Weierstraß-Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Technical Report ISSN 1618 – 7776

Analysing fMRI experiments with the fmri package in R. Version 1.0 - A users guide.

Jörg Polzehl[1] , Karsten Tabelow [1] [2]

submitted: June 27, 2006

[1] Weierstrass Institute
for Applied Analysis and Stochastics,
Mohrenstr. 39, 10117
Berlin, Germany
E-Mail: `polzehl@wias-berlin.de`
E-Mail: `tabelow@wias-berlin.de`

No. 10

Berlin 2006

[2]Supported by the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin
2000 *Mathematics Subject Classification.* 62G05.

Key words and phrases. adaptive weights; local structure; local polynomial regression, propagation, separation.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
10117 Berlin
Germany

Fax:	+ 49 30 2044975
E-Mail:	preprint@wias-berlin.de
World Wide Web:	http://www.wias-berlin.de/

Contents

Abstract

This document describes the usage of the R package *fmri* to analyse functional Magnetic Resonance Imaging (fMRI) data with structure adaptive smoothing procedures (Propagation-Separation (PS) approach) as described in [7].

1 Analysing fMRI data with the fmri package

Functional Magnetic Resonance Imaging (fMRI) is a non-invasive tool for studying the functionality of the brain and localizing cognitive functions. It has become increasingly important in neurosciences as well as for clinical applications such as presurgical planning. In a typical fMRI experiment with block design the patient has to perform one or several tasks alternated by some period of rest. However, activation in brain is not subject to direct measurement. fMRI-experiments therefore detect a change in blood oxygenation (BOLD response) [3, 4] instead. A higher oxygenation level is associated with increased neuronal activity necessary to solve the task.

The *fmri* package was build to provide an **easy-to-use** interface for the analysis of fMRI data. The approach is based on a linear model for the hemodynamic response according to the Blood Oxygen Level Dependent (BOLD) effect and structural adaptive spatial smoothing of the resulting Statistical Parametric Map (SPM). The package requires R (version ≥ 2.2) [6] (binaries available for Linux **and** Windows) to be installed. 3D visualization needs the R-package *tkrplot*.

Sections marked with a $\star$ are not required in order to understand how to write an R-script based on the package and can thus be dropped at first reading. However, they contain information on the implementation of the algorithm if this is not straightforward.

The statistical modelling implemented with this software is described in [7].

NOTE! This software comes with absolutely **no warranty**! It is **not** intended for clinical use, but for research 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 of this manual we describe the consecutive steps of the analysis. This is accompanied by a complete documentation of the user-level functions of the package (also available through R's help system).

```
# load the package
library(fmri)

# 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)

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

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

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

2 Reading the data

The *fmri* package can read AFNI-HEAD/BRIK [2] and ANALYZE [1] files. Use

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

to simply read the data in the file <filename> into the object *data*. 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. The syntax to read them is as follows:

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

Setting the *numbered* argument to TRUE starts reading a list of files. *prefix* contains the string before the number, whereas *postfix* contains the string after the number in the filename. *picstart* is the number of the first file, and *numbpic* the number of files to be read. Note, that read.ANALYZE() requires the file names in the form:

```
<prefix>007<postfix>.hdr/img
```

It may well be that your scanner does not match this convention. If so, you have the possibility to combine the files into a four dimensional data cube with some third-party software first or ask the authors of this package to include more conventions into the package.

Both functions return lists of class "fmridata" with list elements corresponding to the datacube ('ttt'), basic information on the data ('dim', 'delta' etc.), as well as the complete header information ('header'), which is itself a list with elements corresponding to the data format read. A head mask is defined by simply considering a 75% quantile of the data grey levels as cut-off. This will only be used to provide improved spatial correlation estimates for the head in fmri.lm().

`read.AFNI` *I/O function*

Description

Read HEAD/BRIK file.

Usage

