fmri: A Package for Analyzing fmri Data

by J. Polzehl and K. Tabelow

This article describes the usage of the R package **fmri** to analyze single time series BOLD fMRI (blood-oxygen-level dependent functional magnetic resonance imaging) data using structure adaptive smoothing procedures (Propagation- Separation approach) as described in (Tabelow et al., 2006). See (J. Polzehl and K. Tabelow, 2006) for an extended documentation.

Analysing fMRI data with the fmri package

The approach implemented in the **fmri** package is based on a linear model for the hemodynamic response and structural adaptive spatial smoothing of the resulting Statistical Parametric Map (SPM). The package requires R (version $\geq$ 2.2). 3D visualization needs the R package **tkrplot**.

The statistical modeling implemented with this software is described in (Tabelow et al., 2006).

> **NOTE!** This software comes with absolutely **no warranty**! It is **not** intended for clinical use, but for evaluation purposes only. Absence of bugs can not be guaranteed!

We first start with a basic script for using the **fmri** package in a typical fmri analysis. In the following sections we describe the consecutive steps of the analysis.

```
# read the data
data <- read.AFNI("afnifile")
# or read sequence of ANALYZE files
# analyze031file.hdr ... analyze137file.hdr
# data <- read.ANALYZE("analyze",
#              numbered = TRUE, "file", 31, 107)

# create expected BOLD signal and design matrix
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
x <- fmri.design(hrf)

# generate parametric map from linear model
spm <- fmri.lm(data, x)

# adaptive smoothing with maximum bandwith hmax
spmsmooth <- fmri.smooth(spm, hmax = 4)

# calculate p-values for smoothed parametric map
pvalue <- fmri.pvalue(spmsmooth)

# write slicewise results into a file or ...
plot(pvalue, maxpvalue = 0.01, device = "jpeg",
                           file = "result.jpeg")
# ... use interactive 3D visualization
plot(pvalue, maxpvalue = 0.01, type = "3d")
```

Reading the data

The **fmri** package can read ANALYZE (Mayo Foundation, 2001), AFNI- HEAD/BRIK (Cox, 1996), NIFTI and DICOM files. Use

```
data <- read.AFNI(<filename>)
data <- read.ANALYZE(<filename>)
```

to create the object `data` from the data in file 'filename'. Drop the extension in 'filename'. While AFNI data is generally given as a four dimensional datacube in one file, ANALYZE format data often comes in a series of numbered files. A sequence of images in ANALYZE format can be imported by

```
data <- read.ANALYZE(prefix = "", numbered = TRUE,
                     postfix = "", picstart = 1,
                     numbpic = 1)
```

Setting the argument `numbered` to TRUE allows to read a list of image files. The image files are assumed to be ordered by a number, contained in the filename. `picstart` is the number of the first file, and `numbpic` the number of files to be read. The increment between consecutive numbers is assumed to be 1. `prefix` specifies a string preceeding the number, whereas `postfix` identifies a string following the number in the filename. Note, that `read.ANALYZE()` requires the file names in the form: '<prefix>number<postfix>.hdr/img'.

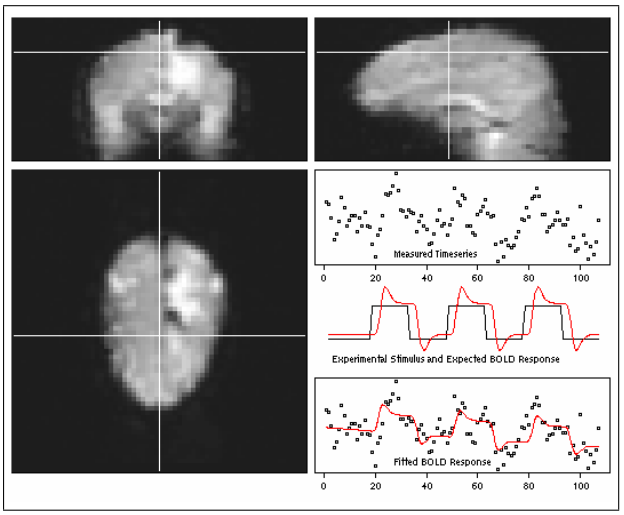

Figure 1: View of a typical fMRI dataset. Data dimension is 64x64x26 with a total of 107 scans. A high noise level is typical. The time series is described by a linear model containing the experimental stimulus (red) and quadratic drift. Data: courtesy of H. Voss, Weill Medical College of Cornell University.

