Package ‘fmri’

February 19, 2015

Version 1.5-1
Date 2014-06-18
Title Analysis of fMRI experiments
Suggests tcltk, tkrplot, fastICA, adimpro, oro.nifti
Description The package contains R-functions to perform an fmri analysis as described in
K. Tabelow, J. Polzehl, H.U. Voss, and V. Spokoiny,
Analysing fMRI experiments with structure adaptive smoothing procedures,
J. Polzehl, H.U. Voss, K. Tabelow, Structural adaptive segmentation
for statistical parametric mapping, NeuroImage, 52:515-523 (2010) and
K. Tabelow, J. Polzehl, Statistical Parametric Maps for
Functional MRI Experiments in {R}: The Package {fmri}),
License GPL (>= 2)
Copyright This package is Copyright (C) 2006-2014 Weierstrass
Institute for Applied Analysis and Stochastics.
URL http://www.wias-berlin.de/software/imaging/
Note This software comes with NO warranty! It is NOT intended to be
used in clinical applications! For evaluation only!
Author Karsten Tabelow [aut, cre],
Joerg Polzehl [aut],
Brandon Whitcher [ctb]
Maintainer Karsten Tabelow <tabelow@wias-berlin.de>
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-06-18 13:01:44
**R topics documented:**

- Convert Between fmridata and oro.nifti ........................................ 2
- cutroi .............................................................................. 3
- extract.data ................................................................. 4
- fmri.design ................................................................. 5
- fmri.detrend ............................................................... 6
- fmri.gui ........................................................................ 7
- fmri.lm .......................................................................... 8
- fmri.pvalue ................................................................. 10
- fmri.smooth ................................................................. 12
- fmri.stimulus ............................................................... 14
- fmriica ......................................................................... 16
- hvred ............................................................................ 17
- ngca ............................................................................. 18
- plot.fmridata ............................................................... 19
- print.fmridata ............................................................. 21
- read.AFNI ................................................................. 22
- read.ANALYZE ............................................................. 23
- read.DICOM .............................................................. 25
- read.NIFTI ................................................................. 26
- summary.fmridata ....................................................... 27
- write.AFNI ................................................................. 28
- write.ANALYZE .......................................................... 30
- write.NIFTI ............................................................. 31

**Index** ........................................................................ 33

---

**Convert Between fmridata and oro.nifti**

*Convert Between fmridata and oro.nifti Objects*

**Description**

NIfTI data can be converted between fmridata S3 objects (from the fmri package) and nifti S4 objects.

**Usage**

```r
oro2fmri(from, value = NULL, level = 0.75, setmask = TRUE)
fMRI2oro(from, value = NULL, verbose = FALSE, reorient = FALSE,
         call = NULL)
```
Arguments

- `from`: is the object to be converted.
- `value`: NULL
- `level`: is the quantile level defining the mask.
- `setmask`: is a logical variable (default = TRUE), whether to define a suitable mask based on level.
- `verbose`: is a logical variable (default = FALSE) that allows text-based feedback during execution of the function.
- `reorient`: is a logical variable (default = TRUE) that enforces Qform/Sform transformations.
- `call`: keeps track of the current function call for use in the NIfTI extension.

Details

These functions enhance the capabilities of `fmri` by allowing the exchange of data objects between `nifti` and `fmridata` classes.

Value

The function `oro2fmri` produces an S3 object of class `fmridata`. The function `fmri2oro` produces an S4 object of class `nifti`.

Author(s)

Brandon Whitcher <bwhitcher@gmail.com>

See Also

- `read.NIFTI`

Description

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

Usage

```r
cutroi(data, xind = 1:dim[1], yind = 1:dim[2],
        zind = 1:dim[3], tind = 1:dim[4])
```
extract.data

Arguments

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

Details
Cut a region of interest from fmridata.

Value
Corresponding cutted fmridata object.

Author(s)
Karsten Tabelow <tabelow@wias-berlin.de>

See Also
read.AFNI, read.ANALYZE, read.NIFTI

Examples

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

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>an object of class 'fmridata'</td>
</tr>
<tr>
<td>what</td>
<td>either &quot;data&quot; or &quot;residuals&quot;.</td>
</tr>
</tbody>
</table>
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**

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

**Arguments**

- `hrf`: matrix containing expected BOLD response(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>