```
read.AFNI(file)
```

Arguments

`file`	name of the file (without extension)

Details

The function reads a HEAD/BRIK file.

Value

Object of class "fmridata" with the following list entries:

`ttt`	4 dimensional datacube, the first three dimensions are voxel dimensions, the fourth dimension denotes the time
`header`	header information list
`format`	data source. string "HEAD/BRIK"
`delta`	voxel size in mm
`origin`	position of the datacube origin
`orient`	data orientation code. see AFNI documentation
`dim`	dimension of the datacube
`weights`	weights vector coding the relative voxel sizes in x, y, z-direction.
`mask`	head mask

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

References

R. W. Cox (1996). AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages. Computers and Biomed. Res. 29:162-173.

See Also

`write.AFNI`, `read.ANALYZE`

Examples

```
## Not run: afni <- read.AFNI("afnifile")
```

`read.ANALYZE` *I/O Functions*

Description

Read fMRI data from ANALYZE file(s).

Usage

```
read.ANALYZE(prefix = "", numbered = FALSE, postfix = "",
             picstart = 1, numbpic = 1)
```

Arguments

`prefix`	string. part of the file name before the number
`numbered`	logical. if `FALSE` only `prefix` is taken as file name (default).
`postfix`	string. part of the file name after the number
`picstart`	number of the first image to be read.
`numbpic`	number of images to be read

Details

This function reads fMRI data files in ANALYZE format. It takes the strings in `prefix` and `postfix` and a number of the form ”007” in between to create the file name.

If `numbered` is `FALSE`, only the string in `prefix` is used for file name (default).

The number is assumed to be 3 digits (including leading zeros). First number is given in `picstart`, while `numbpic` defines the total number of images to be read. Data in multiple files will be combined into a four dimensional datacube.

Value

Object of class ”fmridata” with the following list entries:

`ttt`	four dimensional data cube, the first three dimensions are voxel dimensions, the fourth dimension denotes the time
`header`	header information of the data
`format`	data source. string ”ANALYZE”
`delta`	voxel size in mm
`origin`	position of the datacube origin
`orient`	data orientation code
`dim`	dimension of the datacube
`weights`	weights vector coding the relative voxel sizes in x, y, z-direction
`mask`	head mask

Note

Since numbering and naming of ANALYZE files widely vary, this function may not meet your personal needs. See Details section above for a description.

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

References

Biomedical Imaging Resource (2001). Analyze Program. Mayo Foundation.

See Also

`write.ANALYZE`, `read.AFNI`

Examples

```
## Not run: analyze <- read.ANALYZE("analyze",TRUE,"file",31,107)
```

3 Create the expected BOLD response and design matrix

The creation of the design includes the creation of the expected BOLD response due to experimental stimuli. For the hemodynamic response function $h(t)$ the difference of two gamma functions is used:

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

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. The expected BOLD response is given as a discrete convolution of this function with the task indicator function. No sub-sampling is done. Use

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

to create 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. Create as much stimuli as you need (pay attention that the number of scans is equal in all of them). Once finished with this, create a design matrix:

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

This will include polynomial drift terms up to quadratic order. If you want to deviate from this default, add a second argument with the order of the polynomial drift you want to specify. Use cbind() to combine several stimulus vectors into a matrix of stimuli before calling fmri.design(). The design matrix specifies the linear model used to construct the SPM.

The hemodynamic response function may have an unknown latency [11]. To model this effect one can create an additional explanatory variable from the first 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.

`fmri.stimulus` *Linear Model for FMRI Data*

Description

Create the expected BOLD response for a given task indicator function.

Usage

```
fmri.stimulus(scans = 1, onsets = c(1), length = 1, rt = 3,
              mean = TRUE,
              a1 = 6, a2 = 12, b1 = 0.9, b2 = 0.9, cc = 0.35)
```

Arguments

`scans`	number of scans
`onsets`	vector of onset times (in scans)
`length`	length of ON stimulus (in scans)
`rt`	time between scans in seconds (TR)
`mean`	logical. if TRUE the mean is substracted from the resulting vector
`a1`	parameter of the hemodynamic response function (see details)
`a2`	parameter of the hemodynamic response function (see details)
`b1`	parameter of the hemodynamic response function (see details)
`b2`	parameter of the hemodynamic response function (see details)
`cc`	parameter of the hemodynamic response function (see details)

Details

The functions calculates the expected BOLD response for the task indicator function given by the argument as a convolution with the hemodynamic response function. The latter is modelled by the difference between two gamma functions as given in the reference (with the defaults for a1, a2, b1, b2, cc given therein):

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

The parameters of this function can be changed through the arguments `a1`, `a2`, `b1`, `b2`, `cc`.

The dimension of the function value is set to `c(scans,1)`.

If `mean` is TRUE (default) the resulting vector is corrected to have zero mean.

Value

Vector with dimension `c(scans,1)`.

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

References

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

See Also

`fmri.design`, `fmri.lm`

Examples