Both functions return lists of class "fmridata" with components containing the datacube ('ttt'). See figure 1 for an illustration. The data cube is stored

in ('raw') format to save memory. The data can be extracted from the list using `extract.data()`. Additionally the list contains basic information like the size of the data cube ('dim') and the voxel size ('delta'), as well as the complete header information ('header'), which is itself a list with components depending to the data format read. A head mask is defined by simply using a 75% quantile of the data grey levels as cut- off. This is only be used to provide improved spatial correlation estimates for the head in `fmri.lm()`.

Expected BOLD response

In voxel affected by the experiment the observed signal is assumed to follow the expected Blood Oxygenation Level Dependent (BOLD) signal. This signal depends on both the experimental stimulus, described by a task indicator function and a hemodynamic response function $h(t)$. We define $h(t)$ as the difference of two gamma functions

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

with $a_1 = 6$, $a_2 = 12$, $b_1 = 0.9$, $b_2 = 0.9$, and $d_i = a_i b_i (i = 1, 2)$, $c = 0.35$ where t is the time in seconds, see (Glover, 1999). The expected BOLD response is given as a discrete convolution of this function with the task indicator function. Use

```
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
```

to create the expected BOLD response for a stimulus with 107 scans, onset times at the 18th, 48th, and 78th scan with a duration of the stimulus of 15 scans, and a time $TR = 2$s between two scans. In case of multiple stimuli the results of `fmri.stimulus()` for the different stimuli should be arranged as columns of a matrix `hrf` using `cbind()`.

The hemodynamic response function may have an unknown latency (Worsley and Taylor, 2005). This can be modeled creating an additional explanatory variable as the first numerical derivative of any experimental stimulus:

```
dhrf <- (c(0,diff(hrf)) + c(diff(hrf),0))/2
```

See the next section for how to include this into the linear model.

Construction of the SPM

We adopt the common view of a linear model for the time series $Y_i = (Y_{it})$ in each voxel i after reconstruction of the raw data and motion correction.

$$Y_i = X\beta_i + \varepsilon_i, \quad (1)$$

where X denotes the design matrix. The design matrix is created by

```
x <- fmri.design(hrf) .
```

This will include polynomial drift terms up to quadratic order. To deviate from this default, the order of the polynomial drift can be specified by a second argument. Use `cbind()` to combine several stimulus vectors into a matrix of stimuli before calling `fmri.design()`.

The first q columns of X contain values of the expected BOLD response for the different stimuli evaluated at scan acquisition times. The other $p - q$ columns are chosen to be orthogonal to the expected BOLD responses and to account for a slowly varying drift and possible other external effects. The error vector ε_i has zero expectation and is assumed to be correlated in time. In order to access the variability of the estimates of β_i correctly we have to take the correlation structure of the error vector ε_i into account. We assume an AR(1) model to be sufficient for commonly used MRI scanners. The autocorrelation coefficients ρ_i are estimated from the residual vector $r_i = (r_{i1}, \ldots, r_{iT})$ of the fitted model (1) as

$$\bar{\rho}_i = \sum_{t=2}^{T} r_{it} r_{i(t-1)} / \sum_{t=1}^{T} r_{it}^2.$$

This estimate of the correlation coefficient is biased due to fitting the linear model (1). We therefore apply the bias correction given by (Worsley et al., 2002) leading to an estimate $\tilde{\rho}_i$.

