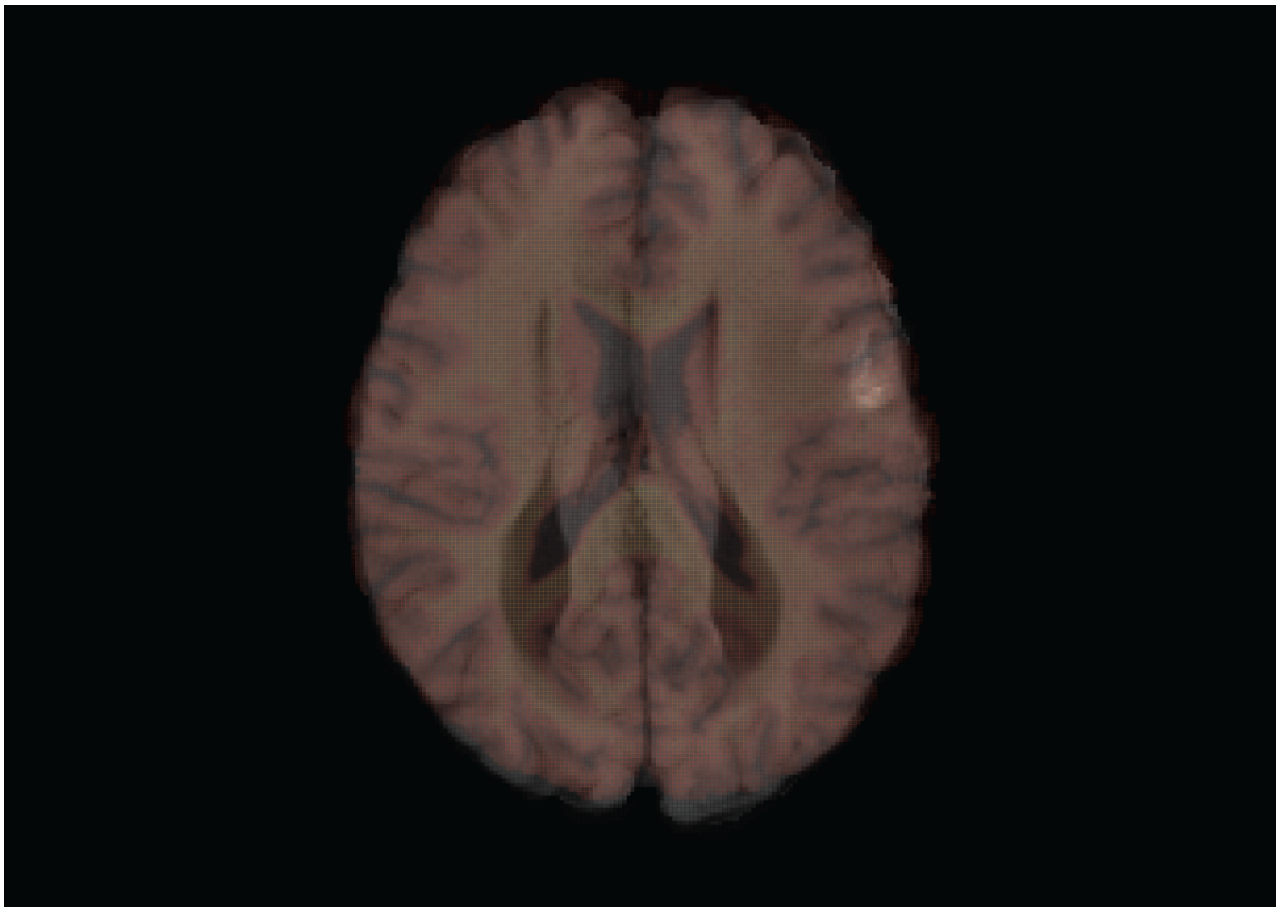
```
ortho2(aff_brain, template,col.y=alpha(hotmetal(),0.35))
```

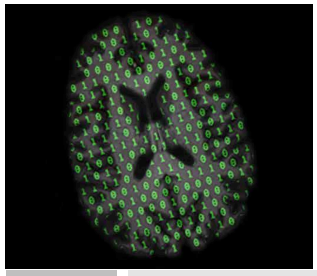




Affine T1 Registration to Template Results: Overlay One Slice

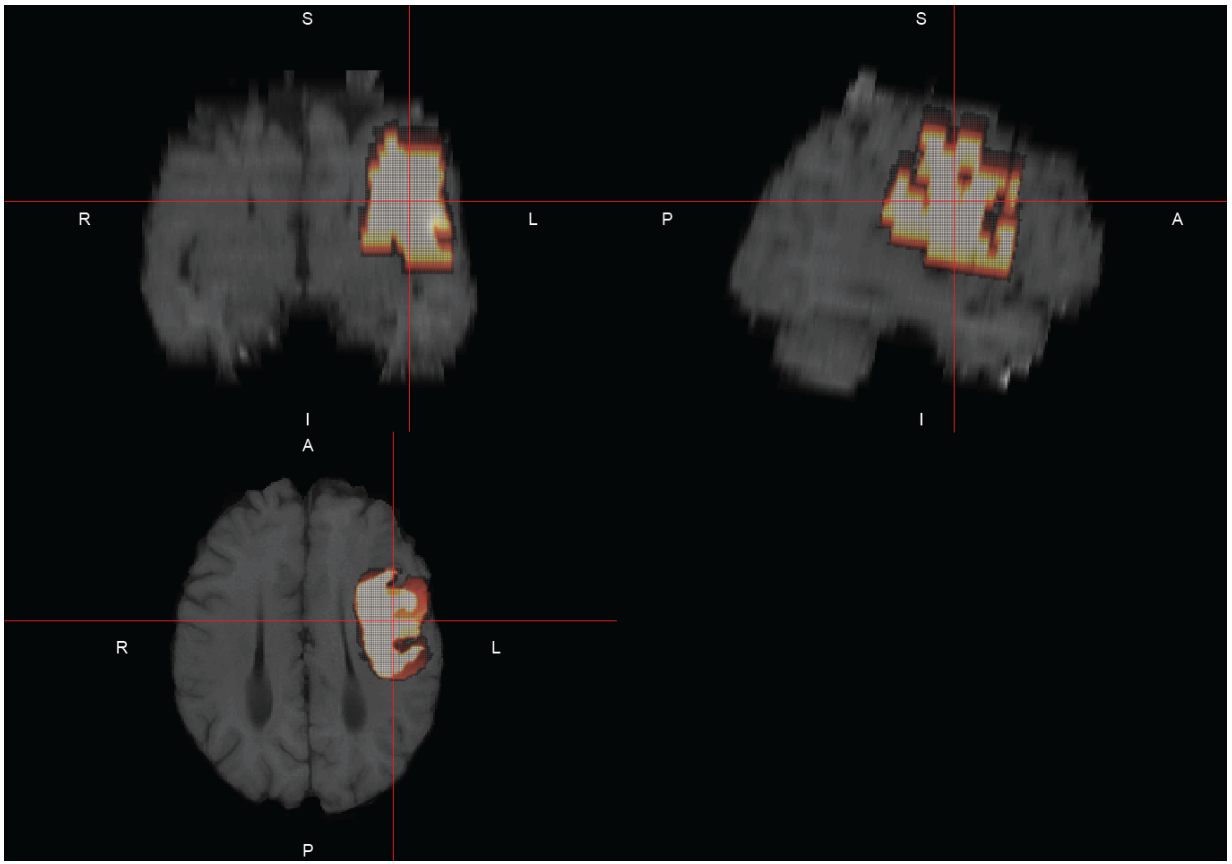
```
ortho2(aff_brain, template,z=ceiling(dim(template)[3]/2),
plot.type= "single",col.y=alpha(hotmetal(),0.35))
```

