Coordinate-Based Meta-Analysis of fMRI Studies with R

by Andrea Stocco

Abstract This paper outlines how to conduct a simple meta-analysis of neuroimaging foci of activation in R. In particular, the first part of this paper reviews the nature of fMRI data, and presents a brief overview of the existing packages that can be used to analyze fMRI data in R. The second part illustrates how to handle fMRI data by showing how to visualize the results of different neuroimaging studies in a so-called orthographic view, where the spatial distribution of the foci of activation from different fMRI studies can be inspected visually.

Functional MRI (fMRI) is one of the most important and powerful tools of neuroscientific research. Although not as commonly used for fMRI analysis as some specific applications such as SPM (Friston et al., 2006), AFNI (Cox and Hyde, 1997), or FSL (Smith et al., 2004), R does provide several packages that can be employed in neuroimaging research. These packages deal with a variety of topics, ranging from reading and manipulating fMRI datasets, to implementing sophisticated statistical models.

The goal of this paper is to provide a brief introduction to fMRI analysis, and the various R packages that can be used to carry it out. As an example, it will show how to use simple R commands to read fMRI images and plot results from previous studies, which can then be visually compared. This is a special form of meta-analysis, and a common way to compare results from the existing literature.

A brief introduction to fMRI data analysis

Before getting into the details of the R code, it is important to have a clear picture of what type of data and analyses are common in fMRI research. An fMRI study consists of a set of 3D images that capture brain activity at fixed intervals of time while participants are performing a given task inside an MRI scanner. The interval between image acquisitions is known as *repetition time* or *TR*, and is typically 2 seconds. Each image is made of thousands of almost-cubic elements called *voxels*. The size of a voxel (typically on the order of $3 \times 3 \times 3$ mm) represents the lower limit of the spatial resolution of an image. Not all the voxels are acquired at the same time: each image is typically created by acquiring each horizontal 2D plane (or "slice") in serial order. Thus, voxels belonging to different slices are acquired at different times within the same 2-second TR.

The MRI scanner captures brain "activity" only indirectly, by measuring the so-called Blood Oxygen-Level Dependent (BOLD) signal—that is, the amount of oxygenated blood present in every position in space. The BOLD signal is known to vary as a function of neural activity. Specifically, the BOLD response to an increase of neural activity at time $t = 0$ follows a specific time course, which is known as the hemodynamic response function $h(t)$, and commonly modeled as the difference between two gamma functions (Worsley et al., 2002):

$$h(t) = \left(\frac{t}{d_1}\right)^{a_1} \times e^{-\frac{t-d_1}{b_1}} - c \left(\frac{t}{d_2}\right)^{a_2} \times e^{-\frac{t-d_2}{b_2}}$$

Figure 1 illustrates the typical shape of the hemodynamic response function, as returned by the function `fmri.stimulus` of the **fmri** package (Polzehl and Tabelow, 2007; Tabelow and Polzehl, 2011) using the following code.

```
library(fmri)
t <- seq(0, 30, 0.1) # from 0 to 30 secs, in increments of 100ms
h <- fmri.stimulus(301, durations = c(0),     # h(t) for instantaneous event at t=0,
                   onsets = c(1), rt = 0.1),  # Sampled at TR = 0.1 secs
plot(t, h, type = "l", lwd = 2, col = "red",
     xlab = "Time (secs)", ylab = "h(t)", main = "Hemodynamic Response Function")
```

Note how the function peaks only about 6 seconds *after* the event that triggered neural activity: The sluggishness of this response is one of the major drawbacks of functional neuroimaging, as it must be accounted for when designing an experiment and analyzing the data.

After collection, fMRI images need to be preprocessed to remove various sources of noise. A typical preprocessing procedure (e.g., Friston et al., 2006) consists of the following steps:

1. *Spatial realignment*, whereby head motion from one image to another is removed by means of rigid-body transformation;

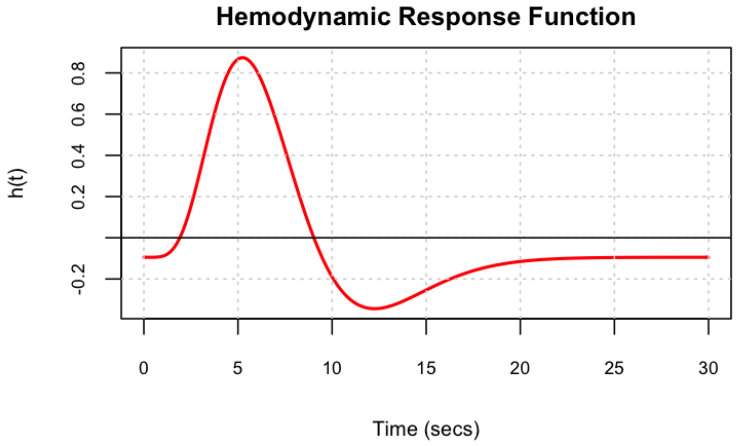

Figure 1: The hemodynamic response function, as implemented in the **fmri** package.

