Package ‘fmri’

September 15, 2016

Version 1.7-2
Date 2016-09-09
Title Analysis of fMRI Experiments
Depends R (>= 2.14.0), awsMethods (>= 1.0-1), nlme
Imports parallel, metafor, methods
Suggests tcltk, tkplot, fastICA, adimpro, oro.nifti
Description Contains R-functions to perform an fMRI analysis as described in
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)
Copyright This package is Copyright (C) 2006-2016 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!
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 2016-09-15 12:30:35

R topics documented:

Convert Between fmridata and oro.nifti ........................................ 2
cutroi ............................................................................. 3
extract.data ..................................................................... 4
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, 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.
The cutroi function 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 function 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 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.
```

---

### extract.data

*Extract data or residuals from a fmridata object*

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

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

**Arguments**
- `z`: an object of class 'fmridata'
- `what`: either "data" or "residuals".

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

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

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

Usage

```r
fmri.design(stimulus, order = 2, cef = NULL, verbose = FALSE)
```

Arguments

- `stimulus`: matrix containing expected BOLD response(s) for the linear model as columns
- `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>

References

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

```r
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`).
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


See Also

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

Examples

```r
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, 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


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.gui**  
*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**

```r
## Not run: fmri.gui()
```

---

**fmri.lm**  
*Linear Model for fMRI data*

**Description**

Estimate the parameters and variances in a linear model.

**Usage**

```r
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 Designmatrix specifying the expected BOLD response(s) and additional components for trend and other effects.

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-withened using the estimated (and possibly smoothed) AR-coefficients. Parameter and variance estimates are then obtained from the pre-withened 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 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

res raw (integer size 2) vector containing residuals of the estimated linear model up to scale factor resscale.

resscale resscale*extract.data(object,"residuals") are the residuals.

dim dimension of the data cube and residuals

arfactor estimated autocorrelation parameter
**rxyz**
array of smoothness from estimated correlation for each voxel in resel space (for analysis without smoothing)

**scorr**
array of spatial correlations with maximal lags 5, 5, 3 in x,y and z-direction.

**bw**
vector of bandwidths (in FWHM) corresponding to the spatial correlation within the data.

**weights**
ratio of voxel dimensions

**vwghts**
ratio of estimated variances for the stimuli given by vvector

**mask**
head mask.

**df**
Degrees of freedom for t-statistics.

**hrf**
expected BOLD response for contrast

**Note**
The argument vvector is no longer supported.

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

**References**


**See Also**

`fmri.lm`, `fmri.stimulus`

**Examples**

```r
## Not run:
# Example 1
data <- list(ttt=writeBin(rnorm(32*32*32*107), raw(), 4),
   mask=array(1, c(32, 32, 32)), dim=c(32, 32, 32, 107))
class(data) <- "fmridata"
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
z <- fmri.design(hrf,2)
model <- fmri.lm(data, z, verbose=TRUE)
plot(extract.data(data)[16, 16, 16,])
lines(extract.data(data)[16, 16, 16,] - extract.data(model, "residuals")[16, 16, 16,], col=2)
```

## End(Not run)
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

```r
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.design` 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:

* cbetaL cbetaR  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*extract.data(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


See Also

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

Examples

```R
# 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] <- TRUE

ds1 <- list(ttt=writeBin(1.8*runif(dx*dy*dz*dt)) + as.vector(stim),
            raw(), 4), mask=mask, dim=c(dx, dy, dz, dt))
ds2 <- list(ttt=writeBin(1.7*runif(dx*dy*dz*dt)) + as.vector(stim),
            raw(), 4), mask=mask, dim=c(dx, dy, dz, dt))
ds3 <- list(ttt=writeBin(0.8*runif(dx*dy*dz*dt)) + as.vector(stim),
            raw(), 4), mask=mask, dim=c(dx, dy, dz, dt))
ds4 <- list(ttt=writeBin(1.2*runif(dx*dy*dz*dt)) + as.vector(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)] <- extract.data(ds1)
Bold[1:dx,1:dy,1:dz,(dt*1+1):(dt*2)] <- extract.data(ds2)
Bold[1:dx,1:dy,1:dz,(dt*2+1):(dt*3)] <- extract.data(ds3)
Bold[1:dx,1:dy,1:dz,(dt*3+1):(dt*4)] <- extract.data(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),
```
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`.