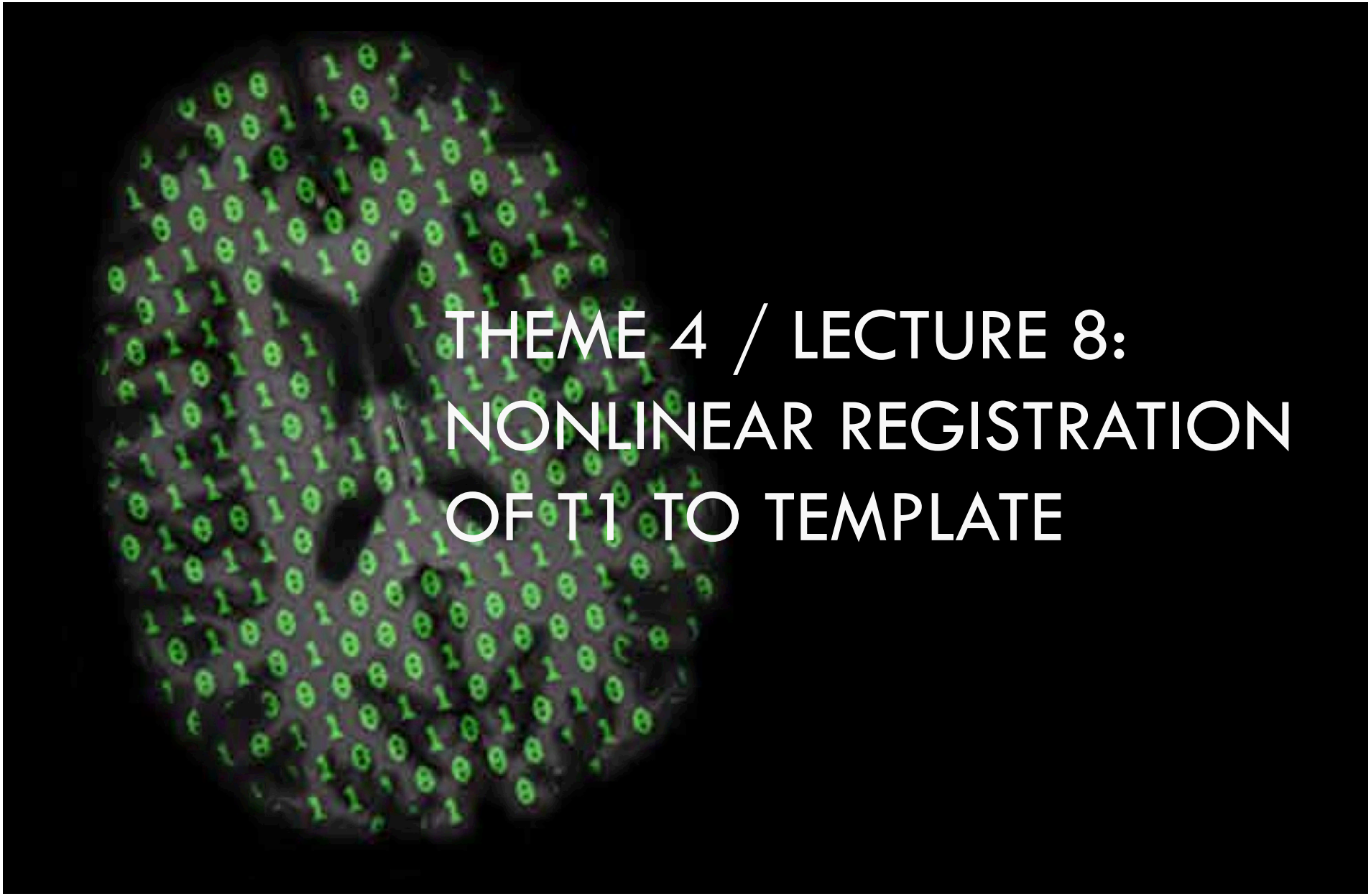




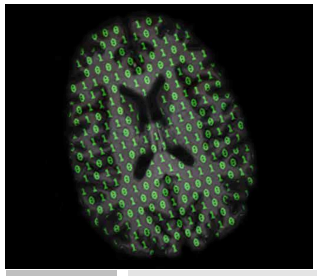
Affine T1 Registration to Template Results: ROI Overlay

```
ortho2(aff_brain, aff_roi,col.y=alpha(hotmetal(),0.35),
xyz=xyz(aff_roi))
```

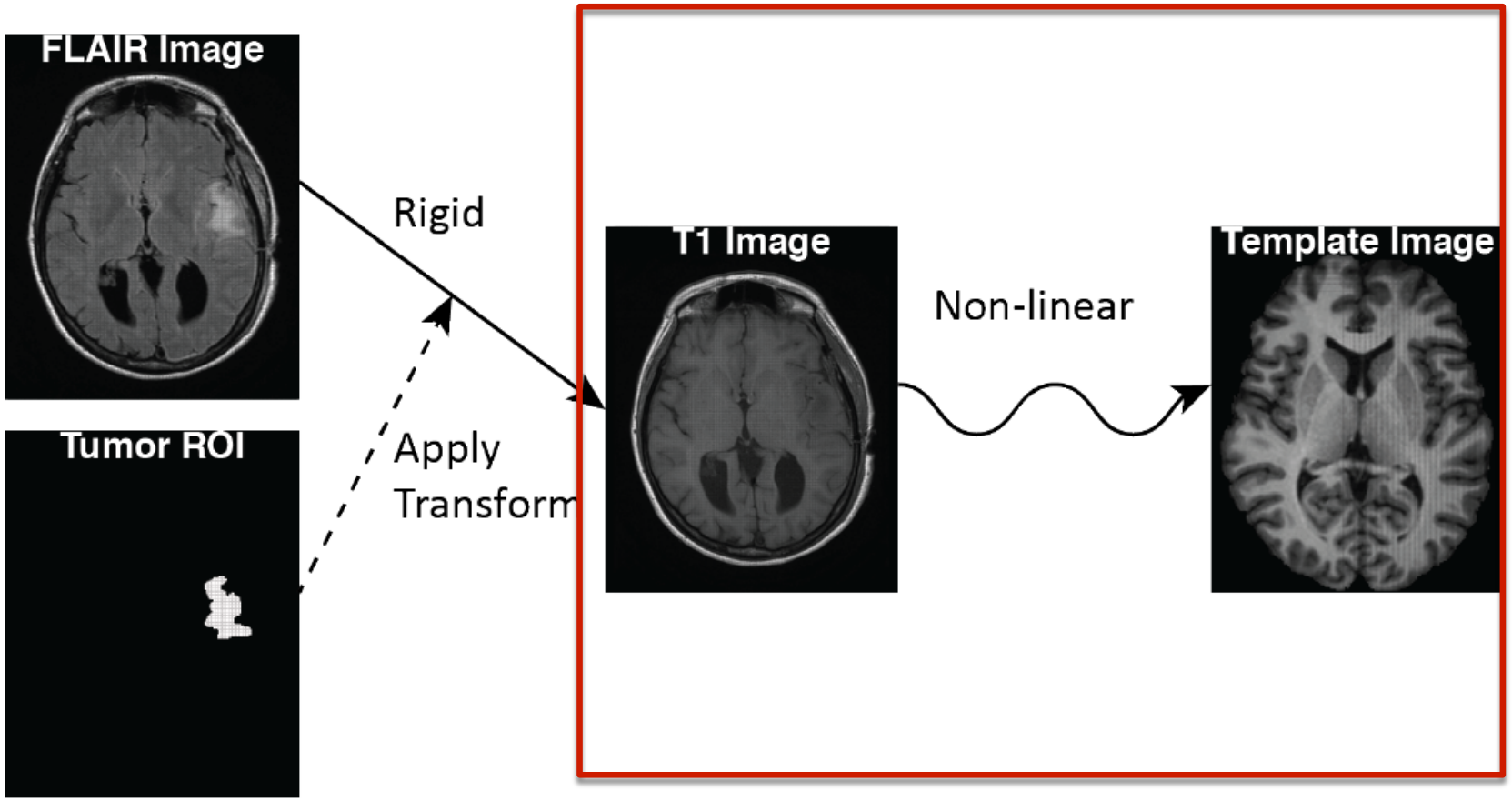


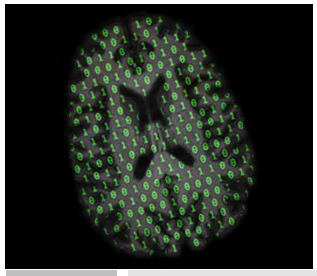


THEME 4 / LECTURE 8: NONLINEAR REGISTRATION OF T1 TO TEMPLATE



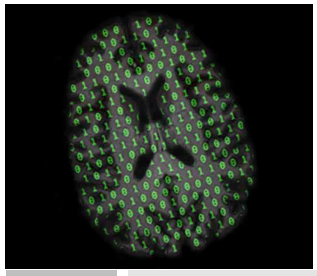
Nonlinear Registration of T1 to Template





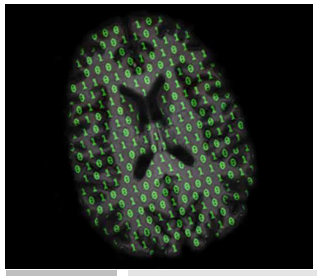
Non-linear Registration: SyN

- Using symmetric image normalization (SyN) (Avants, 2008)
  - A symmetric diffeomorphic registration technique
  - Registration is non-linear and matches the image to the template well
- SyN performs an affine registration first
  - We will use the skull-stripped image to the MNI T1 Brain template