2. *Slice timing correction*, whereby the temporal differences in the time at which voxels belonging to different 2D slices are removed;
3. *Normalization*, where individual brains are warped onto a common anatomical template by means of a non-linear transformation. This step is crucial to aggregate and compare data collected from anatomically different brains;
4. *Spatial smoothing*, which is typically performed with a 3D Gaussian kernel and is aimed at reducing noise and removing residual differences in individual anatomy between participants.

Specialized R packages exists that perform the first and the last of these steps. Specifically, package **RNiftyReg** (Clayden, 2013) provides functions for realigning images in the NIFTI file format, while spatial smoothing is available in the **AnalyzeFMRI** package (Bordier et al., 2011). In addition, the **fmri** package implements an advanced smoothing procedure that adapts to the constraints of the underlying anatomical structures.

To the best of my knowledge, and as confirmed by Tabelow and Polzehl (2011), there are currently no R packages that support the second and third preprocessing steps, which must still be performed using other software.

Statistical analysis of fMRI data

FMRI data is typically analyzed with a *mass-univariate* approach (Friston et al., 2006), where a single model is fitted to the timecourse of hemodynamic activity of each voxel independently. This is done by first creating a $n \times m$ matrix of regressors $\mathbf{X}$, where the m columns represent the time courses of different experimental conditions and the n rows represent the discrete time points at which the images were acquired. To account for the sluggishness of the BOLD response (Figure 1), the matrix $\mathbf{X}$ is created by convolving each column of a matrix $\mathbf{S}$ describing onsets and durations of the experimental stimuli with the hemodynamic response function h, so that $\mathbf{X} = h * \mathbf{S}$.

Data from all the voxels $\mathbf{Y}$ of a single participant is then analyzed by fitting the linear model $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$, and thus finding the values of $\boldsymbol{\beta}$ that minimize the residual sum of squares $\epsilon^T\epsilon$ for each voxel. However, the experimenter is typically interested in examining *differences* between conditions, rather than the estimates of the conditions *per se*. These differences are often referred to as "contrasts", and can be expressed as the product between a contrast vector $\mathbf{c}$ (usually a binary vector), and the vector of estimates $\boldsymbol{\beta}$, i.e., $\mathbf{c}^T\boldsymbol{\beta}$. Thus, the null hypothesis H_0 that there is no difference between conditions corresponds to the case $\mathbf{c}^T\boldsymbol{\beta} = 0$.

The result of a neuroimaging analysis is a set of three-dimensional images. Each image is made of tens of thousands of *voxels*, and each voxel contains the statistical value of a test of a contrast vector $\mathbf{c}$. These parametric images are typically thresholded at a level that corresponds to a significant value of the corresponding statistical test, corrected for multiple comparisons (Friston et al., 2006). This procedure yields a set of 3D *clusters* of spatially adjacent voxels, whose activity is similarly affected by the experimental manipulations.

While R does not provide many tools for preprocessing, there are many R packages that can be used to analyze preprocessed fMRI datasets. For instance, the **fmri** package contains functions to fit statistical models to fMRI datasets in the way described above. In particular, the design matrix $\mathbf{X}$ is generated using the function `fmri.design`, and a linear model can be fit using `fmri.lm`. In

addition, both the **AnalyzeFMRI** and the **fmri** packages also contain functions to perform Independent Component Analysis in both the spatial and temporal domains, as well as several procedures for multiple-error correction, such as Bonferroni, False Discovery Rate, and various instances of family-wise corrections. The package **RfmriVC** (Bothmann and Kalus, 2013) permits users to modulate the effects of the hemodynamic response function with other variables that vary within a single session, such as time or fatigue. The package **arf3DS4** (Weeda et al., 2011) uses the estimated β- or t-images to identify active brain regions and their patterns of functional connectivity. The package **BHMSMAfMRI** (Sanyal and Ferreira, 2014) implements Bayesian multi-subject analysis. Finally, the package **neuRosim** (Welvaert et al., 2011) contains functions that simulate brain imaging data for testing purposes. A comprehensive list of these and related packages is maintained in the CRAN task view on Medical Image Analysis (`http://CRAN.R-project.org/view=MedicalImaging`).

Describing the functionalities of these packages is beyond the scope of this paper; the interested reader can check the recent and extensive review by Eloyan et al. (2014) on the subject. Instead, the remainder of this article will introduce some simple code lines that demonstrate how to read imaging data, plot them over a brain image, and visually compare results from different neuroimaging studies. Incidentally, this type of comparison is a common way to familiarize oneself with a new field in neuroimaging research, and is quite often an important step in justifying hypotheses about the functions of one brain region.

An example of coordinate-based meta-analysis

To understand the coordinate-based meta-analysis, we need to take a step back. Remember that the result of an fMRI analysis is a set of clusters of voxels that are jointly above a certain statistical threshold. Because a single test might identify multiple clusters, neuroimaging papers typically focus only on those that are deemed the most interesting, and relegate the full list of clusters to separate tables. These tables detail various characteristics of each cluster, including its size (i.e., the number of voxels it spans) and position within the brain.

