Package ‘fmri’

July 24, 2024

Version 1.9.12.1

Date 2024-07-24

Title Analysis of fMRI Experiments

Depends R (>= 3.3.0)

Imports grDevices, graphics, stats, utils, nlme, parallel, metafor, methods, aws (>= 2.5.1), oro.nifti

Suggests tcltk, tkrplot, fastICA, adimpro (>= 0.9)

LazyData true

Description Contains R-functions to perform an fMRI analysis as described in Polzehl and Tabelow (2019) <DOI:10.1007/978-3-030-29184-6>, Tabelow et al. (2006) <DOI:10.1016/j.neuroimage.2006.06.029>, Polzehl et al. (2010) <DOI:10.1016/j.neuroimage.2010.04.241>, Tabelow and Polzehl (2011) <DOI:10.18637/jss.v044.i11>.

License GPL (>= 2)

URL `https://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!

RoxygenNote 7.2.3

NeedsCompilation yes

Author Karsten Tabelow [aut, cre],
Joerg Polzehl [aut],
Brandon Whitcher [ctb],
Dames Sibylle [ctb]

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

Repository CRAN

Date/Publication 2024-07-24 14:40:02 UTC

Contents

Convert Between fmridata and oro.nifti . . . 3
cutroi . . . 4
extractData . . . 5
fmri.cluster . . . 6
fmri.design . . . 7
fmri.designG . . . 8
fmri.detrend . . . 10
fmri.lm . . . 11
fmri.lmePar . . . 13
fmri.metaPar . . . 18
fmri.pvalue . . . 23
fmri.searchlight . . . 25
fmri.sgroupICA . . . 26
fmri.sICA . . . 27
fmri.smooth . . . 28
fmri.stimulus . . . 31
gen_fmridata . . . 33
getSearchlightPattern . . . 33
hvred . . . 34
ICAfingerprint . . . 34
niftiImage2fmri . . . 36
plot.fmridata . . . 37
plot.fmriICA . . . 39
plot.fmripvalue . . . 40
print.fmridata . . . 41
read.AFNI . . . 42
read.ANALYZE . . . 43
read.DICOM . . . 45
read.NIFTI . . . 46
setmask . . . 47
sincfilter . . . 48
slicetiming . . . 49
summary.fmridata . . . 50
write.AFNI . . . 51
write.ANALYZE . . . 52
write.NIFTI . . . 53

Index **55**

`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

```
oro2fmri(from, value = NULL, level = 0.75, mask=NULL, 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.
`mask`	array or nifti-object containing the mask. If set this replaces the mask defined by argument level.
`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`

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$dim[3], tind = 1:data$dim[4])
```

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 frmidata.

Value

Corresponding cut 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.
```

`extractData` *Extract data or residuals from a fmridata object*

Description

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

Usage

```
extractData(z, what = "data", maskOnly = FALSE)
expandfMRI(z)
condensefMRI(z, mask)
```

Arguments

`z`	an object of class 'fmridata'
`what`	either `"data"` or `"residuals"`.
`maskOnly`	logical: if TRUE only values within the brain mask will be returned.
`mask`	logical brain mask

Details

The function `extractData` extracts data stored as raw within an object of class 'fmridata'. Functions `expandfMRI` and `condensefMRI` change the way data and residuals are stored between full 3D data and data within a brain mask. `condensefMRI` can also be used to set a more restrictive brain mask.

Value

In case of function `extractData` an array of dimension `data$dim` containing either the fmri-data or residuals. The other two functions return an object of class 'fmridata'.

Author(s)

Joerg Polzehl `<polzehl@wias-berlin.de>`

See Also

`fmri.lm`

`fmri.cluster` *Cluster thresholding.*

Description

Detection of activated regions using cluster thresholding.

Usage

```
fmri.cluster(spm, alpha = 0.05, ncmin = 2, ncmax=ncmin,
             minimum.signal = 0, verbose = FALSE)
```

Arguments

`spm`	`fmrispm` object
`alpha`	multiple test (over volume and cluster sizes) adjusted significance level used for thresholds.
`ncmin`	minimal cluster size used. An activation is detected if for any clustersize in `nvmin:20` the size specific threshold is exceeded.
`ncmax`	maximal cluster size used. An activation is detected if for any clustersize in `ncmin:ncmax` the size specific threshold is exceeded.
`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.
`verbose`	intermediate diagnostics

Details

Approximate thresholds for the existence of a cluster with spm-values exceeding a `1-beta` threshold `k_{nc,na:ne}` for cluster size `nc` are based on a simulation study under the hypothesis and adjusted for number of voxel in mask and spatial correlation. `beta` is chosen such that under the hypothesis the combined (over cluster sizes `ncmin:ncmax`) test has approximate significance level `alpha`.

Value

Object with class attributes "fmripvalue" and "fmridata"

`pvalue`	cluster based p-values for voxel that were detected for any cluster size, a value of `1` otherwise.
`mask`	mask of detected activations
`weights`	voxelsize ratio
`dim`	data dimension
`hrf`	expected BOLD response for contrast (single stimulus only)

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

See Also

`fmri.lm`, `fmri.pvalue`, `fmri.searchlight`

Examples

```
## Not run: fmri.cluster(fmrispmobj)
```

`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(stimulus, order = 2, cef = NULL, verbose = FALSE)
```

Arguments

`stimulus`	matrix containing expected BOLD response(s) for the linear model as columns or list of expected BOLD responses containing matrices of dimension `scans`, `number of slices` as returned by function `fmri.stimulus`.
`order`	order of the polynomial drift terms
`cef`	confounding effects
`verbose`	Report more if `TRUE`

Details

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

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

Confounding effects can be included in a matrix `cef`.

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

Value

design matrix of the linear model

Author(s)

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

References

Polzehl, J. and Tabelow, K.(2007). *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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.designG *Design matrix for fMRI group analysis*

Description

This function returns a design matrix for multi-subject fMRI data to fit a Linear Mixed-effects Model (one-stage procedure) with given stimuli, polynomial drift terms and a set of known population parameters.

Usage

```
fmri.designG(hrf, subj = 1, runs = 1, group = NULL, XG = NULL)
```

Arguments

hrf	vector or matrix containing expected BOLD response(s) for one session, typically a fmri.stimulus object.
subj	number of subjects in the study.
runs	number of repeated measures within subjects.
group	optional vector to define groups. It is expected one value per subject. A grouping factor can also be part of XG.
XG	optionally, a group-level design matrix of class "data.frame", which contains population parameters such as ages or gender corresponding to the subjects. It is expected one value per subject.

Details

Based on the dimensionality of the `hrf` object, which provides the total number of scans (time-points) within each session, the entered number of subjects and repeated measures the auxiliary variables: "subj", "run", "scan" and "session" are generated as first part of the returned design matrix.

If no `group` argument is specified, only one population will be assumed; otherwise group labels are replicated within sessions of the same subject.

First a design matrix for a single run is created by calling: `x <- fmri.design(hrf, order = 2)`. Hence the polynomial drift terms are defined orthogonal to the stimuli (see `fmri.design`). This matrix is replicated blockwise to all sessions assuming the same experimental design for all runs. The first drift term, a column of ones, is called "drift0" and models an intercept.

