fslr: Connecting the FSL Software with R

by John Muschelli, Elizabeth Sweeney, Martin Lindquist, and Ciprian Crainiceanu

Abstract We provide the package **fslr**, a set of R functions that interface with FSL (FMRIB Software Library), a commonly-used open-source software package for processing and analyzing neuroimaging data. **fslr** performs operations on `nifti` image objects in R using command-line functions from FSL, and returns R objects back to the user. **fslr** allows user to develop image processing and analysis pipelines based on FSL functionality while interfacing with the functionality provided by R. We present an analysis example of structural magnetic resonance images, which demonstrates how R users can leverage the functionality of FSL without switching to shell commands.

Introduction

FSL (FMRIB Software Library) is a commonly-used software for processing and analyzing neuroimaging data (Jenkinson et al., 2012). This software provides open-source command-line tools and a graphical user interface (GUI) for image processing such as image smoothing, brain extraction (Smith, 2002), bias-field correction, segmentation (Zhang et al., 2001), and registration (Jenkinson and Smith, 2001; Jenkinson et al., 2002). Many of these functions are used extensively in medical imaging pipelines.

R contains a large number of packages for reading and manipulating imaging data, including **AnalyzeFMRI** (Bordier et al., 2011), **RNiftyReg** (Clayden, 2013), and **fmri** (Tabelow and Polzehl, 2011) (see the Medical Imaging CRAN task view `http://cran.r-project.org/web/views/MedicalImaging.html` for more information). Although these packages are useful for imaging-related analysis, much of the fundamental functionality FSL and other imaging software provide is not implemented in R. Instead of re-implementing FSL functions in R, we propose to develop a user-friendly interface between R and FSL that preserves all the functionality of FSL and complements with R functionality. This will allow R users to implement imaging pipelines without learning software-specific syntax.

The **fslr** package relies heavily on the **oro.nifti** (Whitcher et al., 2011) package implementation of the "nifti" S4 class for storage of images that are in the Neuroimaging Informatics Technology Initiative (NIfTI) format and other common image formats such as ANALYZE. **oro.nifti** also provides useful functions for plotting and manipulating images. In addition to interfacing FSL and R, **fslr** expands on the **oro.nifti** package by adding additional functions for manipulation of `nifti` objects.

fslr Workflow

The general workflow for most **fslr** functions that interface with FSL is as follows:

1. Pass filename or `nifti` object to **fslr** function.
2. FSL command is created within **fslr** function and executed using the `system` command.
3. Output is written to disk and/or read into R and returned from the function.

From the user's perspective, the input/output process has been within R, where R objects are input into the function and R objects are returned. The advantage of this process is that a user can read in an image, do manipulations of the `nifti` object using standard syntax for arrays, and then pass this object into the **fslr** function without using FSL-specific syntax written in a shell language. Also, one can perform image operations using FSL, perform operations on the `nifti` object in R that would be more difficult using FSL, and then perform additional operations using FSL by passing that object to another **fslr** command. Thus, users can create complete pipelines for analysis using FSL written using only `fslr` commands.

fslr Setup

To use **fslr**, a working installation of FSL is required. **fslr** must also have the path of FSL specified. If using R from a shell environment, and the `FSLDIR` environment variable is set (which can be done when installing FSL), **fslr** will use this as the path to FSL. If using R through a graphical user interface (GUI) such as RStudio (RStudio, Boston, MA), environmental variables and paths are not explicitly exported. Therefore, `FSLDIR` is not set, and we can specify the path to FSL using `options(fsl.path="/path/to/fsl")`.

fslr also requires an output type for the format of images returned from FSL. Some **fslr** functions produce intermediate files that the user may want removed after the command is executed and

the extension for the file is required. Again, if working in a shell environment, **fslr** will use the environment variable for output type `FSLOUTPUTTYPE`. If working in a GUI, the default is given by `NIFTI_GZ`, which returns compressed NIfTI images, ending in ".nii.gz". This can be changed by setting the `fsl.outputtype` option (`options(fsl.outputtype="OUTPUTTYPE")`). See http://fsl.fmrib.ox.ac.uk/fsl/fsl-4.1.9/fsl/formats.html for a description of FSL output types.

```
library(fslr)
options(fsl.path="/usr/local/fsl")
options(fsl.outputtype = "NIFTI_GZ")
```

Image Preprocessing with fslr

We will present an analysis of structural magnetic resonance imaging (MRI) studies completely within **fslr** and R. Images were obtained from a patient with multiple sclerosis (MS) at 2 different visits (SWEENEY CITE). At each visit, the image modalities obtained were T1, T2, fluid-attenuated inversion recovery (FLAIR), and proton density (PD). We will co-register scans within a visit, perform a MRI bias-field correction using FAST (FMRIB's Automated Segmentation Tool) (Zhang et al., 2001), co-register scans within visits to the T1 image of that visit, and register T1 images between visits. Once these operations have been performed, one can take within-modality difference images to see the changes between visits. We will also register all images to a template, as this is common in population-based analyses.

Bias-Field Correction

Imaging acquired using MRI have good properties such as good contrast between soft tissue classes, but intensity inhomogeneities in the radio frequency (RF) field can cause different ranges of tissue types at different spatial locations. These inhomogeneities can cause problems with algorithms based on histograms, quantiles, or raw intensities (Zhang et al., 2001). Therefore, corrrection for image homogeneties is a crucial step in many analyses. FSL implements the bias-field correction from Guillemaud and Brady (1997) in its FAST segmentation pipleine (Zhang et al., 2001).