```
# Example 1
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
z <- fmri.design(hrf,2)
par(mfrow=c(2,2))
for (i in 1:4) plot(z[,i],type="l")
```

`fmri.design` *Linear Model for FMRI Data*

Description

Return a design matrix for a linear model with given stimuli and possible polynomial drift terms.

Usage

```
fmri.design(hrf, order = 2)
```

Arguments

`hrf`	matrix containing expected BOLD repsonse(s) for the linear model as columns
`order`	order of the polynomial drift terms

Details

The stimuli given in `hrf` are used as first columns in the design matrix.

The order of the polynomial drift terms is given by `order`, which defaults to 2.

The polynomials are defined orthogonal to the stimuli given in `hrf`.

Value

design matrix of the linear model

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

See Also

`fmri.stimulus`, `fmri.lm`

Examples

```
# Example 1
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
z <- fmri.design(hrf,2)
par(mfrow=c(2,2))
for (i in 1:4) plot(z[,i],type="l")
```

4 Estimation of parameters in the linear model

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.

$$Y_i = X\beta_i + \varepsilon_i, \tag{1}$$

where X denotes the design matrix. 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. Here 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 $r_i = (r_{i1}, \dots, r_{iT})$ 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 [9] 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

$$\tilde{Y}_i = \tilde{X}_i \beta_i + \tilde{\varepsilon}_i \tag{2}$$

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 Var $\varepsilon_i = \sigma_i^2 I_T$. Finally least squares estimates $\tilde{\beta}_i$ are obtained from model (2) as

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

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

$$\text{Var } \tilde{\beta} = \tilde{\sigma}^2 (\tilde{X}^T \tilde{X})^{-1}.$$

This is, in the simplest case, done 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().

To consider more than one stimulus and to estimate an effect

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

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))
spm1 <- fmri.lm(data, x, contrast = c(1,0)) # stimulus 1 only
spm2 <- fmri.lm(data, x, contrast = c(0,1)) # stimulus 2 only
spm3 <- fmri.lm(data, x, contrast = c(1,-1)) # contrast between both
```

If the argument "vvector" is set, the list element "cbeta" of the fmri.lm() value 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 into 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 an estimate of the ratio of the variances of these parameters is calculated and used for smoothing. See fmri.smooth() for details about smoothing this parametric map. Do not mix with the *contrast* argument described above since unexpected side-effects may occur.

The function returns an object with class attributes "fmridata" and "fmrispm". This is again a list with elements that contain the estimated parameter contrast ('cbeta'), and its estimated variance ('var'), as well as estimated spatial correlations in all directions (see documentation of the function fmri.lm for details).

Algorithm*

This section describes the implementation of the function in more detail.

Estimation of the parameters β and its variance in the linear model (1) is given by

$$\hat{\beta} = (X^T X)^{-1} X^T Y = BY$$

and

$$\text{Var } \hat{\beta} = \sigma^2 (X^T X)^{-1}.$$

We use the singular value decomposition of the design matrix $X = U \Lambda V^T$ to calculate the estimates. Thus:

$$B = V \Lambda^{-2} U^T$$

and therefore we have

$$\begin{aligned} \hat{\beta} &= V \Lambda^{-2} U^T Y \\ \text{Var } \hat{\beta} &= \sigma^2 V \Lambda^{-2} V^T \end{aligned}$$

We now calculate the residuals r_i of the model for each voxel i as

$$r = Y - X\beta$$

The errors ϵ in the linear model are correlated in time. We use an AR(1) model here, and estimate its parameter ρ_i at voxel i with bias reduction according to [9]. Therefore

$$\tilde{\rho_i} = \frac{v_1}{v_0}$$

with

$$v = M^{-1} a$$

and

$$a = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}, \qquad M = \begin{pmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \end{pmatrix}$$

and finally

$$a_l = \sum_{t=l+1}^{T} r_{it} r_{it-l}$$

and

$$m_{lj} = \begin{cases} tr(RD_l) & j = 0, \\ tr(RD_l R(D_j + D_j^T)) & j = 1 \end{cases}$$

where D_l is a matrix of zeros with ones on the lth upper off-diagonal.

