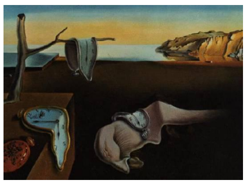

Journal of Statistical Software

October 2011, Volume 44, Issue 10. http://www.jstatsoft.org/

neuRosim: An R Package for Generating fMRI Data

Marijke Welvaert
Ghent University

Joke Durnez
Ghent University

Beatrijs Moerkerke
Ghent University

Geert Verdoolaege
Ghent University

Yves Rosseel
Ghent University

Abstract

Studies that validate statistical methods for functional magnetic resonance imaging (fMRI) data often use simulated data to ensure that the ground truth is known. However, simulated fMRI data are almost always generated using in-house procedures because a well-accepted simulation method is lacking. In this article we describe the R package **neuRosim**, which is a collection of data generation functions for neuroimaging data. We will demonstrate the possibilities to generate data from simple time series to complete 4D images and the possibilities for the user to create her own data generation method.

Keywords: fMRI, data generation, simulation.

1. Introduction

Despite optimization of experimental designs and significant improvements in scanner technology, functional magnetic resonance imaging (fMRI) data still contain a considerable amount of noise. Statistics are needed to infer information from the data. However, a major problem is that the ground truth of fMRI data (i.e., *where* and *when* the activation is located) is unknown and can only be measured with very invasive techniques (i.e., intracranial electroencephalography) that are almost always unethical to perform with humans (David *et al.* 2008). Therefore, when researchers try to establish the validity of a new statistical method, or when they want to assess the sensitivity and the specificity of an existing method, they need to know the ground truth. As a solution, simulation studies have gained great interest as a validation tool because in these studies, the data themselves are generated under a known model.

Although the necessity of knowing the ground truth is acknowledged, a standard simulation procedure for fMRI data is lacking. In the literature, two major categories of computational simulations can be distinguished, namely (1) generating time series based on an experimental design and (2) simulating the magnetic signal by solving the Bloch equations (Bloch 1946). Unfortunately, the first category in itself has no common method. Most researchers model the activation in the time series as the convolution of a haemodynamic response function and a stimulus vector. Additionally, some noise is added ranging from pure random Gaussian noise (Lei *et al.* 2010; Liao *et al.* 2008; Lin *et al.* 2010), over temporally correlated noise (Grinband *et al.* 2008; Locascio *et al.* 1997; Bullmore *et al.* 1996; Purdon and Weisskoff 1998) to *real* noise derived from empirically acquired resting state scans (Bianciardi *et al.* 2004; Lange 1999; Weibull *et al.* 2008; Lee *et al.* 2008; Lange *et al.* 1999; Hansen *et al.* 2001; Skudlarski *et al.* 1999). Furthermore, all simulations are done using in-house software routines. As a consequence, convergence of the simulation methods is impossible as long as fMRI simulators are not available. In contrast, the second method (Drobnjak *et al.* 2006), using the Bloch equations, is embedded in a simulator as part of the software package **FSL** (Smith *et al.* 2004). However, the simulator is rarely used for validation studies. Probably, this is due to the fact that solving the Bloch equations is computationally very intensive and it takes, for example, about a month to generate a 4D dataset of 100 scans including all artefacts using a PC with a 3.4 GHz processor. By developing our package `neuRosim`, we want to respond to the current lack of fMRI simulators. Our package is by no means intended to provide *the* fMRI data generation method. The aim of the package is to provide a tool for simulating fMRI data that can initiate the search for more established and validated simulation methods for fMRI data such that the results of simulation studies can be generalized.

The package **neuRosim** for R (R Development Core Team 2011) is created with two types of users in mind. The first type is the practical researcher who uses the fMRI scanner as a tool to acquire data that hopefully support her theory. This researcher normally would not think of generating fMRI data. However, by generating some data before the actual scanning process is started, this researcher can check the effectiveness of her design without almost any cost, both in time and money. In this way, the most effective design for a particular research question can be tested and adjusted.[1] Secondly, the more theoretical researcher (e.g., a statistician) can validate both existing and new methods based on the generated data. Because the data generation in **neuRosim** is fairly fast, the generation process can easily be embedded in large simulation studies.

fMRI data are in fact the result of a Fourier transformation of the *k*-space and are, as a result, complex-valued data (Rowe and Logan 2004). However, in most fMRI studies the data analysis is done for the magnitude data and not for the phase data. In the current version of **neuRosim**, only the generation of fMRI magnitude data is considered. Therefore, all assumptions that are made to model the data apply only to the characteristics of magnitude data. The generation of magnitude fMRI data is seen as an additive source problem (Bellec *et al.* 2009) in which two main sources are distinguished, namely (1) the activation caused by an experimental design or resting state activation, and (2) the noise. **neuRosim** contains several functions to model both sources. These functions are regarded as low-level functions, meaning that they generate only a specific part of the data and are mostly used as building blocks to construct higher-level functions. For beginning users, it will be more convenient to start with the high-level functions that are described in Section 3. However, advanced users

[1]It should be noted that **AFNI** also contains algorithms for design optimization in the function `3dDeconvolve` without the need for data (Cox 1996).

can use the high-level functions as a basis for their completely customized simulations. In Section 2, we will give an overview of the different models in the low-level functions.

Further, it should be noted that the data generated by **neuRosim** are considered to be pre-processed data. This implies that several artefacts (e.g., head motion, magnetic field inhomogeneity) that are normally removed during the pre-processing stage of the data are not explicitly modeled. However, it is possible to incorporate some residual effects of these artefacts under the assumption that the artefacts are not completely removed by the pre-processing analysis. For example, **neuRosim** data can contain task-related noise that can account for residual head movements.

2. Features and examples of low-level functions

2.1. Experimental activation and design

To generate BOLD (blood oxygen level dependent) activation, **neuRosim** uses a stimulus function that is part of the experimental design. A BOLD response is only generated if the function indicates the presence of a stimulus. Block designs, as well as event-related designs (or a combination of both) can be defined based on the onsets and the durations of the task as defined by the user. The function `stimfunction` uses these arguments to generate a 0-1 valued time vector where 1 indicates that the stimulus is present. Note that for a single event, the duration of the stimulus should be defined as 0. For example, to generate a stimulus function for a 20-second ON/OFF block design of 200s with a microtime resolution of 0.1s:

```
R> totaltime <- 200
R> onsets <- seq(1, 200, 40)
R> dur <- 20
R> s <- stimfunction(totaltime = totaltime, onsets = onsets,
+    durations = dur, accuracy = 0.1)
```

The resulting stimulus function is shown as a dashed line in Figure 1. To simulate the BOLD signal caused by the task, the stimulus function is convoluted with a haemodynamic response function (HRF). The role of the microtime resolution is to ensure a high-precision convolution with the specified HRF. In the current version of **neuRosim**, three different response functions are implemented.