The `fsl_biascorrect` from **fslr** will create the corrected images. We pass in the filename in the `file` argument, any additional options, such as `-v` for **v**erbose diagnostic outputs, and the **out**put **file**name (`outfile`), which is our inhomogeneity-corrected image.

```
head(df, 2)
```

```
fsl_biascorrect(file = "01-Baseline_T1.nii.gz", opts= "-v",outfile= "01-Baseline_T1_FSL_BiasCorrect",)
```

```
for (ifile in seq_along(df)){
  bias_file = df$bias_file[ifile]
  file = df$file[ifile]
  fsl_biascorrect(file, opts= "-v", outfile=bias_file)
}
```

We can observe the difference in voxel values from the baseline T1 image compared to the bias-corrected version in Figure 1. In panel a we display the T1 image, and in panel b we display the bias-corrected T1 image. The T1 image looks brighter in the middle of the image while the bias-corrected image looks more uniform in the white matter (brighter regions). This difference may be hard to distinguish visually, so we present the scatterplot of these images in Figure 1c, using the **ggplot2** package (Wickham, 2009). Note, the scales are in arbitrary units (a.u.).

The blue line in Figure 1c represents the 45° diagonal line, where the original and bias-corrected image intensities are equal, and the pink represents a generalized additive model (GAM) (Hastie and Tibshirani, 1990) smoother to estimate the shape of the relationship using the **mgcv** package (Wood, 2011). We see that for values in the low range of the data (< 40), the T1 values and bias-corrected T1 values, on average, fall along the diagonal, but values in the higher range are lower in the bias-corrected T1 values.

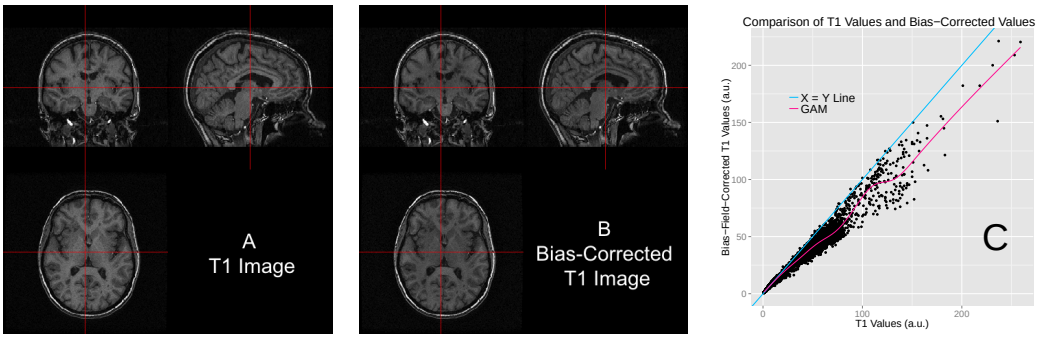

Figure 1: Results of Inhomogeneity Correction. We present the original T1 image (a), bias-corrected T1 image (b), and the scatterplot of the sampled values comparing the values from the T1 image and bias-corrected values (c). We see in panel c for values in the low range of the data (< 40), the T1 values and bias-corrected T1 values, on average, fall along the diagonal (blue line), but values > 40 are lower in the bias-corrected T1 values shown by a generalized additive model (GAM) smoother (pink line).

Within-Visit Co-registration

All subsequent steps will be done on the bias-corrected images. We will first co-register the images within each separate visit to the T1 image from that visit. This registration allows us to investigate joint distributions of voxel intensities of different image modalities. We will register images within a visit using FMRIB's Linear Image Registration Tool (FLIRT) (Jenkinson and Smith, 2001; Jenkinson et al., 2002). As the images are from the same individual, we may assume that the overall shape of the brain has not changed, but each scan may have undergone a translation and/or rotation in space. Therefore, we will use a rigid-body transformation, with 6 degrees of freedom (DOF).

We will use another `data.frame` with image input and output names, including the T1 image of that visit, the image to be registered, the output transformation matrix and output filename.

```
head(reg_df, 2)
```

The **fslr** command `flirt` will call the FSL command `flirt`, taking in the input image (`infile`) and the reference image to be registered to (`reffile`). Any additional options can be passed to the FSL command using the `opts` argument. We will use the defaults (i.e. trilinear interpolation) and the `-v` option for diagnostic messages to be printed. We are doing a rigid-body transformation so we set the degrees of freedom (`dof`) to 6. Here we present the code for registering the baseline T1 image to the baseline T2 image; we will do this process the T2, FLAIR and PD images, for the baseline and followup scans.

```
flirt(reffile = "01-Baseline_T1_FSL_BiasCorrect",
        infile = "01-Baseline_T2_FSL_BiasCorrect",
        omat = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1.mat",
        outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1",
        dof = 6, opts = "-v")
```

The resulting image transformation will be stored in the file name passed to the `omat` (**o**utput **mat**rix) argument. This matrix can be used to transform other images that were in the same space as the input image to the reference image space. After co-registration, one could compare images of different modalities at the same voxels, e.g. T1 versus FLAIR images.

In the previous example, we present a rigid-body transformation, using the default parameters. `flirt` has options for different cost functions to optimize over, interpolation operators to estimate voxel intensity, and additional degrees of freedom for performing affine transformations. These options can be passed to the FSL `flirt` command using the `opts` argument in `flirt` in **fslr**.

