freesurfer: Connecting the Freesurfer Software with R

by John Muschelli, Elizabeth M. Sweeney, Ciprian M. Crainiceanu

Abstract We present the package **freesurfer**, a set of R functions that interface with Freesurfer, a commonly-used open-source software package for processing and analyzing structural neuroimaging data. The **freesurfer** package performs operations on `nifti` image objects in R using command-line functions from Freesurfer, and returns R objects back to the user. **freesurfer** allows users to process neuroanatomical images and provides functionality to convert and read the output of the Freesurfer pipelines. We present an example of the analysis of structural magnetic resonance images, which demonstrates how R users can interface with Freesurfer and analyze the results of the complete Freesurfer analysis pipeline.

Introduction

Freesurfer is a commonly-used software for processing and analyzing anatomical neuroimaging data (Fischl, 2012), developed by the Laboratory for Computational Neuroimaging at the Athinoula A. Martinos Center for Biomedical Imaging. This software provides open-source, command-line tools for image processing tasks such as brain extraction/skull-stripping (Ségonne et al., 2004), bias-field correction (Sled et al., 1998), segmentation of structures within the brain (Fischl et al., 2002, 2004), and image registration (Fischl et al., 1999; Reuter et al., 2010). Many of these functions are used extensively in medical imaging pipelines and Freesurfer has a complete pipeline packaged with its software as well.

We have previously published a similar adaptation of the FSL imaging software (Jenkinson et al., 2012) to R, called **fslr** (Muschelli et al., 2015). Again, we note that there exist a number of R packages for reading and manipulating image data, including **AnalyzeFMRI** (Bordier et al., 2011), **RNiftyReg** (Clayden, 2015), 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 performing image analysis, much of the fundamental functionality of image preprocessing and processing that Freesurfer provides are not currently implemented in R. The **ANTsR** package (`https://github.com/stnava/ANTsR`) is a currently unpublished R package that interfaces with the ANTs (advanced normalization tools) software suite (Avants et al., 2011), where a lot of additional functionality has been implemented, but this package has not been released onto CRAN. Moreover, having multiple options for image processing through R allows for users to compare methods and the flexibility of using multiple packages to achieve a working data processing pipeline.

In particular, we provide an interface to users to the state-of-the-art anatomical processing implemented in Freesurfer, as well as a suite of tools that simply the analysis of the output of Freesurfer. This will allow R users to implement complete imaging analysis without necessarily learning Freesurfer-specific syntax.

R imaging objects

The **freesurfer** package relies on the **oro.nifti** (Whitcher et al., 2011) package implementation of images (referred to as `nifti` objects) that are in the Neuroimaging Informatics Technology Initiative (NIfTI) format, as well as other common image formats such as ANALYZE. Some Freesurfer functions require other formats, such as MINC (`http://www.bic.mni.mcgill.ca/ServicesSoftware/MINC`). The Freesurfer installation provides functions to convert from MINC to NIfTI formats and there are implemented in functions such as `nii2mnc` and `mnc2nii` in R. Moreover, the `mri_convert` Freesurfer function has been interfaced in the `freesurfer` package (same function name), which allows for a more general conversion tool of imaging types for R users than currently implemented in native R.

Reconstruction pipeline in Freesurfer

The Freesurfer pipeline and analysis workflow for neuroanatomical images is based on a structural magnetic resonance image (MRI) of the brain. The specific type of image commonly used in this software is a T1-weighted image, a specific MRI sequence commonly taken. The full pipeline is implemented in the Freesurfer `recon-all` function, where the "recon" stands for reconstruction (`https://surfer.nmr.mgh.harvard.edu/fswiki/recon-all`). Using the `-all` flag in the the `recon-all` function performs over 30 different steps and takes 20-40 hours to fully process a subject when performing all the steps. This process is the common way of fully processing an T1-weighted image in Freesurfer.