1. The stimulus function is convoluted with a gamma-variate HRF as implemented in the function `gammaHRF` with a user-defined full width at half maximum (FWHM) value (Buxton *et al.* 2004). The function is defined as

$$h(t) = \frac{1}{k\tau_h(k-1)!}\left(\frac{t}{\tau_h}\right)^k \exp\left(-\frac{t}{\tau_h}\right),$$

   with $k = 3$. To provide the desired FWHM, the time constant τ_h is given by $\tau_h = 0.242 \cdot \mathrm{FWHM}$ (Buxton *et al.* 2004, p. S227).

```
R> gamma <- specifydesign(totaltime = 200, onsets = list(onsets),
+    durations = list(dur), effectsize = 1, TR = 2, conv = "gamma")
```

To modulate the strength of the activation in each condition, the argument `effectsize` in the function `specifydesign` should be specified. The values, provided in this argument, are used to increase (values larger than 1) or decrease (values smaller than 1) the amplitude of the generated BOLD response.

2. The stimulus function is convoluted with a double-gamma HRF via `canonicalHRF`, which models an initial dip and an undershoot of the BOLD signal (Friston *et al.* 1998),

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

where a_1 and a_2 model the delay of the response and the undershoot relative to the onset, b_1 and b_2 model the dispersion of the response and the undershoot, c models the scale of the undershoot, and d_1 and d_2 model the time to peak of the response and the undershoot. The default values of the parameters are $d_i = a_i b_i$, $a_1 = 6$, $a_2 = 12$, $b_i = 0.9$ and $c = 0.35$ (Glover 1999).

```
R> canonical <- specifydesign(totaltime = 200, onsets = list(onsets),
+    durations = list(dur), effectsize = 1, TR = 2,
+    conv = "double-gamma")
```

3. The stimulus function is used as the input for the balloon model implemented in the `balloon` function (Buxton *et al.* 2004). The solving of the differential equations in the model is based on the Runge-Kutta solver in the R package **deSolve** (Soetaert *et al.* 2010). The parameters of the model can be modulated via the `param` argument, which should be a list containing values for all the parameters in the model. If not specified, the default values as described by Buxton *et al.* (2004) are used.

```
R> balloon <- specifydesign(totaltime = 200, onsets = list(onsets),
+    durations = list(dur), effectsize = 1, TR = 2, conv = "Balloon")
```

The spatial location of the activation is specified as regions using the function `specifyregion`. A region can be modeled in three ways, namely (1) as a cube, (2) as a sphere or (3) manually. The first two forms can be modeled by defining two arguments, namely the coordinates of the center of the region and the distance from the center to the edge of the region in voxels. For example, to define an activated sphere (the result is displayed in Figure 2)

```
R> a <- specifyregion(dim = c(64, 64), coord = c(20, 20), radius = 10,
+    form = "sphere", fading = 0.5)
```

To define the form manually, the coordinates of all voxels that are part of the region should by specified as a matrix with columns corresponding to their (x, y)-coordinates.

```
R> coord <- matrix(c(rep(20, 20), rep(26:30, each = 2), 20:27, 20:27,
+    rep(28, 6), 21:40, 30:21, rep(31, 8), rep(40, 8), 33:38),
+    ncol = 2, byrow = FALSE)
R> head(coord)
```

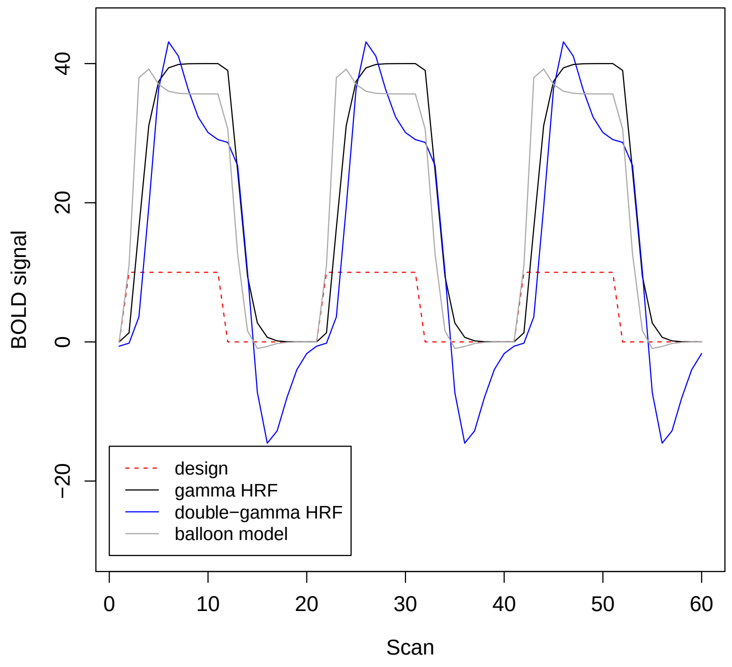

Figure 1: The BOLD signals based on the three convolution functions for a 20-second ON/OFF block design.

```
     [,1] [,2]
[1,]   20   21
[2,]   20   22
[3,]   20   23
[4,]   20   24
[5,]   20   25
[6,]   20   26
```

```
R> b <- specifyregion(dim = c(64, 64), coord = coord, form = "manual")
```

The resulting activated slice is shown in Figure 2.

Additionally, it is possible to differentiate the strength of the measured activation between voxels in the activated region. This can be the case if, for example, the BOLD response to a certain stimulus is of different size in some parts of the activated region. A first method to include this variability is to divide the activated region into separate subregions and specify separate parameters of the HRF for each subregion in `specifydesign`. The subregions can than be merged together using the high-level function `simprepSpatial` (see Section 3). Secondly, if the region is defined as a cube or a sphere, the `fading` option can be used to require that the region has the largest effect in the center and smaller activation towards the edges (Logan and Rowe 2004). This fading of the BOLD response is modeled as an exponential decay depending on the distance of the activated voxel to the center of the region. The decay