If given, further subject characteristics are filled in the design matrix.

Value

A design matrix as a data frame, which contains the following variables:

`subj`	consecutive subject number: 1 to `subj` specified as factor
`run`	consecutive run number within the subjects: 1 to `runs` specified as factor
`scan`	consecutive scan number: 1 to T within each session
`session`	consecutive experiment number: 1 to `(subj*runs)` specified as factor
`group`	grouping variable specified as factor, one group by default
`hrf, hrf2, ...`	replicated expected BOLD-response(s)
`drift0, drift1, drift2`	replicated polynomial drift terms created with `fmri.design(hrf, order = 2)` orthogonal to the stimuli given in `hrf`
`...`	further expanded between-subject factors and covariates

Author(s)

Sibylle Dames

References

Polzehl, J. and Tabelow, K.(2007). *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17.

See Also

`fmri.stimulus`, `fmri.design`, `fmri.lmePar`

Examples

```
subj <- 6
runs <- 1
scans <- 121
times <- c(12, 48, 84, 120, 156, 192, 228, 264)
duration <- 24
```

```
tr <- 2.5

hrf <- fmri.stimulus(scans, times, duration, tr, times = TRUE)
x.group <- fmri.designG(hrf, subj = subj, runs = runs)
# View(x.group)
```

---

fmri.detrend *Detrend fMRI time series*

Description

Detrend fMRI dataset with a polynomial of given degree

Usage

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

Arguments

`data`	fMRI dataset of class "`fmridata`"
`degree`	Degree of the polynomial used to detrend the data. defaults to 1 (linear trends).
`nuisance`	Matrix of additional nuisance parameters to regress against.
`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

Polzehl, J. and Tabelow, K. (2007). *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17.

See Also

`fmri.lm`

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"
data <- fmri.detrend(data,2)
```

---

fmri.lm *Linear Model for fMRI data*

Description

Estimate the parameters and variances in a linear model.

Usage

```
fmri.lm(ds, z, mask = NULL,
        actype = c("smooth", "noac", "ac", "accalc"),
        contrast = c(1), verbose = FALSE)
```

Arguments

`ds`	Data object of class "fmridata"
`z`	Design matrix specifying the expected BOLD response(s) and additional components for trend and other effects. This can either be a matrix (in case that no slice timing is required at this stage) or an 3D - array with 3rd dimension corresponding to the slice number. It can be interpreted as stacked array of of design matrices for the individual slices.
`mask`	Array of dimensionality of the data describing a (brain) mask the computation should be restricted to. The default is the mask given with the data.
`actype`	String describing the type of handling autocorrelation of time series. One of "smooth", "nonac", "ac", "accalc".
`contrast`	Contrast vector for the covariates.
`verbose`	Verbose mode, default is `FALSE`.

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-coefficients are corrected for bias. If `actype=="smooth"` the estimated AR-coefficients are spatially smoothed. If `actype %in% c("ac", "smooth")` the linear model is pre-whitened using the estimated (and possibly smoothed) AR-coefficients. Parameter and variance estimates are then obtained from the pre-whitened data. The argument `keep` describes the amount of data which is returned. The estimated effects

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

and their estimated variances are returned as well as the residuals and temporal autocorrelation. cbeta then contains the corresponding parameter estimates and thus is a vector of corresponding length in each voxel.

If z is an 3-dimensional array the third component is assumed to code the design matrix information for the corresponding slice, i.e. design matrices to differ with respect to slice timing effects. Note that if motion correction needs to be performed in preprocessing slice time correction may be better carried out on the data before image registration using, e.g., function slicetiming.

If warning "Local smoothness characterized by large bandwidth" occurs, check scorr elements. If correlation drops with lag towards zero, data has been pre-smoothed. Adaptive smoothing the SPM 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
residuals	raw (integer size 2) vector containing residuals of the estimated linear model up to scale factor resscale.
resscale	resscale*extractData(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 stimuli given by vvector
mask	head mask.
df	Degrees of freedom for t-statistics.
hrf	expected BOLD response for contrast

Author(s)

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

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

See Also

fmri.design, fmri.stimulus

Examples

```
  ## Not run:
  # Example 1
  data <- list(ttt=writeBin(rnorm(32*32*32*107), raw(), 4),
               mask=array(TRUE, 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, verbose=TRUE)
  plot(extractData(data)[16, 16, 16,])
 lines(extractData(data)[16, 16, 16, ] - extractData(model, "residuals")[16, 16, 16, ], col=2)

## End(Not run)
```

`fmri.lmePar` *Linear Mixed-effects Model for fMRI data*

Description

Group maps are directly estimated from the BOLD time series data of all subjects using `lme` from R package **nlme** to fit a Linear Mixed-effects Model with temporally correlated and heteroscedastic within-subject errors. Voxel-wise regression analysis is accelerated by optional parallel processing using R package **parallel**.

Usage