To facilitate comparisons across studies, cluster positions are given in terms of three-dimensional coordinates in predefined stereotactic spaces (which will be discussed in the next section). Because a cluster spans many voxels, different conventions can be applied to determine which coordinates best represent a cluster's position. For instance, some authors might report the coordinates of the cluster's "peak" (i.e., the highest parameter value), while others report the cluster's geometrical center or its center of mass. Fortunately, these values tend not to differ much from each other, and for simplicity we can consider them as functionally equivalent.

In theory, and assuming they are available and comparable, one could directly enter contrast or statistical parameter images from different studies into a meta-analysis. This procedure is known as *image-based* meta-analysis, and is generally considered the most reliable (Salimi-Khorshidi et al., 2009). However, because cluster sizes are dependent on the statistical error correction procedures that were adopted, most meta-analyses of fMRI data, like the Activation Likelihood Estimate (Turkeltaub et al., 2002), focus instead on the consistency of cluster *positions*, i.e., how often a given region is reported, and how the spatial distribution of cluster peaks is affected by the study parameters. This procedure is known as *coordinate-based* meta-analysis, and has been successfully used to identify possible subdivisions within the same anatomical brain regions (for example, the rostral prefrontal cortex, Burgess et al. 2007; or the anterior cingulate cortex, Bush et al. 2000), or to suggest the involvement, in a certain cognitive function, of a brain structure that had been previously overlooked (e.g., the basal ganglia in language switching, Stocco et al. 2014). The remainder of this paper will introduce the packages and the R code that will be used to visualize foci of activation on a standard brain template. The final result will be an image like the one in Figure 2.

Although simple, the code provided herein is useful in its own right. Images like Figure 2 have been published in many scientific papers, such as Bush et al. (2000) and Burgess et al. (2007). This very same code, in fact, has been used to generate two figures published in Stocco et al. (2014) and Buchweitz and Prat (2013).

Displaying the MNI template

The spatial coordinates of a cluster of voxels are given in reference to an *ideal brain template*, oriented in a given *stereotactic space*. Two stereotactic spaces are commonly used, the Talairach-Tournoux and the Montreal Neurological Institute (MNI) spaces. In this paper, we will assume that all the coordinates are in the MNI space, which is preferred by the majority of researchers (Poldrack et al., 2011). Talairach-Tournoux coordinates can be converted to their MNI equivalent using non-linear transformations, such as those proposed by Brett et al. (2001).

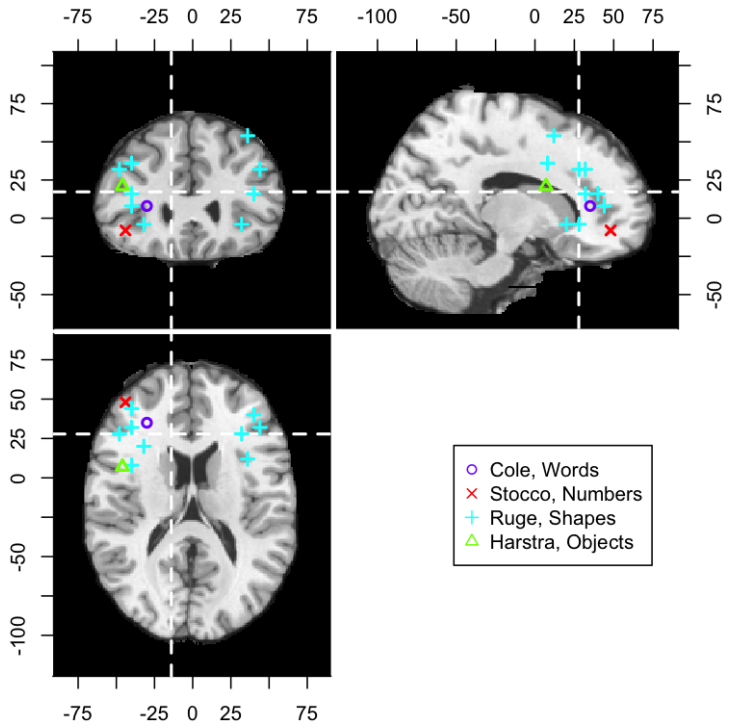

Figure 2: A visual summary of the results of four neuroimaging studies.

In the MNI space the x, y, and z axes correspond to the left-right, front-back, and up-down orientation of a brain. The origin of the space is located in a point in the middle of the brain, so that the brain volume spans both positive and negative coordinates along all axes.

The first step to producing Figure 2 is to visualize the MNI template brain on an R plot. Many versions of the MNI template exist; in this paper, I am going to use the so-called "Colin27" version (Holmes et al., 1998). This template is available on various websites, and is included in many common software packages for fMRI, such as MRICron (Rorden et al., 2000). This example will use the template that is available at `http://packages.bic.mni.mcgill.ca/mni-models/colin27/mni_colin27_1998_nifti.zip`.