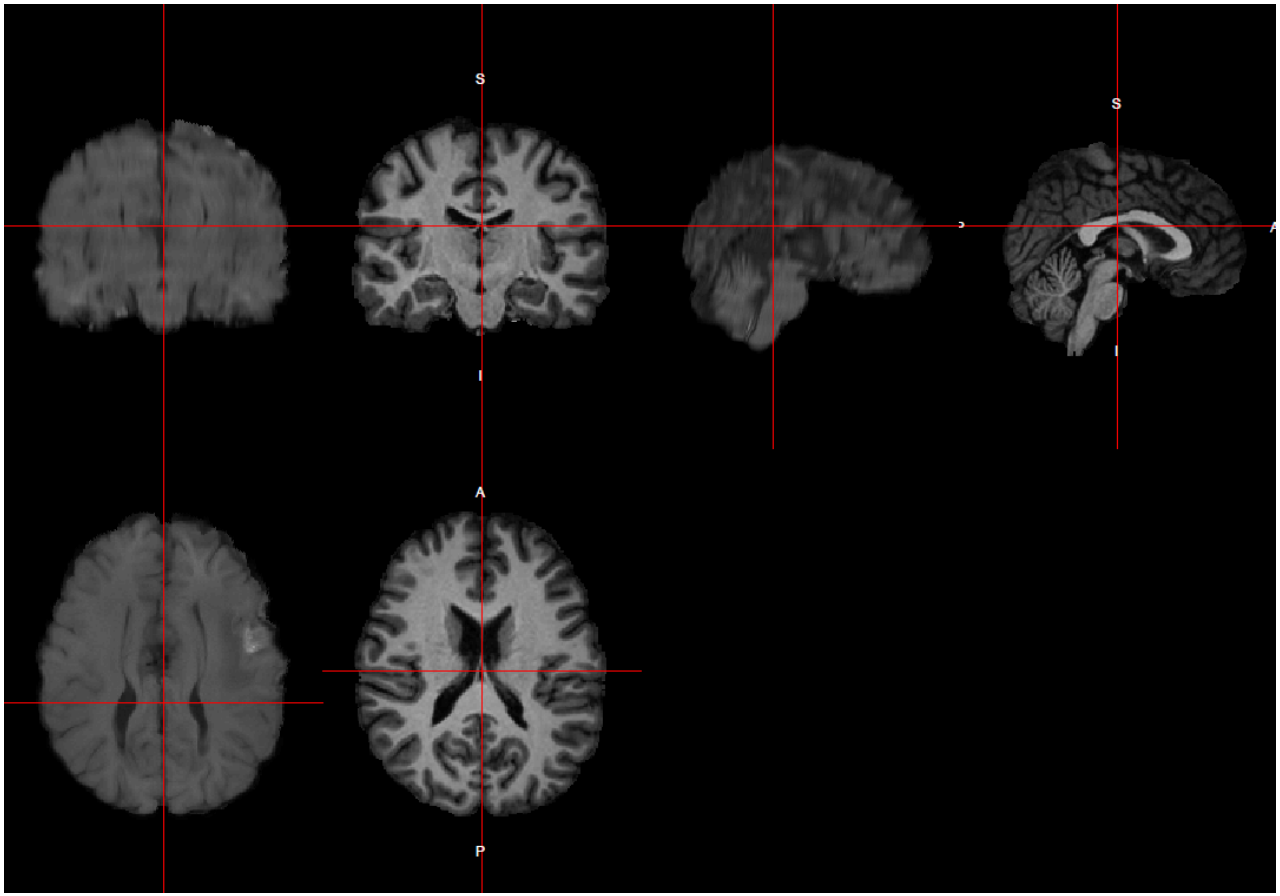
SyN T1 Registration to Template

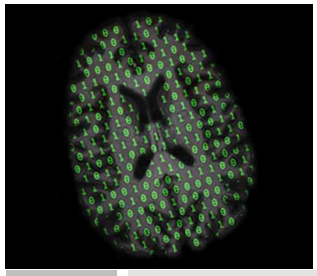
```
template.file = file.path(neurodir,
"Template","JHU_MNI_SS_T1_brain.nii.gz")
syn_t1_outfile = file.path(mridir,"T1_SyntoEve.nii.gz")
syn_roi_outfile = file.path(mridir,
"ROI_regToT1_SyntoEve.nii.gz")

syn_brain = ants_regwrite(filename = brain,
                          outfile = syn_t1_outfile,
                          other.files = reg_roi,
                          other.outfiles = syn_roi_outfile,
                          template.file = template.file,
                          typeofTransform = "SyN",
                          verbose = FALSE)

syn_roi = readNIfTI(aff_roi_outfile, reorient = FALSE)
```



SyN T1 Registration to Template Results

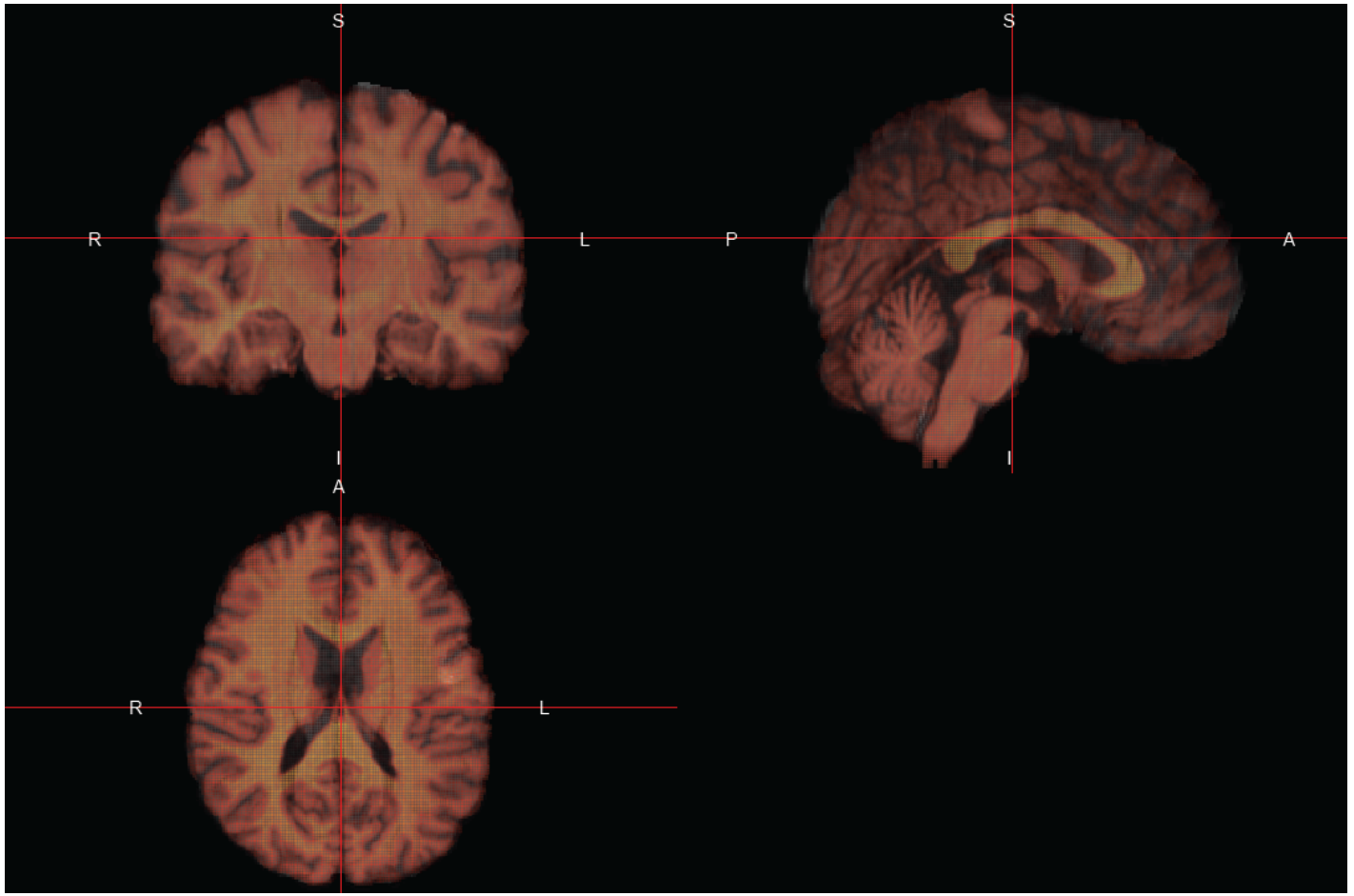
```
double_ortho(syn_brain, template)
```