```
fmri.lmePar(bold, z, fixed = NULL, random = NULL, mask = NULL,
            ac = 0.3, vtype = "individual", cluster = 2,
            wghts = c(1, 1, 1))
```

Arguments

`bold` a large 4D-Array with the aggregated fMRI data of all subjects that were previously registered to a common brain atlas. Be careful with the assembly of this array, the order of the data sets has to be compatible with the design matrix: `"z"`. If not the whole brain but a region is analyzed, vectors with region-indices can be preserved by adding as attributes (e.g. `attr(bold, "xind") <- xind`).

`z` a design matrix for a multi-subject and/or multi-session fMRI-study of class `"data.frame"` specifying the expected BOLD response(s) and additional components for trend and other effects. Typically a `fmri.designG` object. This data frame contains all variables named in the model. There are some indispensable variables: *"group"*, *"subj"*, *"session"* and *"run"*, which define the different strata. That information will be used for setting up the residual variance structure.

`fixed` optionally, a one-sided linear formula describing the fixed-effects part of the model. Default settings are: `fixed <- ~ 0 + hrf + session + drift1:session + drift2:session` in case of one detected group, and the same but `"hrf"` replaced with `"hrf:group"` if two group levels in `z` are found. Since an intercept would be a linear combination of the session factor-variable modeling session-specific intercepts, it is excluded.

`random` optionally, a one-sided formula of the form `~ x1 + ... + xn | g1/.../gm`, with `~ x1 + ... + xn` specifying the model for the random effects and `g1/.../gm` the grouping structure.

Default is always the basic model without covariates, i.e.
`random <- ~ 0 + hrf|subj` if no repeated measures in `z` are found (`nlevels(z$run)==1`),
`random <- ~ 0 + hrf|subj/session` if repeated measures and
`random <- ~ 0 + hrf|session` if repeated measures but one subject only.
In case of two independent groups:
`random <- list(subj = pdDiag(~ 0 + hrf:group))` is used.

`mask` if available, a logical 3D-Array of dimensionality of the data (without time component) describing a brain mask. The computation is restricted to the selected voxels.

`ac` if available, a numeric 3D-Array of dimensionality of the data (without time component) with spatially smoothed autocorrelation parameters should be used in the AR(1) models fitted in each voxel, e.g. locally estimated and smoothed AR(1)-coefficients from `fmri.lm` applied to the first subject. Alternatively, a global approach with uniform value can be used. In this case enter a number between 0 and 1. Default is `0.3` applied to all voxels.

`vtype` a character string choosing the residual variance model. If `"equal"`, homoscedastic variance across subjects is assumed setting weights argument in function `lme` to zero, whereas `"individual"` allows different within-subject variances. Default method is `"individual"` that means subject-specific error variances using formula: `weights <- varIdent(form =~ 1|subj)`.

`cluster` number of threads for parallel processing, which is limited to available multicore CPUs. If you do not know your CPUs, try: `detectCores()` from **parallel** package. Presets are 2 threads. `cluster = 1` does not use **parallel** package.

`wghts`	a vector of length 3 specifying ratio of voxel dimensions. Isotropic voxels (e.g. MNI-space) are set as default.

Details

`fmri.lmePar()` fits the configured Linear Mixed-effects Model separately at each voxel and extracts estimated BOLD contrasts, corresponding squared standard errors and degrees of freedom as well as the residuals from resulting `lme` objects to produce a statistical parametric map (SPM) for the group(s). Voxel-by-voxel analysis is performed by either the function `apply` or `parApply` from **parallel** package, which walks through the `bold` array.

If one group is analyzed, from each fitted model the first fixed-effects coefficient and corresponding parameters are stored in results object. This should be the first specified predictor in the fixed-effects part of the model (verify the attribute of `"df"` in returned object). However, in two-sample case this principle does not work. The order changes, estimated session-specific intercepts now comes first and the number of these coefficients is not fixed. Therefore in current version it has explicitly been looked for the coefficient names: *"hrf:group1"* and *"hrf:group2"*. Available functions within the **nlme** package to extract estimated values from `lme` objects do not operate at contrast matrices.

Spatial correlation among voxels, e.g. through the activation of nearby voxels, is ignored at this stage, but corrects for it, when random field theory define a threshold for significant activation at inference stage.

It is recommended to check your model syntax and residuals choosing some distinct voxels before running the model in loop (see Example, step 1); especially for more advanced designs! Error handling default is to stop if one of the threads produces an error. When this occurs, the output will be lost from any voxel, where the model has fitted successfully.

Value

An object of class `"fmrispm"` and `"fmridata"`, basically a list with components:

`cbeta`, `cbeta2`	estimated BOLD contrast parameters separated for the groups 1 and 2
`var`, `var2`	estimated variance of the contrast parameters separated for the groups 1 and 2
`mask`	brain mask
`res`, `res2`	raw (integer size 2) vector containing residuals of the estimated Linear Mixed-effects Model up to scale factor `resscale` separated for the groups 1 and 2
`resscale`, `resscale2`	`resscale*extractData(object,"residuals")` are the residuals of group 1 and group 2 respectively.
`arfactor`	autocorrelation parameters used in AR(1)-model
`rxyz`, `rxyz2`	array of smoothness from estimated correlation for each voxel in resel space separated for the groups 1 and 2 (for analysis without smoothing)
`scorr`, `scorr2`	array of spatial correlations with maximal lags 5, 5, 3 in x, y and z-direction separated for the groups 1 and 2
`bw`, `bw2`	vector of bandwidths (in FWHM) corresponding to the spatial correlation within the data separated for the groups 1 and 2
`weights`	ratio of voxel dimensions

`dim`, `dim2`	dimension of the data cube and residuals separated for the groups 1 and 2
`df`, `df2`	degrees of freedom for t-statistics reported in `lme` objects for the extracted regression coefficients separated for the groups 1 and 2. The name of the coefficient belonging to this df-value appears as attribute.
`subjects`	number of subjects in the study
`subj.runs`	number of repeated measures within subjects
`sessions`	number of total sessions that were analyzed
`groups`	number of groups in the study
`fixedModel`	fixed-effects model
`randomModel`	random-effects model
`VarModel`	assumption about the subject error variances
`cluster`	number of threads run in parallel
`attr(*, "design")`	design matrix for the multi-subject fMRI-study
`attr(*, "approach")`	one-stage estimation method

Note

Maybe the computing power is insufficient to carry out a whole brain analysis. You have two opportunities: either select and analyze a certain brain area or switch to a two-stage model.

Current Limitations
The function cannot handle experimental designs with:

- more than two independent groups
- more than one stimulus (task)
- paired samples with varying tasks
- user defined contrasts

Author(s)

Sibylle Dames

References

Pinheiro J. and Bates D. (2000). Mixed-Effects Models in S and S-Plus. Springer.

Pinheiro J., Bates D., DebRoy S., Sarkar D. and the R Core team (2014). *nlme: Linear and Nonlinear Mixed Effects Models* R package version 3.1-117.

See Also

`lme`, `fmri.designG`, `fmri.design`, `fmri.stimulus`, `fmri.metaPar`

Examples

```
## Not run: ## Generate some fMRI data sets: noise + stimulus
dx <- dy <- dz <- 32
dt <- 107
hrf <- fmri.stimulus(dt, c(18, 48, 78), 15, 2)
stim <- matrix(hrf, nrow= dx*dy*dz, ncol=dt, byrow=TRUE)
mask <- array(FALSE, c(dx, dy, dz))
mask[12:22,12:22,12:22] <- TRUE

ds1 <- list(ttt=writeBin(1.0*rnorm(dx*dy*dz*dt) + as.vector(5*stim),
            raw(), 4), mask=mask, dim=c(dx, dy, dz, dt))
ds2 <- list(ttt=writeBin(1.7*rnorm(dx*dy*dz*dt) + as.vector(3*stim),
            raw(), 4), mask=mask, dim=c(dx, dy, dz, dt))
ds3 <- list(ttt=writeBin(0.8*rnorm(dx*dy*dz*dt) + as.vector(1*stim),
            raw(), 4), mask=mask, dim=c(dx, dy, dz, dt))
ds4 <- list(ttt=writeBin(1.2*rnorm(dx*dy*dz*dt) + as.vector(2*stim),
            raw(), 4), mask=mask, dim=c(dx, dy, dz, dt))
class(ds1) <- class(ds2) <- class(ds3) <- class(ds4) <- "fmridata"

## Construct a design matrix for a multi-subject study
subj <- 4
runs <- 1
z <-fmri.designG(hrf, subj = subj, runs = runs)

## Assembly of the aggregated BOLD-Array
Bold <- array(0, dim = c(dx,dy,dz,subj*runs*dt))
Bold[1:dx,1:dy,1:dz,1:(dt*1)] <- extractData(ds1)
Bold[1:dx,1:dy,1:dz,(dt*1+1):(dt*2)] <- extractData(ds2)
Bold[1:dx,1:dy,1:dz,(dt*2+1):(dt*3)] <- extractData(ds3)
Bold[1:dx,1:dy,1:dz,(dt*3+1):(dt*4)] <- extractData(ds4)