The template is encoded in a particular file format known as NIfTI, which is, by far, the most common file format for functional neuroimages. All the packages mentioned above contain functions to read and write NIfTI files. In this example, I will use the **oro.nifti** package (Whitcher et al., 2011), which has been developed specifically for this purpose. Assuming that the template's URL is stored in the variable `colin.url`, the image can be easily loaded with this code:

```
library(oro.nifti)
temp <- file.path(tempdir(), "mni_colin27_1998_nifti.zip")
colin.url <-
  "http://packages.bic.mni.mcgill.ca/mni-models/colin27/mni_colin27_1998_nifti.zip"
download.file(colin.url, dest = temp)
unzip(temp, exdir = tempdir())
colin <- readNIfTI(file.path(tempdir(), "colin27_t1_tal_lin.nii"))
```

The `colin` object is simply a 3D matrix. The matrix dimensions are of $181 \times 217 \times 181$ voxels, and can be accessed by calling the `dim` function: `dim(colin)`. Each slice of this matrix is in itself a 2D image, and can be visualized with any standard 2D plotting tools in R. For instance, a professional-looking sagittal (i.e., lateral) view of the brain can be visualized by calling the *image* function on the 80th 2D slice across the x axis, e.g., `image(colin[80, , ], col = grey(0:255 / 255))`, as shown in Figure 3.

Notice that the original template, as shown in Figure 3, contains the brain as well as a surrounding skull. For clarity purposes, the skull has been omitted in Figure 2. The Colin27 archive that had been downloaded in the previous step also contains a "brain mask", i.e., a binary image that spans the entire brain but leaves out the skull. An skull-stripped image of the brain can be generated as the point-wise product of the `colin` image and the mask:

```
mask <- readNIfTI(file.path(tempdir(), "colin27_t1_tal_lin_mask.nii"))
colin <- colin * mask
```

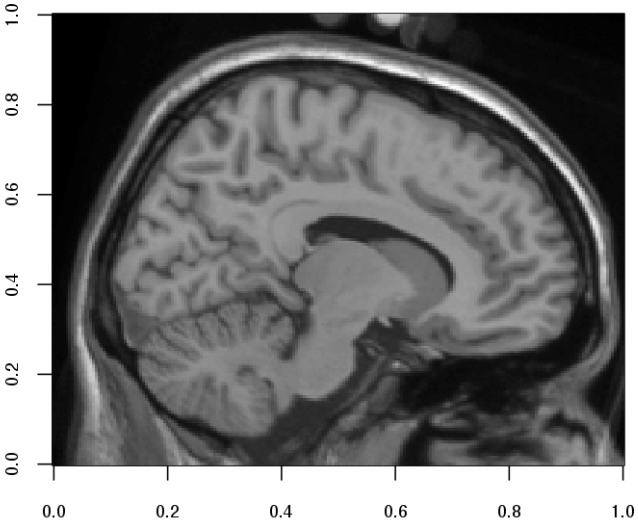

Figure 3: The 80th 2D sagittal slice of the `colin` template.

The three axes of the `colin` object are already aligned with the three axes of the `colin` template. However, the coordinate system of the image and the coordinate system of the `colin` image have different centers. In the `colin` object, the point of coordinates $x = 0$, $y = 0$, $z = 0$ is located at the leftmost voxel of the first row of the bottom slice. In the MNI space, however, the origin is located in the middle of the brain. The MNI origin corresponds to the voxel of coordinates $x = 91$, $y = 126$, $z = 72$ in the image space. Thus, the MNI coordinates need to be translated in space to be visualized correctly. Because each voxel in `colin` is a $1 \times 1 \times 1$ mm cube, this affine translation can be easily implemented using `sweep`:

```
# Origin of MNI space
origin <- c(x = 91, y = 126, z = 72)

# Converting back and forth to MNI space
mni2xyz <- function(x) sweep(x, 2, as.double(origin), "+")
xyz2mni <- function(x) sweep(x, 2, as.double(origin), "-")
```

Figure 2 is a rendition of the brain in so-called *orthographic view*, which is particularly helpful for visualizing the relative position of foci within the brain. The orthographic view consists of three orthogonal brain slices that intersect in a specific point. We will refer to this point as the center C of the orthographic view, with coordinates x_C, y_C, z_C. The three views are produced by simply selecting the 2D slice of the 3D image with the corresponding values of C. The top left view in Figure 2 is called *coronal view*, and corresponds to the case where $y = y_C$. The top right view in Figure 2 is called *sagittal view*, and corresponds to the case where $x = x_C$. Finally, the bottom-left view is known as *axial view*, and corresponds to the case where $z = z_C$.

The package **oro.nifti** has its own method (`orthographic`) to visualize a brain in orthographic view. In this example, however, we will implement the procedure from scratch, as it is fairly easy and allows for more control on the proper visualization.

The first step towards producing the image in Figure 2 is to create a the plot window and divide it into four quadrants. R's default `layout` method is a convenient way to do so:

```
layout(matrix(1:4, ncol = 2, byrow = TRUE),
       heights = c(181, 217),
       widths = c(181, 217))
layout.show(4)
```

The heights and widths parameters in the code are used to assign non-uniform heights and widths to the four quadrants. This is needed, in turn, because the colin template is not cubic, and it spans 181 voxels along the x and z axis, but 217 on the z axis.