With the map of autocorrelation coefficients $\tilde{\rho}_i$ at hand we perform the prewhitening. We have for the variance of ε:

$$\text{Var } \varepsilon = \sigma^2 \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \cdots & \rho^1 \end{pmatrix}$$

We now consider the Choleski decomposition of this correlation matrix:

$$\text{Var } \varepsilon = \sigma^2 SS^T$$

If we choose $A = S^{-1}$ we find (assuming $E\varepsilon = 0$)

$$\text{Var } (A\varepsilon) \quad = \quad A\text{Var } \varepsilon A^T = \sigma^2 I$$

Thus $A\varepsilon$ are uncorrelated. A plug-in estimate with ρ replaced by $\tilde{\rho}_i$ is used for prewithening.

The fmri.lm() function provides the possibility to neglect prewhitening as well as to smooth the map of autocorrelation parameters ρ, with a bandwidth $FWHM_{filter}$ to achieve a target degree of freedom *df* for the random t-field given by the parameter estimate divided by its estimated standard deviation [8]. This is controlled by the argument ”actype” (see documentation for details).

However, since the *fmri* package performs data smoothing on the map of estimated parameters rather than on the original data, we achieve a sufficiently high number of degrees of freedom for the resulting random field to be gaussian through the smoothing of the SPM [7]. It therefore is not necessary to smooth the autocorrelation coefficients. However, we provide this possibility, if one wants to apply the package for spatially smoothed data, without using adaptive weights smoothing for the statistical parameter map.

`fmri.lm` *Linear Model For FMRI Data*

Description

Estimate the parameters and variances in a linear model.

Usage

```
fmri.lm(data, z, actype = "accalc", hmax = 3.52, vtype = "var",
        step = 0.01, contrast = c(1), vvector = c(1),
        keep = "essential")
```

Arguments

`data`	object of class "fmridata"
`z`	designmatrix specifying the expected BOLD response(s) and additional components for trend and other effects.
`actype`	string describing the type of handling autocorrelation of time series. "nonac", "ac", "accalc", "smooth"
`hmax`	bandwidth for smoothing autocorrelation parameter if actype = "smooth"
`vtype`	method of estimating residual variance (only "var" implemented)
`step`	step size for binning autocorrelations (see details)
`contrast`	contrast vector
`vvector`	vector defining the parameters for which the covariance matrix is returned as well as the corresponding length of the vector `cbeta` in each voxel
`keep`	string describing the amount of data returned: "essential", "diagnostic", "all"

Details

This function performs parameter estimation in the linear model. It implements a two step procedure. After primary estimation of the parameters in the first step residuals are obtained. If `actype %in% c("ac", "accalc", "smooth")` an AR(1) model is fitted, in each voxel, to the time series of residuals. The estimated AR-coefficient is corrected for bias. If `actype=="smooth"` the estimated AR-coefficients are spatially smoothed using bandwidth `hmax`. If `actype %in% c("ac", "smooth")` the linear model is prewithened using the estimated (smoothed) AR-coefficients. Parameter and variance estimates are then obtained from the prewithened data. The argument `keep` describes the amount of data which is returned. If "essential" only the estimated effects

$$\tilde{\gamma}_i = C^T \tilde{\beta}_i$$

and their estimated variances are returned. "all" gives the full data, including residuals, temporal autocorrelation. If `vvector` is given and has length greater than 1, the covariance matrix for the stimuli given therein are returned (`varm`) and `vwghts` contains an estimate for the ratio of the variances of the parameter for the stimuli indicated in `vvector`. `cbeta` then contains the corresponding parameter estimates and thus is a vector of corresponding length in each voxel.

