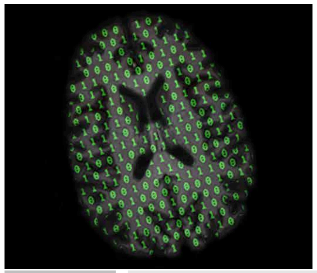
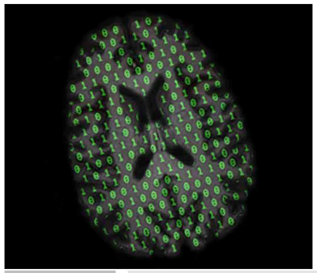
THEME 3: PRE-PROCESSING



Theme 3: Pre-Processing

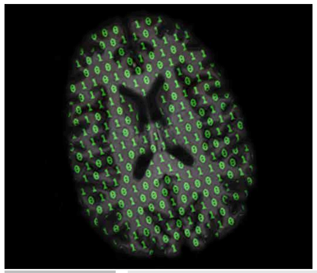
- Requirements
- Pre-processing pipelines
- Inhomogeneity correction
- Skull stripping
- Rigid and affine registration
- Non-linear registration
- Intensity normalization

THEME 3 / LECTURE 1: PRE-PROCESSING PIPELINES



Pre-Processing Pipeline

- Definitions
- Basic components
- Pipeline Tools



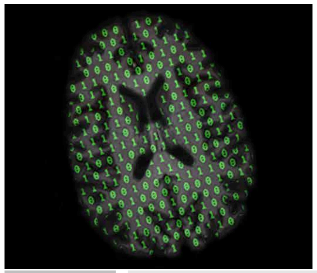
Definition: Image Pre-Processing

Pre-processing is a collection of transformations applied to an original image to obtain data in a pre-specified analytic format

Different pipelines result in different files

We divide **image pre-processing** into four main conceptual steps

1) inhomogeneity correction
2) spatial interpolation
3) skull stripping
4) spatial registration



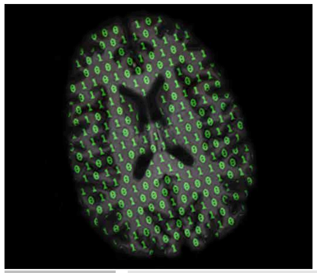
Definition: Pipeline

A **pipeline** is a choice of a particular set and sequence of image pre-processing steps that can be applied to many images

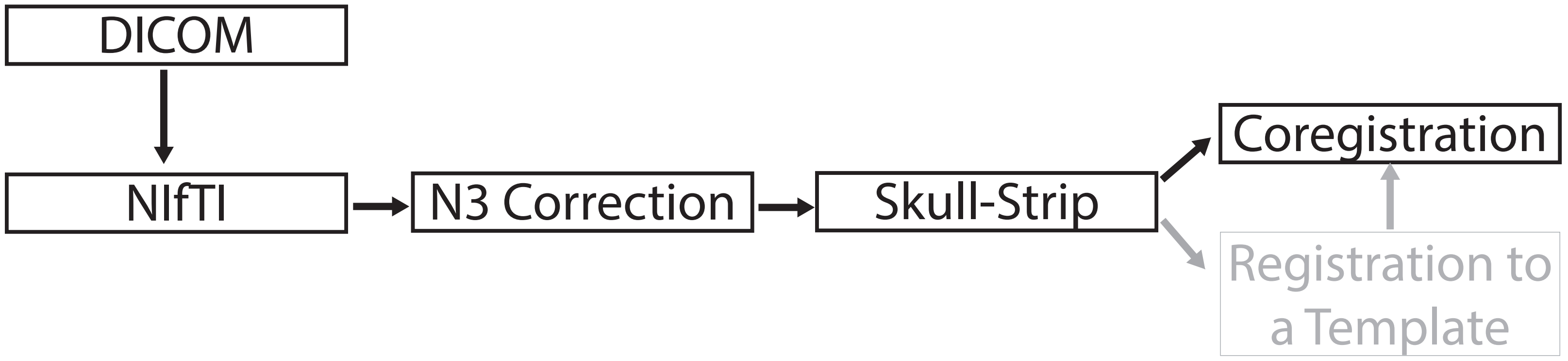
Pipelines should be scriptable and reproducible

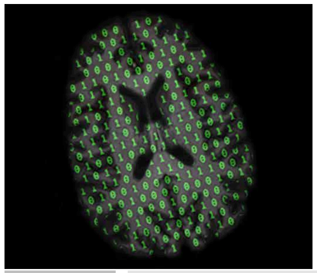
Pipeline scripts should be published with the paper

Current publication standards for image pipelining are inadequate

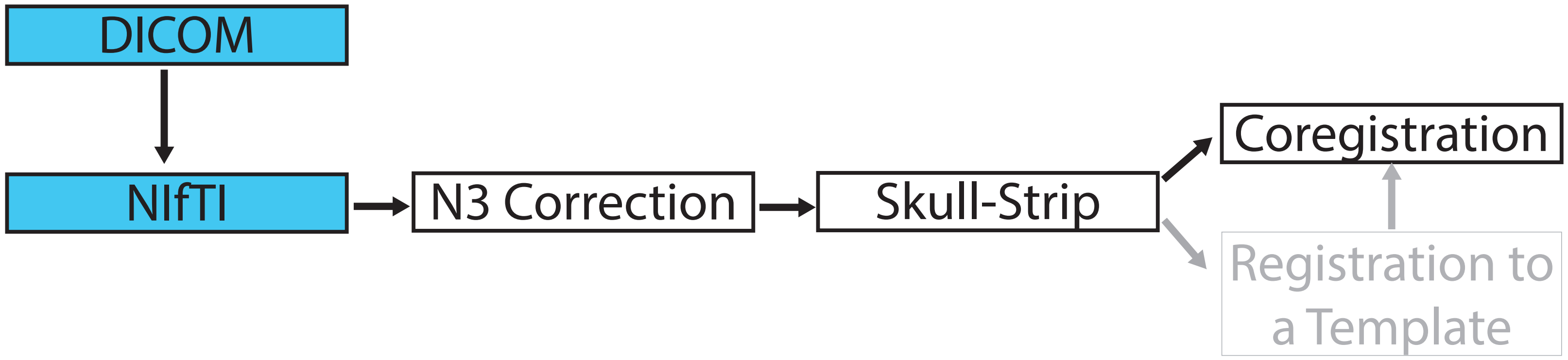


A Basic Pre-Processing Pipeline



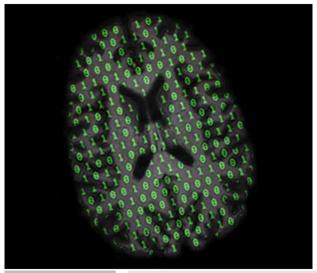


File Types

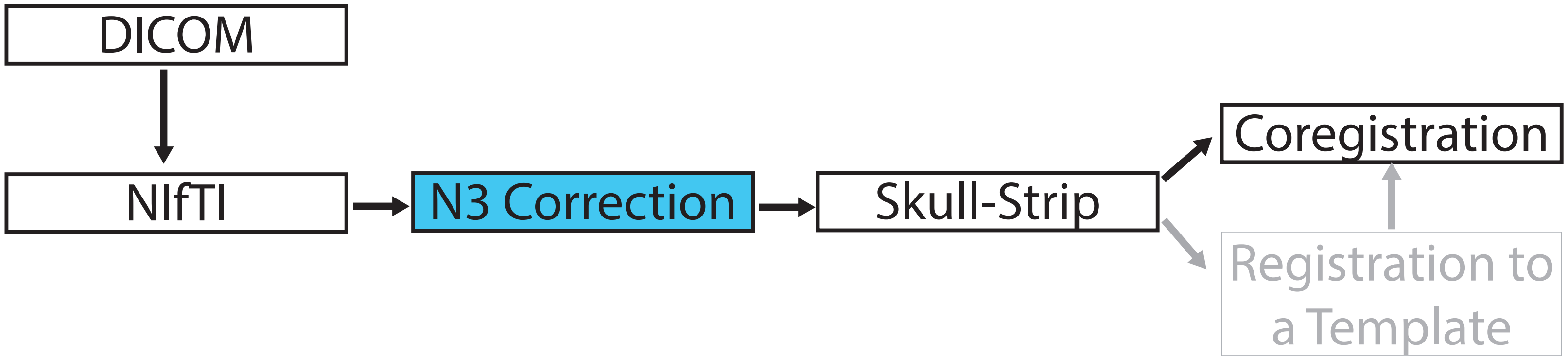


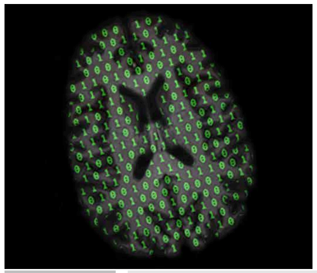
- dcm2nii
  http://www.mccauslandcenter.sc.edu/mricro/mricron/dcm2nii.html

- R Package oro.dicom
  http://cran.r-project.org/web/packages/oro.dicom/index.html

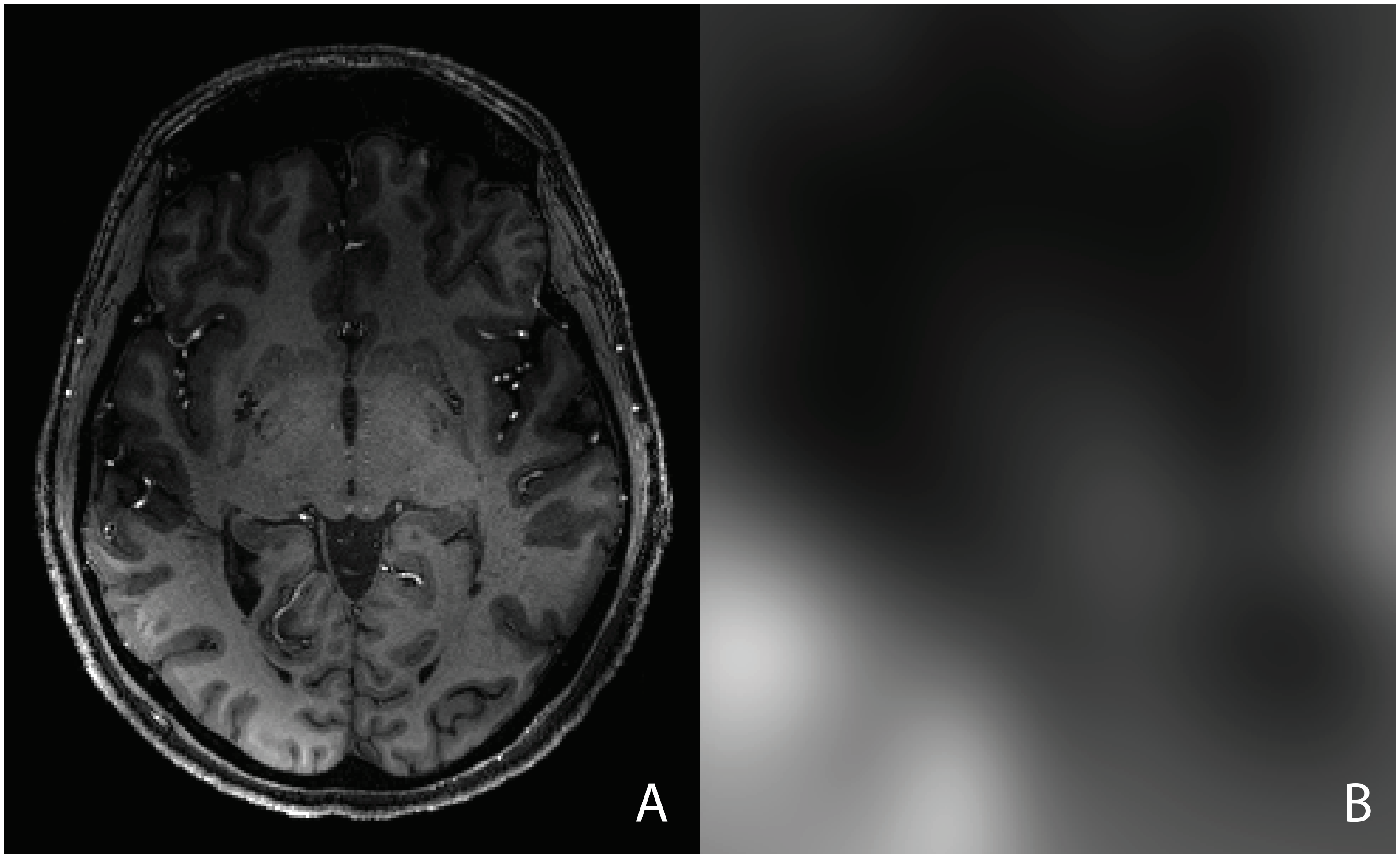


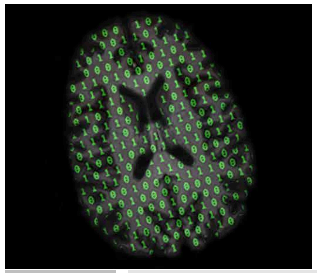
Inhomogeneity Correction





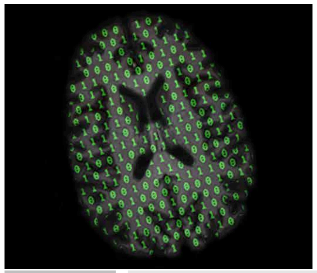
Inhomogeneity Correction



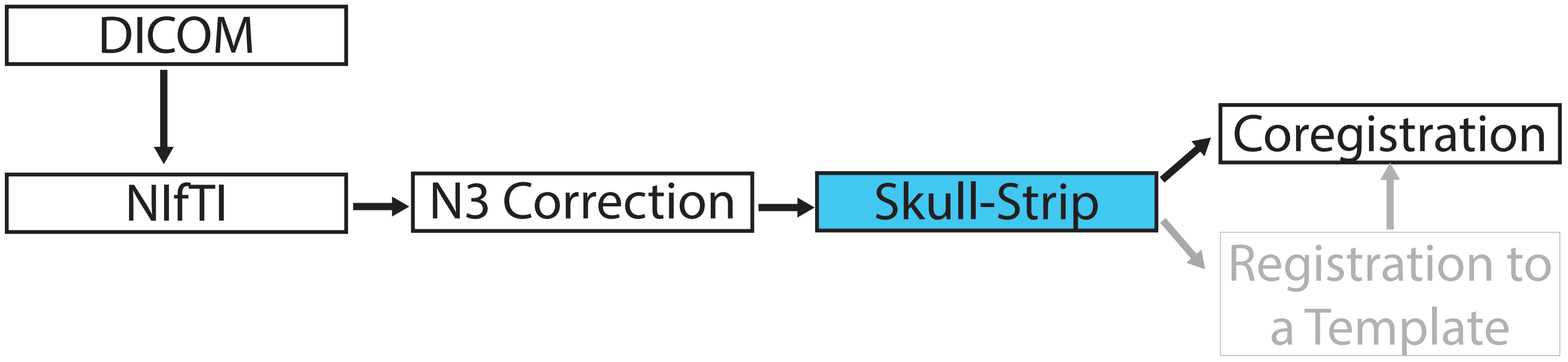


Inhomogeneity: Specifics

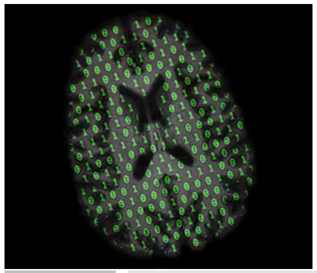
- Technical definition
  - The (probability) distribution function of tissue class intensities should not depend on the spatial localization of the tissue
  - Simple checks:
    - run an aggressive smoother over the image
    - obtain and plot the tissue-specific histogram intensities as a function of location
- Intuition:
  - white matter intensities in the superior part of the brain should not be darker than the white matter intensities in the inferior part of the brain
  - gray matter intensities on the left side of the brain should not be lighter than the gray matter intensities on the right side of the brain
  - white matter intensities should not be darker than gray matter intensities in certain parts of the brain and lighter in others