Next, we need to define our center point C. Initially, I will simply assume that the center of the orthographic view is also the origin in MNI space, i.e., the point of coordinates $x = 0$, $y = 0$, $z = 0$. The coronal, sagittal, and axial views of the orthographic figure can be simply generated by subsequent

calls of the default method `image`:

```
center <- origin
greys  <- grey(1:max(colin) / max(colin))
par(mar = c(2, 2, 2, 2))
image(colin[, center["x"], ], col = greys) # Coronal
image(colin[center["y"], , ], col = greys) # Sagittal
image(colin[, , center["z"]], col = greys) # Axial
plot.new()  # Empty quadrant
```

In the code above, the `greys` variable is used to create as many shades of grey as there are values of intensity in the Colin27 template. These colors as passed as an argument to visualize the brain slices in the canonical greyscale (instead of R's default heat colors) like in in Figure 3. The `mar` variable sets the plot margins for each image to 2 lines of text, which is the minimum amount necessary to visualize the image axes in each view. Finally, the call to `plot.new` is required because R expects one plot for each quadrant defined by `layout`.

Plotting the foci

Because each image in the orthographic view is a figure in itself, the individual foci can be visualized by plotting points on top of each image with R's `points` function, before proceeding to the next image. In the example of Figure 2, the individual foci of activation were stored in a '`data.frame`' object called `foci`.

```
> foci
     x  y  z   Study     Type
1  -30 35  8    Cole    Words
2  -46  7 21 Harstra  Objects
3  -40  8 36    Ruge   Shapes
4   36 12 54    Ruge   Shapes
5  -48 28 32    Ruge   Shapes
6   44 32 32    Ruge   Shapes
7  -40 32 16    Ruge   Shapes
8  -40 44  8    Ruge   Shapes
9   40 40 16    Ruge   Shapes
10 -32 20 -4    Ruge   Shapes
11  32 28 -4    Ruge   Shapes
12 -44 48 -8  Stocco  Numbers
```

This list of foci was created by examining the results of four publications that, between 2010 and 2012, defined a rule-learning paradigm called Rapid Instructed Task Learning (Cole et al., 2013), which can be used to study how the brain encodes abstract rules. Specifically, the foci of activation in this list come from an analysis of the brain regions that were significantly more engaged when encoding completely *novel* rules than rules that had been practiced before. For simplicity's sake, only foci appearing in the frontal lobe were considered. Interestingly, these four studies used rules of similar length and complexity, but of different nature. Specifically, the rules used in the four studies applied to pictures of common objects (Hartstra et al., 2011), geometric shapes (Ruge and Wolfensteller, 2010), words (Cole et al., 2010), and numbers (Stocco et al., 2012). By comparing the distribution of results across these studies, we can observe whether the *nature* of the rules performed determines the *location* where the corresponding rule is represented. For example, the distribution of foci in Figure 2 seems to suggest that rules that encode more abstract materials (such as words and numbers) activate more anterior regions than rules that encode more concrete materials (such as pictures of common objects).

In the `foci` data frame, the `x`, `y`, and `z` columns represent, of course, the corresponding MNI coordinates of each focus of activation. The fourth column `Study` serves to identify the specific study the foci belong to. The last column `Type` defines the type of stimuli the rules were applied to in each study.

Before plotting the foci, their coordinates need to be transformed from the three-dimensional MNI space to the bidimensional space of each image. The procedure requires two consecutive steps. First, the coordinates need to be transformed from MNI space to image space (using the `mni2xyz` function above). Second, the image coordinates need to be properly scaled to the fit within the range $[0, 1]$ of the axes generated by the *image* function. Once we have the foci coordinates in the proper space, we can plot them by using the appropriate coordinates as the values of the `x` and `y` parameters of `points`. The following code illustrates the example.

```
# Transforms the coordinates
im.foci <- mni2xyz(foci[1:3])
```

```
# Normalizes
im.foci <- sweep(im.foci, 2, dim(colin), "/")

image(colin[, center["y"], ], col = greys) # Coronal
points(x = im.foci[,"x"], y = im.foci[,"z"], lwd = 2, col = "red")

image(colin[center["x"], , ], col = greys) # Sagittal
points(x = im.foci[,"y"], y = im.foci[,"z"], lwd = 2, col = "red")

image(colin[, , center["z"]], col = greys) # Axial
points(x = im.foci[,"x"], y = im.foci[,"y"], lwd = 2, col = "red")
plot.new()
```

This code, however, does not efficiently convey the position of the foci. This is because, while the three views are centered at the origin of the MNI coordinates, the foci are instead mostly distributed in the left hemisphere and in the frontal lobe. The easiest way to solve this problem is to use the center C_F of the distribution of foci, and not the MNI origin, as the center of the orthographic view. The coordinates of C_F are simply the mean of each focus' x, y, and z coordinates: $x_{C_F} = \overline{x}$, $y_{C_F} = \overline{y}$, $z_{C_F} = \overline{z}$. In R, these values can be easily calculated with one call of the `colMeans` function, and turned into integers by using `round`.

```
im.foci <- mni2xyz(foci[1:3])
center <- round(colMeans(im.foci), 0)
```