**References**

See Also

* fmri.stimulus, fmri.lm

Examples

```r
# 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.detrend

Detrend fMRI time series

Description

Detrend fMRI dataset with a polynomial of given degree

Usage

fmri.detrend(data, degree = 1, accoef = 0)

Arguments

data fMRI dataset of class "fmridata"
degree Degree of the polynomial used to detrend the data. defaults to 1 (linear trends).
accoef Coefficient of AR(1) model used for prewhitening. default 0.

Details

The function can be used to detrend the time series of an fMRI dataset data (of class "fmridata"
using polynomials. If the argument degree is larger than 0 (default: 1) the polynomial trends up
to the given degree are removed from the data. If the argument accoef is larger than 0 (default: 0)
prewhitening using an AR(1) model is performed.

Value

Detrended data object of class "fmridata".

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References

Graphical user interface

Description

The function provides a graphical user interface that guides through the analysis of single subject fmri analysis from assessing the image data to visualization of results.

Usage

fmri.gui()

Value

Results of the analysis are stored in a file or saved in the global workspace.

Author(s)

Devy Hoffmann and Karsten Tabelow <tabelow@wias-berlin.de>

See Also

read.AFNI, read.ANALYZE, read.DICOM, fmri.design, fmri.stimulus, fmri.stimulus, fmri.lm, fmri.smooth, fmri.pvalue, plot.fmridata, print.fmridata, write.AFNI, write.ANALYZE, write.NIFTI

Examples

## Not run: fmri.gui()
Description

Estimate the parameters and variances in a linear model.

Usage

`fmri.lm(data, z, actype = "smooth", 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"
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 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

\[ \hat{\gamma}_i = C^T \hat{\beta}_i \]

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 vwgghts 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? discard 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
- vwgghts: ratio of estimated variances for the stimuli 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


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) <- "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(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, alpha= 0.05)

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. Othehwise 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 too conservative, this case needs further improvements.

alpha Significance level in case of mode="FDR"

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 slightly larger p-values. The alternative is mode="FDR" specifying signal detection by False Discovery Rate (FDR) with significance level specified by alpha.

Value

Object with class attributes "fmripvalue" and "fmridata"

pvalue p-value. use with plot for thresholding.

weights voxelsize ratio

dim data dimension

hmf 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 been produced for every voxel.

Author(s)

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


See Also

*fmri.smooth, plot.fmridata*

Examples

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

---

**fmri.smooth**

*Smoothing Statistical Parametric Maps*

**Description**

Perform the adaptive weights smoothing procedure

**Usage**

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

**Arguments**

- `spm`: object of class `fmrispm`
- `hmax`: maximum bandwidth to smooth
- `adaptation`: character, type of adaptation. If "none" adaptation is off and non-adaptive kernel smoothing with `lkern` and bandwidth `hmax` is used. Other values are "aws" for adaptive smoothing using an approximative correction term for spatial smoothness in the penalty (fast), "fullaws" for adaptive smoothing using variance estimates from smoothed residuals in the penalty (CPU-time about twice the time compared to adaptation="aws" and "segment" for a new approach based on segmentation using multi-scale tests.
- `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 Epanechnikov 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 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. lkernel="Plateau" and lkernel="Triangle" allow for much faster computation (saves up to 50% CPU-time). lkernel="Plateau" produces a less random weighting scheme.

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.

Further internal arguments for the smoothing algorithm usually not to be set by the user. Allows e.g. for parameter adjustments by simulation using our propagation condition. Usefull exceptions can be used for adaptation="segment": Specifically alpha (default 0.05) defines the significance level for the signal detection. It can be choosen between 0.01 and 0.2 as for other values we did not determine the critical values for the statistical tests. delta (default 0) defines the minimum signal which should be detected. restricted determines if smoothing for voxel detected to be significant is restricted to use only voxel from the same segment. The default is restricted=FALSE. restricted slightly changes the behaviour under the alterenative, i.e. not the interpretation of results.

Details

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

hmax is the (maximal) bandwidth used in the last iteration. Choose adaptation as "none" for non adaptive smoothing. lkernel can be used for specifying the localization kernel. For comparison with non adaptive methods use "Gaussian" (hmax times the voxelsize in x-direction will give the FWHM bandwidth in mm), for better adaptation use "Plateau" or "Triangle" (default, hmax given in voxel). For lkernel="Plateau" and lkernel="Triangle" thresholds may be inaccurate, due to a violation of the Gaussian random field assumption under homogeneity. lkernel="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", or "fmrisegment" and "fmridata" for segmentation choice

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
vweights: 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

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**

```r
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):

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

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


See Also

fmri.design, fmri.lm
Examples

```r
# 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")
```

Description

The function performs Independent Component Analysis.