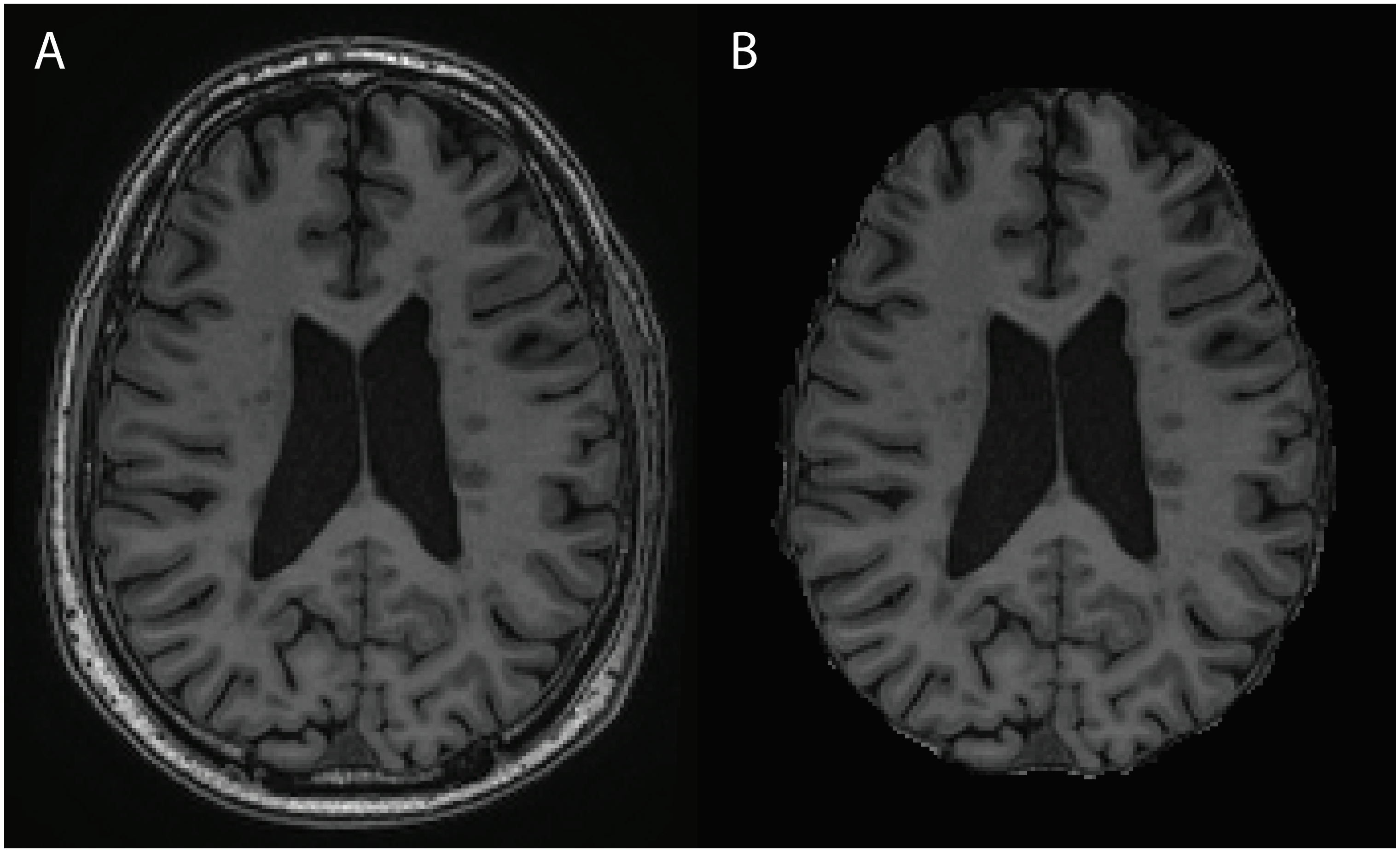
Skull-stripping



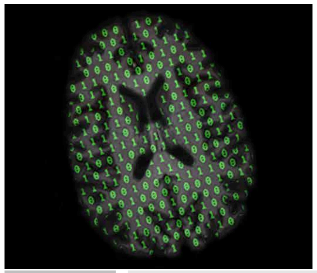
Skull-stripping removes extra-cerebral voxels from the volume



Skull-Stripping

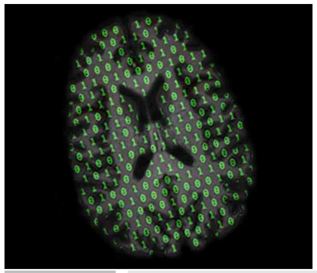


THEME 3 / LECTURE 2: REGISTRATION

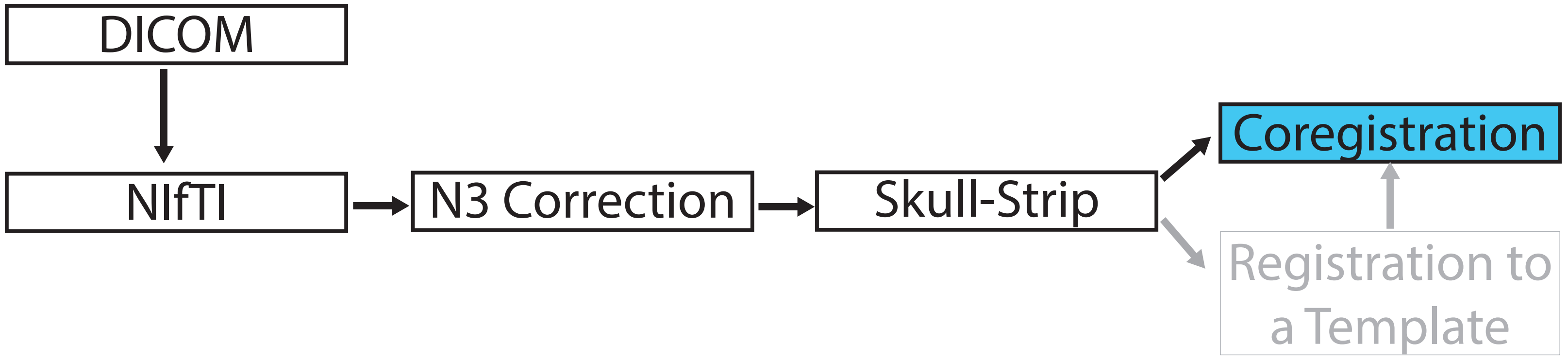


Registration

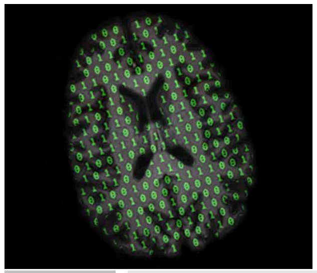
Registration is a spatial transformation of one or multiple images with the goal of making locations (voxels, ROIs) have the same or similar interpretation



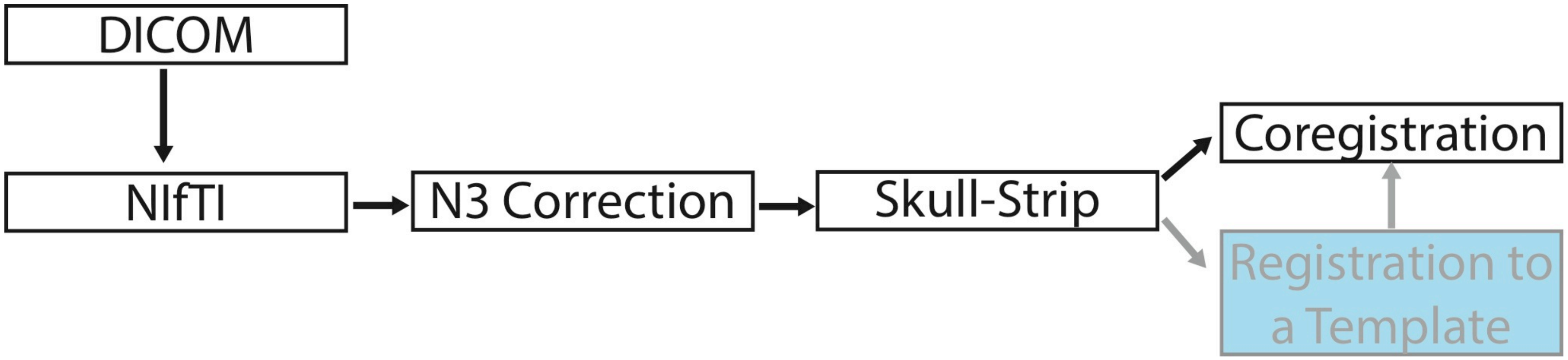
Co-Registration



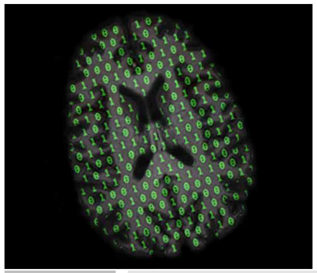
Coregister volumes from different modalities to one another (for example, register the FLAIR to the T1-w volume or register a baseline to a follow-up study)



Registration to a Template

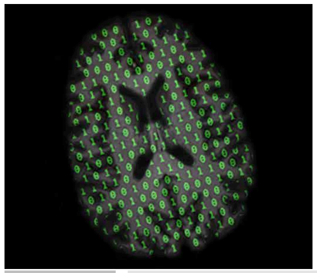


You can also register to a group template (such as the MNI T1-w template)



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 (e.g. MNI template stored in …/data/ MNI152_T1_1mm.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.

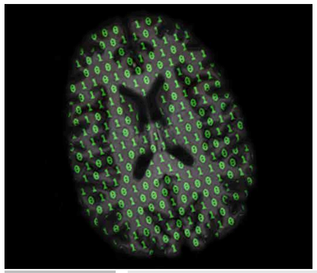
$$T_{\mathrm{rigid}}(v) = Rv + t$$

- 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

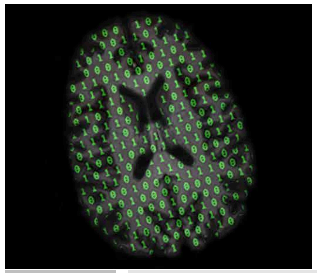
$$t = (t_x, t_y, t_z)$$



Linear Registration: Affine

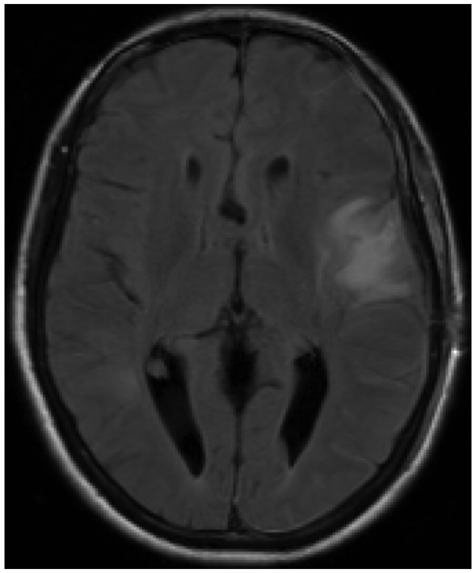
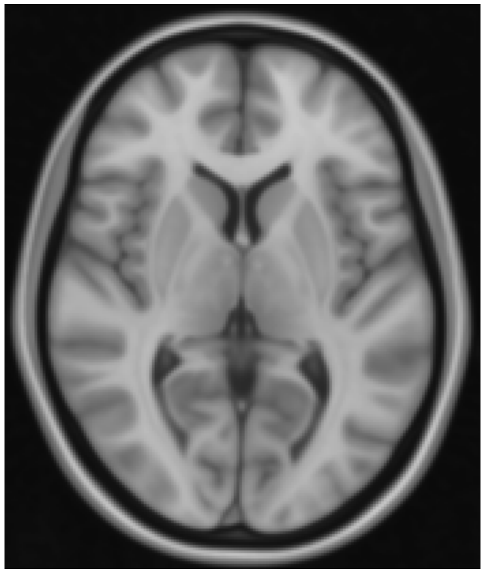
- Affine registration has 12 degrees of freedom
- Same form as the rigid, but the matrix A is not constrained to be a rotation matrix
- A has 9 entries (3×3 matrix) and the translation vector has 3 entries: total "12 degrees of freedom"

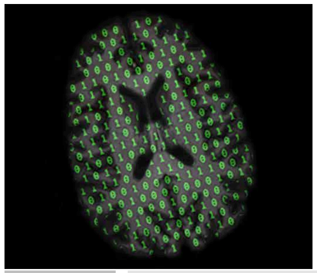
$$T_{\text{affine}}(v) = Av + t$$



Nonlinear registration

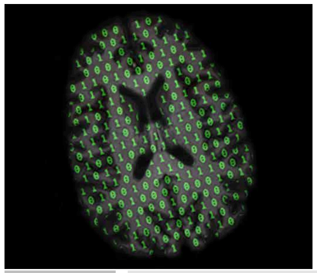
- Co-registration, registration to a template, or from one subject to another