Note that each **fslr** function has a corresponding help function, which is the **fslr** command appended with `.help()`, which will print out the FSL help page for that function. For example, users can see which options can be changed in the `flirt` command using the `flirt.help()` function. Additional non-linear registration techniques are presented in section "Registration to the MNI Template".

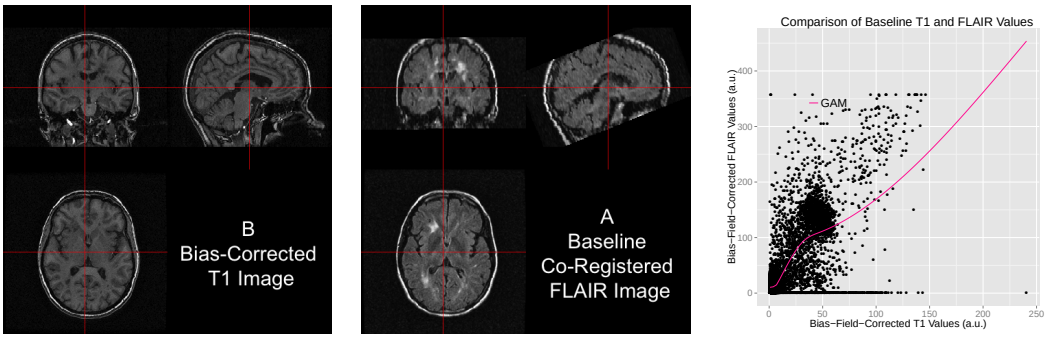

Figure 2: Results of Within-Visit Co-registration. We present the bias-corrected T1 image (a), co-registered bias-corrected FLAIR image (b), and the scatterplot of the sampled values comparing the values from the T1 image and bias-corrected values (c). We see in panel c a generalized additive model (GAM) smoother (pink line).

Between-Visit Co-registration

Though within-visit, across-modality comparisons can be achieved by within-visit co-registration, across-visit registration is required for assessing within-modality differences between longitudinal scans. To compute difference images, we co-register followup images to the baseline images within each modality. Similar to the within-visit co-registration, we use a rigid-body transformation. We present 4 options for between-visit registration:

1. Co-register images in the space they were acquired (referred to as native space),
2. Register the T1 images from baseline and followup, and use this transformation to compare the co-registered-to-T1 images from above, within modality,
3. Co-register the co-registered-to-T1 images to each other, within modality,
4. Co-register the T1 images and use the composition of the transformation from registration to the T1 and the T1-to-T1 registration to register within-modality, across-visit images.

(SWEENEY REFS)

The difference between options 2, 3, 4 are subtle in this case. In options 2 and 3, the co-registered-to-T1 images have had the intensities of voxels interpolated (i.e. locally averaged) when transformed to the T1 space. In option 2, we estimate one transformation from the T1 images for each modality, which saves time computationally, but may not be the best transformation for a given modality. Option 3 allows for a modality-specific transformation, but images within the same visit are no longer necessarily aligned in the same space. Option 4 is similar to option 1 in that it only has one interpolation done, but can put all the transformed images in the same space. Each of these options have different advantages, depending on the desired analysis.

Though two interpolations are done in option 2 and may not be optimal for within-modality comparisons, we will present that option here as we have already obtained the co-registered-to-T1 images in the section "Within-Visit Co-registration" and only one additional transformation needs to be performed. This operation also demonstrates how to use transformation matrices in **fslr**. We will register the followup T1 image to the baseline T1 image, again using a rigid-body transformation (6 degrees of freedom):

```
flirt(reffile = "01-Baseline_T1_FSL_BiasCorrect",
        infile = "01-Followup_T1_FSL_BiasCorrect",
      omat = "01-Followup_T1_FSL_BiasCorrect_rigid_to_BaseT1.mat",
      dof = 6,
      outfile = "01-Followup_T1_FSL_BiasCorrect_rigid_to_BaseT1",
      opts = '-v')
```

Now, both T1 images are aligned in the space of the baseline T1 image. We can observe the results in Figure 3: the bias-corrected baseline T1 image is presented in a and the co-registered bias-corrected followup T1 is presented in b. We observe that the images displayed at the same cross section correspond to the same brain area, indicating a good registration.

Using the `flirt_apply` function from **fslr**, we can apply the transformation matrix to the T2, PD, and FLAIR images from the followup visit, previously co-registered to the T1 from followup, to transform them to the baseline T1 image space. Here we present the case of the T2 followup image:

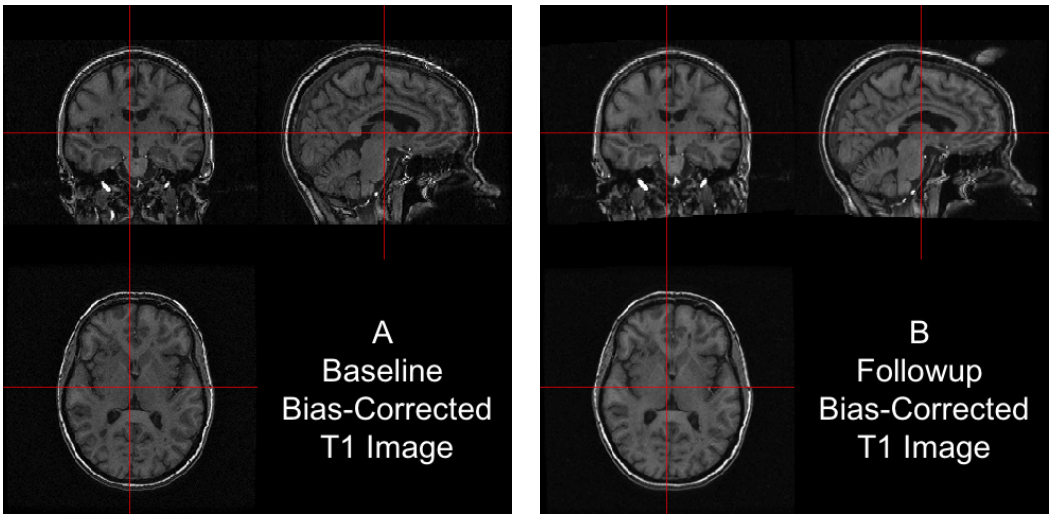

Figure 3: Results from FLIRT. We present the followup T1 co-registered to the to baseline T1, each bias-corrected. The baseline T1 is presented in (a) and the registered followup T1 is presented in (b), each displayed at the same intersection. We observe that the images displayed correspond to the same brain area, indicating a good registration.

```
flirt_apply(reffile = "01-Baseline_T1_FSL_BiasCorrect", # register to this
            infile = "01-Followup_T2_FSL_BiasCorrect_rigid_to_T1", # reg to Followup T1
            initmat = "01-Followup_T1_FSL_BiasCorrect_rigid_to_BaseT1.mat", #transform
            outfile = "01-Followup_T2_FSL_BiasCorrect_rigid_to_BaseT1" # output file
            )
```

In Figure 4, we display each image after FLIRT has been applied. Each image is in the baseline T1 image space, displayed at the same cross section. Each panel show the same brain areas across modalities, indicating an adequate registration. We see that some areas of the brain are cropped from the field of view, which may be problematic if relevant brain areas are removed. We have registered all images with the skull and extracranial tissue included in the image. A better process may be doing registration with brain tissues only. In order to analyze only brain tissues, we must perform brain extraction.

Brain Extraction

In many applications, researchers are only interested in brain tissues. The process of extracting the brain, referred to as brain extraction or skull stripping, is a crucial step in many analyses. We will perform brain extraction using FSL's brain extraction tool (BET) (Smith, 2002; Jenkinson et al., 2005) using parameters recommended by Popescu et al. (2012), which was derived from patients with MS. No other published R package has brain extraction functionality for brain imaging.

```
fslbet(infile =  '01-Baseline_T1',
       outfile = "01-Baseline_T1_FSL_BiasCorrect_Brain",
       opts = "-B -f 0.1 -v",  # from Popescu et al.
       betcmd = "bet",
       intern=FALSE)
```

We ran BET on the non-corrected T1 image as the `-B` option does inhomogeneity correction from FAST as part of the procedure. The option `-f 0.1` denotes the fractional intensity (FI) parameter in BET: it varies between 0 and 1 and determines the location of the edge of the segmented brain image; smaller values correspond to larger brain masks. We can observe the results from running BET in Figure 5: the bias-corrected T1 image is shown with the brain mask overlaid in red (panel a) and the resulting masked brain (panel b). We see that the brain extraction performed well, not including any areas of the skull or the neck while not discarding areas of the brain. Towards the back of the brain, some areas of the subarachnoid space remain, which may be unacceptable for certain analyses, such as estimation of the volume of brain tissue.

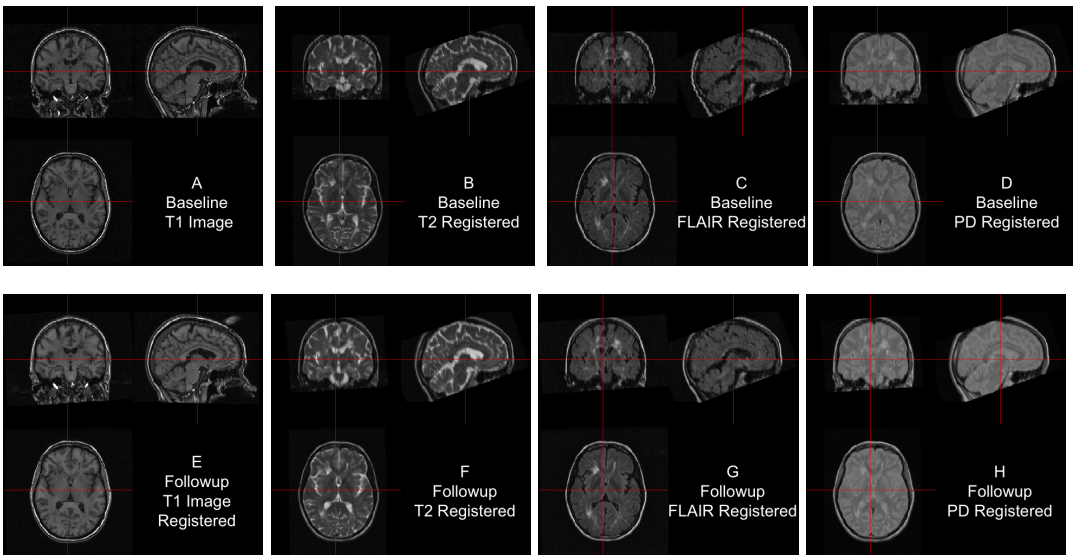

Figure 4: Across-Visit Registration Results Here we present all acquired image modalities, first co-registered within visit to the T1 image of that visit, then registered to the baseline T1 image using the followup T1 to baseline T1 transformation matrix. All registrations used rigid-body transformations.