While this produces a more meaningful figure, it does not resolve all the problems. For instance, the sagittal view of Figure 2 could come from either the left or the right hemisphere. Because the left and right hemispheres are symmetric, there is no way to know from which one a sagittal view comes from. To avoid this ambiguity, orthographic views often outline the two planes that are perpendicular to each view in the form of lines. Thus, each of the three images shows the position of the planes that define the other two images. Because the three images are orthogonal with each other, each plane is visualized as a vertical or a horizontal line (dotted white lines in Figure 2). These lines can be drawn by calling `abline` before invoking the `points` function, and using each plane's coordinates as the `h` or `v` parameter of `abline`. The plane coordinates are, of course, those of the foci's center C_F, normalized to be in the range $[0, 1]$.

```
# Normalizes
im.foci <- sweep(im.foci, 2, dim(colin), "/")
CF <- colMeans(im.foci)

image(colin[, center["y"], ], col = greys) # Coronal
abline(v = CF["x"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[,"x"], y = im.foci[,"z"], lwd = 2, col = "red")

image(colin[center["x"], , ], col = greys) # Sagittal
abline(v = CF["y"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[,"y"], y = im.foci[,"z"], lwd = 2, col = "red")

image(colin[, , center["z"]], col = greys) # Axial
abline(v = CF["x"], h = CF["y"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[,"x"], y = im.foci[,"y"], lwd = 2, col = "red")

plot.new()  # Empty quadrant
```

One important feature of Figure 2 is the use of tick marks and labels representing MNI coordinates, instead of the default $[0, 1]$ axes that are generated otherwise. In addition, the tick marks and labels of Figure 2 are positioned on the outside of the plotting region, bringing the three views closer together. Both of these features increment the readability of the plot.

The precise position of tick marks is controlled by the `at` parameter of the `axis` function, which specifies the location of the tick marks in the axis' native range (in this case, $[0, 1]$). The corresponding labels are similarly controlled by the `labels` parameter.

Let us start by defining a set of "useful" position marks in MNI space, e.g., every 25mm from -100 to 100, along the x, y, and z axis. These labels can then be transformed in 2D plotting coordinates by first converting them into image space and subsequently normalizing them.

```
labels <- seq(-100, 100, by  =  25)
ticks  <- cbind(x = labels, y = labels, z = labels)
```

```
ticks  <- sweep(mni2xyz(ticks), 2, dim(colin), "/")
```

Now that the we have a data frame of tick marks, we can proceed to modify the position of the axes in each image of the orthographic view. To do this, we first need to suppress the `image` method's default axes, which can be done by setting the parameters `ann` (for the labels) and `axes` (for the axes) to `FALSE`. Second, we need to create custom axes with the `axis` function. The axis position is the first argument of the `axis` function, and it is indexed by a number between 1 and 4, in clockwise order beginning from the bottom.

Keeping this scheme in mind, we want to plot axes 1 and 3 on the coronal view; axes 3 and 4 on the sagittal view; and axes 1 and 2 on the axial view. We also want to modify the margins (using `par(mar = ...)`) of each view, so that the outside margins are larger than the inside margins to make room for axes, tick marks, and labels. The resulting code is this:

```
par(mar = c(0.1, 2, 2, 0.1))   # Coronal
image(colin[, center["y"], ], ann = FALSE, axes = FALSE, col = greys)
axis(2, pos = 0, at = ticks[,"z"], labels = labels)
axis(3, pos = 1, at = ticks[,"x"], labels = labels)
abline(v = CF["x"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[,"x"], y = im.foci[,"z"], lwd = 2, col = "red")

par(mar = c(0.1, 0.1, 2, 2))   # Sagittal
image(colin[center["x"], , ], ann = FALSE, axes = FALSE, col = greys)
axis(3, pos = 1, at = ticks[,"y"], labels = labels)
axis(4, pos = 1, at = ticks[,"z"], labels = labels)
abline(v = CF["y"], h = CF["z"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[,"y"], y = im.foci[,"z"], lwd = 2, col = "red")

par(mar = c(2, 2, 0.1, 0.1))   # Axial
image(colin[, , center["z"]], ann = FALSE, axes = FALSE, col = greys)
axis(1, pos = 0, at = ticks[,"x"], labels = labels)
axis(2, pos = 0, at = ticks[,"y"], labels = labels)

abline(v = CF["x"], h = CF["y"], lty = 2, lwd = 2, col = "white")
points(x = im.foci[,"x"], y = im.foci[,"y"], lwd = 2, col = "red")

# Last quadrant
plot.new()
```

Because neuroimaging meta-analysis often aims at identifying functional specializations within or between brain regions, it is helpful to color-code the given foci of activation according to some category. In the example of Figure 2, the foci of activation come from four different studies that dealt with learning rules for different types of stimuli.