We then use prewhitening to transform model (1) into a linear model with approximately uncorrelated errors. The prewhitened linear model is obtained by multiplying the terms in (1) with some matrix A_i depending on $\tilde{\rho}_i$. The prewhitening procedure thus results in a new linear model

$$\tilde{Y}_i = \tilde{X}_i \beta_i + \tilde{\varepsilon}_i \quad (2)$$

with $\tilde{Y}_i = A_i Y_i$, $\tilde{X}_i = A_i X$, and $\tilde{\varepsilon}_i = A_i \varepsilon_i$. In the new model the errors $\tilde{\varepsilon}_i = (\tilde{\varepsilon}_{it})$ are approximately uncorrelated in time t, such that $\operatorname{var} \tilde{\varepsilon}_i = \sigma_i^2 I_T$. Finally least squares estimates $\tilde{\beta}_i$ are obtained from model (2) as

$$\tilde{\beta}_i = (\tilde{X}_i^T \tilde{X}_i)^{-1} \tilde{X}_i^T \tilde{Y}_i.$$

The error variance σ_i^2 is estimated from the residuals $\tilde{r}_i$ of the linear model (2) as $\tilde{\sigma}_i^2 = \sum_1^T \tilde{r}_{it}^2/(T-p)$ leading to estimated covariance matrices

$$\operatorname{var} \tilde{\beta}_i = \tilde{\sigma}_i^2 (\tilde{X}_i^T \tilde{X}_i)^{-1}.$$

Estimates $\tilde{\beta}_i$ and their estimated covariance matrices are, in the simplest case, obtained by

```
spm <- fmri.lm(data, x)
```

where `data` is the data object read by `read.AFNI()` or `read.ANALYZE()`, and `x` is the design matrix created with `fmri.design()`.

See figure 1 for an example of a typical fMRI dataset together with the result of the fit of the linear model to a time series.

To consider more than one stimulus and to estimate an effect

$$\tilde{\gamma} = c^T \tilde{\beta} \tag{3}$$

defined by a vector of contrasts c set the argument `contrast` of the function `fmri.lm()` correspondingly

```
hrf1 <- fmri.stimulus(214, c(18, 78, 138), 15, 2)
hrf2 <- fmri.stimulus(214, c(48, 108, 168), 15, 2)
x <- fmri.design(cbind(hrf1, hrf2))

# stimulus 1 only
spm1 <- fmri.lm(data, x, contrast = c(1,0))

# stimulus 2 only
spm2 <- fmri.lm(data, x, contrast = c(0,1))

# contrast between both
spm3 <- fmri.lm(data, x, contrast = c(1,-1))
```

If the argument `vvector` is set, the component "cbeta" of the object returned by `fmri.lm()` contains a vector with the parameters corresponding to the non- zero elements in `vvector` in each voxel. This may be used to include unknown latency of the hemodynamic response function in the analysis. First define the expected BOLD response for a given stimulus and its derivative and then combine them into the design matrix

```
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
dhrf <- (c(0,diff(hrf)) + c(diff(hrf),0))/2
x <- fmri.design(cbind(hrf, dhrf))
spm <- fmri.lm(data, x, vvector = c(1,1)) .
```

The specification of `vvector` in the last statement results in a vector of length 2 containing the two parameter estimates for the expected BOLD response and its derivative in each voxel. Furthermore the ratio of the variance estimates for these parameters is calculated as a prerequisite for the smoothing procedure. See `fmri.smooth()` for details about smoothing this parametric map.

The function returns an object with class attributes "fmridata" and "fmrispm". This is again a list with components containing the estimated parameter contrast ('cbeta'), and its estimated variance ('var'), as well as estimated spatial correlations in all directions.

Structure Adaptive Smoothing (PS)

Smoothing of SPM's is applied in this context to improve the sensitivity of signal detection. This is expected to work since neural activations extends over regions of adjacent voxels. Averaging over such regions allows us to reduce the variance of the parameter estimates without compromising their mean. Introducing a spatial correlation structure also reduces the number of independent decisions made for signal detection and therefore eases the multiple test problem. Adaptive smoothing as implemented in this package also allows to improve the specificity of signal detection, see figure 2 for an illustration.

The parameter map is smoothed with

```
spmsmooth <- fmri.smooth(spm, hmax = hmax)
```

where `spm` is the result of the function `fmri.lm()`. `hmax` is the maximum bandwidth for the smoothing algorithm. For `lkern="Gaussian"` the bandwidth is given in units of FWHM, for any other localization kernel the unit is voxel. `hmax` should be chosen as the expected maximum size of the activation areas. As adaptive smoothing automatically adapts to different sizes and shapes of the activation areas, oversmoothing is not to be expected.