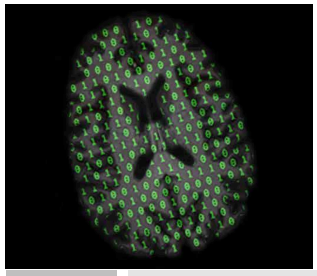




Syn T1 Registration to Template Results: Overlay

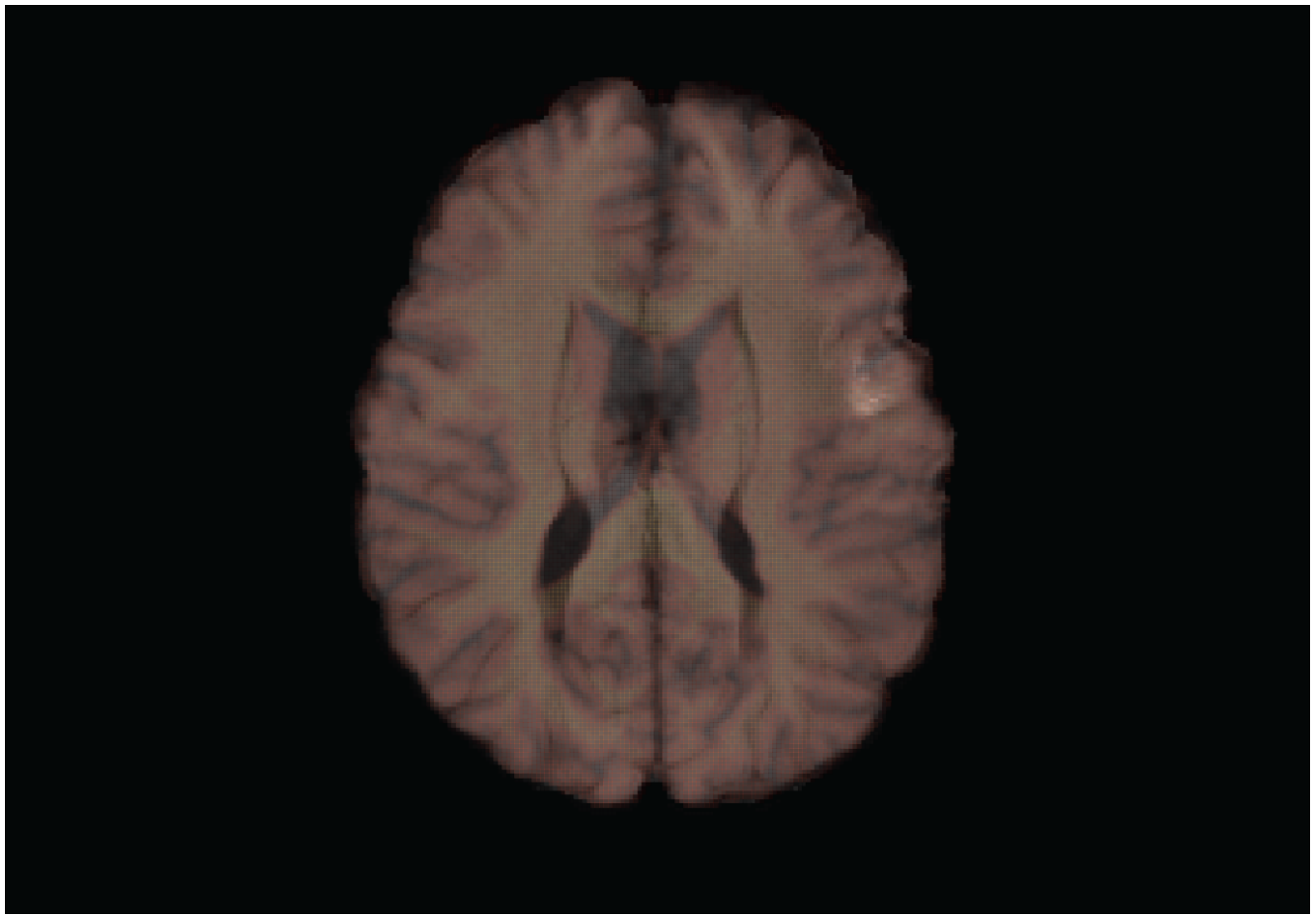
```
ortho2(syn_brain, template, col.y=alpha(hotmetal(),0.35))
```

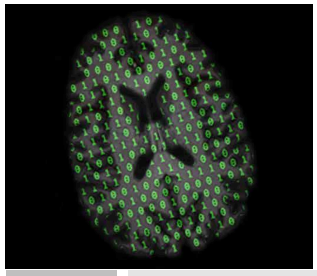




SyN T1 Registration to Template Results: Overlay One Slice

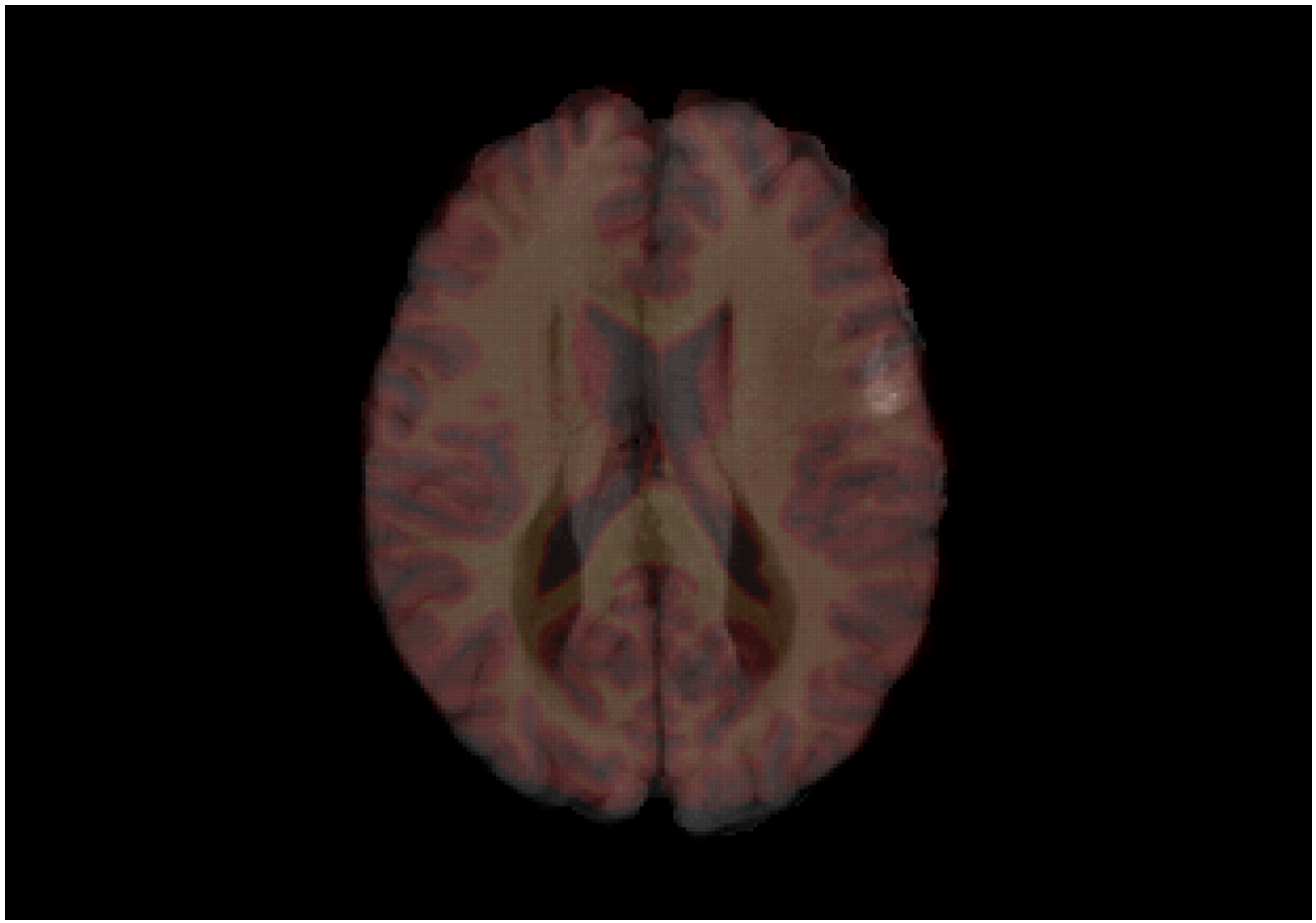
```
ortho2(syn_brain, template,z=ceiling(dim(template)[3]/2),
plot.type= "single",col.y=alpha(hotmetal(),0.35))
```