## Step 1: Check the model
y <- Bold[16, 16, 16, ] # choose one voxel
M1.1 <-  lme(fixed = y ~ 0 + hrf + session + drift1:session + drift2:session,
            random = ~ 0 + hrf|subj,
            correlation = corAR1(value = 0.3, form = ~ 1|subj/session, fixed=TRUE),
            weights = varIdent(form =~ 1|subj),
            method ="REML",
            control = lmeControl(rel.tol=1e-6, returnObject = TRUE),
            data = z)
summary(M1.1)

# Residual plots
plot(M1.1, resid(.,type = "response") ~ scan|subj)
qqnorm(M1.1, ~resid(.,type = "normalized")|subj, abline = c(0,1))

# Testing the assumption of homoscedasticity
M1.2 <- update(M1.1, weights = NULL, data = z)
anova(M1.2, M1.1)

# Model fit: observed and fitted values
fitted.values <- fitted(M1.1)
```

```
plot(y[1:dt], type="l", main = "Subject 1", xlab = "scan",
     ylab = "BOLD-signal", ylim = c(-5,5))
lines(fitted.values[names(fitted.values)==1],lty=1,lwd=2)

plot(y[(dt+1):(2*dt)], type="l", main = "Subject 2", xlab = "scan",
     ylab = "BOLD-signal", ylim = c(-5,5))
lines(fitted.values[names(fitted.values)==2],lty=1,lwd=2)

plot(y[(2*dt+1):(3*dt)], type="l", main = "Subject 3", xlab = "scan",
     ylab = "BOLD-signal", ylim = c(-5,5))
lines(fitted.values[names(fitted.values)==3],lty=1,lwd=2)

plot(y[(3*dt+1):(4*dt)], type="l", main = "Subject 4", xlab = "scan",
     ylab = "BOLD-signal", ylim = c(-5,5))
lines(fitted.values[names(fitted.values)==4],lty=1,lwd=2)

## Step 2: Estimate a group map
## without parallelizing
spm.group1a <- fmri.lmePar(Bold, z, mask = mask, cluster = 1)
# same with 4 parallel threads
spm.group1b <- fmri.lmePar(Bold, z, mask = mask, cluster = 4)
## Example for two independent groups
group <- c(1,1,4,4)
z2 <- fmri.designG(hrf, subj = subj, runs = runs, group = group)
spm.group2 <- fmri.lmePar(Bold, z2, mask = mask, cluster = 4)
## End(Not run)
```

`fmri.metaPar` *Linear Mixed-effects Meta-Analysis model for fMRI data*

Description

Group maps are estimated from BOLD effect estimates and their variances previously determined for each subject. The function `rma.uni` from R package **metafor** is used to fit mixed-effects meta-analytic models at group level. Voxel-wise regression analysis is accelerated by optional parallel processing using R package **parallel**.

Usage

```
fmri.metaPar(Cbold, Vbold, XG = NULL, model = NULL, method = "REML",
             weighted = TRUE, knha = FALSE, mask = NULL, cluster = 2,
             wghts = c(1, 1, 1))
```

Arguments

`Cbold` a 4D-Array with the aggregated individual BOLD contrast estimates in standard space, e.g. all `cbeta` maps obtained from single-session analysis with `fmri.lm` may put together. Dimensions 1 to 3 define the voxel space, dimension 4 indicates a subject. If not the whole brain but a region is analyzed, vectors with region-indices can be preserved by adding as attributes (e.g. `attr(Cbold, "xind") <- xind`).

`Vbold` a 4D-Array with the aggregated variance estimates for the contrast parameters in `Cbold`, e.g. all `var` maps obtained from single-session analysis with `fmri.lm` may put together. Dimensions 1 to 3 define the voxel space, dimension 4 indicates a subject.

`XG` optionally, a group-level design matrix of class `"data.frame"` to include one or more moderators in the model. By default, an intercept is added to the model.

`model` optionally, a one-sided formula of the form: `model <- ~ mod1 + mod2 + mod3` describing a model with moderator variables. Adding `"-1"` removes the intercept term.

`method` a character string specifying whether a fixed- (method = "FE") or a random/mixed-effects model (method = "REML", default) should be fitted. Further estimators for random/mixed-effects models are available, see documentation of `rma.uni` function for more details.

`weighted` logical indicating whether weighted (`weighted = TRUE`, default) or unweighted estimation should be used to fit the model.

`knha` logical specifying whether the method by Knapp and Hartung (2003) should be used for adjusting standard errors of the estimated coefficients (default is FALSE). The Knapp and Hartung adjustment is only meant to be used in the context of random- or mixed-effects models.

`mask` if available, a logical 3D-Array of dimensionality of the data (without 4th subject component) describing a brain mask. The computation is restricted to the selected voxels.

`cluster` number of threads for parallel processing, which is limited to available multi-core CPUs. If you do not know your CPUs, try: `detectCores()` from **parallel** package. Presets are 2 threads. `cluster = 1` does not use **parallel** package.

`wghts` a vector of length 3 specifying ratio of voxel dimensions. Isotropic voxels (e.g. MNI-space) are set as default.

Details

`fmri.metaPar()` fits the configured linear mixed-effects meta-analytic (MEMA) model separately at each voxel and extracts the first regression coefficient (usually the overall group mean), corresponding squared standard errors and degrees of freedom as well as the residuals from resulting `rma.uni` objects, to obtain a statistical parametric map (SPM) for the group. Voxel-by-voxel analysis is performed by either the function `apply` or `parApply` from **parallel** package, which walks through the `Cbold` array.

This two-stage approach reduces the computational burden of fitting a full linear mixed-effects (LME) model, `fmri.lmePar` would do. It assumes first level design is same across subjects and normally distributed not necessarily homogeneous within-subject errors. Warping to standard space has been done before first-stage analyses are carried out. Either no masking or a uniform brain mask should be applied at individual subject analysis level, to avoid loss of information at group level along the edges.

At the second stage, observed individual BOLD effects from each study are combined in a meta-analytic model. There is the opportunity of weighting the fMRI studies by the precision of their respective effect estimate to take account of first level residual heterogeneity (`weighted = TRUE`). This is how to deal with intra-subject variability. The REML estimate of cross-subject variability (tau-squared) assumes that each of these observations is drawn independently from the same Gaussian distribution. Since correlation structures cannot be modeled, multi-subject fMRI studies with repeated measures cannot be analyzed in this way.

Spatial correlation among voxels, e.g. through the activation of nearby voxels, is ignored at this stage, but corrects for it, when random field theory define a threshold for significant activation at inference stage.

It is recommended to check your model syntax and residuals choosing some distinct voxels before running the model in loop (see Example). Error handling default is to stop if one of the threads produces an error. When this occurs, the output will be lost from any voxel, where the model has fitted successfully.

Value

An object of class `"fmrispm"` and `"fmridata"`, basically a list with components:

`beta`	estimated regression coefficients
`se`	estimated standard errors of the coefficients
`cbeta`	estimated BOLD contrast parameters for the group. Always the first regression coefficient is taken.
`var`	estimated variance of the BOLD contrast parameters
`mask`	brain mask
`residuals`	raw (integer size 2) vector containing residuals of the estimated linear mixed-effects meta-analytic model up to scale factor `resscale`
`resscale`	`resscale*extractData(object,"residuals")` are the residuals.
`tau2`	estimated amount of (residual) heterogeneity. Always 0 when `method = "FE"`.
`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
`dim`	dimension of the data cube and residuals
`df`	degrees of freedom for t-statistics, df = (n-p-1)
`sessions`	number of observations entering the meta-analytic model, n
`coef`	number of coefficients in the meta-analytic model (including the intercept, p+1)
`method`	estimator used to fit the meta-analytic model. In case of "FE", it is weighted or unweighted least squares.
`weighted`	estimation with inverse-variance weights
`knha`	Knapp and Hartung adjustment
`model`	meta-analytic regression model

cluster number of threads running in parallel

attr(*, "design")
group-level design matrix

attr(*, "approach")
two-stage estimation method

Note

Meta analyses tend to be less powerful for neuroimaging studies, because they only have as many degrees of freedom as number of subjects. If the number of subjects is very small, then it may be impossible to estimate the between-subject variance (tau-squared) with any precision. In this case the fixed effect model may be the only viable option. However, there is also the possibility of using a one-stage model, that includes the full time series data from all subjects and simultaneously estimates subject and group levels parameters (see fmri.lmePar). Although this approach is much more computer intensive, it has the advantage of higher degrees of freedom (> 100) at the end.

Current Limitations

The function cannot handle:

- experimental designs with a within-subject (repeated measures) factor
- paired samples with varying tasks, unless the contrast of the two conditions is used as input

Author(s)

Sibylle Dames

References

Chen G., Saad Z.S., Nath A.R., Beauchamp M.S., Cox R.W. (2012). FMRI group analysis combining effect estimates and their variances. NeuroImage, 60: 747-765.

Knapp G. and Hartung J. (2003). Improved tests for a random effects meta-regression with a single covariate. Statistics in Medicine, 22: 2693-2710.

Viechtbauer W. (2005). Bias and efficiency of meta-analytic variance estimators in the random-effects model. Journal of Educational and Behavioral Statistics, 30: 261-293.

Viechtbauer W. (2010). Conducting meta-analyses in R with the metafor package. Journal of Statistical Software, 36(3): 1-48

Viechtbauer W. (2015). *metafor: Meta-Analysis Package for R* R package version 1.9-7.

See Also

rma.uni, fmri.lm, fmri.lmePar

Examples