Value

object with class attributes "fmrispm" and "fmridata"

`beta`	estimated parameters
`cbeta`	estimated contrast of parameters
`var`	estimated variance of the contrast of parameters.
`varm`	covariance matrix of the parameters given by `vvector`
`res`	residuals of the estimated linear model
`arfactor`	estimated autocorrelation parameter
`scorr`	spatial correlation of data
`weights`	ratio of voxel dimensions
`vwghts`	ratio of estimated variances for the stimululi given by `vvector`
`rxyz`	array of smoothness from estimated correlation for each voxel in resel space (for analysis without smoothing)
`hrf`	expected BOLD response for contrast

Note

`vvector` is intended to be used for delay of the HRF using its first derivative. Do not mix with the `contrast` argument, since unexpected side effects may occur. Look out for updates of this package.

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

References

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

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

See Also

`fmri.design`, `fmri.stimulus`

Examples

```
# Example 1
data <- list(ttt=array(rnorm(32*32*32*107),c(32,32,32,107)),
             mask=array(1,c(32,32,32)))
class(data) <- "fmri.data"
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
z <- fmri.design(hrf,2)
model <- fmri.lm(data,z,keep="all")
plot(data$ttt[16,16,16,])
lines(data$ttt[16,16,16,] - model$res[16,16,16,],col=2)
```

5 Structure Adaptive Smoothing

The parameter map is smoothed with

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

where *spm* is the result of the fmri.lm() function. *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 size of 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 expected.

In [7] the use of a spatial adaptive smoothing procedure derived from the Propagation-Separation approach [5] 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 $\boldsymbol{E}\tilde{\gamma} \equiv Const.$, 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 is preserved.

For a formal description of this algorithm, a discussion of its properties and theoretical results we refer to [5] and [7]. 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 how to do) these will be smoothed according to their estimated variance ratio, see [7] for details on the smoothing procedure.

`fmri.smooth` *Smoothing Statistical Parametric Maps*

Description

Perform the adaptive weights smoothing procedure

Usage

```
fmri.smooth(spm, hmax = 4, adaptive=TRUE,
            lkern="Triangle", skern="Triangle")
```

Arguments

`spm`	object of class `fmrispm`
`hmax`	maximum bandwidth to smooth
`adaptive`	logical. TRUE (default) for adaptive smoothing
`lkern`	`lkern` specifies the location kernel. Defaults to "Triangle", other choices are "Gaussian", "Quadratic", "Cubic" and "Uniform". Note that the location kernel is applied to `(x-x_j)^2/h^2`, i.e. the use of "Triangle" corresponds to the Epanechnicov kernel in nonparametric kernel regression.
`skern`	`skern` specifies the kernel for the statistical penalty. Defaults to "Triangle", the alternative is "Exp". `lkern="Triangle"` allows for much faster computation (saves up to 50%).

Details

This function performs the smoothing on the Statistical Parametric Map spm.

`hmax` is the (maximal) bandwidth used in the last iteration. Choose `adaptive` as `FALSE` for non adaptive smoothing. `lkern` can be used for specifying the localization kernel. For comparison with non adaptive methods use "Gaussian" (hmax given in FWHM), for better adaptation use "Triangle" (default, hmax given in voxel). `skern` can be used for specifying the kernel for the statistical penalty.

The function handles zero variances by assigning a large value (1e20) to these variances.

Value

object with class attributes "fmrispm" and "fmridata"

`cbeta`	smoothed parameter estimate
`var`	variance of the parameter
`hmax`	maximum bandwidth used
`rxyz`	smoothness in resel space. all directions
`rxyz0`	smoothness in resel space as would be achieved by a Gaussian filter with the same bandwidth. all directions
`scorr`	spatial correlation of original data
`weights`	ratio of voxel dimensions
`vwghts`	ratio of estimated variances for the stimuli given by `vvector`
`hrf`	Expected BOLD response for the specified effect

Author(s)

Joerg Polzehl `polzehl@wias-berlin.de`, Karsten Tabelow `tabelow@wias-berlin.de`

References

Tabelow, K., Polzehl, J., Voss, H.U., and Spokoiny, V. (2005). *Analysing fMRI experiments with structure adaptive smoothing procedures*, NeuroImage, accepted (2006).