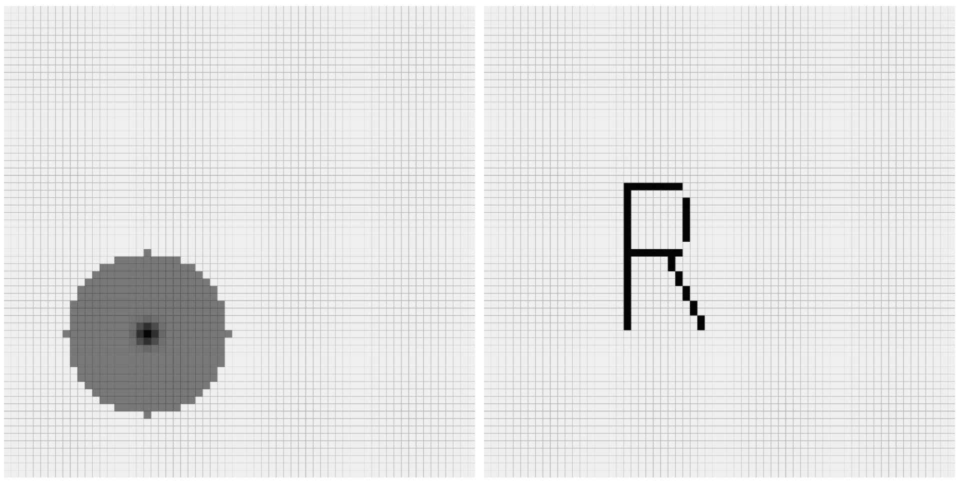

Figure 2: Example of an activated slice: on the left, the activation is modeled as a sphere, on the right, the activated voxels are defined manually.

rate λ can vary between 0 and 1 with 0 meaning no decay and 1 indicating the strongest decay. In 3D this corresponds to

$$A(i,j,k) = \frac{1}{6}\left\{3 \cdot \exp\left[\frac{(i-i')^2 + (j-j')^2 + (k-k')^2}{\lambda}\right] + 3\right\},$$

where (i', j', k') are the (x, y, z)-coordinates of the voxel in the center of the region, λ is the decay rate and the activation is scaled to be 1 in the center of the region. An example of an activated sphere with fading ($\lambda = 0.5$) is presented in Figure 2.

2.2. Noise

The noise present in fMRI data is caused by different sources, such as for example the scanner and the subject. **neuRosim** offers a bundle of functions to model noise from one of these sources. The noise functions can be divided into four categories, namely (1) white noise, (2) colored noise, (3) temporal noise and (4) spatial noise. The white noise (modeled by the function `systemnoise`) represents the system noise that is part of the fMRI data. Two types of system noise are considered: (1) system noise that is Rician distributed and (2) system noise that is Gaussian distributed. The former is applicable for fMRI magnitude data with low signal-to-noise ratio (SNR), while the latter can be used for higher SNR (about more than 10) (Haacke *et al.* 1999; Gudbjartsson and Patz 1995) . The standard deviation of the noise is user-defined or can be based on the desired SNR defined by the user. In all noise functions, average SNR is defined as

$$\mathrm{SNR} = \frac{\bar{S}}{\sigma_N},$$

where $\bar{S}$ represents the average magnitude of the signal, and σ_N stands for the standard deviation of the noise (Krüger and Glover 2001). For example (the resulting time series is plotted in Figure 3),

```
R> n.white <- systemnoise(dim = 1, nscan = 100, sigma = 15, type = "rician")
```

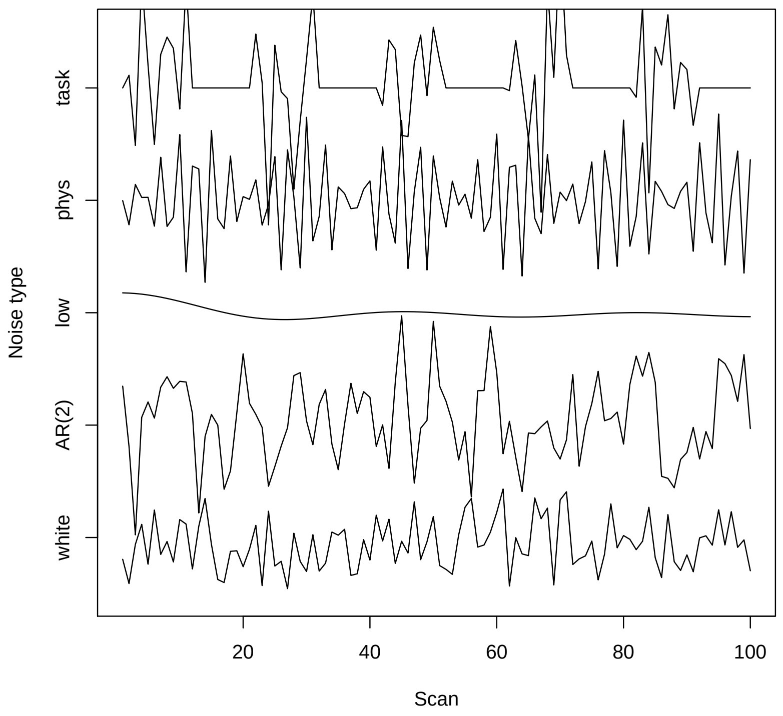

Figure 3: Time series of the noise structures in **neuRosim**.

Colored noise depends on either the signal, the timing or the location. **neuRosim** contains three types of signal-dependent noise, (1) low-frequency drift, (2) physiological noise and (3) task-related noise.

- Low-frequency drift, generated by `lowfreqdrift`, is a consequence of system noise (Smith *et al.* 1999) that can be attributed to slow fluctuations in the scanner hardware (Lazar 2008). The drift is modeled as a basis of discrete cosine functions. The number of functions is determined by the frequency of the drift with a default value of 128 seconds. For example (the resulting time series is plotted in Figure 3),

```
R> n.low <- lowfreqdrift(dim = 1, nscan = 100, TR = 2, freq = 120)
```

- Physiological noise (`physnoise`) is defined as possible cardiac and respiratory artefacts and as such accounts for the variability in the signal that is caused by the heart beat and respiratory rate. These artefacts are often categorized as low-frequency drift. However, we choose to model the physiological noise separately because it is shown that the frequency of these artefacts is often higher than the scanner fluctuations (Smith *et al.* 1999). The physiological noise is modeled as sine and cosine functions with user-defined frequencies. Default values are 1.17 Hz and 0.2 Hz for heart beat and respiratory rate respectively (Biswal *et al.* 1996). For example (the resulting time series is plotted in Figure 3),

```
R> n.phys <- physnoise(dim = 1, nscan = 100, sigma = 15, TR = 2)
```

- Task-related noise accounts for spontaneous neural activity due to the experimental task (Hyde *et al.* 2001) and is operationalized by adding random noise only where and when activation is present. The distribution of this noise can be either Gaussian or Rician. Additionally, the task-related noise can be interpreted as residual noise from head motion that is not removed in the pre-processing stage. For example (the resulting time series is plotted in Figure 3),

  ```
  R> n.task <- tasknoise(act.image = s, sigma = 15)
  ```