Co-Registration

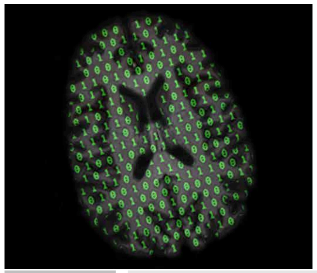
- Works better and requires fewer degrees of freedom
- Easier to register the same brain
- Analysis examples that do not require a reference template
  - Identify location-specific longitudinal changes
  - Segmentation
  - Analysis of intensities



Registration to a Template

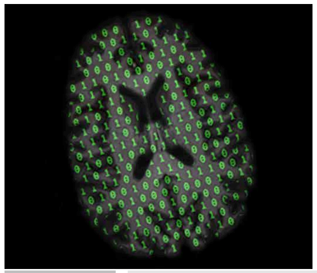
- Assumes that brains can reasonably be morphed to a template space
- Gain anatomical information from the template
- Analysis examples that require a reference template
  - Presenting population level results (e.g. location of lesions)
  - Describing findings at the anatomic level (e.g. the ICH covered more than 30% of thalamus in more than 50% of the patients)
  - Segmentation using multi-atlas label fusion

THEME 3 / LECTURE 3: REQUIREMENTS



Requirements

- Virtual machine: Linux for Windows
- R and fsl installation
- fslr



FSL Installation and Windows Virtual Machine

- **Install FSL:** http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslInstallation

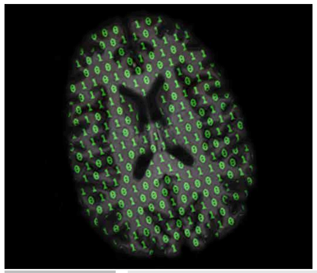
- **Linux/Mac**: follow instructions, you will need to add R packages

- **Windows:** https://smart-stats-tools.org/mooc-2015
  - Recommended approach for Windows
  - Download VirtualBox a free x86 and AMD64/Intel64 virtualization product.
  - The Virtual Machine compressed image (~3.7GB) can be downloaded from here. Make sure you move the archive to a folder of your choice and then decompress it
  - Install the VirtualBox and once that's complete add the virtual machine using the VirtualBox main menu Machine->Add or simply press the CTRL+A combination. A file browser window will open and you need to navigate to the folder created in step 2 and locate the virtual machine image file
  - Once the MOOC virtual machine is added to the VirtualBox start it up by pressing the green Start arrow. The username is fsluser and the password is fsluser, change it if needed
  - This virtual machine comes with all necessary R packages pre-installed (including ANTsR)
  - MOOC sample images were provided and they are located on your virtual machine desktop under "MOOC 2015 data" folder.
  - The virtual machine already has installed the necessary tools for setting up a shared folder with the native operating system (WIN). Please check this easy tutorial that explains the process. Once the folder is set, a reboot of the virtual machine is required and you should be able to access the shared folder by opening the "shared" desktop folder.
  - Data are now loaded in

```
"/home/fsluser/Desktop/MOOC-2015"
```

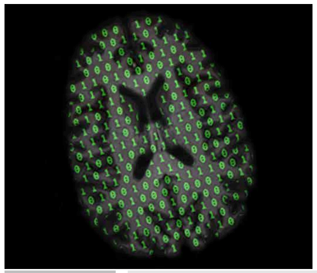
- **Windows**: install FSLvm (FSLvm) and then R

  http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslInstallation/Windows#Installing_FSL



FSL and fslr

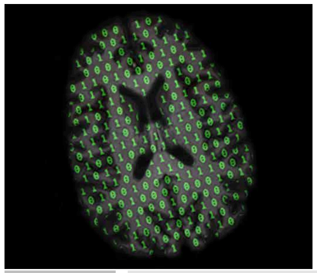
- FSL is a comprehensive library of analysis tools for fMRI, MRI and DTI brain imaging data
  - Collection of routines in C, C++
- fslr: port of FSL into R
- The three functions we focus on are:
  - image inhomogeneity correction (using FAST)
  - skull stripping (using BET)
  - image registration (using FLIRT and FNIRT)



fslr

- fslr is installed on CRAN
- best to install using the devtools package

```
if (!require(devtools)){install.packages('devtools')}
devtools::install_github("muschellij2/fslr")
```



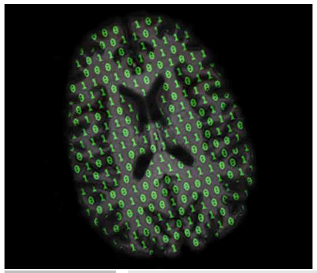
FSL/GUI vs. R Terminal

- GUI-based apps do not inherit the shell environment (if FSLDIR is defined in your terminal Rstudio does not see it)
- fslr requires R to know where the FSL directory was installed

```
Sys.getenv("FSLDIR")
[] “”
library(fslr)
have.fsl()
[] FALSE
```

If `have.fsl()=FALSE` then the fsl path must be specified

```
options(fsl.path= "/usr/local/fsl")
```



Some preliminaries

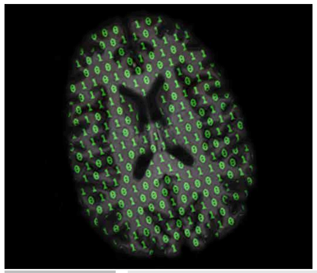
- Linux/Mac: set the working directory to where the data are
- Using the virtual machine: set the data directory and set the working directory accordingly for R

```
setwd("/home/fsluser/Desktop/MOOC-2015/kirby21/visit_1/113")
library(oro.nifti)
nim=readNIfTI("113-01-MPRAGE.nii.gz", reorient=FALSE)
```

- Some statistics using FSL

```
mean(nim)
[1] 143789.2
fslstats(nim, opts= "-m")
fslstats "/tmp/Rtmp4VLZYR/file217f3a830f1a.nii.gz" -m
[1] "143789.231769"
fslstats("113-01-MPRAGE.nii.gz",opts="-m")
fslstats "113-01-MPRAGE.nii.gz" -m
[1] "143789.231769"
fslstats.help()
```

THEME 3 / LECTURE 4: BIAS FIELD CORRECTION USING FSLR



Bias Field Correction Using fslr

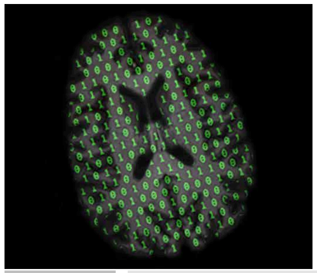
`fslr::fsl_biascorrect` calls `fast` from `FSL` which incorporates the bias field correction by Guillemaud and Brady

This takes a while: be patient

For N3 and N4 correction: use ANTsR

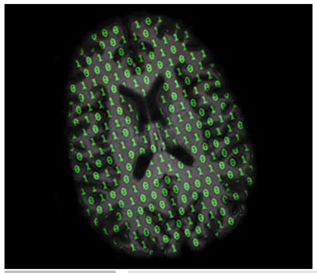
```
fast_img = fsl_biascorrect(nim, retimg=TRUE)
```

Regis Guillemaud and Michael Brady. Estimating the bias field of MR images. In: Medical Imaging, IEEE Transactions on 16.3 (1997), 238-251.

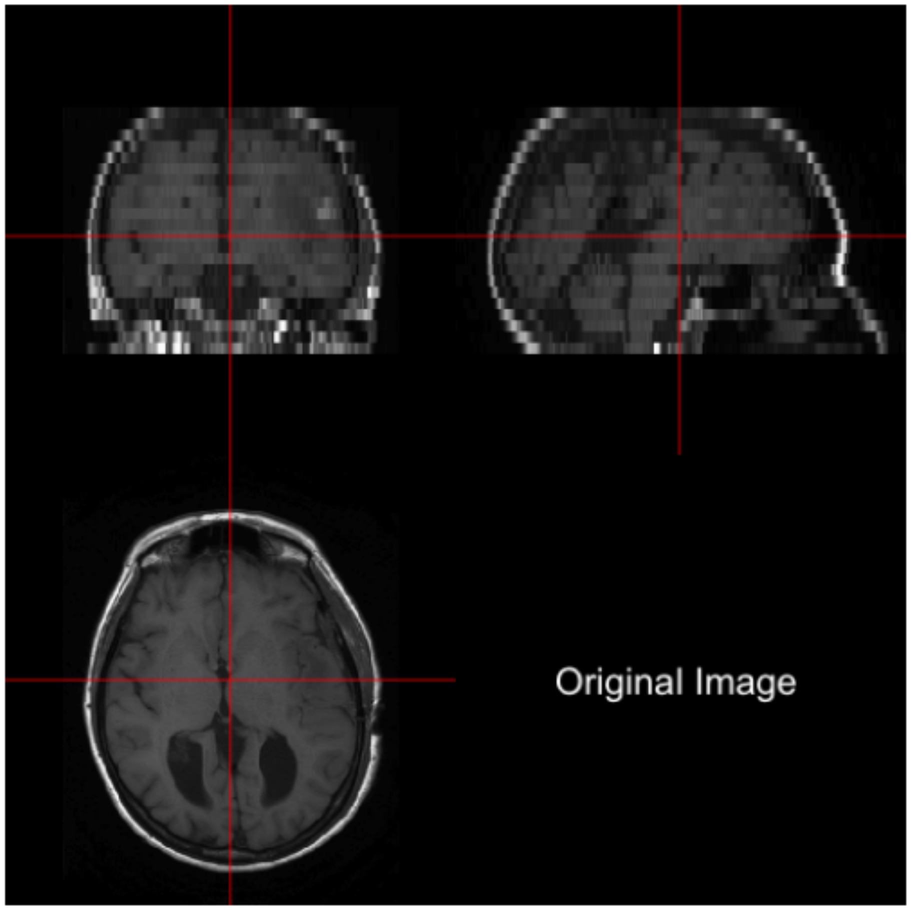


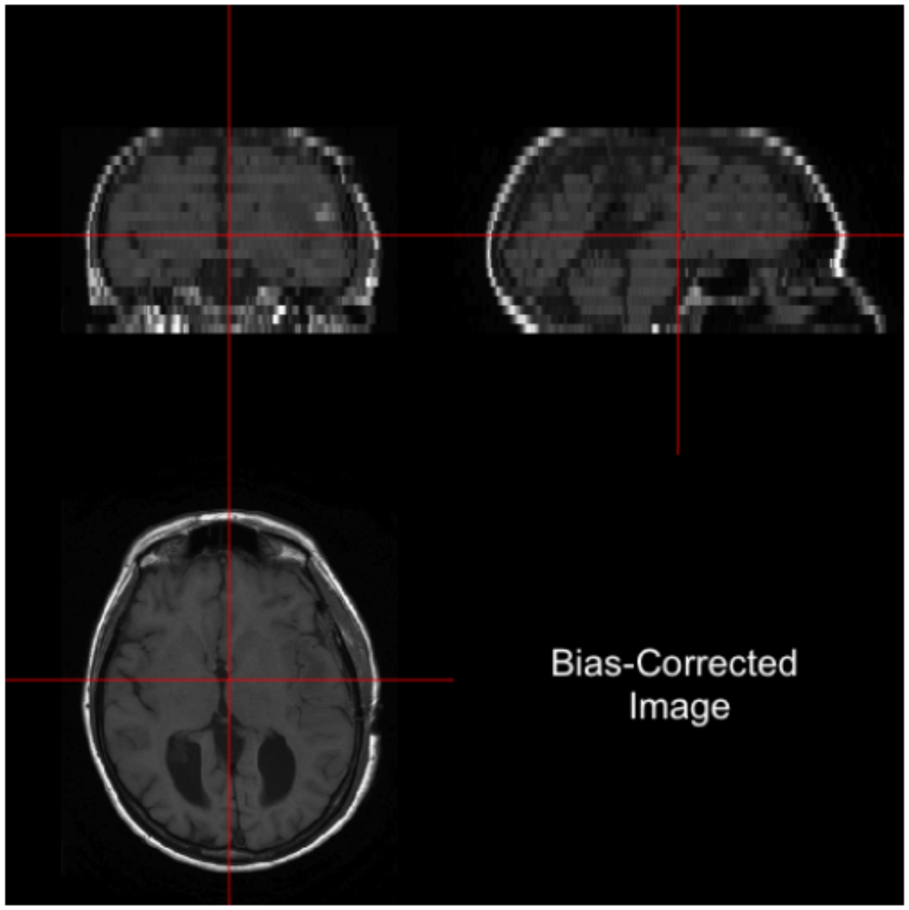
Plotting the Results in R

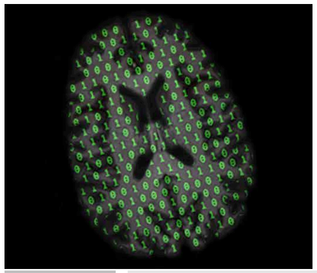
```
orthographic(nim)
orthographic(fast_img)
```