In (Tabelow et al., 2006) the use of a spatial adaptive smoothing procedure derived from the Propagation- Separation approach (Polzehl and Spokoiny, 2006) has been proposed in this context. The approach focuses, for each voxel i, on simultaneously identifying a region where the unknown parameter γ is approximately constant and to obtain an optimal estimate $\hat{\gamma}_i$ employing this structural information. This is achieved by an iterative procedure. Local smoothing is restricted to local vicinities of each voxel, that are characterized by a weighting scheme. Smoothing and characterization of local vicinities are alternated. Weights for a pair of voxels i and j are constructed as a product of kernel weights $K_{\mathrm{loc}}(\delta(i,j)/h)$, depending on the distance $\delta(i,j)$ between the two voxels and a bandwidth h, and a factor reflecting the difference of the estimates $\hat{\gamma}_i$ and $\hat{\gamma}_j$ obtained within the last iteration. The bandwidth h is increased with iterations up to a maximal bandwidth h_{max}.

The name Propagation- Separation is a synonym for the two main properties of this algorithm. In case of a completely homogeneous array $\tilde{\Gamma}$, that is $\mathbb{E}\,\tilde{\gamma} \equiv Const.$, the algorithm delivers essentially the same result as a nonadaptive kernel smoother employing the bandwidth h_{max}. In this case the procedure selects the best of a sequence of almost nonadaptive estimates, that is, it propagates to the one with maximum bandwidth. Separation means that as soon as within one iteration step significant differences of $\hat{\gamma}_i$ and $\hat{\gamma}_j$ are observed the corresponding weight is decreased to zero and the information from voxel j is no longer used to estimate γ_i. Voxels i and j belong to different regions of homogeneity and are therefore separated. As a consequence smoothing is restricted to regions with approximately constant values of γ, bias at the border of such regions is avoided and the spatial structure of activated regions

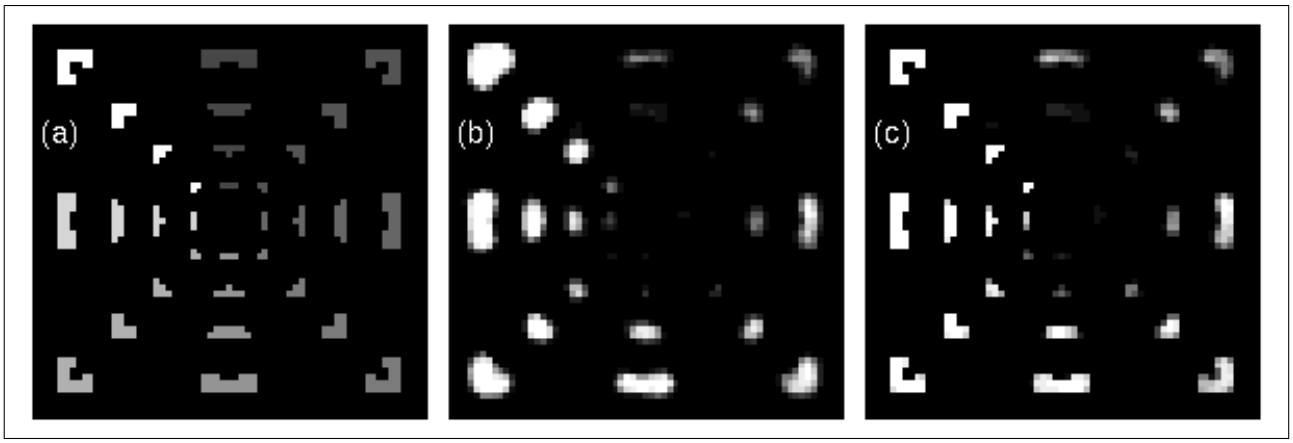

Figure 2: A numerical phantom for studying the performance of PS vs. Gaussian smoothing. (a) Signal locations within one slices. Eight different signal- to- noise ratios, increasing clockwise, are coded by gray values. The spatial extent of activations varies in the radial direction. The data cube contains 15 slices with activation alternated with 9 empty slices. (b) Smoothing with a conventional Gaussian filter. (c) Smoothing with PS. In both (b) and (c), the bandwidth is 3 voxels which corresponds to FWHM = 10 mm for typical voxel size. In (b) and (c) the proportion, over slices containing activations, of voxels that are detected in a given position is rendered by gray values. Black corresponds to the absence of a detection.

is preserved.