Temporal noise accounts for the fact that fMRI data are repeated measurements (Purdon and Weisskoff 1998). The function `temporalnoise` generates noise based on an auto-regressive model of order p (AR(p)) defined as

$$\varepsilon_t = \sum_{i=1}^{p} \rho_i \varepsilon_{t-i} + \chi_t$$

with $\chi_t \sim N(0, \sigma^2)$. For example, the generate temporally correlated noise of order 2 (the resulting time series is plotted in Figure 3),

```
R> n.temp <- temporalnoise(dim = 1, sigma = 15, nscan = 100,
+    rho = c(0.4, -0.2))
```

Finally, spatial noise models the spatial dependencies in fMRI data (Logan and Rowe 2004). Of course, voxels are arbitrary units and neighboring voxels are more likely to be correlated than voxels that are further apart. The function `spatialnoise` incorporates three types of spatial noise models, namely (1) an autoregressive correlation structure, (2) a Gaussian random field and (3) a Gamma random field. The first structure correlates the voxels with each other based on random Gaussian or Rician noise. The strength of the correlation depends on the value of the auto-correlation coefficient (default value is `rho = 0.75`) and the distance between the voxels. If spatial correlation based on random fields is chosen, the full-width-half-maximum (FWHM) of the kernel, which is used to generate the random field, should be provided (default is `FHWM = 4`). Additionally, if the method is `gammaRF`, the shape (default is `gamma.shape = 6`) and rate (default is `gamma.rate = 1`) parameter of the Gamma distribution should be defined as additional arguments. For example, to generate spatially correlated noise for a 20×20 slice

```
R> d <- c(20, 20)
R> n.corr <- spatialnoise(dim = d, sigma = 15, nscan = 100,
+    method = "corr", rho = 0.7)
R> n.gaus <- spatialnoise(dim = d, sigma = 15, nscan = 100,
+    method = "gaussRF", FWHM = 4)
R> n.gamma <- spatialnoise(dim = d, sigma = 15, nscan = 100,
+    method = "gammaRF", FWHM = 4, gamma.shape = 3, gamma.rate = 2)
```

Figure 4 displays the correlation matrices for the generated slices. To generate these images, all voxels were ordered and the correlation matrix of the generated time series was calculated. Therefore, the diagonal represents the perfect correlation of each voxel with itself. We see that voxels that are close to this diagonal, representing neighboring voxels, are also highly

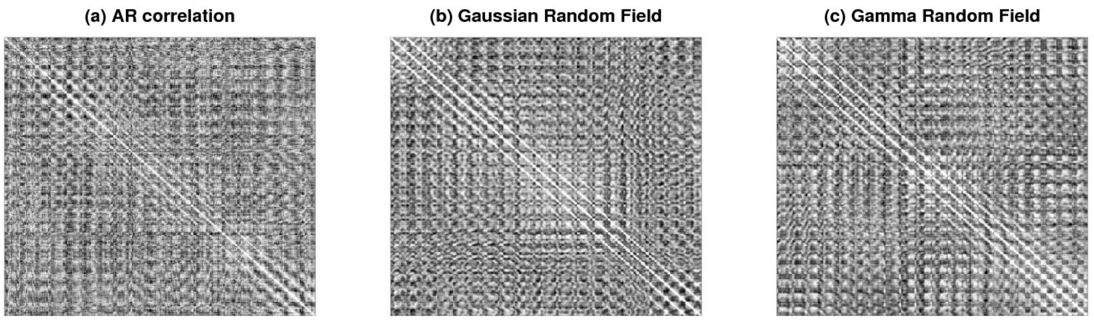

Figure 4: Correlation images for (a) an autoregressive correlation structure, (b) a Gaussian random field and (c) a Gamma random field.

correlated. The block diagonal structure, which can be observed clearly with the Gaussian random field structure (Figure 4b), is the result of reducing the two-dimensional structure of the slice.

Additionally, all noise functions include the functionality that a template or mask can be provided. As such, the noise is only generated for those voxels that are included in the mask. This would allow the user to make for example a distinction between the noise source in the gray matter, the white matter or in the cerebrospinal fluid.

3. Examples of high-level functions

The aim of the high-level functions is to allow the user to generate fMRI data efficiently and transparently. The functions are developed such that they can easily be implemented in a simulation environment. Of course, these functions have limits in their use. Therefore, we refer users who desire more functionality to the low-level functions.

3.1. Generating fMRI time series

The `simTSfmri` function generates fMRI time series for a specified design matrix and with an additive noise structure. The field of the design matrix should be prepared with the `simprepTemporal` function, to ensure that all arguments are in the correct format. As an example, we will generate a time series for a block design with two conditions. The experiment lasts 100 scans with TR = 2 and the first condition has activation blocks of 20s, while the second condition had activation blocks of 7 seconds

```
R> TR <- 2
R> nscan <- 100
R> total <- TR * nscan
R> os1 <- seq(1, total, 40)
R> os2 <- seq(15, total, 40)
R> dur <- list(20, 7)
R> os <- list(os1, os2)
R> effect <- list(3, 10)
```

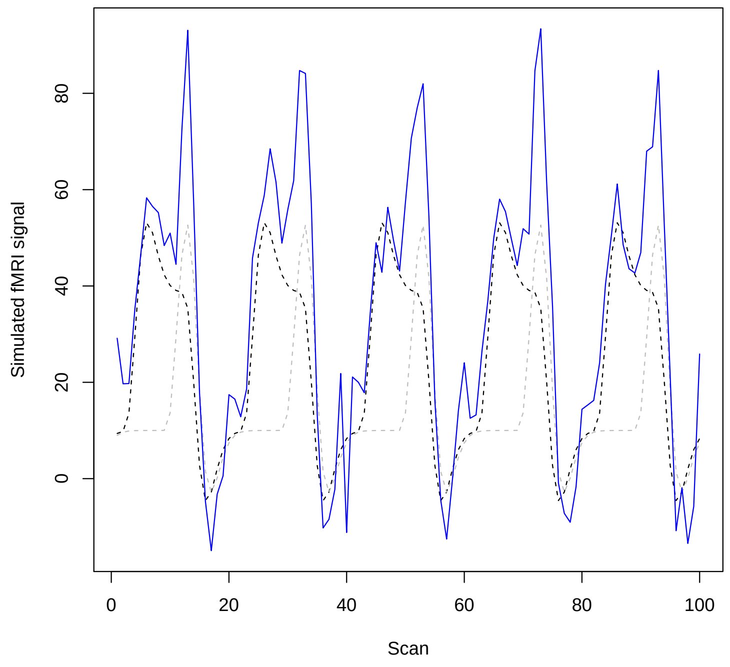

Figure 5: Generated time series (in blue) based on an experiment with two conditions (dashed lines).

```
R> design <- simprepTemporal(totaltime = total, onsets = os, durations = dur,
+    effectsize = effect, TR = TR, hrf = "double-gamma")
```