```
## Not run: ## Generate some fMRI data sets: noise + stimulus
dx <- dy <- dz <- 32
dt <- 107
hrf <- fmri.stimulus(dt, c(18, 48, 78), 15, 2)
stim <- matrix(hrf, nrow= dx*dy*dz, ncol=dt, byrow=TRUE)
mask <- array(FALSE, c(dx, dy, dz))
```

```
mask[12:22,12:22,12:22] <- TRUE

ds1 <- list(ttt=writeBin(1.0*rnorm(dx*dy*dz*dt) + as.vector(5*stim),
           raw(), 4), mask = mask, dim = c(dx, dy, dz, dt))
ds2 <- list(ttt=writeBin(1.7*rnorm(dx*dy*dz*dt) + as.vector(3*stim),
           raw(), 4), mask = mask, dim = c(dx, dy, dz, dt))
ds3 <- list(ttt=writeBin(0.8*rnorm(dx*dy*dz*dt) + as.vector(1*stim),
           raw(), 4), mask = mask, dim = c(dx, dy, dz, dt))
ds4 <- list(ttt=writeBin(1.2*rnorm(dx*dy*dz*dt) + as.vector(2*stim),
           raw(), 4), mask = mask, dim = c(dx, dy, dz, dt))
class(ds1) <- class(ds2) <- class(ds3) <- class(ds4) <- "fmridata"

## Stage 1: single-session regression analysis
x <- fmri.design(hrf, order=2)
spm.sub01 <- fmri.lm(ds1, x, mask, actype = "smooth", verbose = TRUE)
spm.sub02 <- fmri.lm(ds2, x, mask, actype = "smooth", verbose = TRUE)
spm.sub03 <- fmri.lm(ds3, x, mask, actype = "smooth", verbose = TRUE)
spm.sub04 <- fmri.lm(ds4, x, mask, actype = "smooth", verbose = TRUE)

## Store observed individual BOLD effects and their variance estimates
subj <- 4
Cbold <- array(0, dim = c(dx, dy, dz, subj))
Cbold[,,,1] <- spm.sub01$cbeta
Cbold[,,,2] <- spm.sub02$cbeta
Cbold[,,,3] <- spm.sub03$cbeta
Cbold[,,,4] <- spm.sub04$cbeta

Vbold <- array(0, dim = c(dx, dy, dz, subj))
Vbold[,,,1] <- spm.sub01$var
Vbold[,,,2] <- spm.sub02$var
Vbold[,,,3] <- spm.sub03$var
Vbold[,,,4] <- spm.sub04$var

## Stage 2: Random-effects meta-regression analysis
## a) Check your model
library(metafor)
M1.1 <- rma.uni(Cbold[16,16,16, ],
                Vbold[16,16,16, ],
                method = "REML",
                weighted = TRUE,
                knha = TRUE,
                verbose = TRUE,
                control = list(stepadj=0.5, maxiter=2000, threshold=0.001))

# Control list contains convergence parameters later used
# at whole data cube. Values were adjusted to fMRI data.

summary(M1.1)
forest(M1.1)
qqnorm(M1.1)

## b) Estimate a group map
## without parallelizing
```

```
spm.group1a <- fmri.metaPar(Cbold, Vbold, knha = TRUE,
                            mask = mask, cluster = 1)
## same with 4 parallel threads
spm.group1b <- fmri.metaPar(Cbold, Vbold, knha = TRUE,
                            mask = mask, cluster = 4)
## End(Not run)
```

---

fmri.pvalue *P-values*

Description

Determine p-values.

Usage

```
fmri.pvalue(spm, mode="basic", na.rm=FALSE, minimum.signal = 0, alpha= 0.05)
```

Arguments

spm	fmrispm object
mode	type of pvalue definition
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. Otherwise 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.
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. mode="voxelwise" refers to voxelwise tests while mode="Bonferroni" adjusts the significance level for multiple testing. An alternative is mode="FDR" specifying signal detection by False Discovery Rate (FDR) with proportion of false positives level specified by alpha. The other choices apply results on excursion sets of random fields (Worsley 1994, Adler 2003) for smoothed SPM's. "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.

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)
`alpha`	maximal pvalue as scale information
`thresh`	actual threshold used

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

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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

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.lm`, `fmri.smooth`, `plot.fmridata`, `fmri.cluster`, `fmri.searchlight`

Examples

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

`fmri.searchlight` *Searchlight signal detection*

Description

Detection of activated regions using searchlights.

Usage