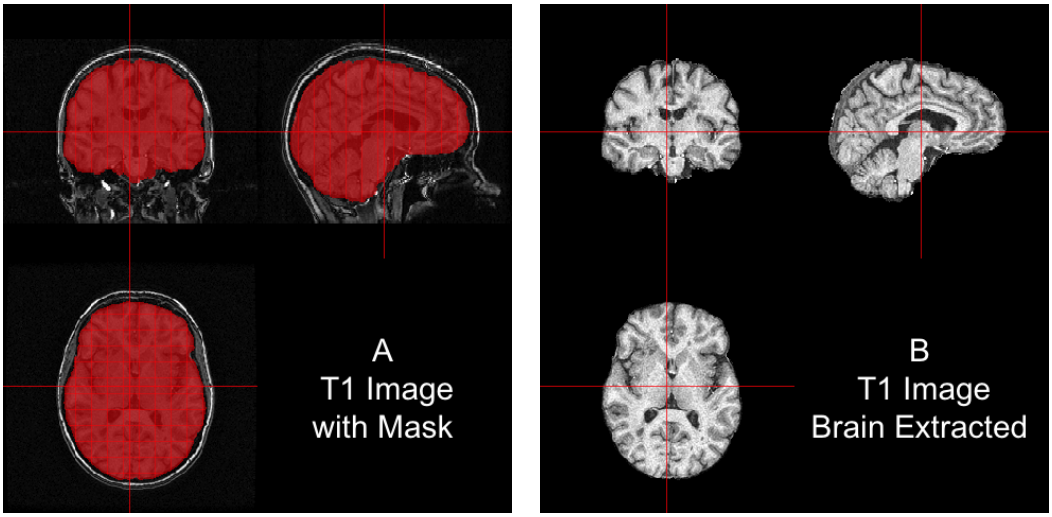

Figure 5: Results from BET. In (a), we show the bias-corrected T1 image is presented with the mask from BET overlaid in red. In (b), we display the extracted brain. We see that the brain extraction performed well, not including any areas of the skull or the neck while not discarding large areas of the brain.

Note that `fslbet` writes both a file containing the brain-extracted image and another image containing the binary brain mask. As all other images are represented in the baseline T1 space, we can use this mask to extract the brain from other images, such as the baseline T2 image, using the **fslr** function `fslmask`.

```
fslmask(file="01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1",
        mask = "01-Baseline_T1_FSL_BiasCorrect_Brain_Mask",
        outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain")
```

We now have all images in the same sterotaxic space with the brain extracted. Many analyses of structural imaging require data to be organized in this representation.

Registration to the MNI Template

In many imaging analyses, however, information is aggregated across a population of images from different participants. For the information to have the same interpretation spatially, images need to be aligned in the same stereotaxic space ("template" space), requiring registration to a template image. A frequently used set of templates are provided by MNI (Montreal Neurological Institute). We have registered the baseline T1 image to the MNI T1 template (Grabner et al., 2006), included with FSL. As an individual's brain does not necessarily have the same size as the template, it is not appropriate to use rigid-body transformations.

We will first register the baseline T1 image to the T1 template using an affine registration, which can scale, shear, and flip the brain. Although an affine transformation has more degrees of freedom than a rigid transformation, it may not be sufficient for analysis. We will then use FNIRT (FMRIB's nonlinear image registration tool) to achieve better overlap of local brain structures(Jenkinson et al., 2012; Andersson et al., 2007). As we are concerned with good overlap only in brain structures and not other areas, such as the skull, we will register our brain-extracted brain images to the brain-only template. The **fslr** function `fnirt_with_affine` will register using `flirt` with an affine transformation and then non-linearly register this image to the template using `fnirt`.

```
fnirt_with_affine(infile = "01-Baseline_T1_FSL_BiasCorrect_Brain",
                  reffile = file.path(fsldir(), "data", "standard", "MNI152_T1_1mm_brain"),
                     flirt.omat = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI.mat",
                     flirt.outfile = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI",
                     outfile = "01-Baseline_T1_FSL_BiasCorrect_Brain_toMNI")
```

The results of the registration can be seen in Figure 6. Each panel represents a different axial slice (25, 45, 92, or 137) in the template space of the template image (a, c, e, g) or the registered T1 image (b, d, f, h). Each slice shows the registered T1 image has similar brain structure represented in the same area as the template image, indicating good registration.

Applying Transformations to Co-registered Data

Since all the data is represented in the baseline T1 image space, we can apply the estimated affine transformation and non-linear warping coefficient field to each image to represent that image in template space. The affine transformation must be applied with `flirt_apply` and the warping coefficient using `fsl_applywarp`.