Figure 5 presents the resulting activation from this design in dashed lines. The following arguments should be specified to ensure a complete definition of the design matrix: the total duration of the experiment in seconds (`total`), the onsets of each condition represented as a list (`onsets`), the duration of the stimulus in each condition represented as a list (`durations`), the repetition time in seconds (`TR`) and the form of the HRF (either `gamma`, `double-gamma` or `balloon`). The noise can be either of the structures described in Section 2, but it is also possible to add a mixture of noise. The different noise components are then weighted with a vector of weights specified by the user. The weights can vary between 0 and 1, however, the weights should sum to one. For example, we will add a mixture of noise to our above specified design. The mixture contains Rician system noise, temporal noise of order 1, low-frequency drift, physiological noise and task-related noise and has a baseline value of 10

```
R> w <- c(0.3, 0.3, 0.01, 0.09, 0.3)
R> ts <- simTSfmri(design = design, base = 10, SNR = 2, noise = "mixture",
+    type = "rician", weights = w, verbose = FALSE)
```

The resulting time series are plotted in Figure 5.

3.2. Generating fMRI volumes

The function `simVOLfmri` is built to generate complete fMRI datasets (i.e., 3D for a slice and 4D for a volume). In this function, some spatial properties of the data are introduced. For this function, not only a design matrix – defining *when* the activation occurs – has to be specified, but also a region – defining *where* the activation takes place – should be provided. Similarly as for the design matrix, a preparation function (`simprepSpatial`) is needed to ensure that all arguments that define the region of activation are in the correct format. Suppose that we wish to simulate 2 activated regions that are part of a small network. We need to call the `simprepSpatial` function as follows

```
R> regions <- simprepSpatial(regions = 2, coord = list(c(10, 5, 24),
+    c(53, 29, 24)), radius = c(10, 5), form = "sphere")
```

The arguments that should be provided in the function are: the number of activated regions (`regions`), a list of coordinates specifying the regions (`coord`), the radius of the region (`radius`, not needed if the region is defined manually) and the shape of the region (`form`) The implemented shapes are `cube` and `sphere`. For any other shape, the coordinates of all voxels in the region should be entered manually (see Section 2 for an example). Further, we will generate the activation in both regions following the same design matrix as for the generation of the time series.

```
R> onset <- list(os, os)
R> duration <- list(dur, dur)
R> effect1 <- list(2, 9)
R> effect2 <- list(14, 8)
R> design2 <- simprepTemporal(regions = 2, onsets = onset,
+    durations = duration, TR = TR, hrf = "double-gamma",
+    effectsize = list(effect1, effect2), totaltime = total)
```

We can now generate an fMRI dataset corresponding to this very simple two-region network. Again, we will add a mixture of noise with the additional possibility that we can add spatially correlated noise.

```
R> w <- c(0.3, 0.3, 0.01, 0.09, 0.1, 0.2)
R> data <- simVOLfmri(dim = c(64, 64, 64), base = 100, design = design2,
+    image = regions, SNR = 10, noise = "mixture", type = "rician",
+    weights = w, verbose = FALSE)
```

The result is a 4D fMRI dataset. To analyze the data with standard fMRI data analysis software like **SPM**, **FSL**, **AFNI**, . . . , the dataset can be exported as a NIfTI file using for example the function `nifti.image.write` in **Rniftilib** (Granert 2010) or the function `writeNIfTI` from **oro.nifti** (Whitcher *et al.* 2011a,b). Note that with `simTSfmri` and `simVOLfmri` it is also possible to simulate data that contain only activation or only noise.

3.3. Simulating and analyzing a 4D fMRI dataset

To further demonstrate the functionality of the package, we present a more real-life example. Consider the data from a repetition priming experiment performed using event-related fMRI

(Henson *et al.* 2002)[2]. The data are the result of a 2×2 factorial study with factors *fame* and *repetition* where famous and non-famous faces were presented twice against a checkerboard (Henson *et al.* 2002, for more details, see). An orthographic overview of the measured data is given on the left side of Figure 6. To generate data using **neuRosim** that are representative for this study, we start by defining the design. First we define some parameters like the dimension of the image space, the number of scans and TR. Then, since we simulate an event-related design, we also assign the onsets for each condition.

```
R> dim <- c(53, 63, 46)
R> nscan <- 351
R> TR <- 2
R> total.time <- nscan * TR
R> onsets.N1 <- c(6.75, 15.75, 18, 27, 29.25, 31.5, 36, 42.75, 65.25,
+    74.25, 92.25, 112.5, 119.25, 123.75, 126, 137.25, 141.75,
+    144, 146.25, 155.25, 159.75, 162, 164.25, 204.75, 238.5) *
+    TR
R> onsets.N2 <- c(13.5, 40.5, 47.25, 56.25, 90, 94.5, 96.75, 135,
+    148.5, 184.5, 191.25, 202.5, 216, 234, 236.25, 256.5, 261,
+    281.25, 290.25, 303.75, 310.5, 319.5, 339.75, 342) * TR
R> onsets.F1 <- c(0, 2.25, 9, 11.25, 22.5, 45, 51.75, 60.75, 63,
+    76.5, 78.75, 85.5, 99, 101.25, 103.5, 117, 130.5, 150.75,
+    171, 189, 227.25, 265.5, 283.5, 285.75, 288, 344.25) * TR
R> onsets.F2 <- c(33.75, 49.5, 105.75, 153, 157.5, 168.75, 177.75,
+    180, 182.25, 198, 222.75, 240.75, 254.25, 267.75, 270, 274.4,
+    294.75, 299.25, 301.5, 315, 317.25, 326.25, 333, 335.25,
+    337.5, 346.5)
```

Next, we have to specify which voxels are activated. We will consider five regions. The first three are general regions that activate when faces are presented, the fourth region is only activated if famous faces are shown, while in the last region adaptation to the repetition of faces is modeled.

```
R> region.1A.center <- c(13, 13, 11)
R> region.1A.radius <- 4
R> region.1B.center <- c(40, 18, 9)
R> region.1B.radius <- 6
R> region.1C.center <- c(10, 45, 24)
R> region.1C.radius <- 3
R> region.2.center <- c(15, 16, 31)
R> region.2.radius <- 5
R> region.3.center <- c(12, 16, 13)
R> region.3.radius <- 5
```

In each region, the same design matrix will be considered. However, the effect size in each condition will vary over conditions.