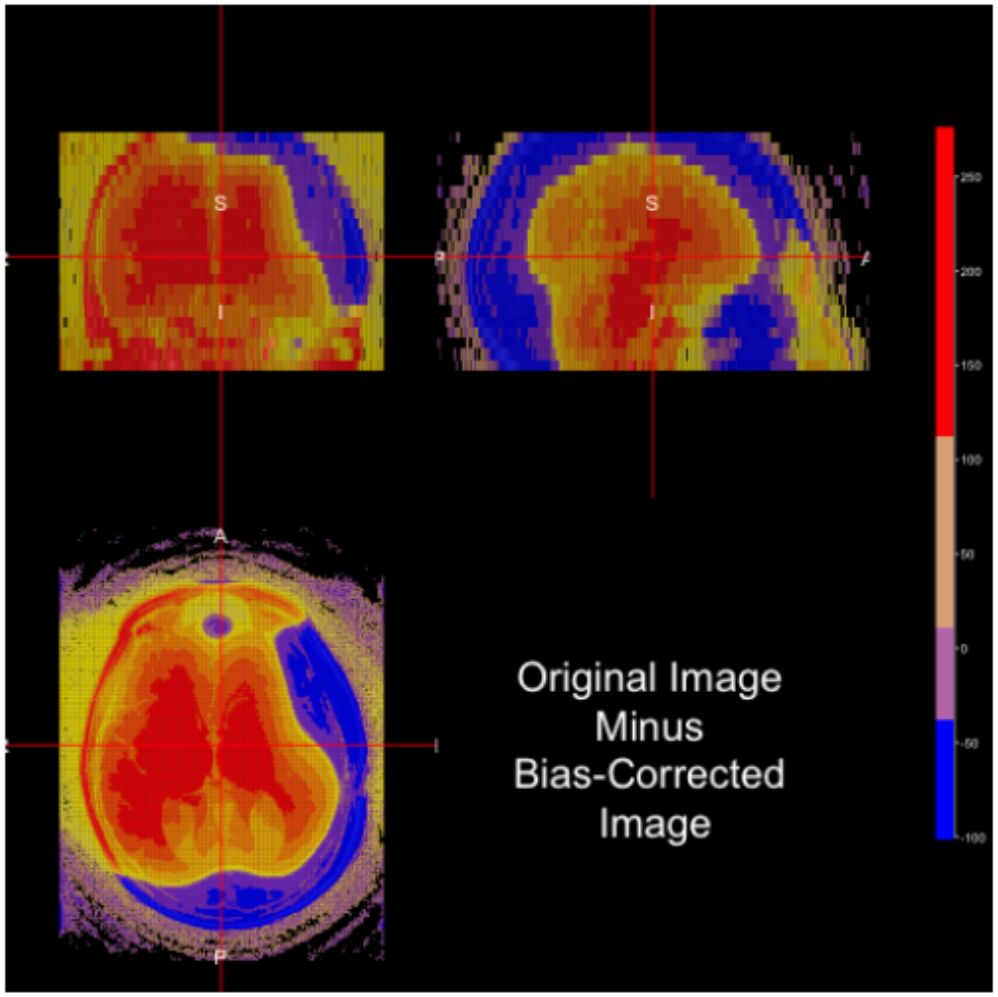
fslr: Bias Field Correction



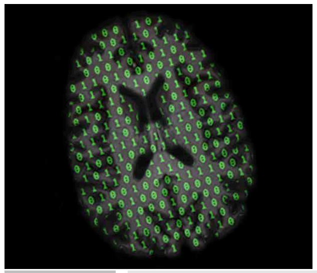




fslr: Original-Bias Field Corrected Image

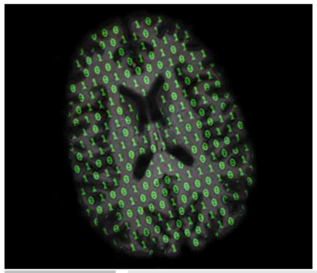


Original Image
Minus
Bias-Corrected
Image

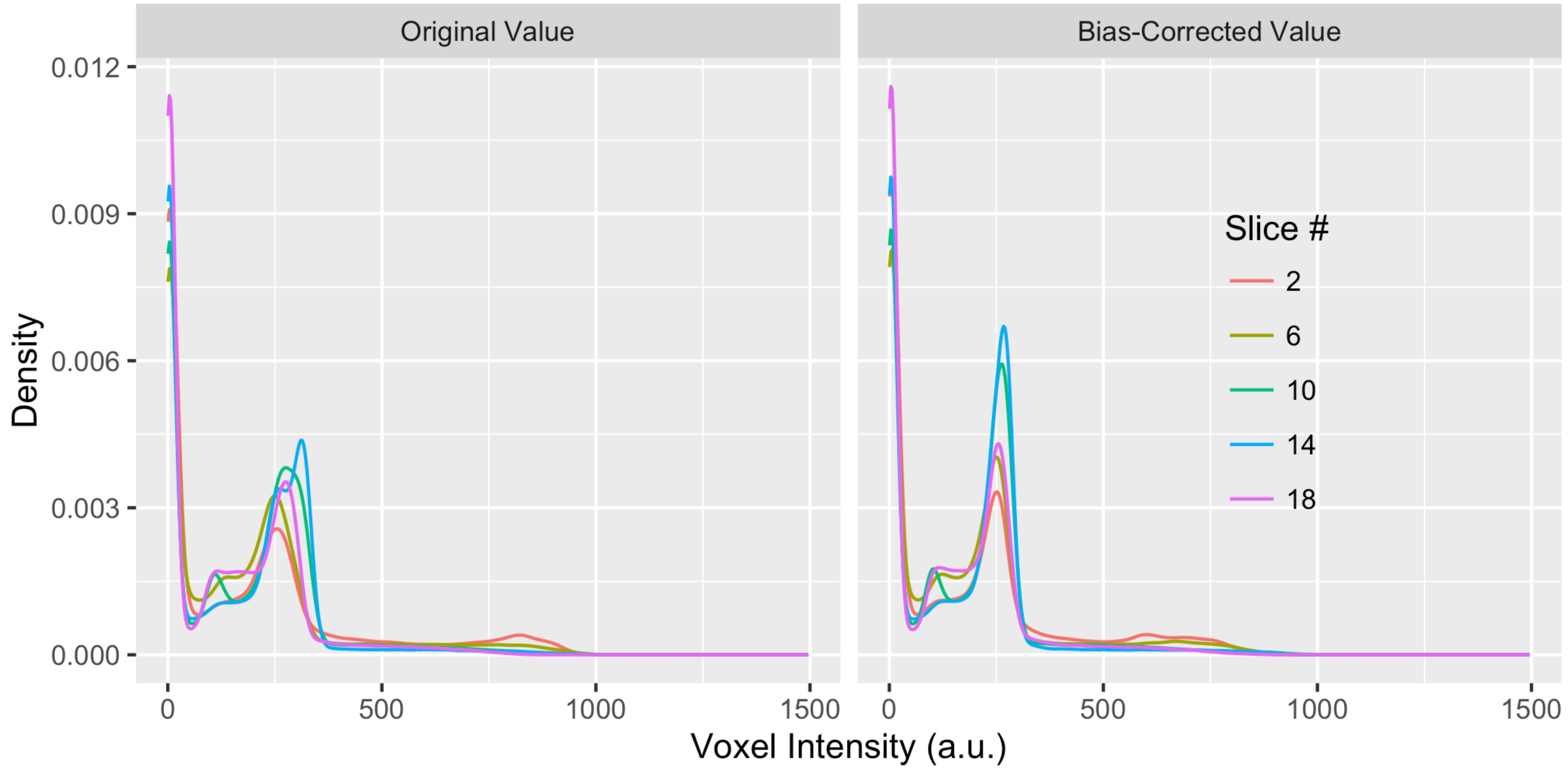


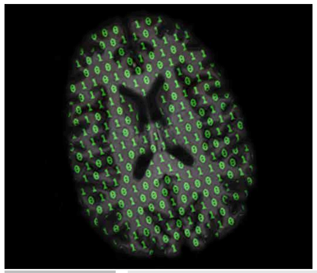
Plotting the Results in R

```
sub.bias <- niftiarr(nim, nim-fast_img)

# quantile the difference image using these as breaks
q=quantile(sub.bias[sub.bias !=0],probs = seq(0,1,by=0.1))

#
install.packages("scales")
library(scales)

# get a diverging gradient palette
fcol=div_gradient_pal(low="blue",mid="yellow",high ="red")


ortho2(nim,sub.bias,col.y = alpha(fcol(seq(0,1, length=10)),
0.5), ybreaks = q, ycolorbar=TRUE, text = paste0("Original
Image Minus N4", "\n Bias-Corrected Image"))
```



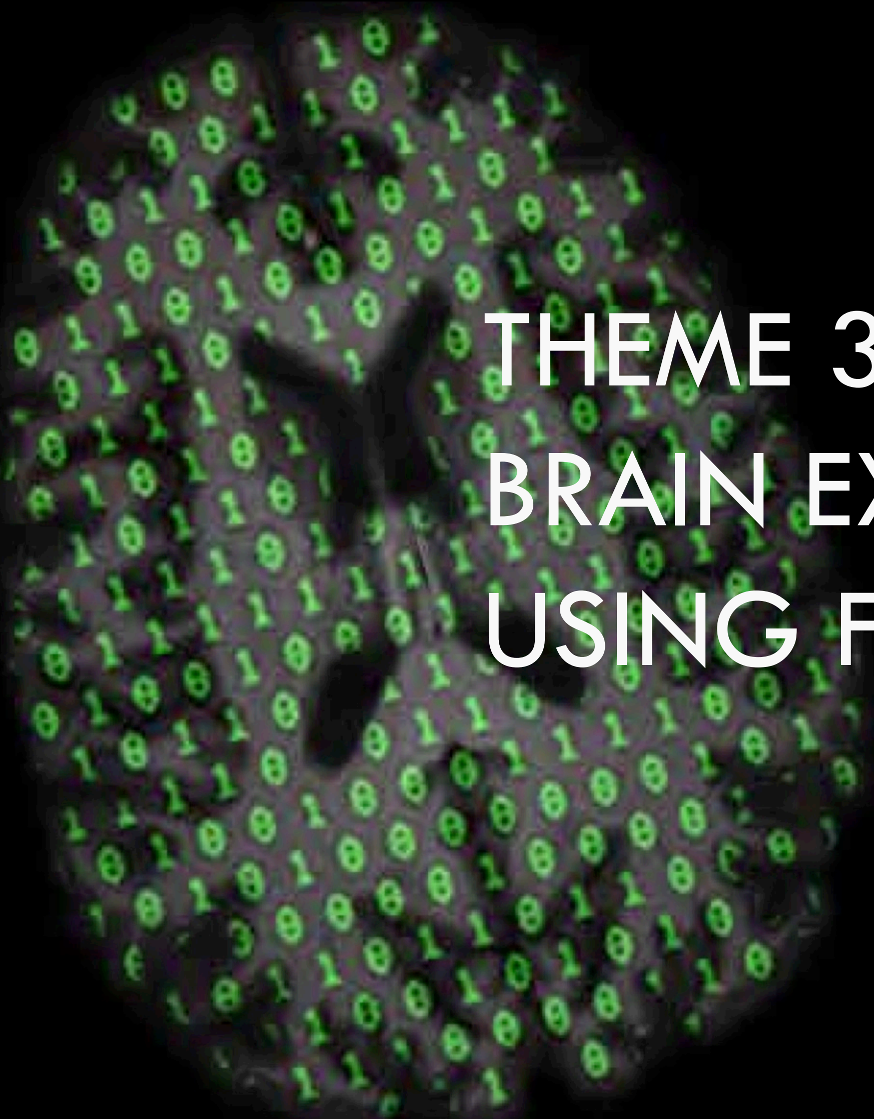
Histogram of Correction



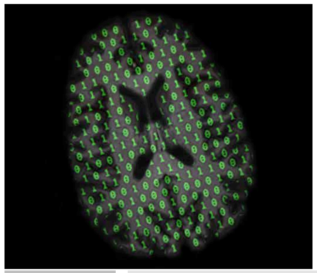


Code for Plotting Histogram

```
slices = c(2, 6, 10, 14, 18)
vals = lapply(slices, function(x) {
    cbind(img = c(nim[,,x]), fast = c(fast_img[,,x]),
          slice = x)
})
vals = do.call("rbind", vals)
vals = data.frame(vals)
vals = vals[ vals$img > 0 & vals$fast > 0, ]
colnames(vals)[1:2] = c("Original Value", "Bias-Corrected Value")
v = melt(vals, id.vars = "slice")
g = ggplot(aes(x = value,
               colour = factor(slice)),
           data = v) + geom_line(stat = "density") +
  facet_wrap(~ variable)
g = g + scale_colour_discrete(name = "Slice #")
```



THEME 3 / LECTURE 5: BRAIN EXTRACTION USING FSLR



Brain Extraction Using fslr

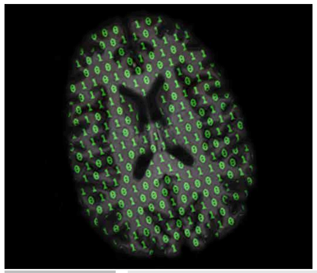
FSL's Brain Extraction Tool (BET) can be used for skull stripping

BET: fast, robust, and popular

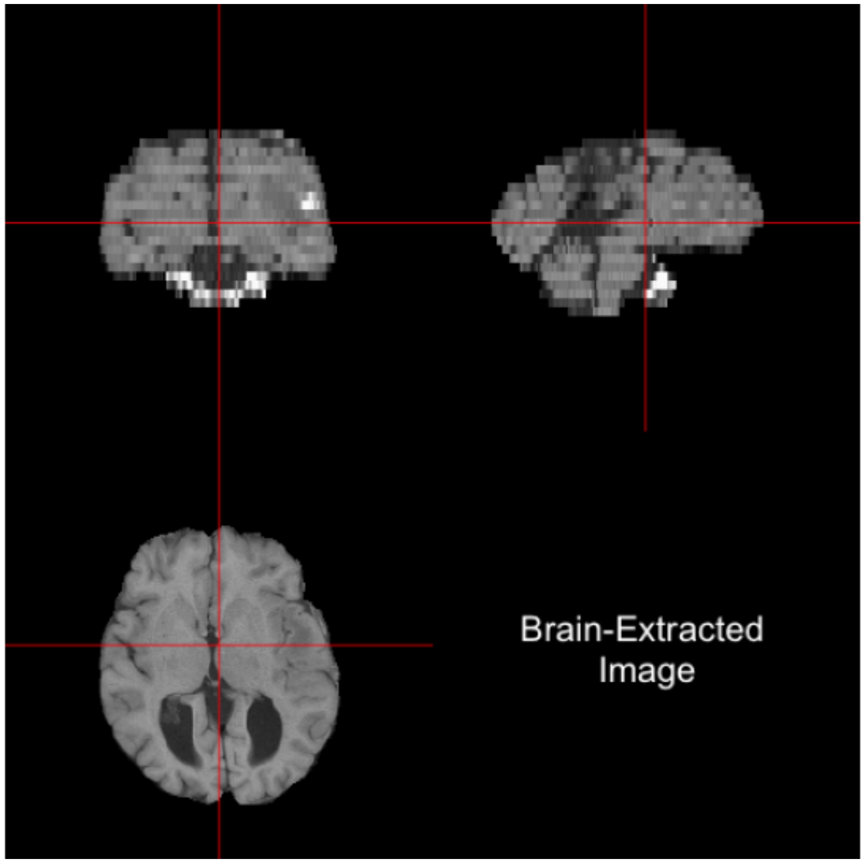
`fslr::fslbet` is used to call the FSL commands

