The fmri Package

February 8, 2009

Version 1.2-6

Date 2008-05-19

Title Analysis of fMRI experiments

Author Karsten Tabelow <tabelow@wias-berlin.de>, Joerg Polzehl <polzehl@wias-berlin.de>

Maintainer Karsten Tabelow <tabelow@wias-berlin.de>

Suggests tkrplot,fastICA

Description The library contains R-functions to perform an fmri analysis as described in Tabelow, K., Polzehl, J., Voss, H.U., and Spokoiny, V. Analysing fMRI experiments with structure adaptive smoothing procedures, NeuroImage, 33:55-62 (2006).

License GPL (>= 2)

URL http://www.wias-berlin.de/projects/matheon_a3/

Note This software comes with NO warranty! It is NOT intended to be used in clinical applications! For evaluation only!

R topics documented:

`cutroi` *I/O function*

Description

This functions cuts a region-of-interest (ROI) from input data.

Usage

```
cutroi(data, xind = 1:data$dim[1], yind = 1:data$dim[2], zind = 1:data$dim[3], tinc
```

Arguments

`data`	Object of class fmridata.
`xind`	list of roi-indices for first data index
`yind`	list of roi-indices for second data index
`zind`	list of roi-indices for third data index
`tind`	list of roi-indices for 4th data index

Details

Cut a region of interest from frmidata.

Value

Corresponding cutted fmridata object.

Author(s)

Karsten Tabelow ⟨tabelow@wias-berlin.de⟩

See Also

`read.AFNI`, `read.ANALYZE`, `read.NIFTI`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--    or do  help(data=index)  for the standard data sets.
```

`extract.data` *Extract data or residuals from a fmridata object*

Description

The function extracts data stored as raw within an object of class 'fmridata'.

Usage

```
extract.data(z, what = "data")
```

Arguments

`z`	an object of class 'fmridata'
`what`	either `"data"` or `"residuals"`.

Details

The function extracts data stored as raw within an object of class 'fmridata'.

Value

an array of dimension `data$dim` containing either the fmri-data or residuals.

Author(s)

Joerg Polzehl ⟨polzehl@wias-berlin.de⟩

See Also

`fmri.lm`

`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")
```

`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 = "all")
```

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. If `keep=="all"` residuals are included in the returned object. This triggers variance estimation in `fmri.smooth` to based on adaptively smoothed residuals. Otherwise variance estimation is based on the estimated smoothness of the measured data. This is less memory and time consuming, but expected to be less accurate.

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.

If warning "Local smoothness characterized by large bandwidth" occurs, check `scorr` elements. If correlation drops with lag towards zero, data has been pre-smoothed. Adaption can then only be of limited use. If correlation does not go to zero, check the residuals of the linear model for unexplained structure (spin saturation in first scans? discuard them!).

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`	raw (integer size 2) vector containing residuals of the estimated linear model up to scale factor resscale.
`resscale`	`resscale*extract.data(object,"residuals")` are the residuals.
`dim`	dimension of the data cube and residuals
`arfactor`	estimated autocorrelation parameter
`rxyz`	array of smoothness from estimated correlation for each voxel in resel space (for analysis without smoothing)
`scorr`	array of spatial correlations with maximal lags 5, 5, 3 in x,y and z-direction.
`bw`	vector of bandwidths (in FWHM) corresponding to the spatial correlation within the data.
`weights`	ratio of voxel dimensions
`vwghts`	ratio of estimated variances for the stimululi given by `vvector`
`mask`	head mask.
`df`	Degrees of freedom for t-statistics.
`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=writeBin(rnorm(32*32*32*107),raw(),4),
             mask=array(1,c(32,32,32)),dim=c(32,32,32,107))
class(data) <- "fmridata"
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
z <- fmri.design(hrf,2)
model <- fmri.lm(data,z,keep="all")
plot(extract.data(data)[16,16,16,])
lines(extract.data(data)[16,16,16,] - extract.data(model,"residuals")[16,16,16,],col=2)
```

---

fmri.pvalue *P-values*

Description

Determine p-values.