```
fmri.searchlight(spm, alpha = 0.05, radius, minimum.signal = 0,
   kind = c("abs", "squared"))
```

Arguments

`spm`	`fmrispm` object
`alpha`	multiple test (over volume) adjusted significance level.
`radius`	radius of searchlight. Value needs to be larger or equal than 1.
`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.
`kind`	Kind of statistics used for aggregation over search light region. `"abs"` specifies averaging of absolute voxelwise t-statistics while `"squared"` corresponds to averaging of squares of these statistics.

Details

The function computes mean statistics (depending on `kind`) over a searchlight region of radius `radius`. Approximate voxelwise p-values are determined with respect an empirical (simulated) distribution of the searchlight statistics under the null hypothesis a central t-distributed spm. Thresholding used `FDR` with rate `alpha`.

Value

Object with class attributes "fmripvalue" and "fmridata"

`pvalue`	voxelwise p-value if exceeding FDR-critical value, 1 otherwise.
`weights`	voxelsize ratio
`dim`	data dimension
`hrf`	expected BOLD response for contrast (single stimulus only)

Author(s)

Joerg Polzehl <`polzehl@wias-berlin.de`>

References

Kriegeskorte, N.; Goebel, R. & Bandettini, P. (2006) *Information-based functional brain mapping*, PNAS 103:3863-3868.

See Also

`fmri.lm`, `fmri.pvalue`, `fmri.cluster`

Examples

```
## Not run: fmri.searchlight(fmrispmobj)
```

`fmri.sgroupICA` *Spatial group ICA for fmri*

Description

Combine ICA results from multiple runs or multiple subjects in group fMRI studies

Usage

```
fmri.sgroupICA(icaobjlist, thresh = 0.75, minsize=2)
```

Arguments

`icaobjlist`	List of results obtained by function `fmri.sICA` for a series of fmri data sets (multiple runs or multiple subjects).
`thresh`	threshold for cluster aggregation. Needs to be in (0,1).
`minsize`	Minimal size of cluster to consider in IC aggregation. Needs to be larger than 1.

Details

The fMRI time series need to be preprocessed and registered before thr ICA decomposition is performed.

The function employs a hierarchical clustering algorithm (complete linkage) on the combined set of spatial independent components obtained from the individual time series. A distance matrix is obtained from correlations of the independent component images. Aggregation of two components from the same fmri series is prevented in the algorithm.

Value

An object of class ”`fmrigroupICA`” with components

`icacomp`	Mean IC’s over cluster members for cluster of size larger or equal `minsize`
`size`	Size of selected clusters
`cl`	Number of selected clusters

`cluster`	Cluster membership corresponding to `thresh`.
`height`	Distance value at which the cluster was created. Elements correspond to elements of cluster.
`hdm`	Object returned by function `hclust`.

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References

F. Esposito et al (2005) Independent component analysis of fMRI group studies by self-organizing clustering, Neuroimage, pp. 193-205.

See Also

`fmri.sICA`, `plot.fmrigroupICA`, `hclust`

`fmri.sICA` *Spacial ICA for fmri data*

Description

Uses fastICA to perform spatial ICA on fMRI data.

Usage

```
fmri.sICA(data, mask=NULL, ncomp=20,
  alg.typ=c("parallel","deflation"), fun=c("logcosh","exp"),
  alpha=1, detrend=TRUE, degree=2, nuisance= NULL, ssmooth=TRUE,
  tsmooth=TRUE, bwt=4, bws=8, unit=c("FWHM","SD"))
```

Arguments

`data`	fMRI dataset of class "`fmridata`"
`mask`	Brain mask, if `NULL` then `data$mask` is used.
`ncomp`	Number of ICA components to compute.
`alg.typ`	Alg. to be used in `fastICA`.
`fun`	Test functions to be used in `fastICA`.
`alpha`	Scale parameter in test functions, see `fastICA`.
`detrend`	Trend removal (polynomial)
`degree`	degree of polynomial trend
`nuisance`	Matrix of additional nuisance parameters to regress against.
`ssmooth`	Should spatial smoothing be used for variance reduction

tsmooth	Should temporal smoothing be be applied
bws	Bandwidth for spatial Gaussian kernel
bwt	Bandwidth for temporal Gaussian kernel
unit	Unit of bandwidth, either standard deviation (SD) of Full Width Half Maximum (FWHM).

Details

If specified polynomial trends and effects due to nuisance parameters, e.g., motion parameters, are removed. If smooth==TRUE the resulting residual series is spatially smoothed using a Gaussian kernel with specified bandwidth. ICA components are the estimated using fastICA based on data within brain mask. The components of the result are related as XKW=scomp[mask,] and X=scomp[mask,]*A.

Value

object of class "fmriICA" list with components

scomp	4D array with ICA component images. Last index varies over components.
X	pre-processed data matrix
K	pre-processed data matrix
W	estimated un-mixing matrix
A	estimated mixing matrix
mask	Brain mask
pixdim	voxelsize
TR	Repetition Time (TR)

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

See Also

plot.fmriICA,ICAfingerprint, fastICA

---

fmri.smooth *Smoothing Statistical Parametric Maps*

Description

Perform the adaptive weights smoothing procedure

Usage

```
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 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.
- `...` — 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. Useful exceptions can be used for `adaptation="segment"`: Specifically `alpha` (default 0.05) defines the significance level for the signal detection. It can be chosen 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 alternative, 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. `lkern` 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 `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", 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
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

Polzehl, J., Voss, H.U., and Tabelow, K. (2010). *Structural Adaptive Segmentation for Statistical Parametric Mapping*, NeuroImage, 52:515-523.

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

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

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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), TR = 2,
              times = FALSE, sliceorder = NULL,
              type = c("canonical", "gamma", "boxcar", "user"),
              par = NULL, scale = 10, hrf = NULL, verbose = FALSE)
```

Arguments

scans	number of scans
onsets	vector of onset times (in scans)
durations	vector of duration of ON stimulus in scans (if `times==FALSE`)) or seconds (if `times==TRUE`))
TR	time between scans in seconds (TR)
times	logical. If TRUE onsets and durations are given in units of time not number of scans. Defaults to FALSE.
sliceorder	order of slice acquisition. If provided separate expected bold responses are calculated for the slices taking slice acquisition times into account. Default: no slice timing.
type	One of `"canonical"`, `"gamma"`, `"boxcar"`, `"user"`
par	Possible parameters to the HRF.
scale	Temporal undersampling factor
hrf	If `type` is `"user"` this should be a function evaluating the hemodynamic response function
verbose	Report more if `TRUE`

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.

If `sliceorder` provides an ordering of slice acquisitions a matrix of expected Bold responses with columns corresponding to the slice number is computed.

For `type` is `"canonical"` 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 a1, a2, b1, b2, cc of this function can be changed through the argument par in this order.

Other choices are a simple gamma function

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

or the "boxcar" stimulus, or a user defined function hrf.

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

If !is.null(times) durations are specified in seconds.

Value

Vector with dimension c(scans, 1) or a matrix with dimension c(scans, number of slices).

Author(s)

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

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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")
```

`gen_fmridata` *Generate fmridata example*

Description

Generate fmridata example

Usage

```
gen_fmridata(signal = 1.5, noise = 20, arfactor = 0.3)
```

Arguments

`signal`	Level of signal in the data
`noise`	Level of noise in the data
`arfactor`	Autoregressive factor

Value

Object of class `fmridata`

Examples

```
gen_fmridata()
```

`getSearchlightPattern` *Extract searchlight pattern from a SPM*

Description

For a provided spm object and a mask of voxel the function extracts the values of the parameter estimates within the searchlight region and for all voxel in the mask.