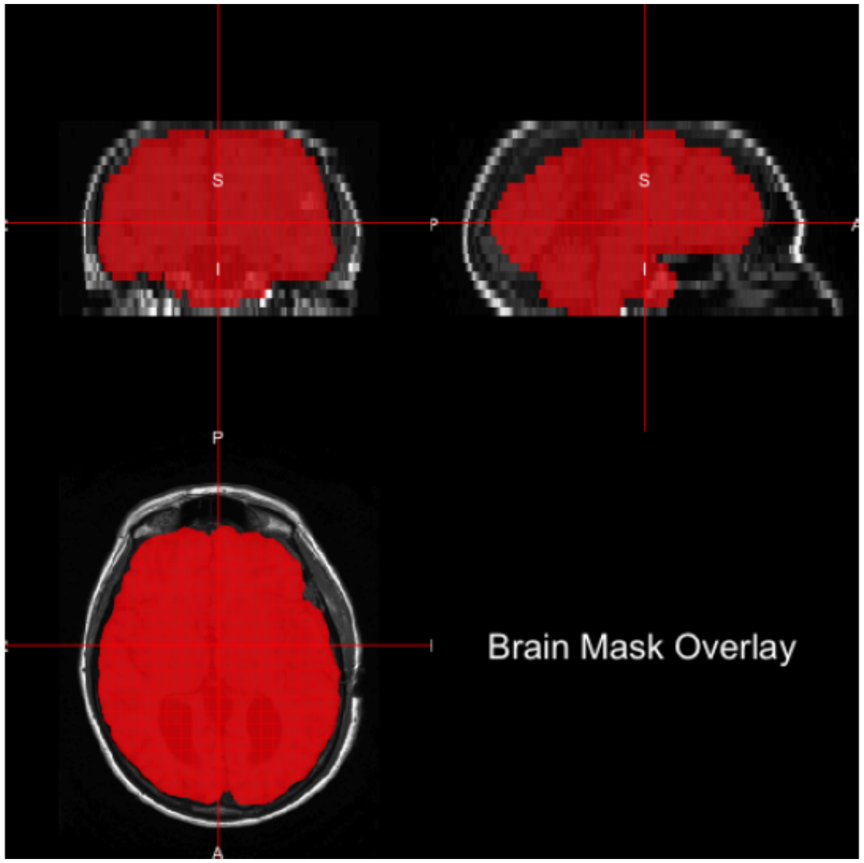
- bet2: does brain extraction
- bet: does brain extraction with additional options

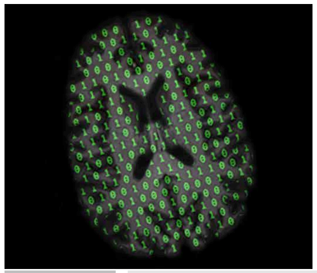
```
bet_fast = fslbet(infile=fast_img, retimg=TRUE)
```



fslr: Brain Extraction Results

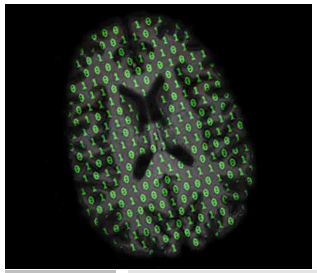




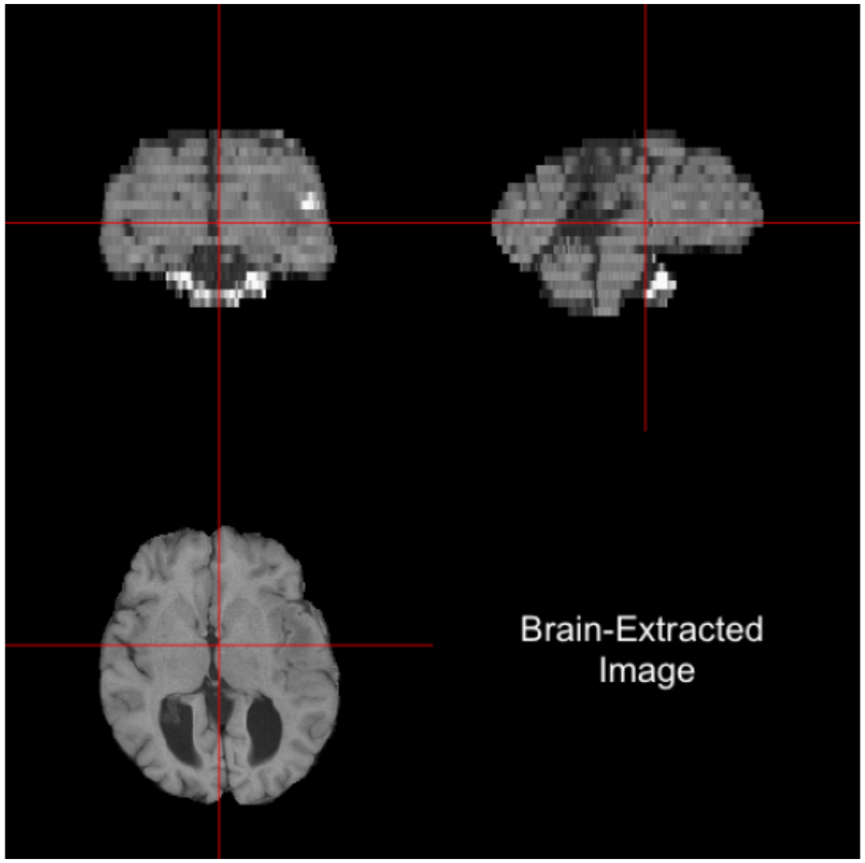


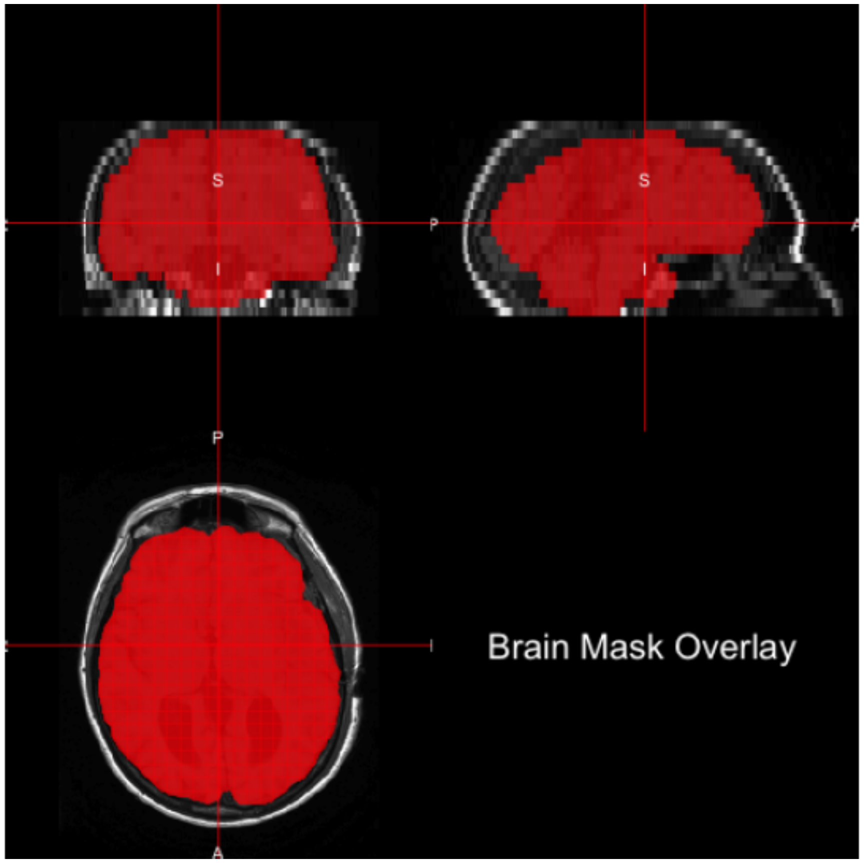
Plotting Extracted Brain on Original Data

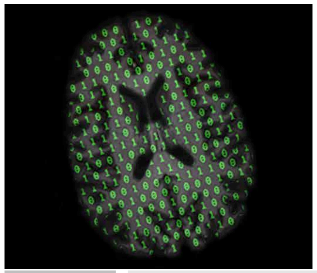
```
bet_fast_mask <- niftiarr(bet_fast, 1)
is_in_mask = bet_fast>0
bet_fast_mask[!is_in_mask]<-NA
orthographic(bet_fast)
orthographic(fast_img,bet_fast_mask)
```



fslr: Brain Extraction Results



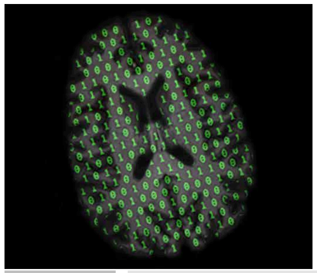




fslr: Improving Brain Segmentation

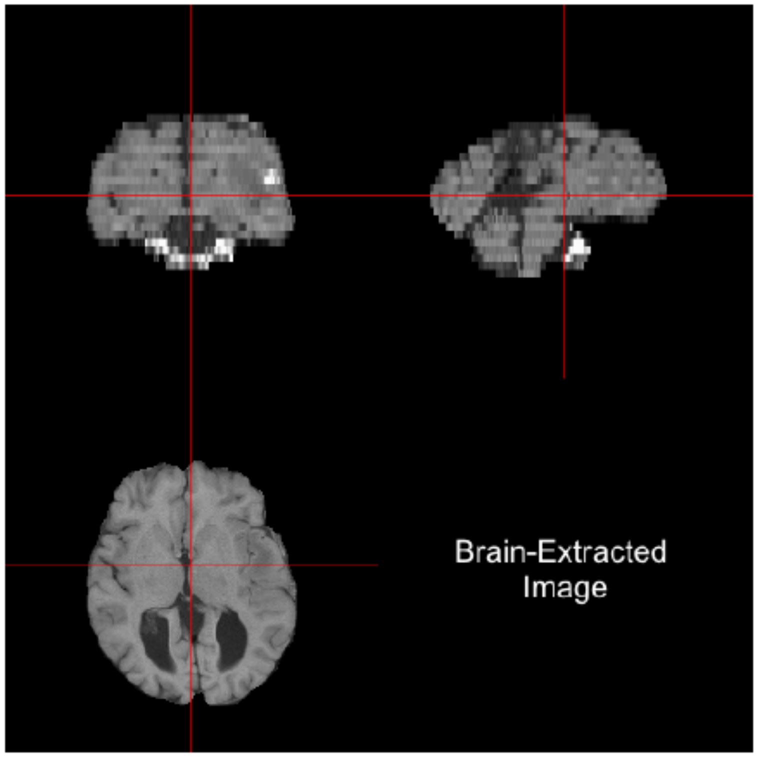
- Some parts of the brain are not segmented
- Possible solution:
  - estimate the center of gravity (COG) from the extracted image
  - re-run bet with the new COG to get a better result

```
cog = cog(bet_fast, ceil=TRUE)
cog = paste("-c", paste(cog, collapse= " "))
bet_fast2 =
fslbet(infile=fast_img,retimg=TRUE,opts=cog)
```

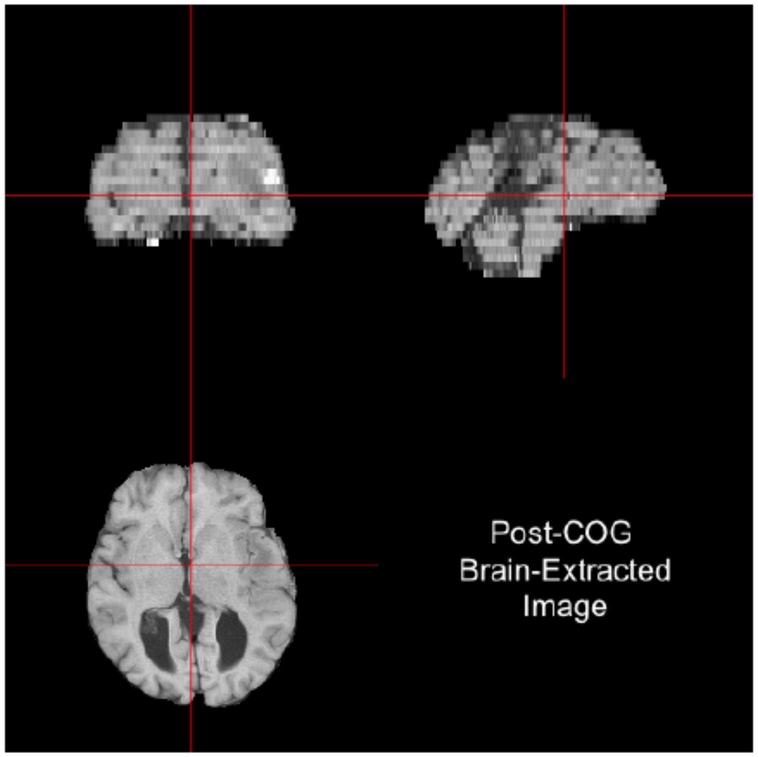


fslr: Improving Brain Segmentation

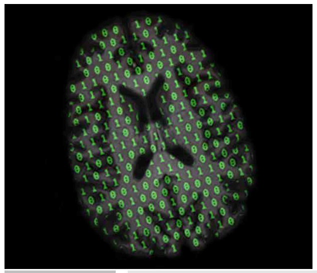
Before COG



After COG



THEME 3 / LECTURE 6: IMAGE REGISTRATION USING FSLR

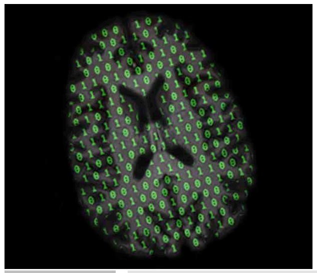


Linear Image Registration

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

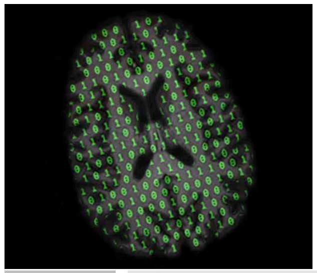
`fslr::flirt` takes in a input filename (or nifti) and a reference filename (or nifti) to transform the infile to:

```
tempdir <- "/home/fsluser/Desktop/MOOC-2015/Template"

template<-readNIfTI(file.path(tempdir, "/
MNI152_T1_1mm_brain.nii.gz"), reorient=FALSE)

registered_fast = flirt(infile=bet_fast2, reffile =
template, dof = 6,retimg = TRUE)
```