[2]The use of the dataset is with permission from the corresponding author and may be downloaded from `http://www.mrc-cbu.cam.ac.uk/people/rik.henson/personal/`.

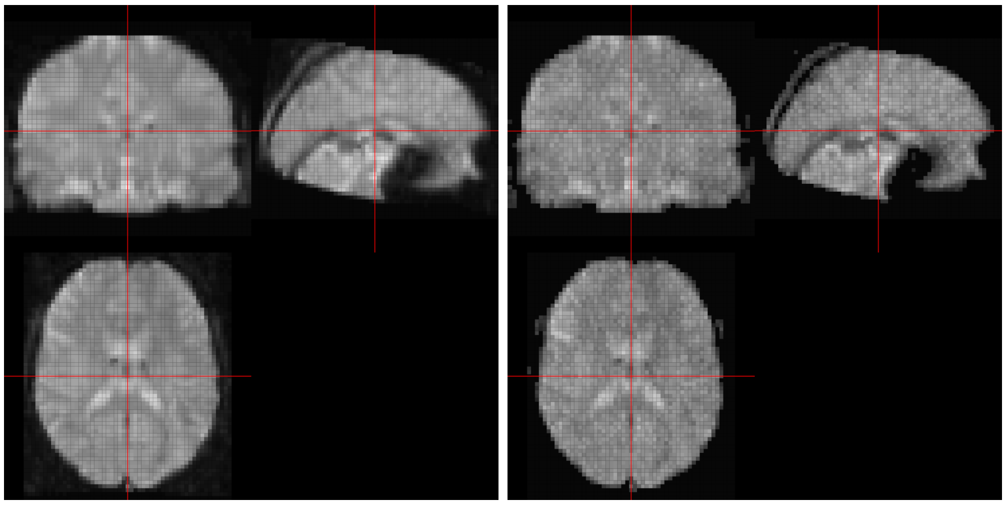

Figure 6: Orthographic view of fMRI data for an event-related repetition priming study. On the left, the data measured by Henson *et al.* (2002) and on the right, the data simulated by **neuRosim**.

```
R> onsets <- list(onsets.N1, onsets.N2, onsets.F1, onsets.F2)
R> onsets.regions <- list(onsets, onsets, onsets, onsets, onsets)
R> dur <- list(0, 0, 0, 0)
R> dur.regions <- list(dur, dur, dur, dur, dur)
R> region.1a.d <- list(160.46, 140.19, 200.16, 160.69)
R> region.1b.d <- list(140.51, 120.71, 160.55, 120.44)
R> region.1c.d <- list(120.53, 120.74, 140.02, 100.48)
R> region.2.d <- list(-0.24, 10.29, 80.18, 160.24)
R> region.3.d <- list(200.81, 50.04, 240.6, 50.83)
R> effect <- list(region.1a.d, region.1b.d, region.1c.d, region.2.d,
+    region.3.d)
```

Additionally, we will consider a baseline image. The baseline value for each voxel is determined as the mean value of the measured time series of that voxel. Non-brain voxels are defined as voxels with an average measured value less than 250 and are fixed to 0 in the baseline image.

```
R> library("oro.nifti")
R> Hensondata <- readNIfTI("preprocessed_face.nii.gz")
R> baseline <- apply(Hensondata, 1:3, mean)
R> baseline.bin <- ifelse(baseline > 250, 1, 0)
R> ix <- which(baseline == 1)
R> baseline[-ix] <- 0
```

Consequently, the anatomical structure of the brain will be incorporated in the simulated data. Now, we can use the functions `simprepTemporal` and `simprepSpatial` to prepare the temporal and spatial structure of our simulated 4D fMRI data.

```
R> design <- simprepTemporal(regions = 5, onsets = onsets.regions,
+    durations = dur.regions, hrf = "double-gamma", TR = TR,
+    totaltime = total.time, effectsize = effect)
```