The study and the stimulus type are encoded in the fourth and fifth column of the `foci` data frame, and are indicated in Figure 2 by the color and point symbol of each focus of activation. Thus, the number of colors and the number of point marks are determined by the number of unique values of the variable `foci$Study` and the variable `foci$Type`. To visualize each point with the proper marker and color, we need to create a color vector and a point mark vector, each of which is as long as the set of foci. To cover all the possible combinations, the color and mark vectors need to be created by substituting each value of the `Rule` and `Learning` columns in the `foci` data frame with the corresponding color and mark.

```
foci.marks  <- as.integer(foci$Learning)
foci.colors <- rainbow(nlevels(foci$Rule))[as.integer(foci$Rule)]
```

Now we can just modify the `points` function used above by specifying the proper parameters—`col` for the colors and `pch` for the marks.

```
# Coronal view code
...
points(x = im.foci[, "x"], y = im.foci[,"z"], lwd = 2, pch = foci.marks,
       col = foci.colors)

# Sagittal view code
...
points(x = im.foci[, "y"], y = im.foci[,"z"], lwd = 2, pch = foci.marks,
       col = foci.colors)
```

```
# Axial view code
...
points(x = im.foci[, "x"], y = im.foci[,"y"], lwd = 2, pch = foci.marks,
       col = foci.colors)
```

The last step to complete Figure 2 is to add the figure legend. Because our color and point mark vectors contain combinations of color and point mark, we need to simplify them appropriately, so that we have only one combination of each. To do so, we will first create a data frame object holding all the legend arguments, and then use `unique` to extract the unique combinations of values. Thus, the final piece of the code will contain the following lines:

```
# Last quadrant
plot.new()
l.args <- data.frame(col = foci.colors, pch = foci.marks,
                     stringsAsFactors = FALSE)
l.args$legend <- paste(foci$Learning, foci$Rule)
l.args <- unique(l.args)
legend(x = "center", legend = l.args$legend, col = l.args$col, pch = l.args$pch)
```

Summary

This paper has discussed the application of R to functional neuroimaging. The first part of this article has given an overview of the basics of functional neuroimaging data acquisition, preprocessing, and analysis. Neuroimaging data requires specific preprocessing steps, such as spatial realignment and normalization, that are currently not covered by R packages. The analysis and modeling of neuroimaging data, on the other hand, is covered by many specialized packages. So, why should one use R instead of, let's say, AfNI or FSL? First and foremost, the existing R packages provide unique and sophisticated features that are not available otherwise. Second, R by itself provides hundreds of statistical packages and models that are *not* specific to neuroimaging, but that can be readily applied to supplement, expand, and innovate the existing approaches. For example, one can imagine to use the **spatstat** package (Baddeley and Turner, 2005) to analyze the three-dimensional spatial distribution of foci in Figure 2, thus giving some statistical bases for our intuition that different types of stimuli affect how rules are represented.

As an introduction on how R can be used to handle neuroimaging data, the second part of this paper has presented an example of R code that visualizes the results of different neuroimaging studies on an orthographic view of the brain. The code permits to visually inspect the distribution of reported cluster of activation across studies, which is an important first step for generating more precise hypotheses about the functions of different brain regions. It uses brain templates and coordinate spaces that are standard in the neuroimaging literature, and can generate journal-quality images for publication. In fact, this very same code has been used to generate figures in published reviews (Stocco et al., 2014; Buchweitz and Prat, 2013).

In summary, although R cannot completely replace specialized software for fMRI data analysis, it can readily integrate them—which is relatively easy after familiarizing oneself with the formats, templates, and conventions of neuroimaging research.

Bibliography

A. Baddeley and R. Turner. spatstat: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6):1–42, 2005. URL http://www.jstatsoft.org/v12/i06. [p13]

C. Bordier, M. Dojat, and P. L. de Micheaux. 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.jstatsoft.org/v44/i09/. [p6]

L. Bothmann and S. Kalus. *RfmriVC: Varying stimulus coefficient fMRI models in R*, 2013. URL http://CRAN.R-project.org/package=RfmriVC. R package version 1.0.4. [p7]

M. Brett, K. Christoff, R. Cusack, and J. Lancaster. Using the Talairach atlas with the MNI template. *Neuroimage*, 5(13):85, 2001. [p7]

A. Buchweitz and C. Prat. The bilingual brain: Flexibility and control in the human cortex. *Physics of Life Reviews*, 10(4):428–443, 2013. [p7, 13]

P. W. Burgess, S. J. Gilbert, I. Dumontheil, P. W. Burgess, S. J. Gilbert, and I. Dumontheil. Function and localization within rostral prefrontal cortex (area 10). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):887–899, 2007. [p7]

G. Bush, P. Luu, and M. I. Posner. Cognitive and emotional influences in anterior cingulate cortex. *Trends in Cognitive Sciences*, 4(6):215–222, 2000. [p7]

J. Clayden. *RNiftyReg: Medical image registration using the NiftyReg library*, 2013. URL http://CRAN.R-project.org/package=RNiftyReg. R package version 1.1.2; based on original code by Marc Modat and Pankaj Daga. [p6]