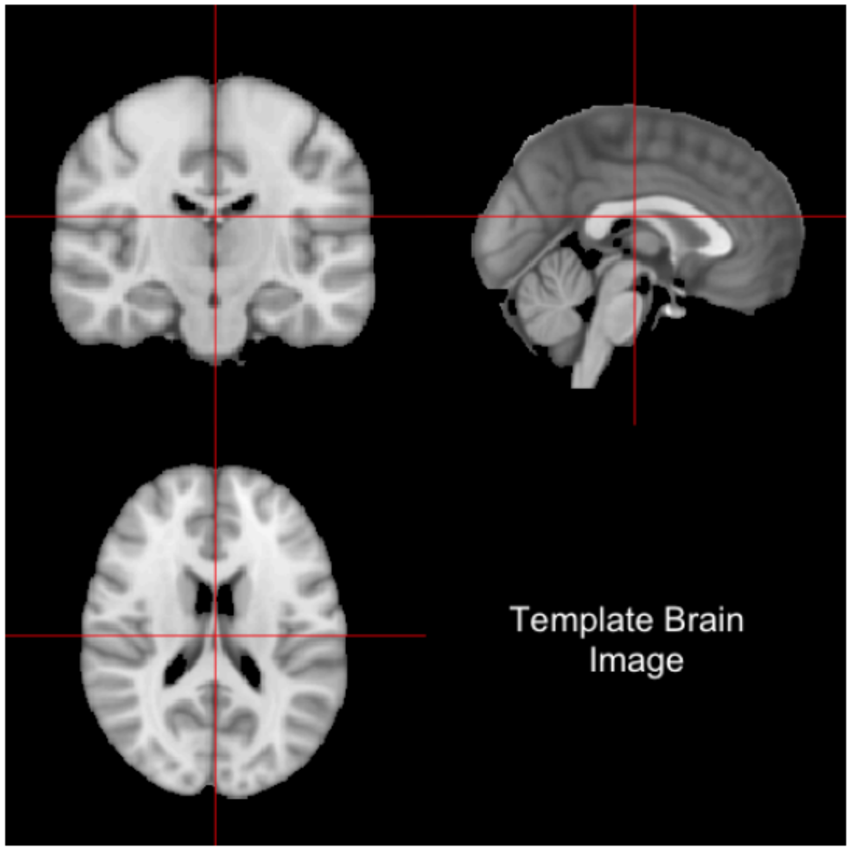


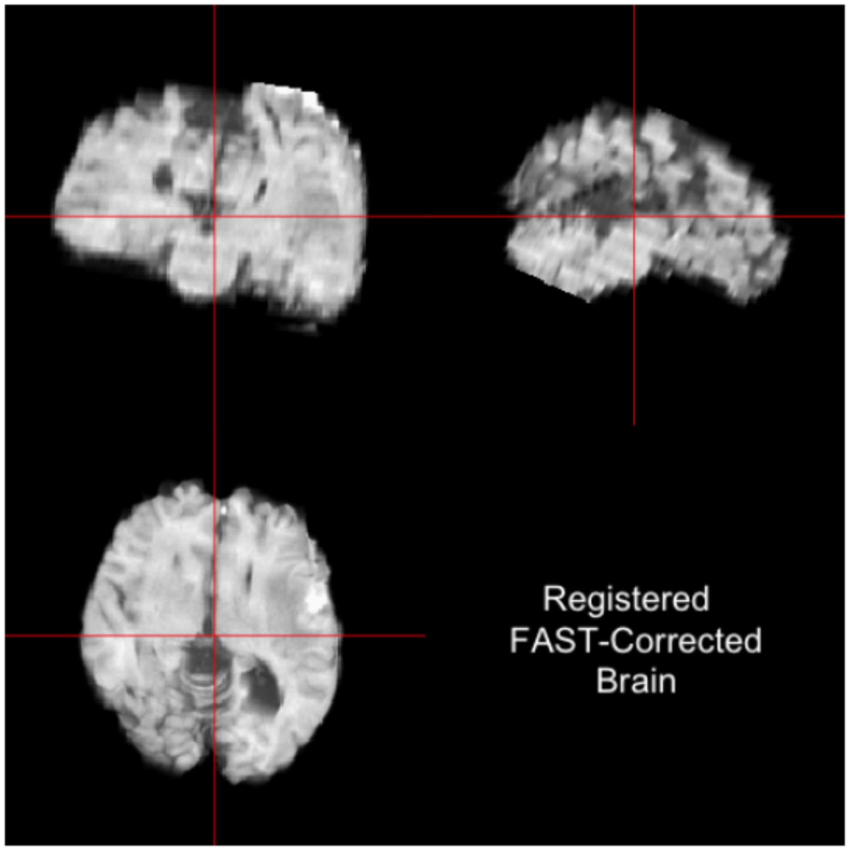
fslr: Image Registration (Rigid) Results

```
orthographic(template)
orthographic(registered_fast)
```



fslr: Image Registration (Rigid) Results





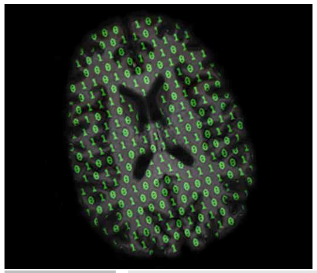
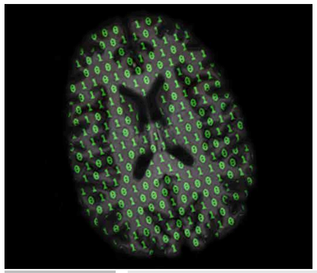


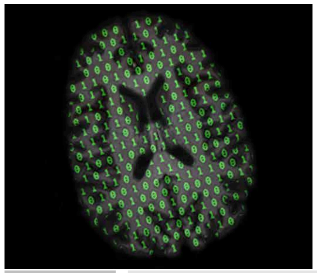
Image dimensions

```
dim(template)
[1] 182 218 182

dim(registered_fast)
[1] 182 218 182

dim(bet_fast2)
[1] 170 256 256
```

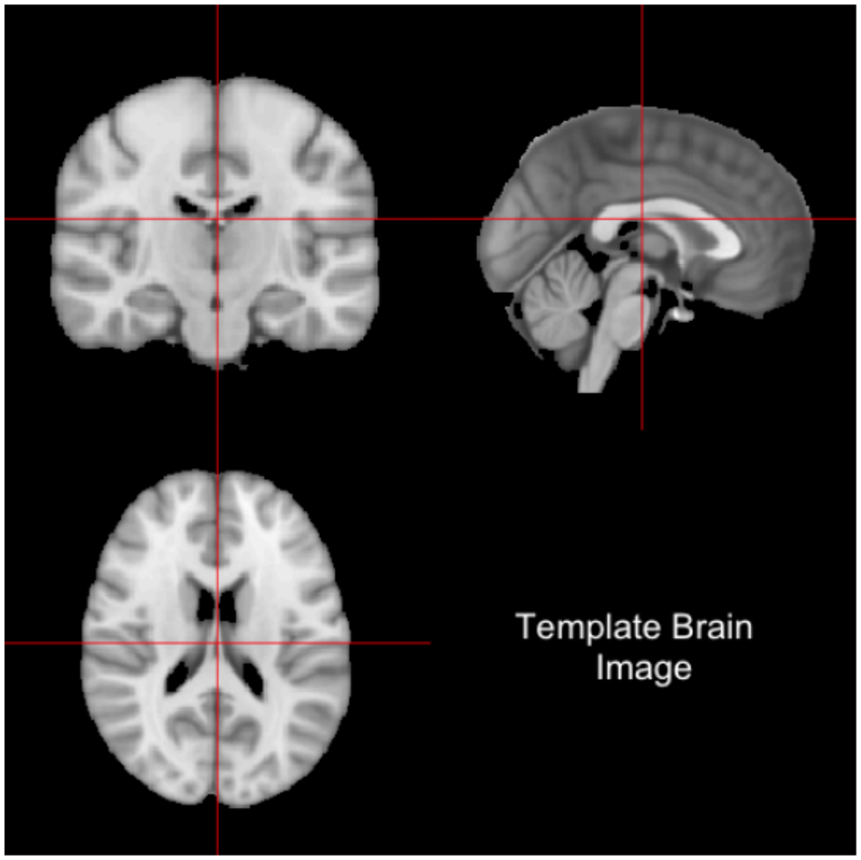


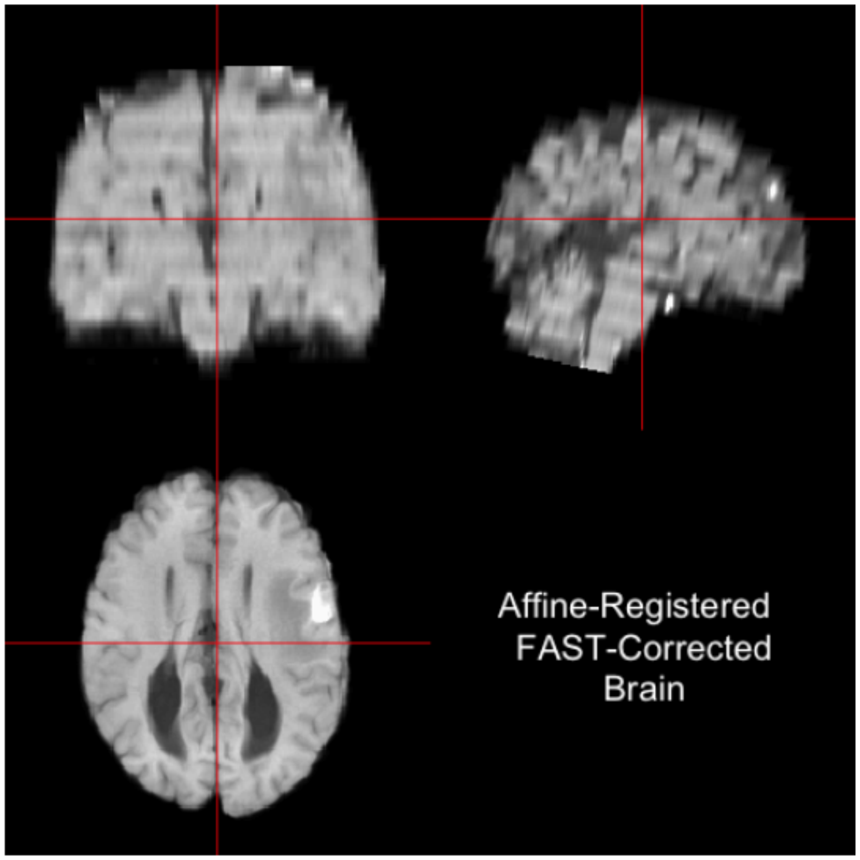
fslr: Affine Image Registration

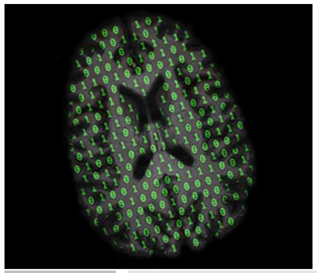
```
reg_fast_affine = flirt(infile=bet_fast2, reffile =
template, dof = 12,retimg = TRUE)
```



fslr: Image Registration (Affine) Results







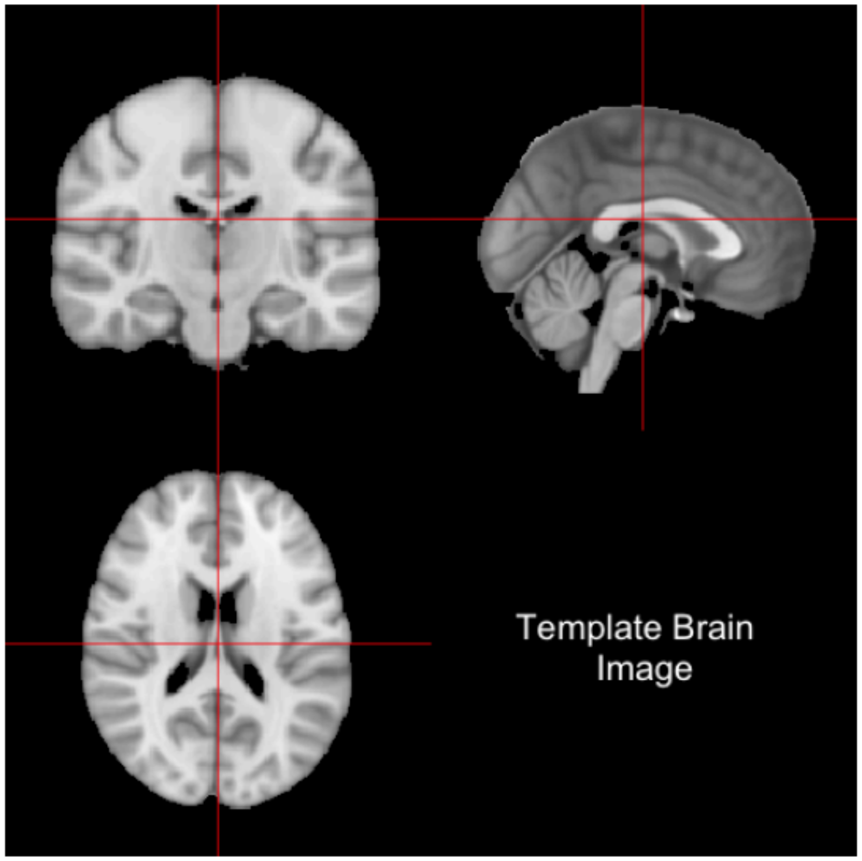
fslr: Nonlinear Image Registration

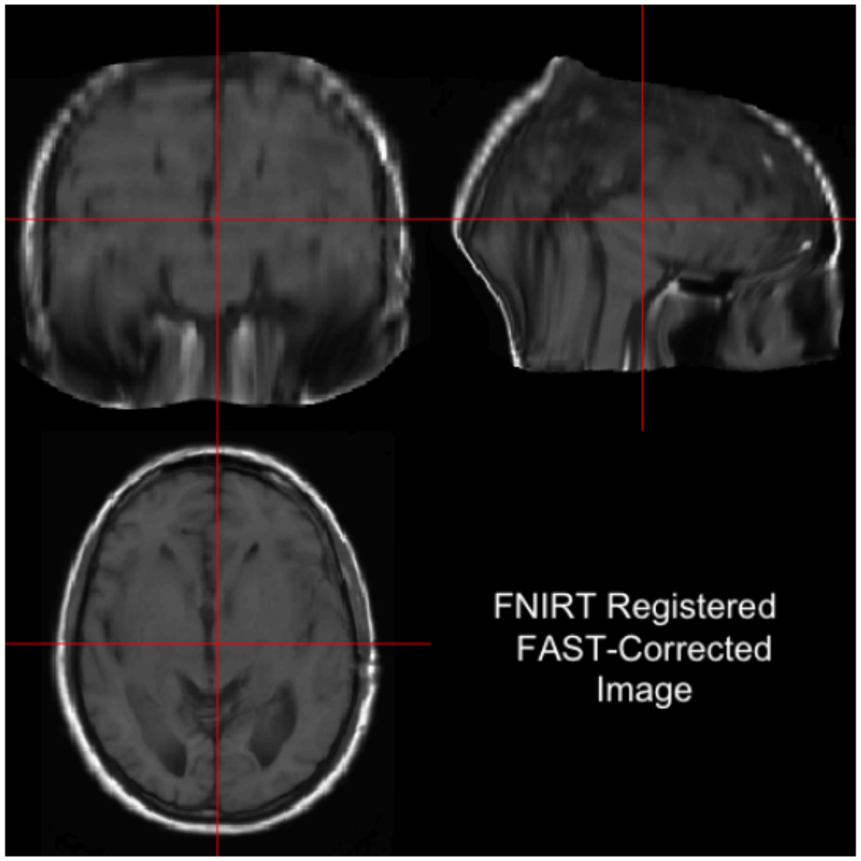
FNIRT performs non-linear registration. An affine registration must be performed before using FNIRT