Usage

```
fmri.pvalue(spm, mode="basic", delta=NULL, na.rm=FALSE, minimum.signal = 0)
```

Arguments

spm	fmrispm object
mode	type of pvalue definition
delta	physically meaningful range of latency for HRF
na.rm	na.rm specifies how NA's in the SPM are handled. NA's may occur in voxel where the time series information did not allow for estimating parameters and their variances or where the time series information where constant over time. A high (1e19) value of the variance and a parameter of 0 are used to characterize NA's. If na.rm=TRUE the pvalue for the corresponding voxels is set to 1. Otehrwise pvalues are assigned according to the information found in the SPM at the voxel.

`minimum.signal`
: allows to specify a (positive) minimum value for detected signals. If minimum.signal >0 the thresholds are to conservative, this case needs further improvements.

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.. *Analysing fMRI experiments with structure adaptive smoothing procedures*, NeuroImage, 33:55-62 (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)
```

`fmri.smooth` *Smoothing Statistical Parametric Maps*

Description

Perform the adaptive weights smoothing procedure

Usage

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

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 "Gaussian", other choices are "Triangle" and "Plateau". 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. "Plateau" specifies a kernel that is equal to 1 in the interval (0,.3), decays linearly in (.5,1) and is 0 for arguments larger than 1.
`skern`	`skern` specifies the kernel for the statistical penalty. Defaults to "Plateau", the alternatives are "Triangle" and "Exp". "Plateau" specifies a kernel that is equal to 1 in the interval (0,.3), decays linearly in (.3,1) and is 0 for arguments larger than 1. `lkern="Plateau"` and `lkern="Triangle"` allow for much faster computation (saves up to 50% CPU-time). `lkern="Plateau"` produces a less random weighting scheme.
`weighted`	`weighted` (logical) determines if weights contain the inverse of local variances as a factor (Weighted Least Squares). `weighted=FALSE` does not employ the heteroscedasticity of variances for the weighting scheme and is preferable if variance estimates are highly variable, e.g. for short time series.

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 "Plateau" or "Triangle" (default, hmax given in voxel). For `lkern="Plateau"` and `lkern="Triangle"` thresholds may be inaccurate, due to a violation of the Gaussian random field assumption under homogeneity. `lkern="Plateau"` is expected to provide best results with adaptive smoothing.

`skern` can be used for specifying the kernel for the statistical penalty. "Plateau" is expected to provide the best results, due to a less random weighting scheme.

The function handles zero variances by assigning a large value (1e20) to these variances. Smoothing is restricted to voxel with `spm$mask`.

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`	array of spatial correlations with maximal lags 5, 5, 3 in x,y and z-direction.
`bw`	vector of bandwidths (in FWHM) corresponding to the spatial correlation within the data.
`dim`	dimension of the data cube and residuals
`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.. *Analysing fMRI experiments with structure adaptive smoothing procedures*, NeuroImage, 33:55-62 (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")
```

`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), durations = c(1), rt = 3,
              times= NULL, 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)
`durations`	vector of duration of ON stimulus in scans or seconds (if `!is.null(times)`)
`rt`	time between scans in seconds (TR)
`times`	onset times in seconds. If present `onsets` arguments is ignored.
`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 `!is.null(times)` durations are specified in seconds.

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")
```

---

fmriica *Independent Component Analysis*

Description

The function performs Independent Component Analysis.

Usage

```
fmriica(data, m = 3, method = "spatial", xind = NULL, yind = NULL, zind = NULL, tir
```

Arguments

`data`	Observation matrix (dimension Nxd)
`m`	Number of independent components.
`method`	Either `"spatial"` or `"temporal"`. Specifies the type of ICA to perform.
`xind`	index of x-coordinates to use
`yind`	index of y-coordinates to use
`zind`	index of z-coordinates to use
`tind`	index of time points to use
`...`	further arguments to fastICA

Details

This is still experimental code based on the package fastICA. The package fastICA seems limited in the data-size it can handle. `xind`, `yind`, `zind` and `tind` may be use to restrict the analysis to a cube in space and certain time points.

Value

The function returns a list with components

`ihat`	Matrix containing the first m ICA directions as columns.
`sdev`	Standard deviations of the principal components of the thresholded ICA directions
`xhat`	first m components of the rotated data
`v`	If `keepv==TRUE` the set of directions `v^{(k)}`
`normv`	If `keepv==TRUE` the norm of each `v^{(k)}`.

Author(s)

Jörg Polzehl polzehl@wias-berlin.de

See Also

`ngca`

---

ngca *Non-Gaussian Component Analysis*

Description

The function performs Non-Gaussian Component Analysis as described in Blanchard et.al. (2005).

Usage