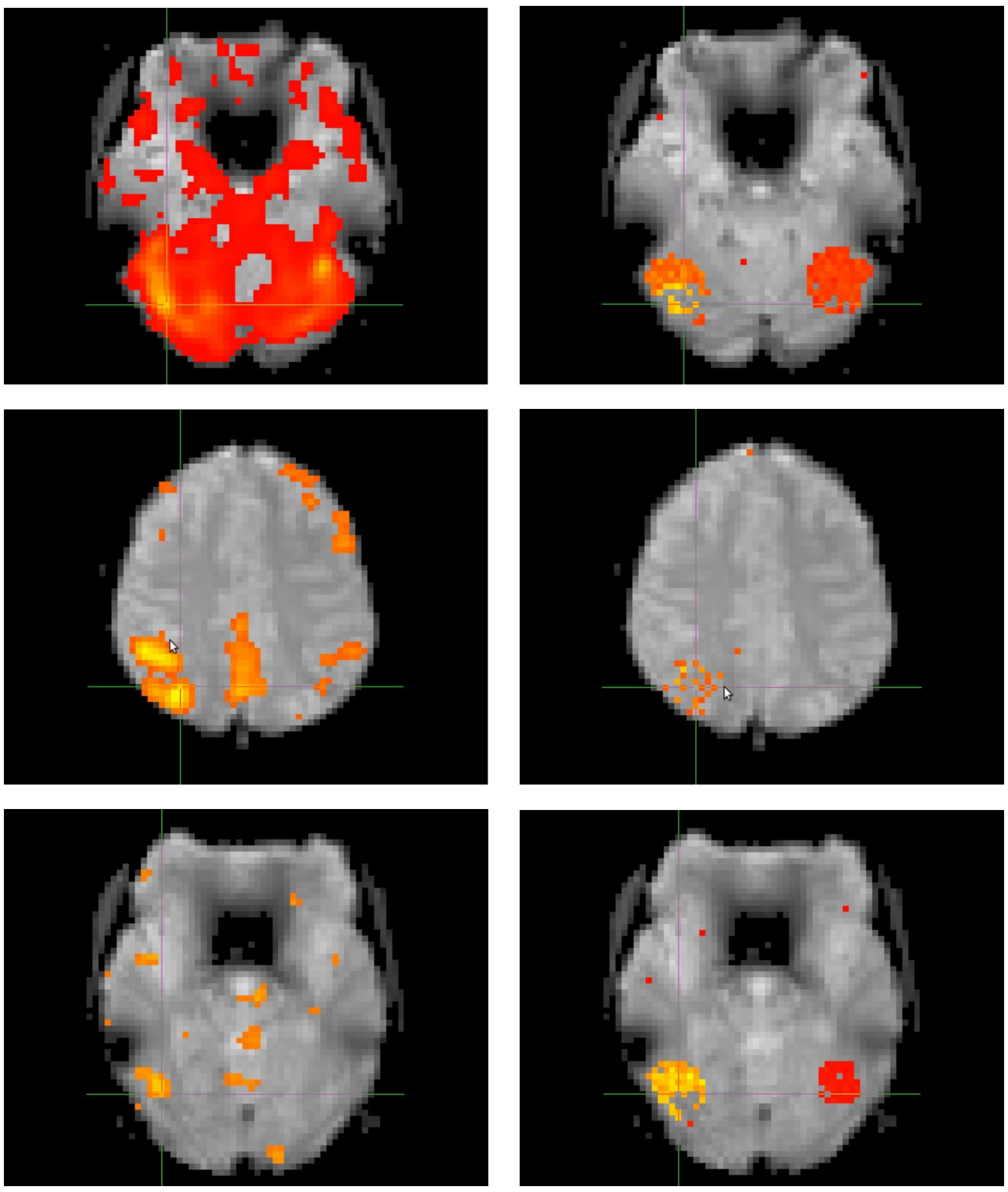

Figure 7: Axial slice view of the activated voxels for the real (left) and simulated data (right): faces versus baseline contrast (top), famous versus non-famous contrast (middle), first versus second presentation contrast (bottom).

```
R> spatial <- simprepSpatial(regions = 5, coord = list(region.1A.center,
+    region.1B.center, region.1C.center, region.2.center, region.3.center),
+    radius = c(region.1A.radius, region.1B.radius, region.1C.radius,
+      region.2.radius, region.3.radius), form = "sphere", fading = 0.01)
```

Finally, we can generate the dataset. Note that the values for the SNR and the temporal autocorrelation coefficients were estimated based on the real data.

```
R> sim.data <- simVOLfmri(design = design, image = spatial, base = baseline,
+    SNR = 3.87, noise = "mixture", type = "rician", rho.temp = c(0.142,
+      0.108, 0.084), rho.spat = 0.4, w = c(0.05, 0.1, 0.01,
+      0.09, 0.05, 0.7), dim = c(53, 63, 46), nscan = 351, vee = 0,
+    template = baseline.bin, spat = "gaussRF")
```

An orthographic overview of the simulated data is given on the right-hand side of Figure 6.

Next, we analyzed the simulated data in **SPM** following the exact description given in the manual (Ashburner *et al.* 2009, Chapter 29). We considered three contrasts, namely: (1) the overall effect of faces versus baseline checkerboard, (2) the effect of famous faces and (3) the effect of repetition. The results were thresholded with $p < 0.05$ (uncorrected), just to demonstrate the detection of the activation. Figure 7 shows a comparison between some of the activated regions that are found in the real data (left-hand) and in the simulated data (right-hand).

4. Conclusions and future work

neuRosim provides a flexible framework for generating fMRI data including a large variety of activation models and noise structures. High-level functions are available to simulate time series or full 4D data in an efficient and transparent way. For more advanced users, the low-level functions create the opportunity to build customized simulation functions. Currently, we are working on an extension of a resting state module such that in future updates it will be possible to have the same functionality for the generation of resting state data as for fMRI data. Other future plans are to include more neurobiological models, for example, the metabolic-hemodynamic model (Sotero and Trujillo-Barreto 2008; Sotero *et al.* 2009) and spatiotemporal BOLD dynamics (Drysdale *et al.* 2010). To extent the generalizability of the data simulated by **neuRosim**, we plan to include the generation of complex-valued fMRI data consisting of both magnitude and phase data. To conclude, it is our hope that **neuRosim** will evolve to a general platform for simulating fMRI data. Simulation studies should be a requisite to publish a statistical validation paper in the field of neuroscience. This will only be possible when standardized and trustworthy simulation methods using validated data generation techniques are available.

References

Ashburner J, Barnes G, Chen CC, Daunizeau J, Flandin G, Friston K, Gitelman D, Kiebel S, Kilner J, Litvak V, Moran R, Penny W, Stephan K, Gitelman D, Henson R, Hutton C, Glauche V, Mattoutee J, Phillips C (2009). ***SPM**8 Manual.* Wellcome Trust Centre for Neuroimaging, University College London.

Bellec P, Perlbarg V, Evans A (2009). "Bootstrap Generation and Evaluation of an fMRI Simulation Database." *Magnetic Resonance Imaging*, **27**, 1382–1396.

Bianciardi M, Cerasa A, Patria F, Hagberg G (2004). "Evaluation of Mixed Effects in Event-Related fMRI Studies: Impact of First-Level Design and Filtering." *NeuroImage*, **22**, 1351–1370.

Biswal B, DeYoe E, Hyde J (1996). "Reduction of Physiological Fluctuations in fMRI Using Digital Filters." *Magnetic Resonance in Medicine*, **35**, 107–113.

Bloch F (1946). "Nuclear Induction." *Physical Review*, **70**(7–8), 460–474.

Bullmore E, Brammer M, Williams S, Rabe-Hesketh S, Janot N, David A, Mellers J, Howard R, Sham P (1996). "Statistical Methods of Estimation and Inference for Functional MR Image Analysis." *Magnetic Resonance in Medicine*, **35**, 261–277.

Buxton R, Uludăg K, Dubowitz D, Liu T (2004). "Modeling the Hemodynamic Response to Brain Activation." *NeuroImage*, **23**, S220–S233.

Cox R (1996). "**AFNI**: Software for Analysis and Visualization of Functional Magnetic Resonance Neuroimages." *Computers and Biomedical Research*, **29**, 162–173.

David O, Guillemain I, Saillet S, Reyt S, Deransart C, Segebarth C, Depaulis A (2008). "Identifying Neural Drivers with Functional MRI: An Electrophysiological Validation." *PLoS Biology*, **6**(12), e315.

Drobnjak I, Gavaghan D, Süli E, Pitt-Francis J, Jenkinson M (2006). "Development of a fMRI Simulator for Modelling Realistic Rigid-Body Motion Artifacts." *Magnetic Resonance in Medicine*, **56**(2), 364–380.

Drysdale P, Huber J, Robinson P, Aquino K (2010). "Spatiotemporal BOLD Dynamics from a Poroelastic Hemodynamic model." *Journal of Theoretical Biology*, **265**(4), 524–34.

Friston K, Fletcher P, Josephs O, Holmes A, Rugg M, Turner R (1998). "Event-Related fMRI: Characterizing Differential Responses." *NeuroImage*, **7**, 30–40.