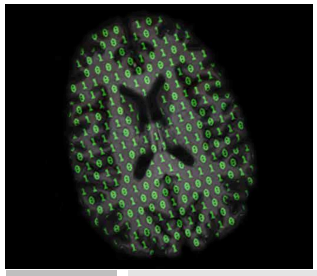




Affine T1 Registration to Template Results: Overlay One Slice

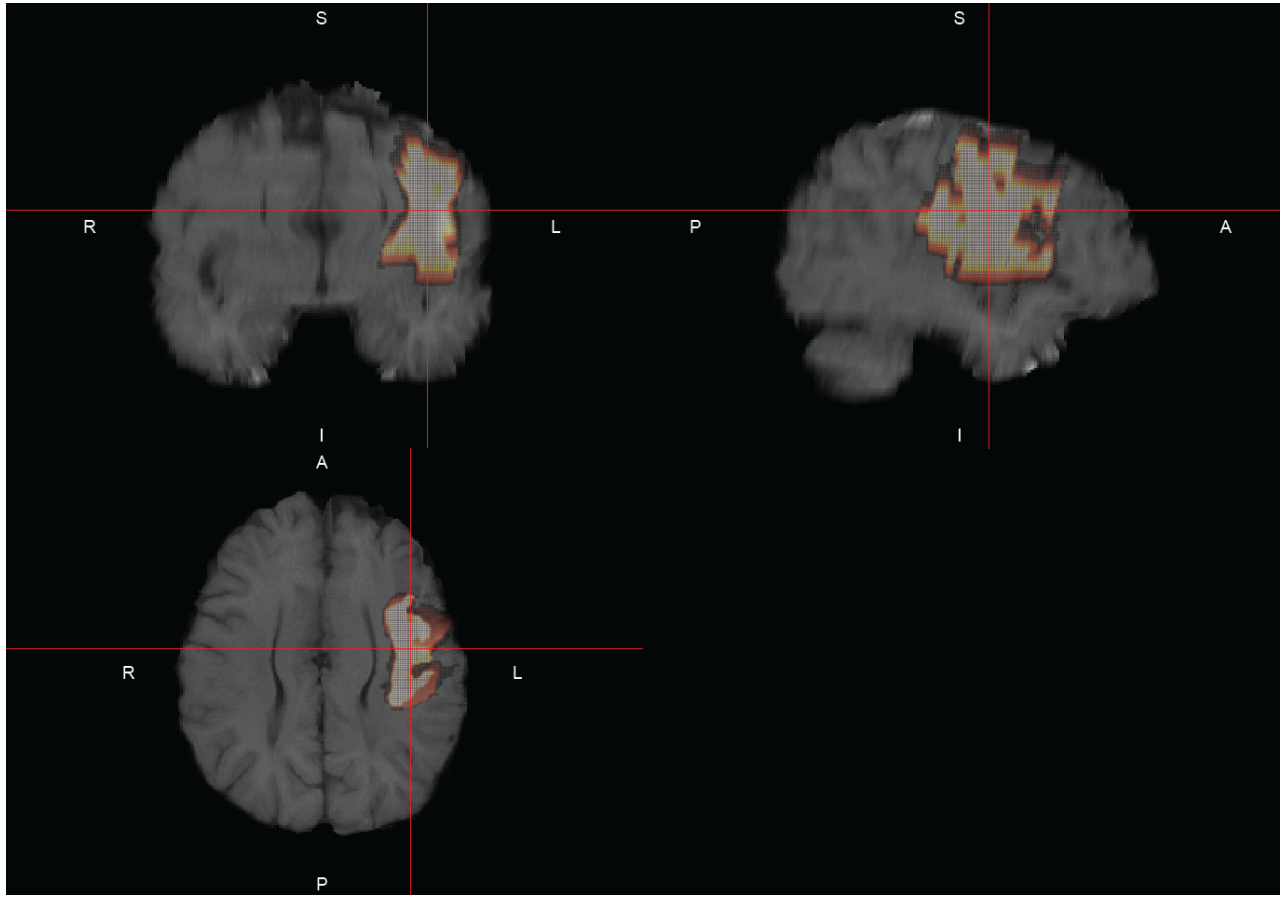
```
ortho2(aff_brain, template,z=ceiling(dim(template)[3]/2),
plot.type= "single",col.y=alpha(hotmetal(),0.35))
```

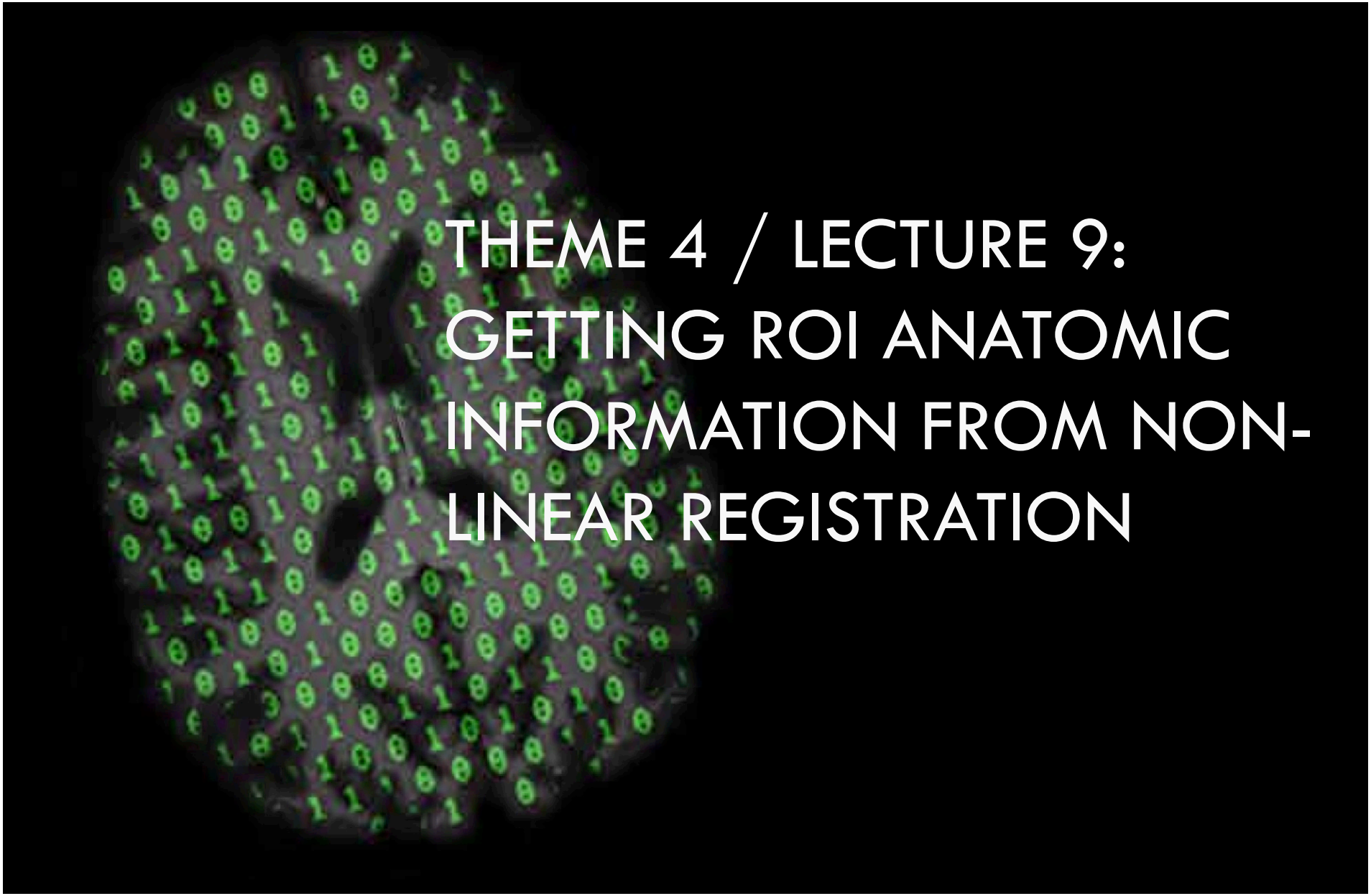




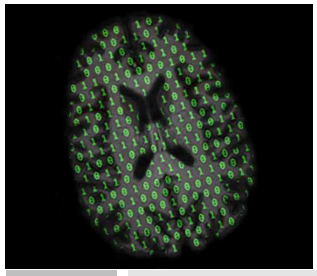
Affine T1 Registration to Template Results: ROI Overlay

```
ortho2(syn_brain, syn_roi, col.y=alpha(hotmetal(),0.35),
xyz=xyz(syn_roi))
```



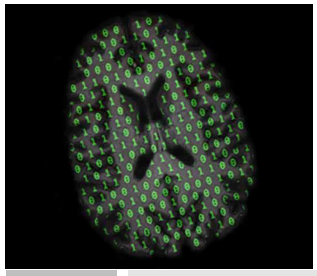


THEME 4 / LECTURE 9: GETTING ROI ANATOMIC INFORMATION FROM NON-LINEAR REGISTRATION



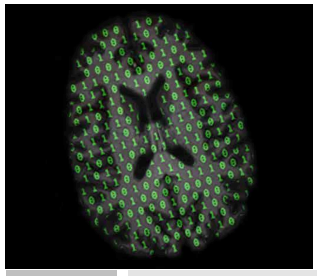
Getting ROI Anatomic Information from Non-Linear Registration

- We have the ROI in the template space
- We want to get information about its location
- This is possible if the atlas is labeled