Here we present the application of the transformations to the baseline T2 image, previously registered to the baseline T1.

```
flirt_apply(infile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain",
            reffile = file.path(fsldir(), "data", "standard", "MNI152_T1_1mm_brain"),
            initmat = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI.mat",
            outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain_toMNI")
fsl_applywarp(infile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain_toMNI",
              reffile = file.path(fsldir(), "data", "standard", "MNI152_T1_1mm_brain"),
              warpfile = "01-Baseline_T1_FSL_BiasCorrect_Brain_affine_toMNI_warpcoef",
              outfile = "01-Baseline_T2_FSL_BiasCorrect_rigid_to_T1_Brain_toMNI")
```

With multiple participants, this process yields a multi-person, multi-modal, longitudinal imaging dataset that can be used for analyses.

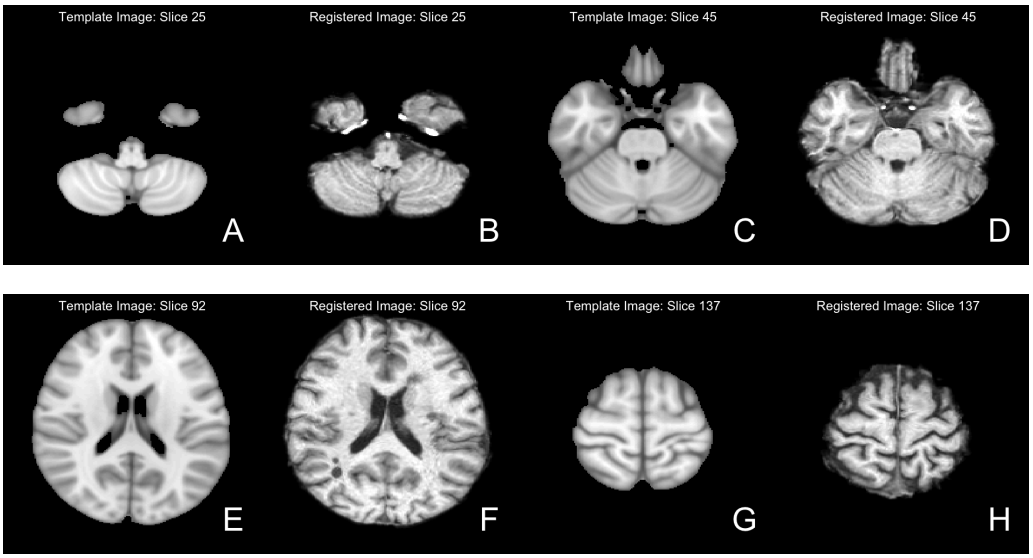

Figure 6: Results from FNIRT. We present different axial slices of the template (a, c, e, g) and the registered T1 image (b, d, f, h). The slices represented are 25 (a, b), 45 (c, d), 92 (e, f) and 137 (g, h). We note that areas of the brain coincide between the template and registered image.

Conclusion

The neuroimaging community has developed a large collection of tools for image processing and analysis. Although R has a number of packages to perform operations on images, much of the fundamental functionality of image processing is not currently available in R. We provide **fslr** to provide R users functions for image processing and analysis that are based in FSL, an established image processing and analysis software suite. Interfacing R with existing, powerful software provides users with already-tested software and an additional community of users, which would not be available if the functions were rewritten in R. **fslr** should be easy to use for any standard R user; the workflow allows R users to manipulate array-like `nifti` objects, pass them to **fslr** functions, which return `nifti` objects. Moreover, as FSL and R are open source and free, this software is readily available to all users.

There has been an increasing popularity of similar interfacing of tools within the Python community such as Nipype (Gorgolewski et al., 2011) (https://qa.debian.org/popcon.php?package=nipype). As many users of R may not know Python or bash scripting, we believe **fslr** provides a lower threshold for use in the R community. Other packages provide R users additional neuroimaging processing functionality. For example, we present one inhomogeneity correction method, yet other inhomogeneity correction methods exist, such as the popular N3 (Sled et al., 1998) and novel N4 (Tustison et al., 2010), which are not implemented in **fslr**. ANTsR (`http://stnava.github.io/ANTsR/index.html`), an R package that interfaces with the ANTs (advanced normalization tools) software suite (Avants et al., 2011), has implementations of these correction methods and an increased set of registration techniques within R.

Most importantly, as **fslr** is based on the R framework, all the benefits of using R are available, such as dynamic doculments, reproducible reports, customized figures, and state-of-the-art statistical software. These tools provide unique functionality compared to other software packages for neuroimaging.

Supplemental Material

Additional fslr Functionality

Although the main goal of **fslr** is to interface R and FSL, there are a set of functions in **fslr** that are not designed to interface with FSL, but rather provide helper functions `nifti` objects form the **oro.nifti** package. We will display 2 example functions: `cal_img`, a function to reset the `cal_min` and `cal_max` slots on a `nifti` object, which are used to determine colors when plotting and `niftiarr`, which copies a `nifti` object and replaces the `.Data` slot with a provided array.