Usage

```
getSearchlightPattern(spm, voxel, radius)
```

Arguments

`spm`	an object of class 'fmrispm'
`voxel`	a mask (logical) with dimensionality compatible to the spm
`radius`	radius of the searchlight

Value

an array of dimension c(nb, nsl, nvox) with nb the number of estimated parameters in spm$beta, nsl the number of voxel in the searchlight and nvox the number of voxel in the mask provided as second argument

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

See Also

fmri.searchlight, fmri.lm~~~

---

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) ...
```

ICAfingerprint *IC fingerprinting*

Description

Implements ICA fingerprinting mainly following De Martino et.al., Neuroimage 2007.

Usage

```
ICAfingerprint(icaobj, nbin = 256, plot = FALSE)
```

Arguments

icaobj	object returned by function fmri.sICA.
nbin	number of bins for entropy estimation
plot	provide results as star plots.

Details

For some characteristics normalization of values differs from De Martino et. al.. Frequency bands are obtained from periodogram estimated instead of using Welch's method.

Value

object of class "fmriICA" list with components

scomp	4D array with ICA component images. Last index varies over components.
X	pre-processed data matrix
K	pre-processed data matrix
W	estimated un-mixing matrix
A	estimated mixing matrix
mask	Brain mask
pixdim	voxelsize
TR	Repetition Time (TR)
fingerprint	matrix of IC characteristics. Columns correspond to IC's .

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

References

De Martino et. al., Classification of fMRI independent components using IC-fingerprints and support vector machine classifiers, Neuroimage 34 (2007) 177-194.

See Also

fmri.sICA, plot.fmriICA, fastICA

`niftiImage2fmri` *Create fmridata object from niftiImage*

Description

Transforms a niftiImage (created by readNifti from package RNiftyReg) into an object with class fmridata

Usage

```
niftiImage2fmri(niftiobj, level = 0.75, mask=NULL, setmask = TRUE, indx = NULL,
   indy = NULL, indz = NULL, avoidnegs = FALSE)
```

Arguments

`niftiobj`	an object of class niftiImage
`level`	quantile used in mask definition
`mask`	array or nifti-object containing the mask. If set this replaces the mask defined by argument level.
`setmask`	if `TRUE` create a brain mask
`indx`	index vector for subcube definition
`indy`	index vector for subcube definition
`indz`	index vector for subcube definition
`avoidnegs`	if `TRUE` change the mean to avoid negative image intensities

Details

This function can be used in connection with readNifti from package RNiftyReg to read large fMRI series from nifti files. The resulting fmridata-object stores the image data as 2 byte integer in raw format, in contrast for the 4 byte real used with other functions.

Value

an object of class fmridata

Author(s)

Joerg Polzehl `<polzehl@wias-berlin.de>`

See Also

`read.AFNI`, `read.DICOM`, `read.ANALYZE`, `read.NIFTI` `plot.fmridata` *I/O functions*

Description

Visualize fMRI data and (intermediate) results.

Usage

```
## S3 method for class 'fmridata'
plot(x, anatomic = NULL, maxpvalue = 0.05,
              spm = TRUE, pos = c(-1, -1, -1), type = "slice",
              slice =  1, view = "axial" ,zlim.u =
              NULL, zlim.o = NULL,col.o = heat.colors(256), col.u =
              grey(0:255/255), cutOff = c(0, 1), ...)
## S3 method for class 'fmrisegment'
plot(x, anatomic = NULL,
              slice =  1, view = c( "axial", "coronal", "sagittal") ,zlim.u =
             NULL, zlim.o = NULL,col.o = c( rainbow( 64, start = 2/6, end = 4/6),
              rainbow( 64, start = 0, end = 1/6)),
              col.u = grey(0:127/127), verbose = FALSE, ...)
```

Arguments

`x`	object of class "fmrisegment", "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 slicewise 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

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

See Also

`fmri.pvalue`

Examples

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

`plot.fmriICA` *Diagnostics plots for objects of class "`fmriICA`"*

Description

The function generates plots for inspecting independent components obtained by spatial independent component analysis.

Usage

```
## S3 method for class 'fmriICA'
plot(x, comp = 1, center = NULL, thresh = 1.5, ...)
## S3 method for class 'fmrigroupICA'
plot(x, comp = 1, center = NULL, thresh = 1.5, ...)
```

Arguments

`x`	object returned by function `fmri.sICA` or preferably function `ICAfingerprinting` in case of `plot.fmriICA` and object returned by function `fmri.sgroupICA` in case of `plot.fmrigroupICA`
`comp`	number of the independent component to inspect.
`center`	coordinates for central point to determine axial, coronal and sagittal slices for display. If NULL the central point of the image cube is selected. center needs to be within the brain mask.
`thresh`	Threshold value
`...`	currently not used

Details

The function generates diagnostic plots for the independent component specified in `comp`. It provides axial, coronal and sagittal images as determined by `center`. Values exceeding the threshold are displayed using a color scale. An IC fingerprint is given as a star plot. Additionally the time series corresponding to the spatial IC and its spectral density are plotted.

Value

nothing returned.

Author(s)

Joerg Polzehl <`polzehl@wias-berlin.de`>

References

De Martino et. al., Classification of fMRI independent components using IC-fingerprints and support vector machine classifiers, Neuroimage 34 (2007) 177-194.

See Also

fmri.sICA, ICAfingerprint, fastICA

---

plot.fmripvalue *Visualize fMRI p-value maps*

Description

Visualize objects created by function fmri.pvalue

Usage

```
## S3 method for class 'fmripvalue'
plot(x, template = NULL, mask = NULL,
      view = c("axial", "coronal", "sagittal", "orthographic"),
      slices = NULL, ncol = 1, nrow = 1, center = NULL, ...)
```

Arguments

x	object of class 'fmripvalue'
template	Anatomical image of same origin and direction as pvalue map in x$pvalue.
mask	optional brain mask
view	Either 'orthographic' or one of 'axial', 'coronal' or 'sagittal'
slices	If view != "orthographic" vector of slice numbers to use. If not provided the ncol*nrow slices with strongest signals are selected
ncol	If view != "orthographic" number of slices per row
nrow	If view != "orthographic" number of rows in display.
center	If view == "orthographic" center of orthographic view. If not provided the center is chosen to provide maximal information.
...	additional parameters (not evaluated)

Value

list with components

comp1	slices, numbers refer to spm
comp2	center, numbers refer to spm

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

See Also

fmri.pvalue, ~~~

`print.fmridata` *I/O functions*

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

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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,mask=NULL,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
`mask`	array or nifti-object containing the mask. If set this replaces the mask defined by argument level.
`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:

`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.

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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, mask=NULL, 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
`mask`	array or nifti-object containing the mask. If set this replaces the mask defined by argument level.
`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

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

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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, mask=NULL, setmask=TRUE)
```

Arguments

filename	name of the NIfTI file
level	Quantile level defining the mask
mask	array or nifti-object containing the mask. If set this replaces the mask defined by argument level.
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:

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>`

References

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

See Also

`read.ANALYZE`, `read.AFNI`

Examples

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

`setmask` *Add or replace mask in an fmridata object*

Description

The function replaces the information in the mask component of an fmridata object.