- `fslr::fnirt_with_affine:` affine registration + FNIRT
- perform this on skull-stripped images
- this may take a while

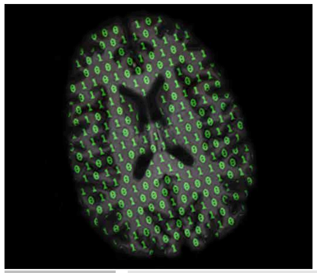
```
fnirt_fast = fnirt_with_affine(infile=bet_fast2,
reffile = template, outfile = "FNIRT_to_Template",
retimg=TRUE)
```

fslr: Image Registration (Nonlinear) Results





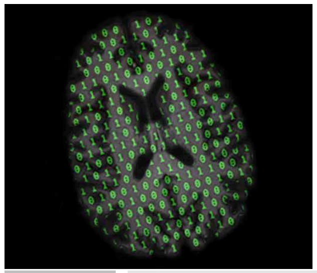
THEME 3 / LECTURE 7: INSTALLING ANTSR



Pre-Processing with ANTsR

- Advanced normalization tools (ANTS): software that can perform many neuroimaging-related functions
  - Collection of routines in C, C++, and some R
- `ANTsR`: port of `ANTS` into `R` using `Rcpp`
- We focus on
  - Image inhomogeneity correction (N3 and N4)
  - Image registration

J.G. Sled, A.P. Zijdenbos, and A.C. Evans. A nonparametric method for automatic correction of intensity nonuniformity in MRI data". In: Medical Imaging, IEEE Transactions on 17.1 (1998), pp. 87-97
N.J. Tustison et al. N4ITK: improved N3 bias correction". In: Medical Imaging, IEEE Transactions on 29.6 (2010), pp. 1310-1320.
http://www.nitrc.org/projects/antsr/

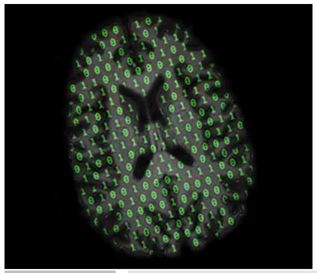


ANTsR installation

- **Linux/Mac**: https://github.com/stnava/ANTsR, GitHub (details to follow)

- **Windows:** https://smart-stats-tools.org/mooc-2015
  - Recommended approach for Windows
  - Download VirtualBox a free x86 and AMD64/Intel64 virtualization product.
  - The Virtual Machine compressed image (~3.3GB) can be downloaded from here. Make sure you move the archive to a folder of your choice and then decompress it
  - Install the VirtualBox and once that's complete add the virtual machine using the VirtualBox main menu Machine->Add or simply press the CTRL+A combination. A file browser window will open and you need to navigate to the folder created in step 2 and locate the virtual machine image file
  - Once the MOOC virtual machine is added to the VirtualBox start it up by pressing the green Start arrow. The username is fsluser and the password is fsluser, change it if needed
  - This virtual machine comes with all necessary R packages pre-installed (including ANTsR)
  - MOOC sample images were provided and they are located on your virtual machine desktop under "MOOC 2015 data" folder.
  - The virtual machine already has installed the necessary tools for setting up a shared folder with the native operating system (WIN). Please check this easy tutorial that explains the process. Once the folder is set, a reboot of the virtual machine is required and you should be able to access the shared folder by opening the "shared" desktop folder.
  - Data are now loaded in

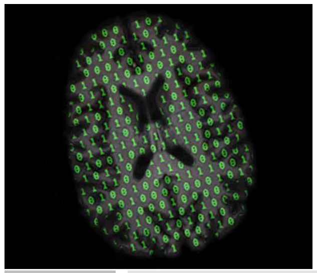
```
"/home/fsluser/Desktop/MOOC-2015"
```



Installing ANTsR (Unix/Mac)

- ANTsR is currently (as of March 23, 2015) hosted on GitHub
- We will install ANTsR using the devtools package
- Overall, any updates to the install process will be located at

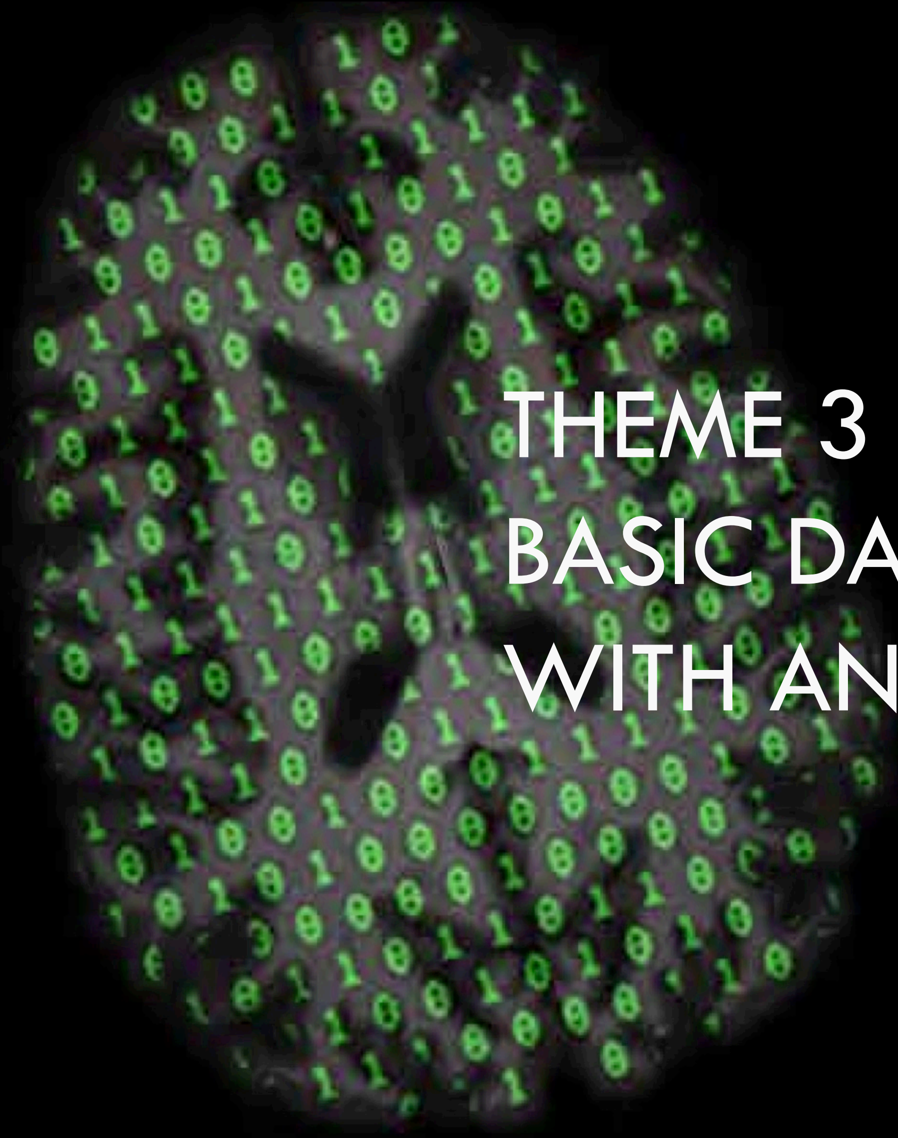
https://github.com/stnava/ANTsR



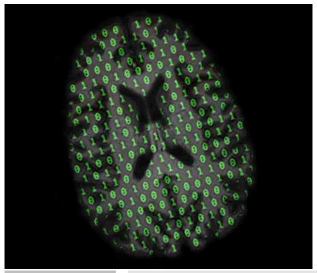
Installing ANTsR

```
if (!require(devtools)){install.packages(devtools)}
devtools::install_github("stnava/cmaker")
devtools::install_github("stnava/ITKR")
devtools::install_github("stnava/ANTsR")
```

This takes ~20 minutes

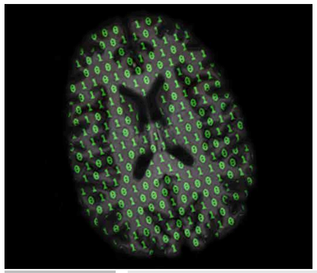


THEME 3 / LECTURE 8: BASIC DATA MANIPULATION WITH ANTSR



Basic Data Manipulation with ANTsR

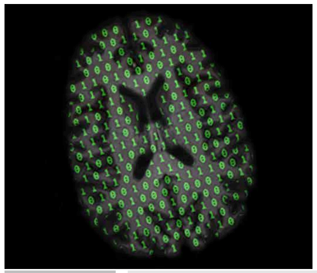
- Reading images
- ANTsR Images
- Basic Statistics
- ANTs image class



Reading Images Using ANTsR

- Requires 2 changes compared to readNIfTI from oro.nifti
  - The extension of the filename (e.g. .nii.gz) must be specified
  - The dimension of the image must be supplied (could be 2, 3, or 4)

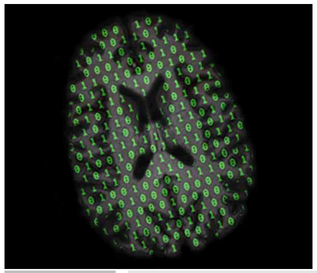
```
library(ANTsR)
aimg=antsImageRead("113-01-MPRAGE.nii.gz",dimension=3)
```



ANTsR Images

- The aimg object is an object of antsImage, which consists of:
  - pixeltype - how is the image stored (integers versus fractional numbers)
  - dimension - how many dimensions does the image have
  - pointer - where the data is stored

```
class(aimg)
[1] "antsImage"
attr(,"package")
[1] "ANTsR"
aimg
antsImage
Pixel Type : float
Pixel Size : 1
Dimensions : 512x512x22
Voxel Spacing: 0.46875x0.46875x5
Origin : 0 0 0
```



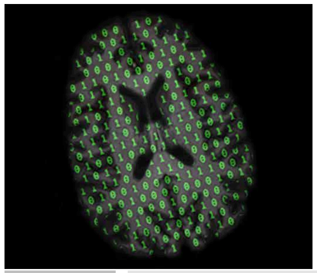
ANTsR Images: Statistics

- Basic data manipulations can be done on the `antsImage`

```
mean(aimg)
[1] 102.4701
mean(aimg[aimg!=0])
[1] 179.4116
```

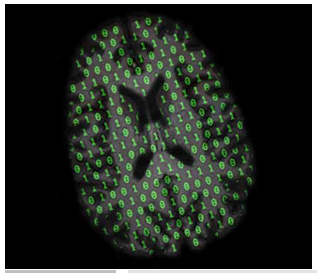
- The image can be obtained from `antsImage` using `as.array`

```
class(as.array(aimg))
[1] "array"
```



Why Discuss the `antsImage` Class?

- The class can be very fast at performing operations
- Some ANTsR functions return object of `antsImage` class
- Some ANTsR functions require an object of `antsImage` class as input

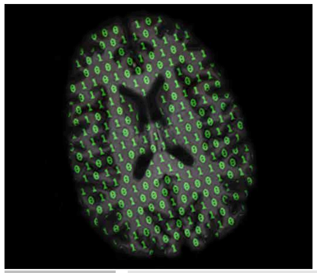


From `antsImage` to `nifti`

- The `extrantsr` (EXTRa ANTsR) package has helper functions to jump from ANTsR to the oro.nifti classes:
- Installing extrantsr:

```
devtools::install_github("muschellij2/extrantsr")
library(extrantsr)
class(nim <- ants2oro(aimg))
[1] "nifti"
attr(,"package")
[1] "oro.nifti"
```