Usage

```r
fmriica(data, m = 3, method = "temporal", xind = NULL, yind =
         NULL, zind = NULL, tind = NULL, filter.time = "None",
         filter.space = FALSE, h.space = 3, h.time = 3, keepv =
         FALSE, ...)
```

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
- `filter.time`: not yet documented
- `filter.space`: not yet documented
- `h.space`: not yet documented
- `h.time`: not yet documented
- `keepv`: not yet documented
- `...`: 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 used 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

---

**hvred**

*Translation between smoothness and bandwidth for Gaussian kernel*

Description

Translation table between smoothness and bandwidth for Gaussian kernel

Usage

data(hvred)

Format

The format is: num [1:500, 1:2] 0.101 0.102 0.103 0.104 0.105 ...

Examples

data(hvred)

## maybe str(hvred) ; plot(hvred) ...
Description

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

Usage

```r
ngca(data, L=c(1000,1000,1000), T=10, m=3, eps=1.5, npca=min(dim(x)[2],
      dim(x)[1]), filter.time="None", filter.space=FALSE, method="temporal",
      dg.trend = 2, h.space=3, h.time=3, keepv=TRUE, delta = NULL)
```

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.
filter.time: Choice of temporal filtering before analysis: "None", "Low", "Both", "High" (default "None")
filter.space: Choice of spatial filtering before analysis: logical, default FALSE
method: Either "spatial" or "temporal". Specifies the type of NGCA to perform.
dg.trend: not yet documented
h.space: bandwidth for spatial filtering, default 3
h.time: bandwidth for temporal filtering, default 3
keepv: if TRUE intermediate results from fast ICA step are kept.
delta: not yet documented

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 \( v^{*}\{k\} \).

...
Arguments

- **x**: object of class "fmrisegment", "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.
- **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
- **col.u**: color scale for anatomical underlay, if anatomic is of "fmridata" class, default grey(0:255/255)
- **col.o**: color scale for functional overlay, if anatomic is of "fmridata" class, default heat.colors(256)
- **cutoff**: not yet documented
- **verbose**: tell something on the progress?
- **...**: 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.

For objects of class "fmrisegment" the smoothed signal size is shown in the activation segments (two-sided test!).

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 display with tkdestroy(object))
Note
3 dimensional plotting requires the tkrplot package.

Author(s)
Karsten Tabelow <tabelow@wias-berlin.de>

References

See Also
fmri.pvalue

Examples

## Not run: plot(pvalue)

print.fmridata

Description
'print' method for class "fmridata".

Usage

## S3 method for class 'fmridata'
print(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>
References


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, setmask=TRUE)

Arguments

filename  name of the file (without extension)
vol  vector of volumes of the dataset to be read
level  Quantile level defining the mask
setmask  Logical (default TRUE), whether to define a suitable mask based on level

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:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttt</td>
<td>raw vector (numeric size 4) containing the four dimensional data cube (the first three dimensions are voxel dimensions, the fourth dimension denotes the time).</td>
</tr>
<tr>
<td>header</td>
<td>header information list</td>
</tr>
<tr>
<td>format</td>
<td>data source. string &quot;HEAD/BRIK&quot;</td>
</tr>
<tr>
<td>delta</td>
<td>voxel size in mm</td>
</tr>
<tr>
<td>origin</td>
<td>position of the datacube origin</td>
</tr>
<tr>
<td>orient</td>
<td>data orientation code. see AFNI documentation</td>
</tr>
<tr>
<td>dim</td>
<td>dimension of the datacube</td>
</tr>
<tr>
<td>weights</td>
<td>weights vector coding the relative voxel sizes in x, y, z-direction.</td>
</tr>
<tr>
<td>mask</td>
<td>head mask</td>
</tr>
</tbody>
</table>
Author(s)

Karsten Tabelow <tabelow@wias-berlin.de>

References


See Also

write.AFNI, read.ANALYZE

Examples

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

---

Description

Read fMRI data from ANALYZE file(s).

Usage

read.ANALYZE(prefix = "", numbered = FALSE, postfix = ",
             picstart = 1, numbpic = 1, level = 0.75,setmask=TRUE)

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

setmask Logical (default TRUE), whether to define a suitable mask based on level
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


See Also

write.ANALYZE, read.AFNI

Examples

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

Read DICOM file.