Usage

```
setmask(fmriobj, mask)
```

Arguments

`fmriobj`	object of class 'fmridata'
`mask`	object of class 'array' or 'nifti'

Details

The dimensions of both objects supplied as arguments need to be compatible.

Value

on object of class 'fmridata'.

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

See Also

oro2fmri, niftiImage2fmri, read.NIFTI, read.AFNI, read.ANALYZE

---

sincfilter *A function for sinc-interpolation*

Description

Performs sinc interpolation for a equidistant time series x to times t.

Usage

```
sincfilter(t, x, wr=8)
```

Arguments

t	vector of new time points
x	observed time series at times 1:length(x).
wr	determines truncation of series expansion

Value

a vector of interpolated values of the time series at time points given in t.

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

See Also

slicetiming

Examples

```
x <- 1:107
y <- rnorm(x)
z <- sincfilter(seq(1,107,.01),y)
plot(x, y, ylim=range(y,z))
lines(seq(1,107,.01),z,col=2)
```

slicetiming — *slicetiming for fmridata-objects*

Description

Perform slicetiming for fMRI data, ideally before preprocessing (registration). Recording times for slices are assumed to be equispaced between scans with argument `sliceorder` providing the order of slice acquisitions. Interpolation between slices is performed using a sinc filter.

Usage

```
slicetiming(fmridataobj, sliceorder = NULL)
```

Arguments

`fmridataobj`	object of class fmridata
`sliceorder`	order of lice acquisitions

Value

an object of class fmridata

Author(s)

Joerg Polzehl <polzehl@wias-berlin.de>

See Also

`fmri.stimulus`, `fmri.design`,`fmri.lm`,~~~

Examples

```
  ## Not run:
# Example 1
  data <- list(ttt=writeBin(rnorm(32*32*32*107), raw(), 4),
               mask=array(TRUE, c(32, 32, 32)), dim=c(32, 32, 32, 107))
  class(data) <- "fmridata"
  data <- slicetiming(data,sliceorder=1:32)
  ## provides data corrected for sequential slice acquisition in linear order

## End(Not run)
```

`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

```
## 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>`

References

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

See Also

read.AFNI,write.ANALYZE

Examples

```
 ## Not run: write.AFNI(tempfile(), 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(tempfile(), 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 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	[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>

References

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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=file.path(tempdir(),"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>`

References

Polzehl, J. and Tabelow, K. (2007) *fmri: A Package for Analyzing fmri Data*, R News, 7:13-17 .

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=file.path(tempdir(),"niftifile"))
```

Index

∗ **IO**
cutroi, 4
niftiImage2fmri, 36
read.AFNI, 42
read.ANALYZE, 43
read.DICOM, 45
read.NIFTI, 46
write.AFNI, 51
write.ANALYZE, 52
write.NIFTI, 53
∗ **datasets**
hvred, 34
∗ **design**
fmri.design, 7
fmri.designG, 8
fmri.stimulus, 31
∗ **hplot**
plot.fmridata, 37
plot.fmriICA, 39
plot.fmripvalue, 40
∗ **htest**
fmri.cluster, 6
fmri.pvalue, 23
fmri.searchlight, 25
∗ **iplot**
plot.fmridata, 37
∗ **manip**
sincfilter, 48
slicetiming, 49
∗ **misc**
Convert Between fmridata and oro.nifti, 3
getSearchlightPattern, 33
setmask, 47
∗ **multivariate**
fmri.sgroupICA, 26
ICAfingerprint, 34
∗ **print**
print.fmridata, 41
summary.fmridata, 50
∗ **regression**
fmri.design, 7
fmri.designG, 8
fmri.detrend, 10
fmri.lm, 11
fmri.lmePar, 13
fmri.metaPar, 18
fmri.stimulus, 31
∗ **smooth**
fmri.smooth, 28
∗ **utilities**
cutroi, 4
extractData, 5
getSearchlightPattern, 33
ICAfingerprint, 34
niftiImage2fmri, 36
plot.fmriICA, 39
read.AFNI, 42
read.ANALYZE, 43
read.DICOM, 45
read.NIFTI, 46
setmask, 47
sincfilter, 48
slicetiming, 49
write.AFNI, 51
write.ANALYZE, 52
write.NIFTI, 53

apply, *15*, *19*

condensefMRI (extractData), 5
Convert Between fmridata and oro.nifti, 3
cutroi, 4

expandfMRI (extractData), 5
extractData, 5

fastICA, *28*, *35*, *40*

fmri.cluster, 6, *24*, *26*
fmri.design, 7, *9*, *13*, *16*, *32*, *49*
fmri.designG, 8, *14*, *16*
fmri.detrend, 10
fmri.lm, *5*, *7*, *8*, *10*, 11, *14*, *18*, *19*, *21*, *24*, *26*, *32*, *34*, *49*
fmri.lmePar, *9*, 13, *19*, *21*
fmri.metaPar, *16*, 18
fmri.pvalue, *7*, 23, *26*, *38*, *40*
fmri.searchlight, *7*, *24*, 25, *34*
fmri.sgroupICA, 26
fmri.sICA, *26*, *27*, 27, *35*, *40*
fmri.smooth, *24*, 28
fmri.stimulus, *7–9*, *13*, *16*, 31, *49*
fmri2oro *(*Convert Between fmridata and oro.nifti*)*, 3

gen_fmridata, 33
getSearchlightPattern, 33

hclust, *27*
hvred, 34

ICAfingerprint, *28*, 34, *40*

lme, *13–16*

niftiImage2fmri, 36, *48*

oro2fmri, *48*
oro2fmri *(*Convert Between fmridata and oro.nifti*)*, 3

parApply, *15*, *19*
plot, *24*
plot.fmridata, *24*, 37
plot.fmrigroupICA, *27*
plot.fmrigroupICA *(*plot.fmriICA*)*, 39
plot.fmriICA, *28*, *35*, 39
plot.fmripvalue, 40
plot.fmrisegment *(*plot.fmridata*)*, 37
print.fmridata, 41, *50*

read.AFNI, *4*, *36*, 42, *44*, *46–48*, *52*
read.ANALYZE, *4*, *36*, *43*, 43, *46–48*, *53*, *54*
read.DICOM, *36*, 45
read.NIFTI, *3*, *4*, *36*, 46, *48*
rma.uni, *18*, *19*, *21*

setmask, 47
sincfilter, 48
slicetiming, *12*, *48*, 49
summary.fmridata, *41*, 50

write.AFNI, *43*, 51, *53*, *54*
write.ANALYZE, *44*, *52*, 52
write.NIFTI, 53