THEME 3 / LECTURE 9: PRE-PROCESSING WITH ANTSR



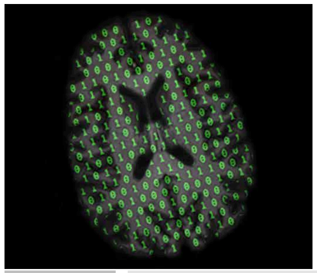
Pre-processing with ANTsR

- Bias field correction (N4)
- Registration

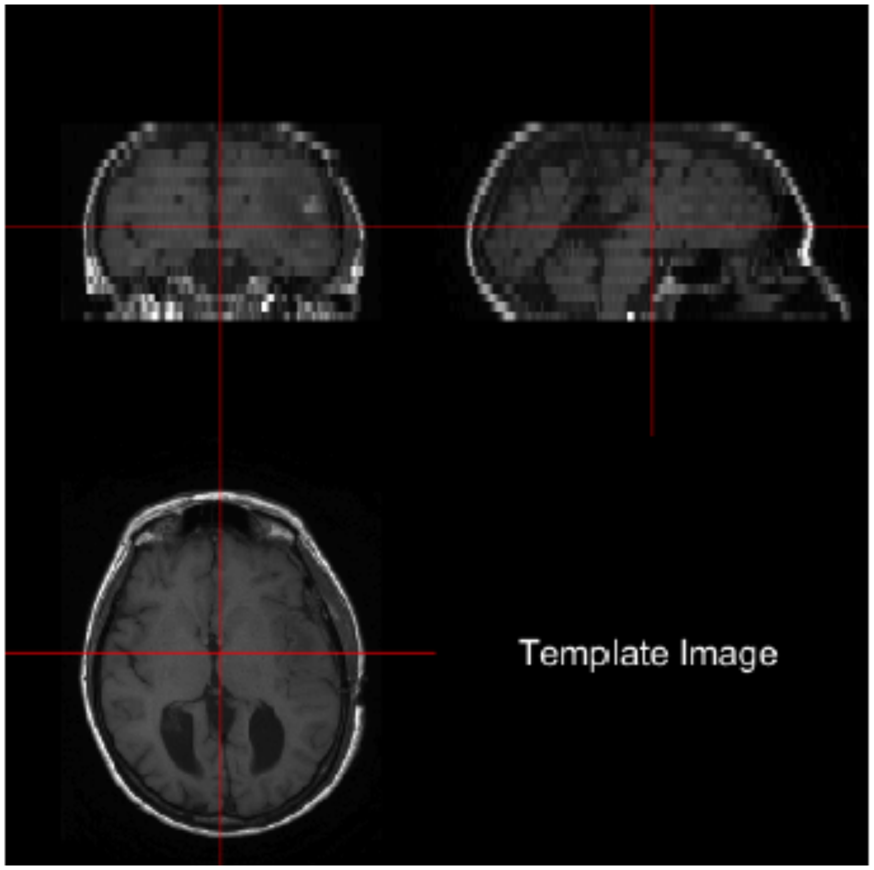
Wrapper Functions in `extrantsr`

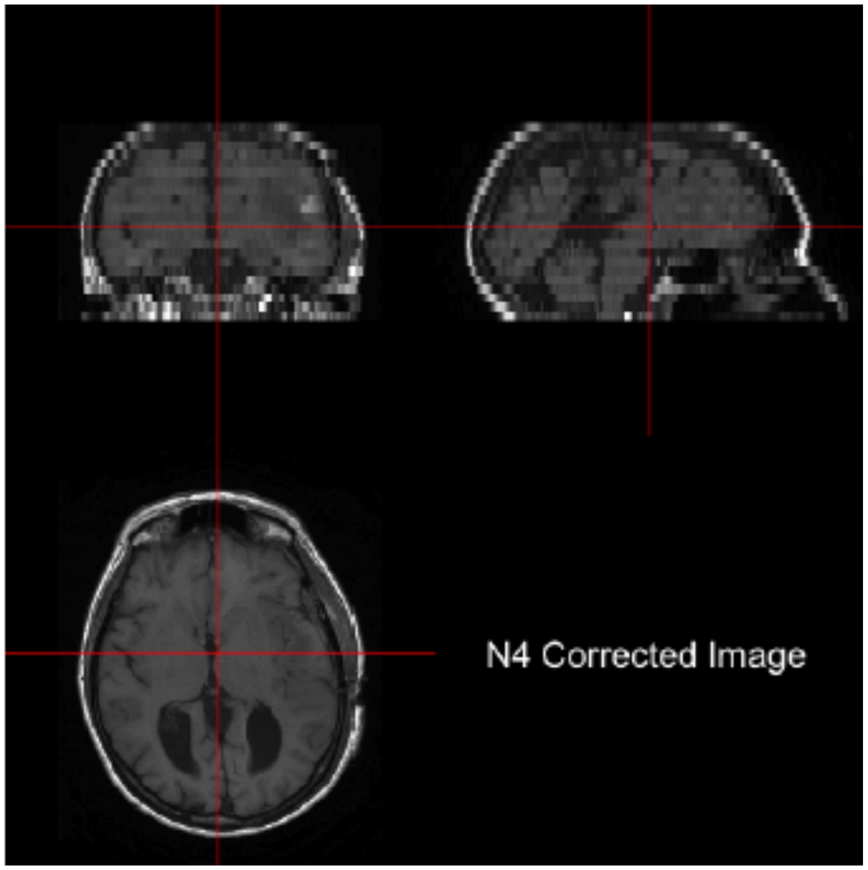
- Bias field correction
- `extrantsr::bias_correct` wraps `n3BiasFieldCorrection` and `n4BiasFieldCorrection` from `ANTsR` for bias field correction:

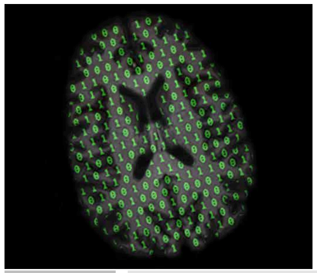
```
n3img = bias_correct(nim, correction = "N3",retimg=TRUE)
n4img = bias_correct(nim, correction = "N4",retimg=TRUE)
```



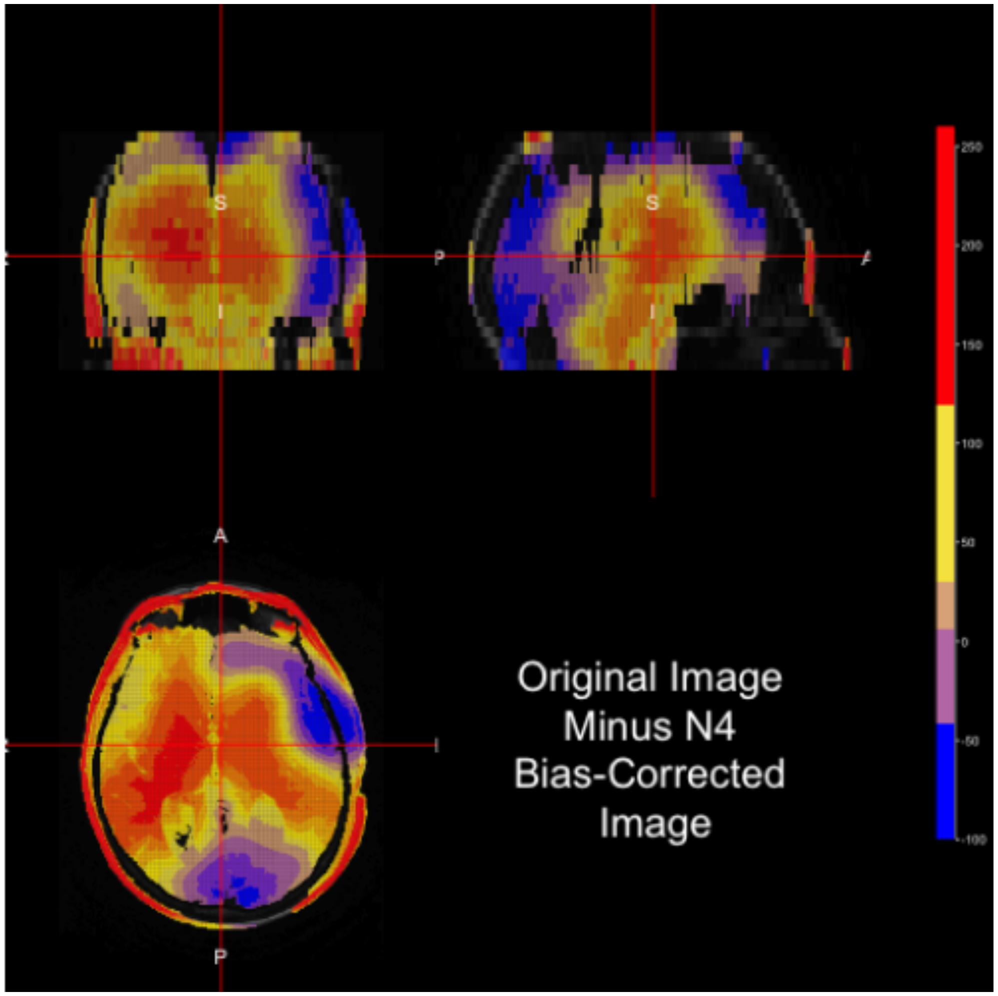
Bias Field Correction with `ANTsR`

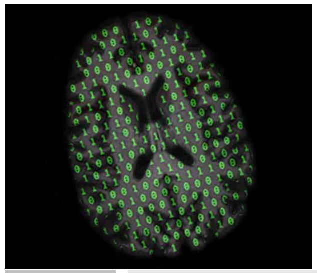






fslr: Original-N4 Bias Field Corrected Image





Wrapper Functions in `extrantsr`

- Image registration
- `ANTsR` **worker function:** `antsRegistration`
- `extrantsr` **worker function:** `ants_regwrite`
- `ants_regwrite` takes in a filename and a template filename. The image in the filename is transformed to the space of the template filename

```
registered_n4 = ants_regwrite(filename=n4img,
template.file = template, remove.warp = TRUE,
typeofTransform = "Rigid")
```

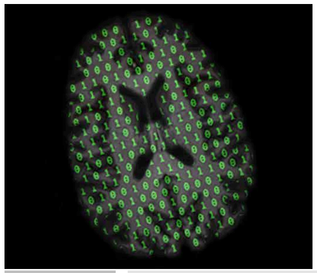
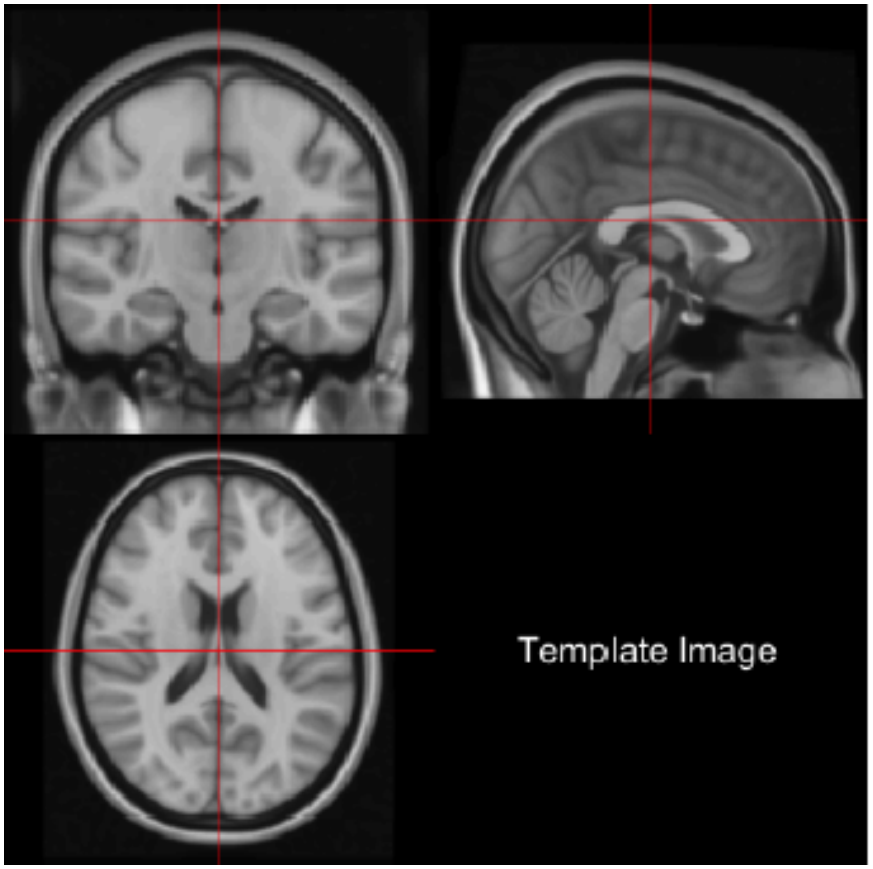
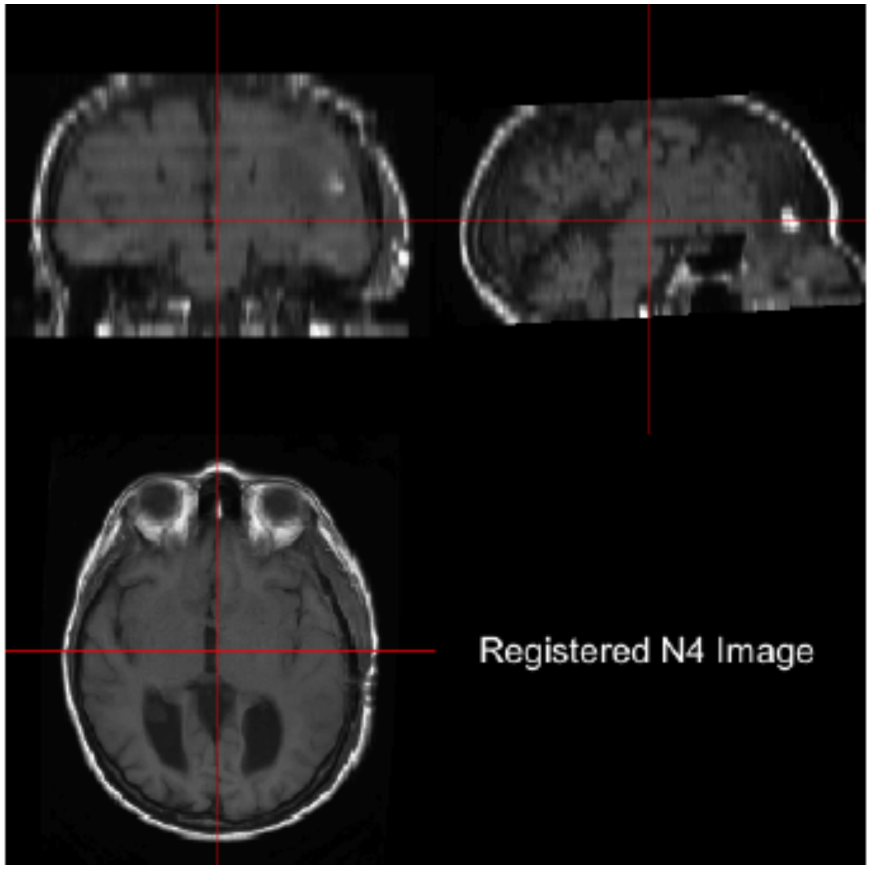
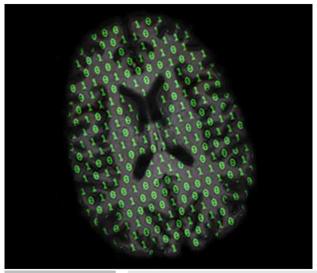


Image Registration with `ANTsR`







Other Options for Registration

```
typeofTransform = "Rigid", "Affine", "SyN"
```