```
ngca(data, L = 1000, T = 10, m = 3, eps = 1.5, npca=min(dim(x)[2],dim(x)[1]),metho
```

Arguments

`data`	Observation matrix (dimension Nxd)
`L`	Number basis functions in each of four classes.
`T`	Number of Fast ICA iterations
`m`	Number of non-Gaussian components.
`eps`	Threshold (defaults to 1.5)
`npca`	Reduce space to `npca` principal components. This can be used to avoid standardizing by numerically singular covariance matrices. In fMRI this allows to reduce the dimensionality assuming that the interesting non-Gaussian directions are also characterized by larger variances.

method	Either "spatial" or "temporal". Specifies the type of NGCA to perform.
sweepmean	either NULL, "none", "global", "spatial" or "spatial". If sweepmean==NULL the value used is determined by method.
keepv	if TRUE intermediate results from fast ICA step are kept.

Details

The function performs Non-Gaussian Component Analysis as described in Blanchard et.al. (2006). The procedure uses four classes of basis functions, i.e. Gauss-Power3, Hyperbolic Tangent and the real and complex part of the Fourier class. See Blanchard et.al. (2005) for details.

Value

The function returns a list with components

ihat	Matrix containing the first m NGCA directions as columns.
sdev	Standard deviations of the principal components of the thresholded ICA directions
xhat	first m components of the rotated data
v	If keepv==TRUE the set of directions v^{(k)}
normv	If keepv==TRUE the norm of each v^{(k)}.
...	

Author(s)

Jörg Polzehl polzehl@wias-berlin.de

References

Blanchard, G., Kawanabe, M., Sugiyama, M., Spokoiny, V. and Müller K.-R. (2005). In Search of Non-Gaussian Components of a High-Dimensional Distribution. Journal of Machine Learning Research. pp. 1-48.

---

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", slice =  1, view = "axial" ,zlim.u
              NULL, zlim.o = NULL,...)
```

Arguments

x	object of class "fmripvalue", "fmrispm" or "fmridata"
anatomic	overlay of same dimension as the functional data, or fmridata object (if of x is fmripvalue object)
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"
slice	number of slice in x, if anatomic is of "fmridata" class
view	"axial", "coronal", or "sagittal", if anatomic is of "fmridata" class
zlim.u	full range for anatomical underlay used for color scale, if anatomic is of "fmridata" class
zlim.o	full range for functional overlay used for color scale, if anatomic is of "fmridata" class
...	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 FALSE 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)
```

`read.AFNI` *I/O function*

Description

Read HEAD/BRIK file.

Usage

```
read.AFNI(filename,vol=NULL,level=0.75)
```

Arguments

`filename`	name of the file (without extension)
`vol`	vector of volumes of the dataset to be read
`level`	Quantile level defining the mask

Details

The function reads a HEAD/BRIK file. If `vol` is given (defaults to `NULL`), only volumes in this vector are read, in order to save memory.

Value

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

`ttt`	raw vector (numeric size 4) containing the four dimensional data cube (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, level = 0.75)
```

Arguments

prefix	string(s). part of the file name before the number or vector of strings for filename (if numbered is FALSE)
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
level	Quantile level defining the mask

Details

This function reads fMRI data files in ANALYZE format. If numbered is FALSE, only the vector of strings in prefix is used for file name (default).

If numbered is TRUE, it takes the first string in prefix and postfix and a number of the form "007" in between to create the file name.