If there are problems with the result of this processing, there are multiple steps users can edit certain parts of the processing, such as skull-stripping, where non-brain tissues are removed from the image. The remainder of the pipeline can be run after these steps. The full pipeline is broken down into 3 separate steps, referred to as `autorecon1`, `autorecon2`, and `autorecon3`, which correspond to the flags in `recon-all` used to initiate these steps. We have written wrapper functions `recon_con1`, `recon_con2`, and `recon_con3`, respectively, for simplicity to the R user.

R function setup

To use **freesurfer**, a working installation of Freesurfer is required. The following code was run using Freesurfer version freesurfer-Darwin-lion-stable-pub-v5.3.0. The Freesurfer version can be accessed using the `fs_version` function. **freesurfer** must also have the path of Freesurfer specified. If using R from a shell environment, and the `FREESURFER_HOME` environment variable is set (which can be done when installing Freesurfer), **freesurfer** will use this as the path to Freesurfer. If using R through a graphical user interface (GUI) such as RStudio (RStudio, Boston, MA), environmental variables and paths are not explicitly exported. Therefore, `FREESURFER_HOME` is not set, `freesurfer` will try the default directories of Mac OSX and Linux. If the user did not perform an standard installation of Freesurfer, the path to Freesurfer can be specified using `options(freesurfer.path="/path/to/freesurfer")`.

We will discuss the setup functions for the **freesurfer** package and how they can be used in analysis and example code. For testing, whether a user has a Freesurfer installation, the `have_fs` function provides a logical indicator as a result. The `fs_dir` function will return the directory of the Freesurfer installation.

Structure of Freesurfer analyses

During the installation of Freesurfer, a series of variables are set up in the user's shell environment. One of these variables is `SUBJECTS_DIR`, which refers to a directory of the output of analysis from all subjects. This setup allows users to simply specify a subject identifier to analyze, rather than a specific path or multiple intermediate files.

This setup may not be desirable if the user prefers to structure the data from multiple studies into different folders. For example, the `asegstats2table` function takes the **a**natomical **seg**mentation **stat**istics and convert it to a table. The default argument for `asegstats2table` is to pass in a subject name rather than a file. The **freesurfer** `asegstats2table` function allows the R user to specify a different subject directory to read in the file, while not overridding the default set by `SUBJECTS_DIR`. This functionality allows users to have separate folders with subjects and read in the data by simply switching the `subj_dir` argument in the R function.

Similarly to the `fs_dir` function, the `fs_subj_dir` function will return the path to the Freesurfer subjects directory if it is set.

Some Freesurfer functions require an image as an input. For those functions, the R **freesurfer** functions that call those Freesurfer functions will take in a filename or a `nifti` object. The R code will convert the `nifti` to the corresponding input required for Freesurfer. From the user's perspective, the input/output process is all within R. The advantage of this approach is that the user can read in an image, do manipulations of the `nifti` object using standard syntax for arrays, and pass this object into the **freesurfer** R function. Thus, users can create complete pipelines for the analysis of imaging data by accessing Freesurfer through **freesurfer**.

Example analyses and use of functions

In the default subjects directory in the Freesurfer installation, there is a subject named "bert", where `recon-all` was run. In the sub-directory for subject bert, there are 3 folders which we will explore the results: "mri", which contain imaging data, "stats", whic containing statistics based on structures of the brain, and "surf", which contain the surface and curvature output from the Freesurfer processing.

Reading in anatomical statistics

The "aseg.stats" in the "stats" folder of subject bert corresponds to measures and statistics from the anatomical segmentation. The `read_aseg_stats` function reads this corresponding file and creates a list of 2 different `data.frames`:

```
file = file.path(fs_subj_dir(), "bert", "stats", "aseg.stats")
out = read_aseg_stats(file)
names(out)
```

The measures element corresponds to global measurements of the brain (e.g.~volume of the brain) as well as measures of gross anatomical structures (e.g.~gray matter).

```
head(out$measures[, c("meaning", "value", "units")])
```

```
                                           meaning          value
1                         brain segmentation volume 1193318.000000
2         brain segmentation volume without ventricles 1174082.000000
3 brain segmentation volume without ventricles from surf 1173867.217735
4            left hemisphere cortical gray matter volume  237947.199463
5           right hemisphere cortical gray matter volume  238312.856735
6                      total cortical gray matter volume  476260.056198
  units
1  mm^3
2  mm^3
3  mm^3
4  mm^3
5  mm^3
6  mm^3
```

The structures element corresponds to a set of measures and statistics for a set of fixed anatomical structures.