For a formal description of this algorithm, a discussion of its properties and theoretical results we refer to (Polzehl and Spokoiny, 2006) and (Tabelow et al., 2006). Numerical complexity, as well as smoothness within homogeneous regions is controlled by the maximum bandwidth $h_{\max}$.

If the argument object contains a parameter vector for each voxel (for example to include latency, see section 4) these will be smoothed according to their estimated variance ratio, see (Tabelow et al., 2006) for details on the smoothing procedure.

Figure 2 provides a comparison of signal detection employing a Gaussian filter and structural adaptive smoothing.

Signal detection

Smoothing leads to variance reduction and thus signal enhancement. It leaves us with three dimensional arrays $\hat{\Gamma}$, $\hat{S}$ containing the estimated effects $\hat{\gamma}_i = c^T\hat{\beta}_i$ and their estimated standard deviations $\hat{s}_i = (c^T \operatorname{var}\hat{\beta}_i c)^{1/2}$, obtained from time series of smoothed residuals. The voxelwise quotient $\hat{\theta}_i = \hat{\gamma}_i/\hat{s}_i$ of both arrays forms a statistical parametric map (SPM) $\hat{\Theta}$. The SPM as well as a map of p-values are generated by

```
pvalue <- fmri.pvalue(spmsmooth)
```

Under the hypothesis, that is, in absence of activation this SPM behaves approximately like a Gaussian Random Field, see (Tabelow et al., 2006). We therefore use the theory of Gaussian Random Fields to assign appropriate p-values as a prerequisite for signal detection. Such p-values can be defined (Worsley et al., 1996) as

$$p_i = \sum_{d=0}^{3} R_d(V(r_x, r_y, r_z))\rho_d(\hat{\theta}_i) \qquad (4)$$

where $R_d(V)$ is the *resel count* of the search volume V and $\rho_d(\hat{\theta}_i)$ is the *EC density* depending only on the parameter $\hat{\theta}_i$. r_x, r_y, r_z denotes the effective FWHM bandwidths that measure the smoothness (in resel space see (Worsley et al., 1996)) of the random field generated by a Gaussian filter that employs the bandwidth from the last iteration of the PS procedure (Tabelow et al., 2006). $R_d(V)$ and ρ_d are given in (Worsley et al., 1996). A signal will be detected in all voxels where the observed p-value is less or equal to a specified threshold.

Finally we provide a statistical analysis including unknown latency of the hemodynamic response function. If `spmsmooth` contains a vector (see `fmri.lm()` and `fmri.smooth()`), a χ^2 statistic is calculated from the first two parameters and used for p-value calculation. If `delta` is given, a cone statistics is used (Worsley and Taylor, 2005).

The parameter `mode` allows for different kinds of p-value calculation. "basic" corresponds to a global definition based on the amount of smoothness achieved by an equivalent Gaussian filter. The propagation condition ensures, that under the hypothesis $\hat{\Theta} = 0$ the adaptive smoothing perform like a non adaptive filter with the same kernel function. "local" corresponds to a more conservative setting, where the p-values are derived from the estimated local resel counts that has been achieved by the adaptive smoothing. "global" takes a global median of these resel counts for calculation.

Figure 3 provides the results of signal detection for an experimental fMRI data set.

Viewing results

Results can be displayed by a generic plot function

```
plot(object, anatomic,
     device="jpeg", file="result.jpeg")
```

`object` is an object of class "fmridata" (and "fmrispm" or "fmripvalue") as returned by `fmri.pvalue()`, `fmri.smooth()`, `fmri.lm()`, `read.ANALYZE()` or `read.AFNI()`. `anatomic` is an anatomic underlay of the same dimension as the functional data. The argument `type="3d"` provides an interactive display to produce 3 dimensional illustrations (requires R- package **tkrplot**) on the screen. The default for `type` is "slice", which can create output to the screen and image files.