Polzehl, J. and Spokoiny, V. (2006). *Propagation-Separation Approach for Local Likelihood Estimation*, Probab. Theory Relat. Fields 135, 335-362.

Examples

```
## Not run: fmri.smooth(spm, hmax = 4, lkern = "Gaussian")
```

6 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 $\hat{s}_i = (c^T \text{Var } \hat{\beta}_i c)^{1/2}$. 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 [7]. 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 [10] as

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

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 [10]) of the random field generated by a Gaussian filter that employs the bandwidth from the last iteration of the PS procedure [7]. It has been given in [10] that

$$\begin{aligned}
R_0(V) &= 1, \\
R_1(V) &= (x-1)r_x + (y-1)r_y + (z-1)r_z, \\
R_2(V) &= (x-1)(y-1)r_x r_y + (y-1)(z-1)r_y r_z + (x-1)(z-1)r_x r_z, \\
R_3(V) &= (x-1)(y-1)(z-1)r_x r_y r_z
\end{aligned} \tag{4}$$

and

$$\begin{aligned}
\rho_0(z) &= \int_z^\infty \frac{1}{(2\pi)^{0.5}} \exp(-u^2/2)du, \\
\rho_1(z) &= \frac{(4\ln 2)^{\frac{1}{2}}}{(2\pi)} \exp(-t^2/2)\ , \\
\rho_2(z) &= \frac{(4\ln 2)}{(2\pi)^{\frac{3}{2}}} \exp(-t^2/2)\ t, \\
\rho_3(z) &= \frac{(4\ln 2)^{\frac{3}{2}}}{(2\pi)^2} \exp(-t^2/2)\ (t^2-1)
\end{aligned} \tag{5}$$

A signal will be detected in all voxels where the observed p-values 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 [11].

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 a 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.

`fmri.pvalue` *P-values*

Description

Determine p-values.

Usage

```
fmri.pvalue(spm, mode="basic", delta=NULL)
```

Arguments

`spm`	`fmrispm` object
`mode`	type of pvalue definition
`delta`	physically meaningful range of latency for HRF

Details

If only a contrast is given in `spm`, we simply use a t-statistic and define p-values according to random field theory for the resulting gaussian field (sufficiently large number of df - see ref.). If `spm` is a vector of length larger than one for each voxel, a chisq field is calculated and evaluated (see Worsley and Taylor (2006)). If `delta` is given, a cone statistics is used.

The parameter `mode` allows for different kinds of p-value calculation. "basic" corresponds to a global definition of the resel counts based on the amount of smoothness achieved by an equivalent Gaussian filter. The propagation condition ensures, that under the hypothesis

$$\hat{\Theta} = 0$$

adaptive smoothing performs like a non adaptive filter with the same kernel function which justifies this approach. "local" corresponds to a more conservative setting, where the p-value is derived from the estimated local resel counts that has been achieved by adaptive smoothing. In contrast to "basic", "global" takes a global median to adjust for the randomness of the weighting scheme generated by adaptive smoothing. "global" and "local" are more conservative than "basic", that is, they generate sligthly larger p-values.

Value

Object with class attributes "fmripvalue" and "fmridata"

`pvalue`	p-value. use with `plot` for thresholding.

`weights`	voxelsize ratio
`dim`	data dimension
`hrf`	expected BOLD response for contrast (single stimulus only)

Note

Unexpected side effects may occur if spm does not meet the requirements, especially if a parameter estimate vector of length greater than 2 through argument `vvector` in `fmri.lm` has beeen produced for every voxel.

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

References

Tabelow, K., Polzehl, J., Voss, H.U., and Spokoiny, V. (2005). *Analysing fMRI experiments with structure adaptive smoothing procedures*, NeuroImage, accepted (2006).

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

See Also

`fmri.smooth`, `plot.fmridata`

Examples

```
## Not run: fmri.pvalue(smoothresult)
```

7 Plot the 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".

`plot.fmridata` *I/O functions*

Description

Visualize fMRI data and (intermediate) results.