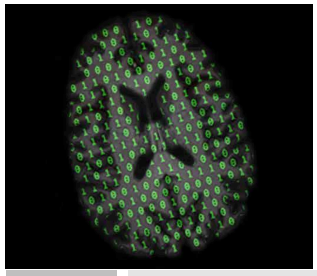
Reading in the Atlas and Look Up Table (LUT)

```
#### extracting JHU Eve atlas Type I and labels###########
atlas = "JHU_MNI_SS_WMPM_Type-I"
txtfile = file.path(neurodir, "Template",
                    paste0(atlas, "_SlicerLUT.txt"))

### read look up table (LUT)
jhut1.df = read.table(txtfile, stringsAsFactors=FALSE)
jhut1.df = jhut1.df[, 1:2]
colnames(jhut1.df) = c("index", "Label")
jhut1.df$index = as.numeric(jhut1.df$index)

jhut1.df[1:4,]
index                          Label
1     0                     Background
2     1 SUPERIOR_PARIETAL_LOBULE_left
3     2          CINGULATE_GYRUS_left
4     3   SUPERIOR_FRONTAL_GYRUS_left
```



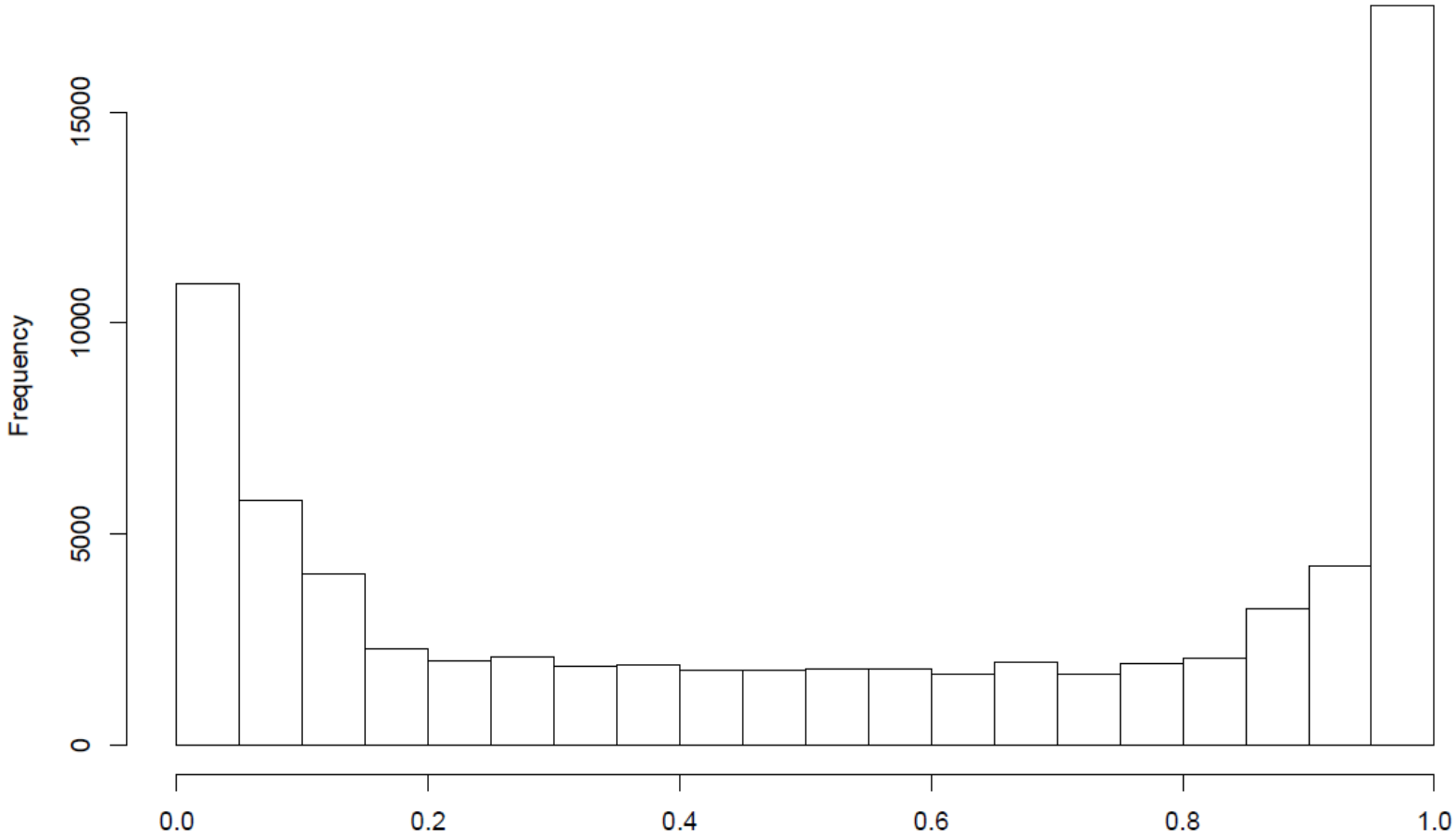
Reading in the Atlas and Look Up Table (LUT)

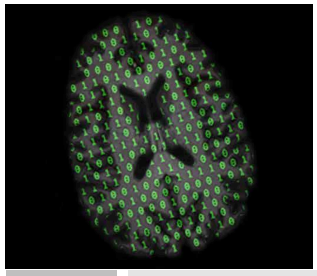
```
## read in the template image
jhut1.img = readNIfTI(file.path(neurodir, "Template",
                        paste0(atlas, ".nii.gz")))

##Obtain the numeric labels from the atlas
uimg = sort(unique(c(jhut1.img)))

##Obtain the numeric labels from the LUT
all.ind = jhut1.df$index

##Check that all numeric labels from the atlas are in LUT
stopifnot(all(uimg %in% all.ind))
```



After Warping and Interpolation, the ROI is No Longer Binary

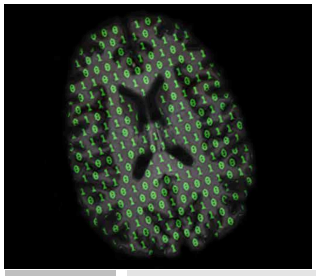
```
hist(c(syn_roi[syn_roi > 0]))
```





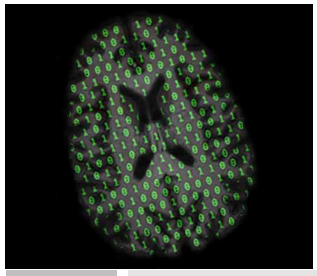
Ways of Dealing with Non-Binary ROI

- Threshold the ROI to binary
- Use a weighted sum over the ROI



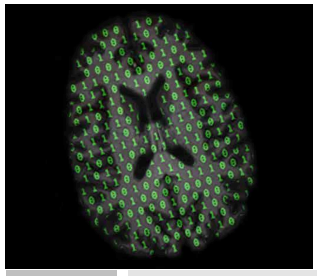
Quantifying the ROI Engagement by Region

```
library(plyr)

##Make a data frame with the index of the atlas
##and the value of the ROI at that voxel
roi.df = data.frame(index = jhut1.img[syn_roi> 0],
                    roi = syn_roi[ syn_roi > 0])

##Obtain the number (sum) of voxels that have an roi
##value >0.5 in the roi by the index of labels
label_sums = ddply(roi.df, .(index), summarize,
sum_roi = sum(roi), sum_roi_thresh = sum(roi > 0.5))
label_sums = merge(label_sums, jhut1.df, by="index")
```



ROI engagement: Tumor by Region

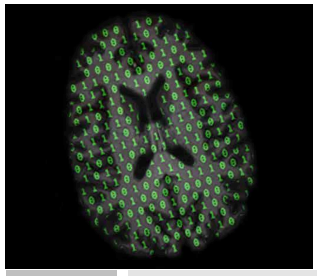
```
sums = label_sums # will use later

# Assign the Labels to the row names
rownames(label_sums) = label_sums$Label
label_sums$Label = label_sums$index = NULL

# Reorder labels from the most to the least engaged
label_sums = label_sums[order(label_sums$sum_roi,
                              decreasing = TRUE), ]
```



ROI Engagement Results: Top 10 Regions

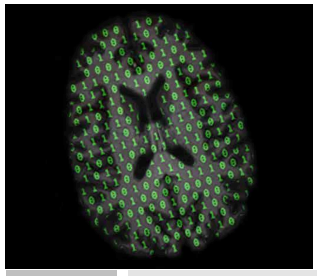
```
# Calculate the percent of the tumor engaging the region
label_pct = t(t(label_sums)/colSums(label_sums)) * 100
head(round(label_pct, 1), 10)
```

	sum_roi	sum_roi_thresh
PRECENTRAL_GYRUS_left	15.6	15.6
PRECENTRAL_WM_left	12.9	12.9
SUPERIOR_TEMPORAL_GYRUS	9.4	9.9
INSULAR	9.2	9.7
INFERIOR_FRONTAL_WM_left	7.4	7.5
Superior_longitudinal_fasciculus_left	7.2	7.3
POSTCENTRAL_GYRUS_left	6.9	6.9
INFERIOR_FRONTAL_GYRUS_left	6.0	6.2
SUPRAMARGINAL_GYRUS	5.7	6.0
MIDDLE_FRONTAL_GYRUS_left	4.8	4.3