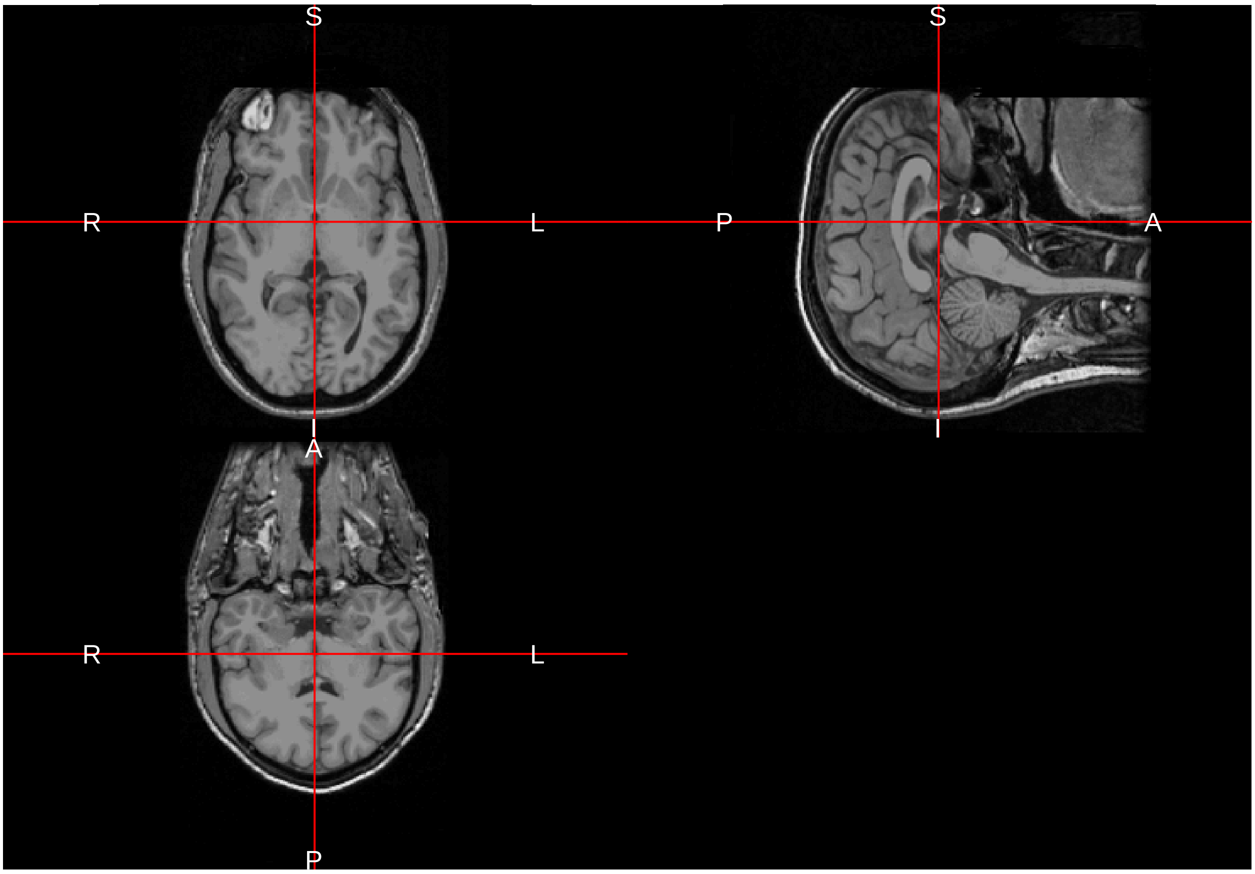
```
head(out$structures)
```

```
  Index SegId NVoxels Volume_mm3                  StructName normMean
1     1     4    6563     6562.6      Left-Lateral-Ventricle  36.0959
2     2     5     228      228.3          Left-Inf-Lat-Vent  54.8842
3     3     7   15708    15708.2 Left-Cerebellum-White-Matter  92.7562
4     4     8   58536    58535.7      Left-Cerebellum-Cortex  77.2709
5     5    10    8150     8150.4        Left-Thalamus-Proper  92.8386
6     6    11    3214     3213.7                Left-Caudate  80.9591
  normStdDev normMin normMax normRange
1    12.2771      16      91        75
2    10.7839      22      87        65
3     5.5123      40     107        67
4     9.9521      17     142       125
5     7.0182      49     109        60
6     8.2079      49     105        56
```