Usage

```
plot.fmridata(x, anatomic = NULL, maxpvalue = 0.05,
              spm = TRUE, pos = c(-1, -1, -1), type = "slice",
              device = "X11", file = "plot.png",...)
```

Arguments

`x`	object of class ”fmripvalue”, ”fmrispm” or ”fmridata”
`anatomic`	overlay of same dimension as the functional data
`maxpvalue`	maximum p-value for thresholding
`spm`	logical. if class is ”fmrispm” decide whether to plot the t-statistics for the estimated effect (`spm=TRUE`) or the estimated effect itself (`spm=FALSE`).
`pos`	voxel to be marked on output
`type`	string. ”slice” for slicewise view and ”3d” for 3d view.
`device`	output device if `type` is `slice`. ”png”, ”jpeg”, ”ppm”, default is ”X11”
`file`	name of output file if `device` is not ”X11”
`...`	additional arguments for plot

Details

Provides a sliceswise view of ”fmridata” objects with anatomic overlay (if appropriate, that is for class ”fmripvalue”). For objects of class ”fmrispm” it plots the t-statistics for the estimated effects if `spm` is `TRUE`, or the estimated effect otherwise. For objects of class ”fmridata” only a plot of the data slices itself is produced. If `device` is specified as ”png”, ”jpeg”, ”ppm” output is done to a file. A grey/color scale is provided in the remaining space.

If `type` is ”3d” a 3 dimensional interactive view opens. Sliders to move in the data cube are given (”x”, ”y”, ”z”, and ”t” if class is ”fmridata” only). Time series are shown if available. For objects of class ”fmrispm” a slider is created to remove information for voxels with smaller signals than a cut-off value from the plot. Use pvalues for statistical evaluation. If `spm` is `TRUE` the estimated BOLD response together with a confidence interval corresponding to `maxpvalue` is drawn. For objects of class ”fmripvalue” the pvalues with overlay are shown.

Value

If 'type' is "3d" the Tk-object is returned. (Remove the diplay with `tkdestroy(object)`)

Note

3 dimensional plotting requires the `tkrplot` package.

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

See Also

`fmri.pvalue`

Examples

```
## Not run: plot(pvalue)
```

`print.fmridata` *I/O functions*

Description

'print' method for class '"fmridata"'.

Usage

```
print.fmridata(x, ...)
```

Arguments

`x`	an object of class `fmridata`, usually, a result of a call to `fmri.lm`, `fmri.smooth`, `fmri.pvalue`, `read.AFNI`, or `read.ANALYZE`.
`...`	further arguments passed to or from other methods.

Details

The method tries to print information on data, like data dimension, voxel size, value range.

Value

none

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

See Also

`summary.fmridata`

Examples

```
## Not run: print(data)
```

`summary.fmridata` *I/O functions*

Description

'summary' method for class '"fmridata"'.

Usage

```
summary.fmridata(object, ...)
```

Arguments

`object`	an object of class `fmridata`, usually, a result of a call to `fmri.lm`, `fmri.smooth`, `fmri.pvalue`, `read.AFNI`, or `read.ANALYZE`.
`...`	further arguments passed to or from other methods.

Details

The method tries to print information on data, like data dimension, voxel size, value range.

Value

A list with the following elements:

`dim`	data dimension
`delta`	voxel dimension, if available
`values`	value range
`z`	design matrix

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

See Also

`print.fmridata`

Examples

```
## Not run: summary(data)
```

8 Writing the results to files

Finally we provide two functions to write out data to standard medical image formats such as HEAD/BRIK and ANALYZE.

```
write.AFNI("afnitest", array(as.integer(65526*runif(10*10*10*20)),
           c(10,10,10,20)), c("signal"), note="random data",
           origin=c(0,0,0), delta=c(4,4,5), idcode="unique ID")

write.ANALYZE(array(as.integer(65526*runif(10*10*10*20)),c(10,10,10,20)),
              list(pixdim=c(4,4,4,5)),
              file="analyzetest")
```

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

write.AFNI *I/O functions*

Description

Write BRIK/HEAD files.