ROI Engagement: Region by Tumor

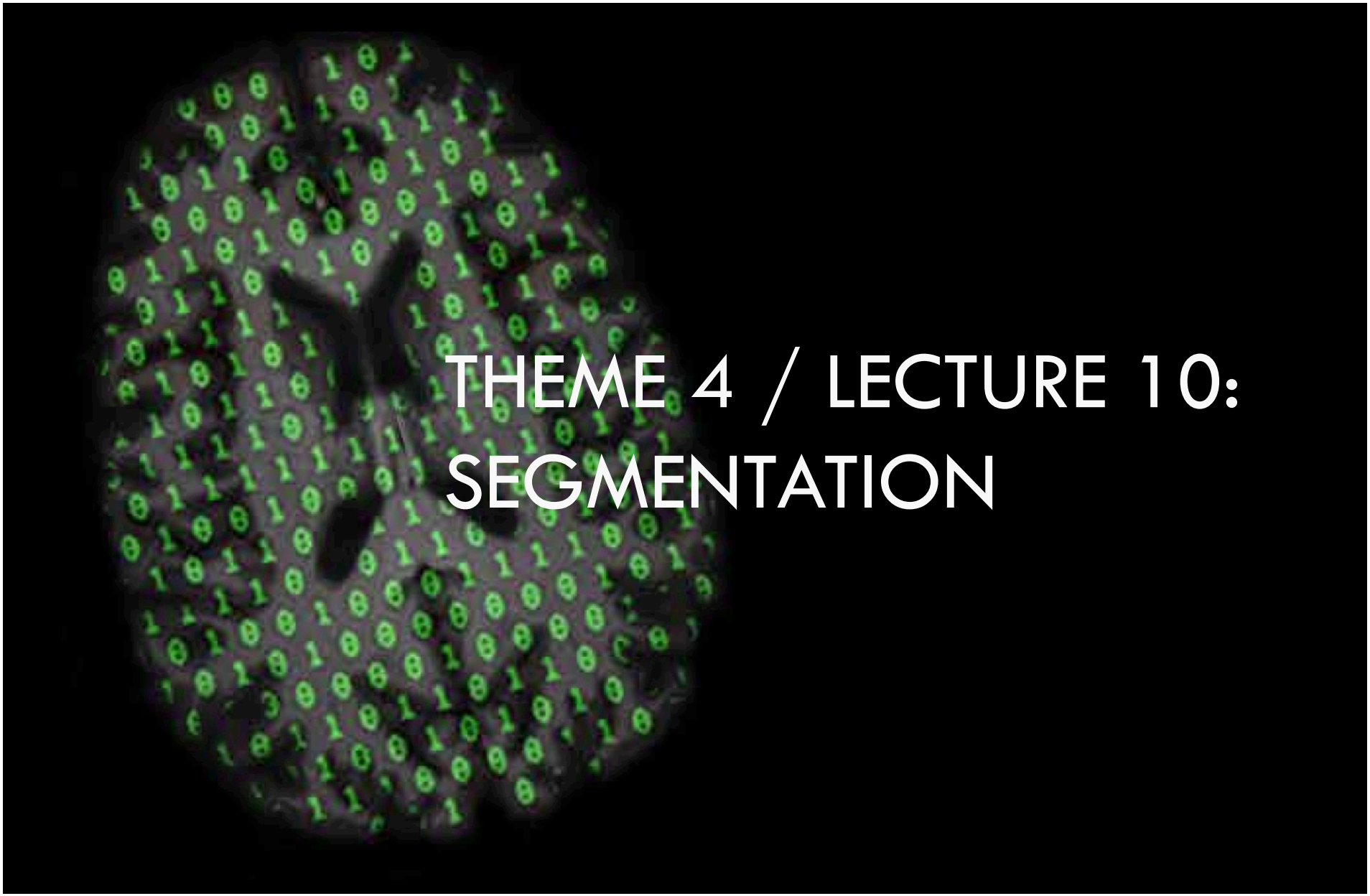
```
# Create a table that contains the number of voxels
#engaged in each labeled region
jhut1.tab = as.data.frame(table(c(jhut1.img)))
colnames(jhut1.tab) = c("index", "size")

# Merge the table of number of voxels per region with the
# table of number of voxels by ROI
region_pct = merge(sums, jhut1.tab, by="index")
rownames(region_pct) = sums$Label

# Calculate the percent of region engaged by the tumor
region_pct$Label = region_pct$index = NULL
region_pct = region_pct/region_pct$size * 100
region_pct$size = NULL

# Reorder regions from the most to the least engaged
region_pct = region_pct[ order(region_pct$sum_roi,
decreasing = TRUE), ]
```



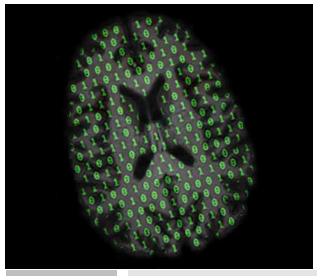
ROI Engagement Results: Top 10 Regions

```
# Calculate the percent of the tumor engaging the region
head(round(region_pct,1),10)
```

	sum_roi	sum_roi_thresh
Superior_longitudinal_fasciculus_left	46.3	46.4
PRECENTRAL_WM_left	34.9	34.5
INSULAR	31.2	32.4
External_capsule_left	27.5	27.5
INFERIOR_FRONTAL_WM_left	25.4	25.4
PRECENTRAL_GYRUS_left	23.8	23.6
SUPRAMARGINAL_GYRUS	12.6	13.0
SUPERIOR_TEMPORAL_GYRUS	11.3	11.6
INFERIOR_FRONTAL_GYRUS_left	11.1	11.3
PUTAMEN_left	11.1	10.2

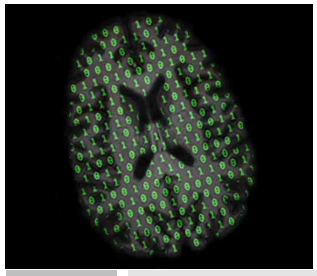


THEME 4 / LECTURE 10: SEGMENTATION



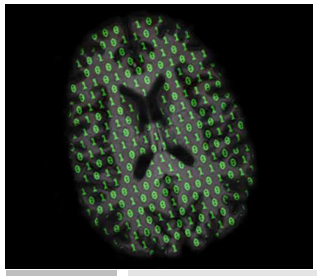
Segmentation

- We show how to use fslr to
  - segment CSF, GM, and WM
  - calculate the volume of CSF, GM, and WM



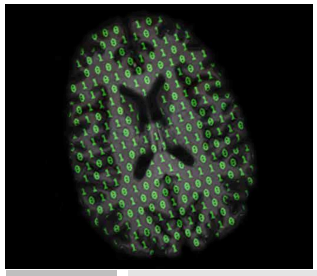
Functions Used in Segmentation

- `readNIfTI`: passes files as NIfTI objects
- `bias_correct`: wraps ANTsR bias field corrections
- `fslbet_robust`: performs skull-stripping
- `fast`: calls fast from FSL, does segmentation
- `ortho2` - does orthographic display