MRI conversion

The typical output format from Freesurfer is MGH/MGZ format, which is explained here: https://surfer.nmr.mgh.harvard.edu/fswiki/FsTutorial/MghFormat. As NIfTI formats are one of the most common formats and has been the common format for analysis in the **oro.nifti** and **fslr** packages, it is useful to convert these files to a NIfTI format to use in R. The mri_convert Freesurfer function will be used for that. Here we will use the T1-weighted image from the "bert" subject, convert it to NIfTI, read it into R, and then plot the image.

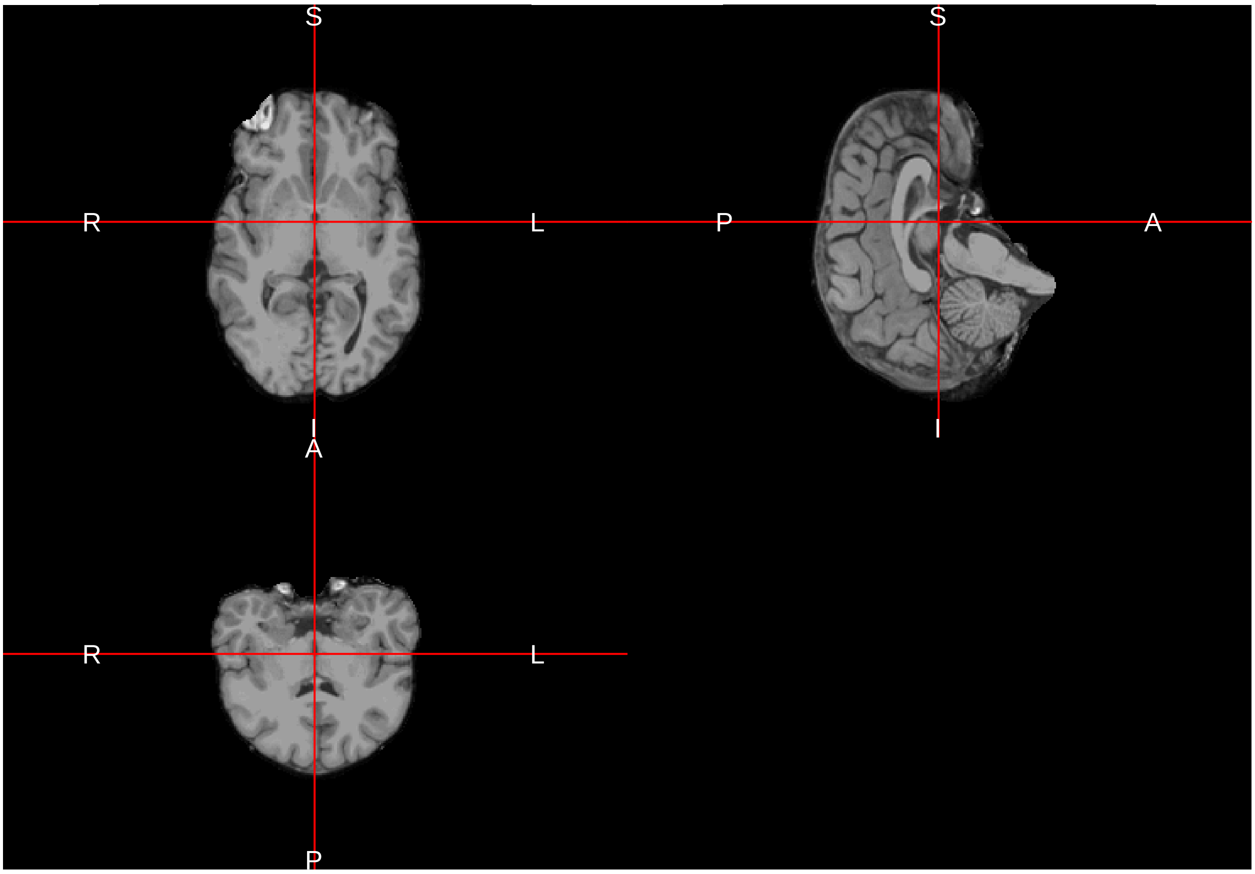
```
library(freesurfer)
t1_mgz = file.path(fs_subj_dir(), "bert", "mri", "T1.mgz")
t1_nii_fname = tempfile(fileext = ".nii.gz")
freesurfer::mri_convert(t1_mgz, t1_nii_fname)
library(fslr)
img = fslr::readnii(t1_nii_fname)
fslr::ortho2(img)
```