We also provide generic functions `summary()` and `print()` for objects with class attribute "fmridata".

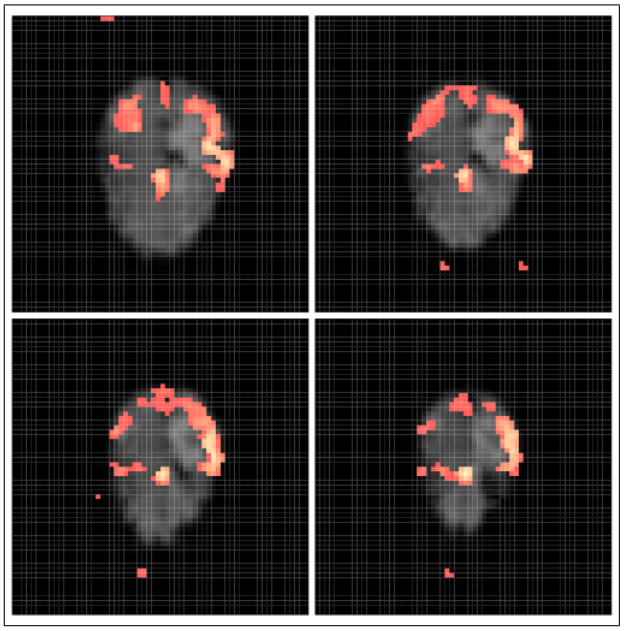

Figure 3: Signal detection in 4 consecutive slices using adaptive smoothing procedure. The shape and extent of the activation areas are conserved much better than with traditional Gaussian filtering. Data: courtesy of H. Voss, Weill Medical College of Cornell University.

Writing the results to files

Finally we provide functions to write data in standard medical image formats such as HEAD/BRIK, ANALYZE, and NIFTI.

```
write.AFNI("afnitest", data, c("signal"),
           note="random data", origin=c(0,0,0),
           delta=c(4,4,5), idcode="unique ID")
write.ANALYZE(data, list(pixdim=c(4,4,4,5)),
              file="analyzetest")
```

Some basic header information can be provided, in case of ANALYZE files as a list with several elements (see documentation for syntax). Any datacube created during the analysis can be written.

Bibliography

Biomedical Imaging Resource. *Analyze Program*. Mayo Foundation, 2001.

R. W. Cox. Afni: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomed. Res.*, 29:162- 173, 1996.

G. H. Glover. Deconvolution of impulse response in event- related BOLD fMRI. *NeuroImage*, 9:416- 429, 1999.

J. Polzehl and V. Spokoiny. Propagation- separation approach for local likelihood estimation. *Probab. Theory and Relat. Fields*, 135:335- 362, 2006.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3- 900051- 07- 0.

K. Tabelow, J. Polzehl, H. U. Voss, and V. Spokoiny. Analyzing fMRI experiments with structural adaptive smoothing procedures. *NeuroImage*, 33:55- 62, 2006.

J. Polzehl and K. Tabelow. Analysing fMRI experiments with the fmri package in R. Version 1.0 - A users guide. *WIAS Technical Report No. 10*, 2006.

K. J. Worsley. Spatial smoothing of autocorrelations to control the degrees of freedom in fMRI analysis. *NeuroImage*, 26:635- 641, 2005.

K. J. Worsley, C. Liao, J. A. D. Aston, V. Petre, G. H. Duncan, F. Morales, and A. C. Evans. A general statistical analysis for fMRI data. *NeuroImage*, 15:1- 15, 2002.

K. J. Worsley, S. Marrett, P. Neelin, K. J. Friston, and A. C. Evans. A unified statistical approach for determing significant signals in images of cerebral activation. *Human Brain Mapping*, 4:58- 73, 1996.

K. J. Worsley and J. E. Taylor. Detecting fMRI activation allowing for unknown latency of the hemodynamic response. *Neuroimage*, 29:649- 654, 2006.

Jörg Polzehl & Karsten Tabelow
Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany
polzehl@wias-berlin.de
tabelow@wias-berlin.de