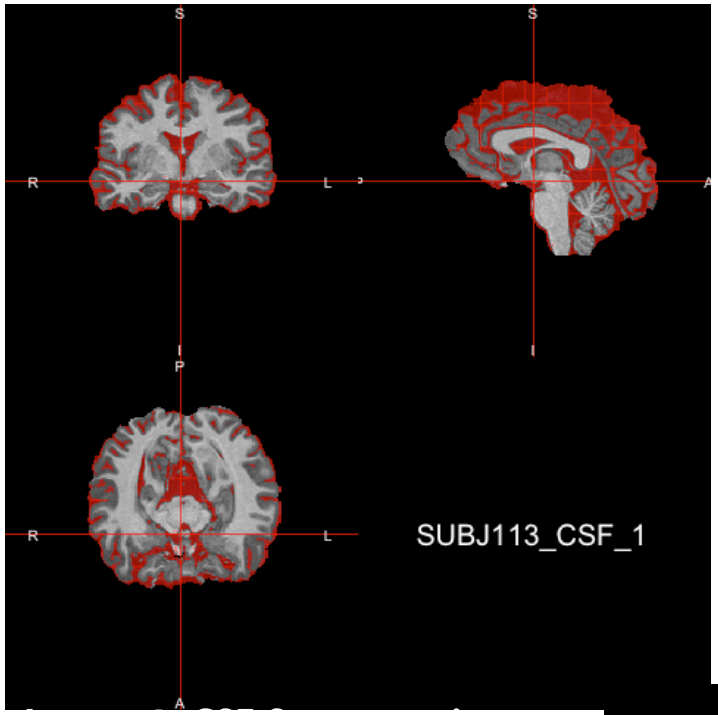
Data Preparation

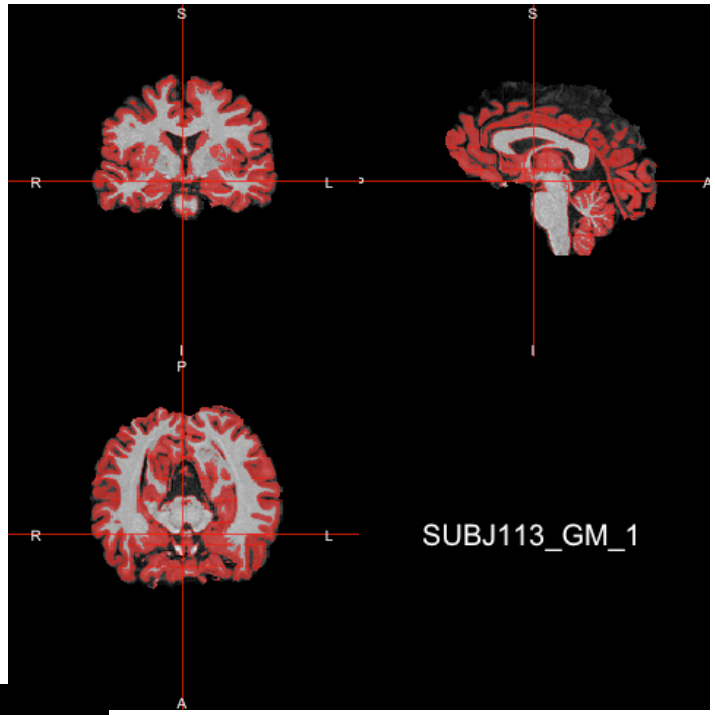
```
mridir = file.path("/home/fsluser/Desktop/MOOC-2015/
kirby21/visit_1/113")
t1_path=file.path(mridir,"113-01-MPRAGE.nii.gz")

#Read the file
nim=readNIfTI(t1_path, reorient=FALSE)

#Conduct bias field correction
fast_img = fsl_biascorrect(nim, retimg=TRUE)


#Perform brain extraction
bet = fslbet(infile=fast_img, retimg=TRUE)
```

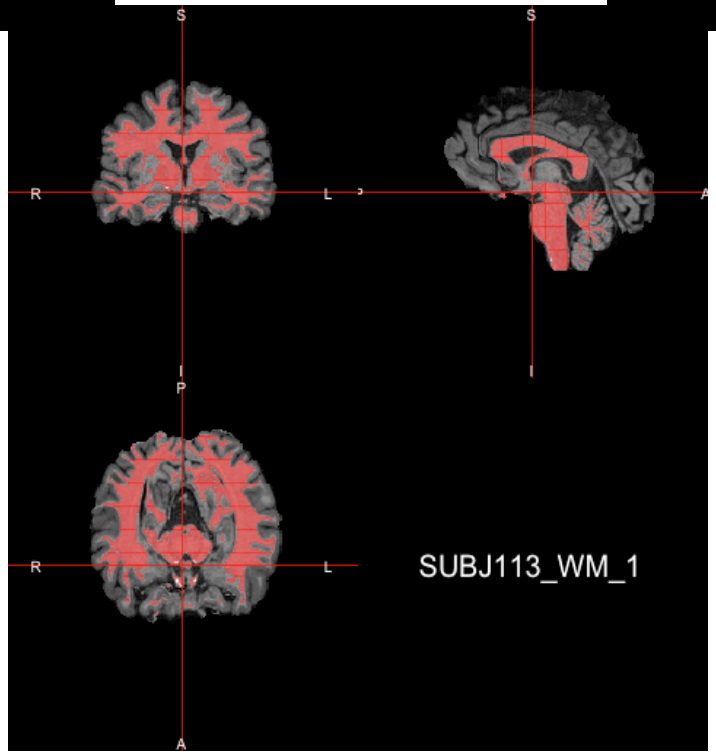


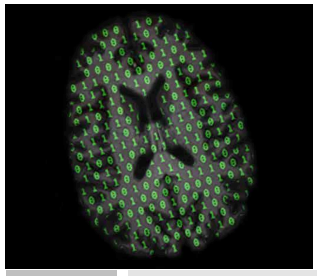
Brain Segmentation

```
#Perform segmentation
fast=fast(file=bet_fast,outfile=file.path(paste0(mridir,"/
113-01-MPRAGE_biascorrected_BET_FAST.nii.gz")))

#Displays CSF segmentation
ortho2(bet,fast==1,col.y=alpha("red",
0.5),text="SUBJ113_CSF_1")


#Displays GM segmentation
ortho2(bet,fast==2,col.y=alpha("red",0.5),text="SUBJ113_GM_1")

#Displays WM segmentation
ortho2(bet,fast==3,col.y=alpha("red",0.5),text="SUBJ113_WM_1")
```


SUBJ113_CSF_1


SUBJ113_GM_1


SUBJ113_WM_1

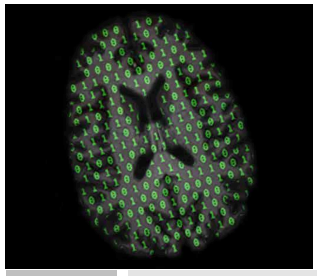


Output Files from fslr Segmentation

113-01-MPRAGE_...AST_mixeltype.nii.gz	Today, 9:03 PM	359 KB	GZip
113-01-MPRAGE_...T_FAST_pve_0.nii.gz	Today, 9:03 PM	719 KB	GZip
113-01-MPRAGE_...T_FAST_pve_1.nii.gz	Today, 9:03 PM	1.2 MB	GZip
113-01-MPRAGE_...T_FAST_pve_2.nii.gz	Today, 9:03 PM	590 KB	GZip
113-01-MPRAGE_..._FAST_pveseg.nii.gz	Today, 9:03 PM	316 KB	GZip
113-01-MPRAGE_N4_BET_FAST_seg.nii.gz	Today, 9:03 PM	307 KB	GZip
113-01-MPRAGE_N4_BET.nii.gz	Today, 8:17 PM	4.1 MB	GZip
113-01-MPRAGE_N4.nii.gz	Today, 8:10 PM	14.9 MB	GZip

Output files:

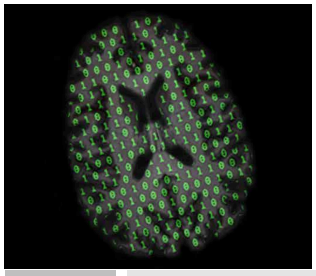
- inhomogeneity correction file
- BET file
- probability maps for corresponding tissue classes: pve_0, pve_1, pve_2
- segmentation files



Output Files from fslr Segmentation

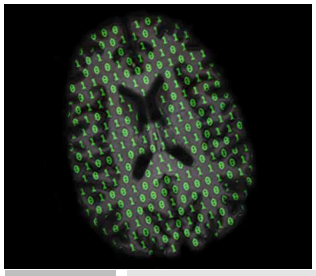
- pve files: probability maps where each voxel is assigned the probability of being in a particular tissue class.
  - pve_0 file corresponds to CSF
  - pve_1 file corresponds to grey matter
  - pve_2 file corresponds to white matter

- Values for the same voxel across the pve_0, pve_1, and pve_2 files sum to 1



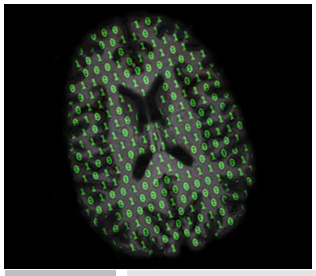
Loading in the pve Files

```
#Reads in the pve file for CSF
pve_CSF=readNIfTI(paste0(mridir,"/113-01-
MPRAGE_N4_BET_FAST_pve_0.nii.gz"))

#Reads in the pve file for GM
pve_GM=readNIfTI(paste0(mridir,"/113-01-
MPRAGE_N4_BET_FAST_pve_1.nii.gz"))

#Reads in the pve file for WM
pve_WM=readNIfTI(paste0(mridir,"/113-01-
MPRAGE_N4_BET_FAST_pve_2.nii.gz"))
```



Calculating CSF, GM, and WM Volumes

```
#Reads in the pve file for CSF
threshold=0.33

#Calculate the product of voxel dimensions (volume)
vdim_CSF=prod(voxdim(pve_CSF))

#Reads in the pve file for WM
nvoxels_CSF=sum(pve_CSF>threshold)

#Calculate the volume of CSF in mL
vol_pveCSF=vdim_CSF*nvoxels_CSF/1000
```



Results

```
#CSF volume in mL
vol_pveCSF
[1] 349.9128

#GM volume in mL
vol_pveGM
[1] 914.67

#WM volume in mL
vol_pveWM
[1] 703.698
```

This would need to be redone using a better brain segmentation, using, for example the cog trick