Brain extraction

The mri_watershed function will segment the brain from the rest of the image. We ca pass in the nifti object and the output is a brain-extracted nifti object.

```
ss = mri_watershed(img)
ortho2(ss)
```



As the result in a `nifti` object, we can create a mask by standard logical operations:

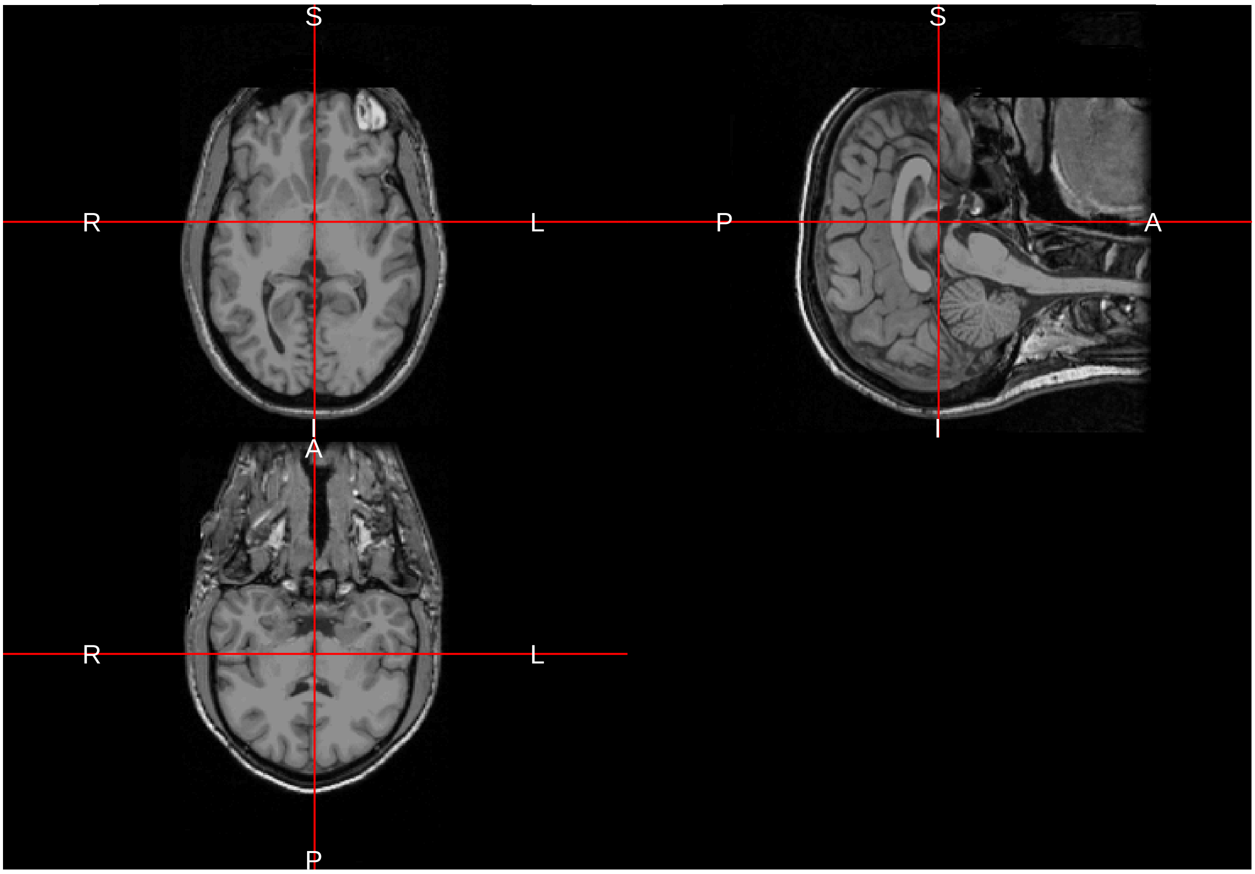
```
mask = ss > 0
```

Bias-field correction

MRI images typically exhibit good contrast between soft tissue classes, but intensity inhomogeneities in the radio frequency field can cause differences in the ranges of tissue types at different spatial locations (e.g.~top versus bottom of the brain) . These inhomogeneities/non-uniformities can cause problems with algorithms based on histograms, quantiles, or raw intensities (Zhang et al., 2001). Therefore, correction for image inhomogeneities is a crucial step in many analyses. The Freesurfer function `nu_correct` performs the non-uniformity correction by Sled et al. (1998) and the **freesurfer** function of the same name will run the correction and return an image.

The Freesurfer `nu_correct` function requires a MNC format (`http://www.bic.mni.mcgill.ca/ServicesSoftware/MINC`). For this to work, you can convert the `nifti` object to a MNC file using `nii2mnc` and pass that file into `nu_correct`. The **freesurfer** `nu_correct` function will run the correction and then convert the output MNC to a NIfTI object.

```
mnc = nii2mnc(img)
print(mnc)
nu_from_mnc = nu_correct(file = mnc)
class(nu_from_mnc)
ortho2(nu_from_mnc)
```