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`	raw vector (numeric size 4) containing the 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)
```

`read.DICOM` *I/O function*

Description

Read DICOM file.

Usage

```
read.DICOM(filename,includedata = TRUE)
```

Arguments

filename	name of the file
includedata	logical. should data be read too? defaults to TRUE.

Details

The function reads a DICOM file.

Value

Object with the following list entries:

header	header information as raw data
ttt	image data if requested. raw vector (numeric size 4) containing the four dimensional data cube (the first three dimensions are voxel dimensions, the fourth dimension denotes the time).
format	data source. string "DICOM"
delta	voxel size in mm
series	series identifier
image	image number within series
dim	dimension of the data if available

Note

Since the DICOM standard is rather complicated, there may be cases where this function cannot read a DICOM file. Known issue: it cannot read header with implicit VR. Return value may change in future version!

Author(s)

Karsten Tabelow ⟨tabelow@wias-berlin.de⟩

References

http://medical.nema.org

See Also

read.AFNI, read.ANALYZE

Examples

```
## Not run: dicom <- read.DICOM("dicomfile")
```

`read.NIFTI` *I/O Functions*

Description

Read fMRI data from NIFTI file(s).

Usage

```
read.NIFTI(filename, level = 0.75)
```

Arguments

`filename`	name of the NIFTI file
`level`	Quantile level defining the mask

Details

This function reads fMRI data files in NIFTI format.

The filename can be given with our without extension. If extension is not included, the function searches for the ".nii" file and then for the "hdr/img" pair.

Value

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

`ttt`	raw vector (numeric size 4) containing the 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 "NIFTI"
`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

Author(s)

Karsten Tabelow ⟨tabelow@wias-berlin.de⟩

See Also

`read.ANALYZE`, `read.AFNI`

Examples

```
## Not run: analyze <- read.NIFIT("niftifile.nii")
```

`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)
```

`write.AFNI` *I/O functions*

Description

Write BRIK/HEAD files.

Usage

```
write.AFNI(filename, ttt, label = NULL, note = NULL, origin = NULL,
               delta = NULL, idcode = NULL, header = NULL, taxis = FALSE)
```

Arguments

`filename`	name of the file
`ttt`	datacube
`label`	labels (BRICK_LABS), depreciated - see header
`note`	notes on data (HISTORY_NOTE), depreciated - see header
`origin`	origin of datacube (ORIGIN), depreciated - see header
`delta`	voxel dimensions (DELTA), depreciated - see header
`idcode`	idcode of data (IDCODE_STRING), depreciated - see header
`header`	This is a list of header information such as DATASET_RANK to be written to the .HEAD file. Arguments `label`, ... are depreciated and to be substituted by a corresponding list entry. For backward compatibility the use of the old arguments is still supported and should give the same results. This will be removed in some future release! Since AFNI does not read any dataset with a header choose carefully what is written. There are some basic tests in this function, but this may not be sufficient.
`taxis`	logical (defaults to `FALSE`. Are the sub-bricks time series? This results in writing TAXIS attributes to the header file.

Details

Write out BRIK/HEAD files as required by AFNI. Most arguments correspond to entries in the HEAD file, but use is depreciated. Use `header` and `taxis` instead!

Value

Nothing is returned.

Author(s)

Karsten Tabelow ⟨tabelow@wias-berlin.de⟩

See Also

read.AFNI,write.ANALYZE

Examples

```
 ## Not run:
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")
## End(Not run)
 write.AFNI("afnifile", array(as.integer(65526*runif(10*10*10*20)),
     c(10,10,10,20)), header=list(HISTORY_NOTE="random data",
     ORIGIN=c(0,0,0), DELTA=c(4,4,5), IDCODE_STRING="unique ID"),taxis=FALSE)
```

---

write.ANALYZE *I/O Functions*

Description

Write a 4 dimensional datacube in ANALYZE file format.

Usage

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

Arguments

ttt	4 dimensional datacube
header	header information
filename	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")
```

`write.NIFTI` *I/O Functions*

Description

Write a 4 dimensional datacube in NIFTI file format.

Usage

```
write.NIFTI(ttt, header=NULL, filename)
```

Arguments

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

Details

Writes the datacube `ttt` to a file named `file` in NIFTI file format. `header` is a list that contains the header information.

See NIFTI 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.NIFTI(array(as.integer(65526*runif(10*10*10*20)),c(10,10,10,20)),
              file="niftifile")
```

Index