M. W. Cole, A. Bagic, R. Kass, and W. Schneider. Prefrontal dynamics underlying rapid instructed task learning reverse with practice. *The Journal of Neuroscience*, 30(42):14245–14254, 2010. [p10]

M. W. Cole, P. Laurent, and A. Stocco. Rapid instructed task learning: A new window into the human brain's unique capacity for flexible cognitive control. *Cognitive, Affective, & Behavioral Neuroscience*, 13(1):1–22, 2013. [p10]

R. W. Cox and J. S. Hyde. Software tools for analysis and visualization of fMRI data. *NMR in Biomedicine*, 10(45):171–178, 1997. [p5]

A. Eloyan, S. Li, J. Muschelli, J. J. Pekar, S. H. Mostofsky, and B. S. Caffo. Analytic programming with fMRI data: A quick-start guide for statisticians using R. *PloS ONE*, 9(2):e89470, 2014. [p7]

K. Friston, J. Ashburner, S. Kiebel, T. Nichols, and W. Penny, editors. *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press, 1 edition, Dec. 2006. ISBN 0123725607. URL http://www.sciencedirect.com/science/book/9780123725608. [p5, 6]

E. Hartstra, S. Kühn, T. Verguts, and M. Brass. The implementation of verbal instructions: An fMRI study. *Human Brain Mapping*, 32(11):1811–1824, 2011. [p10]

C. J. Holmes, R. Hoge, L. Collins, R. Woods, A. W. Toga, and A. C. Evans. Enhancement of MR images using registration for signal averaging. *Journal of Computer Assisted Tomography*, 22(2):324–333, 1998. [p8]

R. A. Poldrack, J. A. Mumford, and T. E. Nichols. *Handbook of Functional MRI Data Analysis*. Cambridge University Press, 2011. [p7]

J. Polzehl and K. Tabelow. fmri: A package for analyzing fmri data. *R News*, 7(2):13–17, 2007. [p5]

C. Rorden, M. Brett, et al. Stereotaxic display of brain lesions. *Behavioural Neurology*, 12(4):191–200, 2000. [p8]

H. Ruge and U. Wolfensteller. Rapid formation of pragmatic rule representations in the human brain during instruction-based learning. *Cerebral Cortex*, 20(7):1656–1667, 2010. [p10]

G. Salimi-Khorshidi, S. M. Smith, J. R. Keltner, T. D. Wager, and T. E. Nichols. Meta-analysis of neuroimaging data: A comparison of image-based and coordinate-based pooling of studies. *Neuroimage*, 45(3):810–823, 2009. [p7]

N. Sanyal and M. A. Ferreira. *BHMSMAfMRI: Bayesian hierarchical multi-subject multiscale analysis of functional MRI data*, 2014. URL http://CRAN.R-project.org/package=BHMSMAfMRI. R package version 1.0. [p7]

S. M. Smith, M. Jenkinson, M. W. Woolrich, C. F. Beckmann, T. E. Behrens, H. Johansen-Berg, P. R. Bannister, M. De Luca, I. Drobnjak, D. E. Flitney, et al. Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage*, 23:S208–S219, 2004. [p5]

A. Stocco, C. Lebiere, R. C. O'Reilly, and J. R. Anderson. Distinct contributions of the caudate nucleus, rostral prefrontal cortex, and parietal cortex to the execution of instructed tasks. *Cognitive, Affective, & Behavioral Neuroscience*, 12(4):611–628, 2012. [p10]

A. Stocco, B. Yamasaki, R. Natalenko, and C. S. Prat. Bilingual brain training: A neurobiological framework of how bilingual experience improves executive function. *International Journal of Bilingualism*, 18(1):67–92, 2014. [p7, 13]

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/. [p5, 6]

P. E. Turkeltaub, G. F. Eden, K. M. Jones, and T. A. Zeffiro. Meta-analysis of the functional neuroanatomy of single-word reading: Method and validation. *Neuroimage*, 16(3):765–780, 2002. [p7]

W. D. Weeda, F. de Vos, L. J. Waldorp, R. P. P. P. Grasman, and H. M. Huizenga. arf3DS4: An integrated framework for localization and connectivity analysis of fMRI data. *Journal of Statistical Software*, 44 (14):1–33, 2011. URL http://www.jstatsoft.org/v44/i14/. [p7]

M. Welvaert, J. Durnez, B. Moerkerke, G. Verdoolaege, and Y. Rosseel. neuRosim: An R package for generating fMRI data. *Journal of Statistical Software*, 44(10):1–18, 2011. URL http://www.jstatsoft.org/v44/i10/. [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/. [p8]

K. J. Worsley, C. Liao, J. Aston, V. Petre, G. Duncan, F. Morales, and A. Evans. A general statistical analysis for fMRI data. *Neuroimage*, 15(1):1–15, 2002. [p5]

Andrea Stocco
Department of Psychology and Institute for Learning and Brain Sciences
University of Washington
Campus Box 357988, Seattle, WA
USA
stocco@uw.edu