Let us go through 2 ways to mask an image. First we will read in the bias-corrected baseline T1

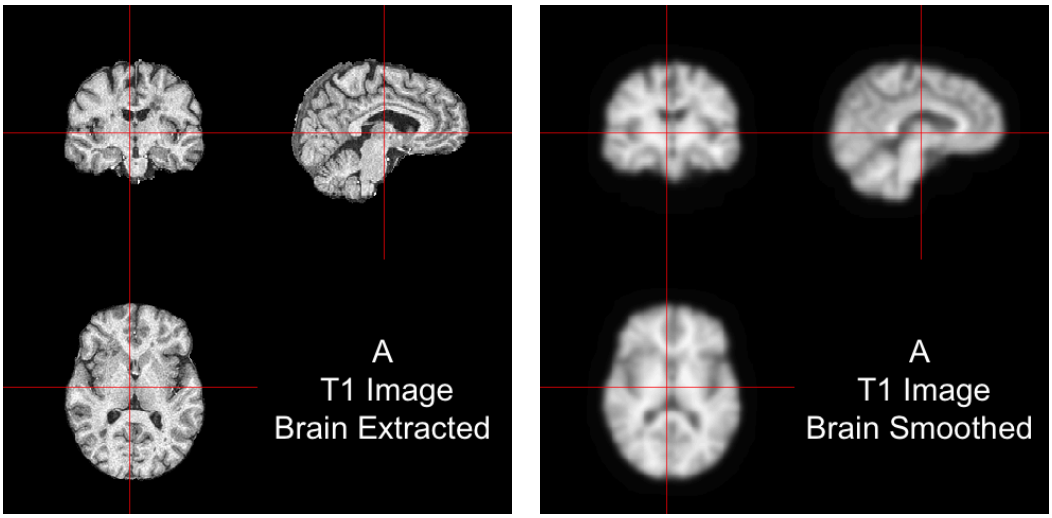

Figure 7: Smoothing the Baseline T1 Brain Image.

and the brain mask from BET:

```
base_t1 = readNIfTI("01-Baseline_T1_FSL_BiasCorrect", reorient=FALSE)
base_t1_mask = readNIfTI("01-Baseline_T1_Brain_Mask", reorient=FALSE)
```

We wish to mask the T1 image with the mask image. One way of masking an image is to multiply the image by the binary mask:

```
base_t1_1 = base_t1 * base_t1_mask
class(base_t1_1)

[1] "array"
```

We see that the resulting object is an array and not a `nifti` object. This may be a problem when trying to plot or manipulate this object using methods for `nifti` objects. To address this problem, the `niftiarr` function in **fslr** inputs a `nifti` object and an `array`, and returns a `nifti` object with the provided `array` in the `.Data` slot, copying over the image information from the input `nifti` object.

```
base_t1_1 = niftiarr(base_t1, base_t1_1)
class(base_t1_1)

[1] "nifti"
attr(,"package")
[1] "oro.nifti"
```

Another way of masking the image is to subset the values of the image that are not in the mask and setting those values to 0 (or some other value).

```
base_t1_2 = base_t1
base_t1_2[base_t1_mask == 0] = 0
class(base_t1_2)

[1] "nifti"
attr(,"package")
[1] "oro.nifti"
```

We see that this correctly returns an object of class `nifti`. One problem is that the we have changed the data in the `nifti` object `base_t1_2` but did not reset the other slots in this object to reflect this change.

The `cal_img` is a simple helper function that will set the `cal_min` and `cal_max` values to the minimum and maximum values, respectively, of the data. The `orthographic` function uses these for

plotting and this equality is required for the `writeNIfTI` function to write out `nifti` objects, from the **oro.nifti**. Let us look at the range of the data and the `cal_min` and `cal_max` slots:

```
range(base_t1_2)

[1]   0.0000 409.3908

c(base_t1_2@cal_min, base_t1_2@cal_max)

[1] 0 0
```

Note, one potential issue with `readNIfTI` function from **oro.nifti** is that the `cal_min` and `cal_max` slots were both read as zero. Let us set these to the range using the `cal_img` command from **fslr**.

```
base_t1_2 = cal_img(base_t1_2)
range(base_t1_2)

[1]   0.0000 409.3908
```

We see that after these operations are done 2 in different ways, the resulting `nifti` objects are equivalent.

```
all.equal(base_t1_1, base_t1_2)

[1] TRUE
```

Additional helper functions are included in **fslr** for plotting and image manipulation.

John Muschelli
PhD Student
Johns Hopkins Bloomberg School of Public Health,
Baltimore, MD 21231
USA jmuschel@jhsph.edu

Elizabeth Sweeney
PhD Student
Johns Hopkins Bloomberg School of Public Health
Baltimore, MD 21231
USA
Special Volunteer Translational Neuroradiology Unit, Neuroimmunology Branch, National Institute of Neurological Disease and Stroke, National Institute of Health
Bethesda, MD 20892
USA emsweene@jhsph.edu

Martin Lindquist
Associate Professor
Johns Hopkins Bloomberg School of Public Health,
Baltimore, MD 21231
USA mlindqui@jhsph.edu

Ciprian Crainiceanu
Professor
Johns Hopkins Bloomberg School of Public Health,
Baltimore, MD 21231
USA ccrainic@jhsph.edu

Bibliography

J. L. Andersson, M. Jenkinson, S. Smith, and others. Non-linear registration, aka spatial normalisation FMRIB technical report TR07ja2. *FMRIB Analysis Group of the University of Oxford*, 2007. URL http://fmrib.medsci.ox.ac.uk/analysis/techrep/tr07ja2/tr07ja2.pdf. [p7]

B. B. Avants, N. J. Tustison, G. Song, P. A. Cook, A. Klein, and J. C. Gee. A reproducible evaluation of ANTs similarity metric performance in brain image registration. *NeuroImage*, 54(3):2033–2044, feb 2011. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2010.09.025. URL http://www.sciencedirect.com/science/article/pii/S1053811910012061. [p8]