Glover G (1999). "Deconvolution of Impulse Response in Event-Related BOLD fMRI." *NeuroImage*, **9**, 416–429.

Granert O (2010). ***Rniftilib**: R Interface to **NIFTICLIB** (V1.1.0).* R package version 0.0-29, URL http://CRAN.R-project.org/package=Rniftilib.

Grinband J, Wager T, Lindquist M, Ferrera V, Hirsch J (2008). "Detection of Time-Varying Signals in Event-Related fMRI Designs." *NeuroImage*, **43**, 509–520.

Gudbjartsson H, Patz S (1995). "The Rician Distribution of Noisy MRI Data." *Magnetic Resonance in Medicine*, **34**(6), 910–914.

Haacke E, Brown R, Thompson M, Venkatesan R (1999). *Magnetic Resonance Imaging: Principles and Sequence Design.* John Wiley & Sons, New York.

Hansen L, Nielsen F, Strother S, Lange N (2001). "Consensus Inference in Neuroimaging." *NeuroImage*, **13**, 1212–1218.

Henson R, Shallice T, Gorno-Tempini ML, Dolan R (2002). "Face Repetition Effects in Implicit and Explicit Memory Tests as Measured by fMRI." *Cerebral Cortex*, **12**, 178–186.

Hyde J, Biswal B, Jesmanowicz A (2001). "High-Resolution fMRI Using Multislice Partial k-Space GR-EPI With Cubic Voxels." *Magnetic Resonance in Medicine*, **46**, 114–125.

Krüger G, Glover G (2001). "Physiological Noise in Oxygenation-Sensitive Magnetic Resonance Imaging." *Magnetic Resonance in Medicine*, **46**, 631–637.

Lange N (1999). "Statistical Thinking in Functional and Structural Magnetic Resonance Neuroimaging." *Statistics in Medicine*, **18**, 2401–2407.

Lange N, Strother S, Anderson J, Nielsen F, Holmes A, Kolenda T, Savoy R, Hansen L (1999). "Plurality and Resemblance in fMRI Data Analysis." *NeuroImage*, **10**, 282–303.

Lazar N (2008). *The Statistical Analysis of Functional MRI Data.* Springer-Verlag.

Lee J, Hu J, Gao J, Crosson B, Peck K, Wierenga C, McGregor K, Zhao Q, White K (2008). "Discriminating Brain Activity from Task-Related Artifacts in Functional MRI: Fractal Scaling Analysis Simulation and Application." *NeuroImage*, **40**, 197–212.

Lei X, Qiu C, Xu P, Yao D (2010). "A Parallel Framework for Simultaneous EEG/fMRI Analysis: Methodology and Simulation." *NeuroImage*, **52**, 1123–1134.

Liao W, Chen H, Yang Q, Lei X (2008). "Analysis of fMRI Data Using Improved Self-Organizing Mapping and Spatio-Temporal Metric Hierarchical Clustering." *IEEE Transactions of Medical Imaging*, **27**, 1472–1483.

Lin QH, Liu J, Zheng YR, Liang H, Calhoun V (2010). "Semiblind Spatial ICA of fMRI Using Spatial Constraints." *Human Brain Mapping*, **31**, 1076–1088.

Locascio J, Jennings P, Moore C, Corkin S (1997). "Time Series Analysis in the Time Domain and Resampling Methods for Studies of Functional Magnetic Resonance Brain Imaging." *Human Brain Mapping*, **5**, 168–193.

Logan B, Rowe D (2004). "An Evaluation of Thresholding Techniques in fMRI Analysis." *NeuroImage*, **22**, 95–108.

Purdon P, Weisskoff R (1998). "Effect of Temporal Autocorrelation Due to Physiological Noise and Stimulus Paradigm on Voxel-Level False-Positive Rates in fMRI." *Human Brain Mapping*, **6**, 239–249.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL `http://www.R-project.org/`.

Rowe D, Logan B (2004). "A Complex Way to Compute fMRI Activation." *NeuroImage*, **23**, 1078–1092.

Skudlarski P, Constable R, Gore J (1999). "ROC Analysis of Statistical Methods Used in Functional MRI: Individual Subjects." *NeuroImage*, **9**, 311–329.

Smith A, Lewis B, Ruttimann U, Ye F, Sinnwell T, Yang Y, Duyn J, Frank J (1999). "Investigation of Low Frequency Drift in fMRI Signal." *NeuroImage*, **9**, 526–533.

Smith S, Jenkinson M, Woolrich M, Beckmann C, Behrens T, Johansen-Berg H, Bannister P, De Luca M, Drobnjak I, Flitney D, Niazy R, Saunders J, Vickers J, Zhang Y, De Stefano N, Brady J, Matthews P (2004). "Advances in Functional and Structural MR Image Analysis and Implementation as **FSL**." *NeuroImage*, **23**, S208–S219.

Soetaert K, Petzoldt T, Setzer RW (2010). "Solving Differential Equations in R: Package **deSolve**." *Journal of Statistical Software*, **33**(9), 1–25. URL http://www.jstatsoft.org/v33/i09/.

Sotero R, Trujillo-Barreto N (2008). "Biophysical Model for Integrating Neuronal Activity, EEG, fMRI and Metabolism." *NeuroImage*, **39**, 290–309.

Sotero R, Trujillo-Barreto N, Jiménez J, Carbonell F, Rodríguez-Rojas R (2009). "Identification and Comparison of Stochastic Metabolic/Hemodynamic Models (sMHM) for the Generation of the BOLD Signal." *Journal of Computational Neuroscience*, **26**, 251–269.

Weibull A, Gustavsson H, Mattsson S, Svensson J (2008). "Investigation of Spatial Resolution, Partial Volume Effects and Smoothing in Functional MRI Using Artificial 3D Time Series." *NeuroImage*, **41**, 346–353.

Whitcher B, Schmid V, Thornton A (2011a). ***oro.nifti**: Rigorous – NIfTI Input / Output*. R package version 0.2.6, URL http://CRAN.R-project.org/package=oro.nifti.

Whitcher B, Schmid VJ, Thornton A (2011b). "Working with the DICOM and NIfTI Data Standards in R." *Journal of Statistical Software*, **44**(6), 1–28. URL http://www.jstatsoft.org/v44/i06/.

Affiliation:

Marijke Welvaert
Department of Data Analysis
Ghent University
H.Dunantlaan 1,
B-9000 Gent, Belgium
E-mail: Marijke.Welvaert@Ugent.be
URL: www.da.ugent.be/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 44, Issue 10 *Submitted:* 2010-10-31
October 2011 *Accepted:* 2011-06-21