**Usage**

```r
cread.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, setmask=TRUE)

Arguments

filename name of the NIFTI file
level Quantile level defining the mask
setmask Logical (default TRUE), whether to define a suitable mask based on level

Details

This function reads fMRI data files in NIFTI format.

The filename can be given with or 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttt</td>
<td>raw vector (numeric size 4) containing the four dimensional data cube (the first three dimensions are voxel dimensions, the fourth dimension denotes the time).</td>
</tr>
<tr>
<td>header</td>
<td>header information of the data</td>
</tr>
<tr>
<td>format</td>
<td>data source. string &quot;NIFTI&quot;</td>
</tr>
<tr>
<td>delta</td>
<td>voxel size in mm</td>
</tr>
<tr>
<td>origin</td>
<td>position of the datacube origin</td>
</tr>
</tbody>
</table>
 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>

References


See Also

read.AFNI, read.ANALYZE

Examples

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

summary.fmridata  I/O functions

Description

'summary' method for class "fmridata".

Usage

## S3 method for class 'fmridata'
summary(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

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

write.AFNI I/O functions

Description
Write BRIK/HEAD files.

Usage

```r
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>

References


See Also

read.AFNI, write.ANALYZE

Examples

```r
## 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)
```
I/O Functions

**Description**

Write a 4 dimensional datacube in ANALYZE file format.

**Usage**

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

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttt</td>
<td>4 dimensional datacube</td>
</tr>
<tr>
<td>header</td>
<td>header information</td>
</tr>
<tr>
<td>filename</td>
<td>file name</td>
</tr>
</tbody>
</table>

**Details**

Writes the datacube `ttt` to a file named `filename` in ANALYZE 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
- [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 ...
- [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.
write.NIFTI

Value
Nothing is returned.

Author(s)
Karsten Tabelow <tabelow@wias-berlin.de>

References

See Also
read.ANALYZE, write.AFNI

Examples

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

Description
Write a 4 dimensional datacube in NIFTI file format.

Usage

```r
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 `filename` 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>

References

See Also
read.ANAYLZE, write.AFNI

Examples
```r
## Example 1
write.NIFTI(array(as.integer(65526*runif(10*10*10*20)),c(10,10,10,20)),
          file="niftifile")
```
Index

*Topic **IO**  
cutroi, 3  
read.AFNI, 22  
read.ANALYZE, 23  
read.DICOM, 25  
read.NIFTI, 26  
write.AFNI, 28  
write.ANALYZE, 30  
write.NIFTI, 31  

*Topic **datasets**  
hvred, 17  

*Topic **design**  
fMRI.design, 5  
fMRI.stimulus, 14  

*Topic **hplot**  
plot.fmridata, 19  

*Topic **htest**  
fMRI.pvalue, 10  

*Topic **iplot**  
plot.fmridata, 19  

*Topic **misc**  
Convert Between fmridata and oro.nifti, 2  

*Topic **multivariate**  
fMRIica, 16  
gPCA, 18  

*Topic **nonparametric**  
fMRIica, 16  
gPCA, 18  

*Topic **print**  
print.fmridata, 21  
summary.fmridata, 27  

*Topic **regression**  
fMRI.design, 5  
fMRI.detrend, 6  
fMRI.lm, 8  
fMRI.stimulus, 14  

*Topic **smooth**  
fMRI.smooth, 12  

*Topic **utilities**  
cutroi, 3  
extract.data, 4  
fMRI.gui, 7  
read.AFNI, 22  
read.ANALYZE, 23  
read.DICOM, 25  
read.NIFTI, 26  
write.AFNI, 28  
write.ANALYZE, 30  
write.NIFTI, 31  
Convert Between fmridata and oro.nifti, 2  
cutroi, 3  
extract.data, 4  
fMRI.design, 5, 7, 10, 15  
fMRI.detrend, 6  
fMRI.gui, 7  
fMRI.lm, 5–7, 8, 11, 15  
fMRI.pvalue, 7, 10, 21  
fMRI.smooth, 7, 12, 12  
fMRI.stimulus, 6, 7, 10, 14  
fMRI2oro (Convert Between fmridata and oro.nifti), 2  
fMRIica, 16  
hvred, 17  
gPCA, 17, 18  
oro2fMRI (Convert Between fmridata and oro.nifti), 2  
plot, 11  
plot.fmridata, 7, 12, 19  
plot.fmrisegment (plot.fmridata), 19  
print.fmridata, 7, 21, 28
INDEX

read.AFNI, 4, 7, 22, 24, 26, 27, 29
read.ANALYZE, 4, 7, 23, 23, 26, 27, 31, 32
read.DICOM, 7, 25
read.NIFTI, 3, 4, 26

summary.fmridata, 22, 27

write.AFNI, 7, 23, 28, 31, 32
write.ANALYZE, 7, 24, 29, 30
write.NIFTI, 7, 31