You can also pass in a `nifti` object in directly, and the **freesurfer** `nu_correct` function will automatically convert any NIfTI input files, and then run the correction. We can also pass in a mask (generated from above) to run the correction only the areas of the brain.

```
nu_nifti = nu_correct(file = img, mask = mask)
class(nu_from_mnc)
```

Image preprocessing with fslr

We present a complete analysis of structural magnetic resonance imaging (MRI) data performed using **fslr** and R. Images were obtained from a patient with multiple sclerosis (MS) at 2 different visits (Sweeney et al., 2013), located at `bit.ly/FSL_Data`. At each visit, the image modalities obtained were T1-weighted (T1), T2-weighted (T2), fluid-attenuated inversion recovery (FLAIR), and proton density (PD). In this example we will 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 common stereotaxic template, as this is common in population-based analyses.

Conclusion

The neuroimaging community has developed a large collection of tools for image processing and analysis. Some of the fundamental functionality of neuroimage processing is being added to R packages, such as the **fslr** and **ANTsR** packages. Additional third-party software still has additional functionality that is not present in R, such as the surface-based registration and processing of Freesurfer. We present **freesurfer** to bridge this gap and provide R users functions from Freesurfer. Interfacing R with existing, powerful software provides users with thoroughly-tested software and an additional community of users, which would not be available if the functions were rewritten in R.