Usage

```
write.AFNI(file, ttt, label, note = "", origin = c(0, 0, 0),
               delta = c(4, 4, 4), idcode = "WIAS_noid")
```

Arguments

`file`	name of the file
`ttt`	datacube
`label`	labels (BRICK_LABS)
`note`	notes on data (HISTORY_NOTE)
`origin`	origin of datacube (ORIGIN)
`delta`	voxel dimensions (DELTA)
`idcode`	idcode of data (IDCODE_STRING)

Details

Write out BRIK/HEAD files as required by AFNI. Most arguments correspond to entries in the HEAD file.

Value

Nothing is returned.

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

See Also

`read.AFNI`,`write.ANALYZE`

Examples

```
 write.AFNI("afnifile", array(as.integer(65526*runif(10*10*10*20)),
     c(10,10,10,20)), c("signal"), note="random data",
     origin=c(0,0,0), delta=c(4,4,5), idcode="unique ID")
```

`write.ANALYZE` *I/O Functions*

Description

Write a 4 dimensional datacube in ANALYZE file format.

Usage

```
write.ANALYZE(ttt, header=NULL, file)
```

Arguments

`ttt`	4 dimensional datacube
`header`	header information
`file`	file name

Details

Writes the datacube `ttt` to a file named `file` in ANLYZE file format. `header` is a list that contains the header information as documented by the Mayo Foundation. We give here a short summary. If a value is not provided, it will be tried to fill it with reasonable defaults, but do not expect fine results, if the entry has a special important meaning (h.i. pixdim).

[1]	datatype1 – 10 byte character	[2]	dbname – 18 byte character
[3]	extents – integer	[4]	sessionerror – integer
[5]	regular – character	[6]	hkey – character
[7]	dimension – 8 integers, dimensions ...	[8]	unused – 7 integers
[9]	datatype – integer, datatype usually ”4”	[10]	bitpix – integer
[11]	dimun0 – integer	[12]	pixdim – 8 floats, voxel dimensions ...
[13]	voxoffset – float	[14]	funused – 3 floats
[15]	calmax – float	[16]	calmin – float
[17]	compressed – float	[18]	verified – float
[19]	glmax – integer	[20]	glmin – integer
[21]	describ – 80 byte character	[22]	auxfile – 24 byte character

[23]	orient – character	[24]	originator – 5 integers
[25]	generated – 10 byte character	[26]	scannum – 10 byte character
[27]	patientid – 10 byte character	[28]	expdate – 10 byte character
[29]	exptime – 10 byte character	[30]	histun0 – 3 byte character
[31]	views – integer	[32]	voladded – integer
[33]	startfield – integer	[34]	fieldskip – integer
[35]	omax – integer	[36]	omin – integer
[37]	smax – integer	[38]	smin – integer

See ANALYZE documentation for details.

Value

Nothing is returned.

Author(s)

Karsten Tabelow `tabelow@wias-berlin.de`

See Also

`read.ANALYZE`, `write.AFNI`

Examples

```
## Example 1
write.ANALYZE(array(as.integer(65526*runif(10*10*10*20)),c(10,10,10,20)),
              file="analyzefile")
```

References

[1] Biomedical Imaging Resource. *Analyze Program.* Mayo Foundation, 2001.

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

[3] N. Lange. Statistical approaches to human brain mapping by functional magnetic resonance imaging. *Stat. in Medicine*, 15:389–428, 1996.

[4] N. Lange and S.L. Zeger. Non-linear Fourier time series analysis for human brain mapping by functional magnetic resonance imaging. *J. Roy. Statist. Soc. Ser. C*, 46:1–29, 1997.

[5] J. Polzehl and V. Spokoiny. Propagation-separation approach for local likelihood estimation. *Probab. Theory and Relat. Fields, in print: http://dx.doi.org/10.1007/s00440-005-0464-1*, 2005.

[6] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0.

[7] K. Tabelow, J. Polzehl, H. U. Voss, and V. Spokoiny. Analyzing fMRI experiments with structural adaptive smoothing procedures. *NeuroImage, accepted*, 2006.

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

[9] 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.

[10] 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.

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