C. Bordier, M. Dojat, P. L. de Micheaux, and others. Temporal and spatial independent component analysis for fMRI data sets embedded in the AnalyzeFMRI r package. *Journal of Statistical Software*, 44(9):1–24, 2011. URL http://www.hal.inserm.fr/inserm-00659425/. [p1]

J. Clayden. *RNiftyReg: Medical image registration using the NiftyReg library, based on original code by Marc Modat and Pankaj Daga*, 2013. URL http://CRAN.R-project.org/package=RNiftyReg. R package version 1.1.2. [p1]

K. Gorgolewski, C. D. Burns, C. Madison, D. Clark, Y. O. Halchenko, M. L. Waskom, and S. S. Ghosh. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. *Frontiers in neuroinformatics*, 5, 2011. URL http://www.ncbi.nlm.nih.gov/pmc/articles/pmc3159964/. [p8]

G. Grabner, A. L. Janke, M. M. Budge, D. Smith, J. Pruessner, and D. L. Collins. Symmetric atlasing and model based segmentation: An application to the hippocampus in older adults. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. P. Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, R. Larsen, M. Nielsen, and J. Sporring, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*, volume 4191, pages 58–66. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-44727-6, 978-3-540-44728-3. URL http://link.springer.com/10.1007/11866763_8. [p7]

R. Guillemaud and M. Brady. Estimating the bias field of MR images. *Medical Imaging, IEEE Transactions on*, 16(3):238–251, 1997. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=585758. [p2]

T. J. Hastie and R. J. Tibshirani. *Generalized additive models*, volume 43. CRC Press, 1990. URL http://books.google.com/books?hl=en&lr=&id=qa29r1Ze1coC&oi=fnd&pg=PR13&dq=generalized+additive+models&ots=j3YSfqBWoR&sig=9WE9tUp7uoXJtjwCmcSLvtvSr_g. [p2]

M. Jenkinson and S. Smith. A global optimisation method for robust affine registration of brain images. *Medical image analysis*, 5(2):143–156, 2001. URL http://www.sciencedirect.com/science/article/pii/S1361841501000366. [p1, 3]

M. Jenkinson, P. Bannister, M. Brady, and S. Smith. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *Neuroimage*, 17(2):825–841, 2002. URL http://www.sciencedirect.com/science/article/pii/S1053811902911328. [p1, 3]

M. Jenkinson, M. Pechaud, and S. Smith. BET2: MR-based estimation of brain, skull and scalp surfaces. In *Eleventh annual meeting of the organization for human brain mapping*, volume 17, 2005. URL http://mickaelpechaud.free.fr/these/HBM05.pdf. [p5]

M. Jenkinson, C. F. Beckmann, T. E. J. Behrens, M. W. Woolrich, and S. M. Smith. FSL. *NeuroImage*, 62(2):782–790, aug 2012. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2011.09.015. URL http://www.sciencedirect.com/science/article/pii/S1053811911010603. [p1, 7]

V. Popescu, M. Battaglini, W. S. Hoogstrate, S. C. J. Verfaillie, I. C. Sluimer, R. A. van Schijndel, B. W. van Dijk, K. S. Cover, D. L. Knol, M. Jenkinson, F. Barkhof, N. de Stefano, and H. Vrenken. Optimizing parameter choice for FSL-brain extraction tool (BET) on 3d t1 images in multiple sclerosis. *NeuroImage*, 61(4):1484–1494, jul 2012. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2012.03.074. URL http://www.sciencedirect.com/science/article/pii/S1053811912003552. [p5]

J. G. Sled, A. P. Zijdenbos, and A. C. Evans. A nonparametric method for automatic correction of intensity nonuniformity in MRI data. *Medical Imaging, IEEE Transactions on*, 17(1):87–97, 1998. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=668698. [p8]

S. M. Smith. Fast robust automated brain extraction. *Human brain mapping*, 17(3):143–155, nov 2002. ISSN 1065-9471. doi: 10.1002/hbm.10062. [p1, 5]

K. Tabelow and J. Polzehl. Statistical parametric maps for functional MRI experiments in r: The package fmri. *Journal of Statistical Software*, 44(11):1–21, 2011. URL http://www.jstatsoft.org/v44/i11/paper. [p1]

N. J. Tustison, B. B. Avants, P. A. Cook, Y. Zheng, A. Egan, P. A. Yushkevich, and J. C. Gee. N4itk: improved n3 bias correction. *Medical Imaging, IEEE Transactions on*, 29(6):1310–1320, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5445030. [p8]

B. Whitcher, V. J. Schmid, and A. Thornton. Working with the DICOM and NIfTI data standards in r. *Journal of Statistical Software*, 44(6):1–28, 2011. URL http://www.jstatsoft.org/v44/i06/paper. [p1]

H. Wickham. *ggplot2: elegant graphics for data analysis*. Springer, 2009. URL http://books.google.com/books?hl=en&lr=&id=bes-AAAAQBAJ&oi=fnd&pg=PR5&dq=ggplot2:+elegant+graphics+for+data+analysis&ots=SA8_QA6NOO&sig=iAb1q3tX5ZkAh0lUHl2uPiySFdI. [p2]

S. N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(1):3–36, 2011. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2010.00749.x/full. [p2]

Y. Zhang, M. Brady, and S. Smith. Segmentation of brain MR images through a hidden markov random field model and the expectation-maximization algorithm. *Medical Imaging, IEEE Transactions on*, 20 (1):45–57, 2001. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=906424. [p1, 2]