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 have experience with Python or bash scripting, we believe **freesurfer** provides a lower threshold for use in the R community.

For example, other inhomogeneity correction methods exist, such as the popular N3 (Sled et al., 1998) and N4 (Tustison et al., 2010), methods which are not implemented in **fslr**. **ANTsR** (http://stnava.github.io/ANTsR/index.html) is a currently unpublished R package that interfaces with the ANTs (advanced normalization tools) software suite (Avants et al., 2011). ANTs has implementations of these correction methods, an increased set of registration techniques, and other methods for image processing. Other packages such as this, along with **fslr**, can create a diverse set of tools for neuroimaging within R, building on preexisting and widely-accepted software.

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

All data and code processed here is located at https://github.com/muschellij2/FSLR_Data.

Summary

This file is only a basic article template. For full details of *The R Journal* style and information on how to prepare your article for submission, see the Instructions for Authors.

Bibliography

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. [p1, 7]

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, Jon Clayden and based on original code by Marc Modat and Pankaj Daga*, 2015. URL http://CRAN.R-project.org/package=RNiftyReg. R package version 1.1.3. [p1]

B. Fischl. Freesurfer. *Neuroimage*, 62(2):774–781, 2012. [p1]

B. Fischl, M. I. Sereno, R. B. Tootell, A. M. Dale, et al. High-resolution intersubject averaging and a coordinate system for the cortical surface. *Human brain mapping*, 8(4):272–284, 1999. [p1]

B. Fischl, D. H. Salat, E. Busa, M. Albert, M. Dieterich, C. Haselgrove, A. Van Der Kouwe, R. Killiany, D. Kennedy, S. Klaveness, et al. Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. *Neuron*, 33(3):341–355, 2002. [p1]

B. Fischl, D. H. Salat, A. J. van der Kouwe, N. Makris, F. Ségonne, B. T. Quinn, and A. M. Dale. Sequence-independent segmentation of magnetic resonance images. *Neuroimage*, 23:S69–S84, 2004. [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/. [p6]

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]

J. Muschelli, E. Sweeney, M. Lindquist, and C. Crainiceanu. fslr: Connecting the fsl software with r. *The R Journal*, 7(1):163–175, 2015. [p1]

M. Reuter, H. D. Rosas, and B. Fischl. Highly accurate inverse consistent registration: a robust approach. *Neuroimage*, 53(4):1181–1196, 2010. [p1]

F. Ségonne, A. Dale, E. Busa, M. Glessner, D. Salat, H. Hahn, and B. Fischl. A hybrid approach to the skull stripping problem in mri. *Neuroimage*, 22(3):1060–1075, 2004. [p1]

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. [p1, 5, 7]

E. M. Sweeney, R. T. Shinohara, C. D. Shea, D. S. Reich, and C. M. Crainiceanu. Automatic Lesion Incidence Estimation and Detection in Multiple Sclerosis Using Multisequence Longitudinal MRI. *American Journal of Neuroradiology*, 34(1):68–73, jan 2013. ISSN 0195-6108, 1936-959X. doi: 10.3174/ajnr.A3172. URL http://www.ajnr.org/content/34/1/68. [p6]

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. [p7]

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]

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. [p5, 6]

John Muschelli
Johns Hopkins Bloomberg School of Public Health
Department of Biostatistics
615 N Wolfe St, Baltimore, MD, 21205
jmuschel@jhsph.edu

Elizabeth M. Sweeney
Rice University
Department of Statistics
6100 Main St, Duncan Hall, Houston, TX, 77005
ems15@rice.edu

Ciprian M. Crainiceanu
Johns Hopkins Bloomberg School of Public Health
Department of Biostatistics
615 N Wolfe St, Baltimore, MD, 21205
ccraini1@jhu.edu