```r
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[(2*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[(3*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[(4*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)
```
Usage

```r
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 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.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
- `res` raw (integer size 2) vector containing residuals of the estimated linear mixed-effects meta-analytic model up to scale factor `resscale`
- `resscale` `resscale*extract.data(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 \textit{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


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))

ds1 <- list(ttt = writeBin(1.0*rrnorm(dx*dy*dz*dt) + as.vector(5*stim),
  raw(), 4), mask = mask, dim = c(dx, dy, dz, dt))
ds2 <- list(ttt = writeBin(1.7*rrnorm(dx*dy*dz*dt) + as.vector(3*stim),
  raw(), 4), mask = mask, dim = c(dx, dy, dz, dt))
ds3 <- list(ttt = writeBin(0.8*rrnorm(dx*dy*dz*dt) + as.vector(1*stim),
  raw(), 4), mask = mask, dim = c(dx, dy, dz, dt))
ds4 <- list(ttt = writeBin(1.2*rrnorm(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$dbeta
Cbold[, , , 2] <- spm.sub02$dbeta
Cbold[, , , 3] <- spm.sub03$dbeta
Cbold[, , , 4] <- spm.sub04$dbeta

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, , ],
  Vbold[, , 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)
```

---

### Description

Determine p-values.

### Usage

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

### Arguments

- **spm**
  - fmrism 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="FDA"
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
- hrf: expected BOLD response for contrast (single stimulus only)

Note

Unexpected side effects may occur if spm does not meet the requirements, especially if a parameter estimate vector of length greater than 2 through argument vvector in fmri.lm has 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` 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 (.3,1) 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. `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` (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 alternative, i.e. not the interpretation of results.

Details

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

\( h_{\text{max}} \) is the (maximal) bandwidth used in the last iteration. Choose adaptation as "none" for non adaptive smoothing. \( l_{\text{kern}} \) can be used for specifying the localization kernel. For comparison with non adaptive methods use "Gaussian" (\( h_{\text{max}} \) times the voxelsize in x-direction will give the FWHM bandwidth in mm), for better adaptation use "Plateau" or "Triangle" (default, \( h_{\text{max}} \) given in voxel). For \( l_{\text{kern}}="\text{Plateau}" \) and \( l_{\text{kern}}="\text{Triangle}" \) thresholds may be inaccurate, due to a violation of the Gaussian random field assumption under homogeneity. \( l_{\text{kern}}="\text{Plateau}" \) is expected to provide best results with adaptive smoothing.

\( sk_{\text{ern}} \) 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 \texttt{spm$mask}.

Value

object with class attributes "fmrispm" and "fmridata", or "fmrisegment" and "fmridata" for segmentation choice

- \texttt{cbeta} smoothed parameter estimate
- \texttt{var} variance of the parameter
- \texttt{hmax} maximum bandwidth used
- \texttt{rxyz} smoothness in resel space. all directions
- \texttt{rxyz0} smoothness in resel space as would be achieved by a Gaussian filter with the same bandwidth. all directions
- \texttt{scorr} array of spatial correlations with maximal lags 5, 5, 3 in x,y and z-direction.
- \texttt{bw} vector of bandwidths (in FWHM) corresponding to the spatial correlation within the data.
- \texttt{dim} dimension of the data cube and residuals
- \texttt{weights} ratio of voxel dimensions
- \texttt{vwghts} ratio of estimated variances for the stimuli given by \texttt{vvector}
- \texttt{hrf} Expected BOLD response for the specified effect

Author(s)

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


Examples

```r
## Not run: fmri.smooth(spm, hmax = 4, likern = "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), TR = 2, times = FALSE, 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 or seconds (if `!is.null(times)`)
- **TR**: time between scans in seconds (TR)
- **times**: onset times in seconds. If present onsets arguments is ignored.
- **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.

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).

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de>

References


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")
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 $v^\ast\{(k)\}$
- `normv`: If `keepv==TRUE` the norm of each $v^\ast\{(k)\}$.
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) ...

---

**ngca**  
*Non-Gaussian Component Analysis*

---

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

**Usage**

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>).

... 

Author(s)

J"org Polzehl polzehl@wias-berlin.de
plot.fmridata

References


plot.fmridata  I/O functions

Description

Visualize fMRI data and (intermediate) results.

Usage

```r
## 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", "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
plot.fmridata

\begin{itemize}
\item \texttt{col.u} color scale for anatomical underlay, if anatomic is of "fmridata" class, default grey(0:255/255)
\item \texttt{col.o} color scale for functional overlay, if anatomic is of "fmridata" class, default heat.colors(256)
\item \texttt{cutoff} not yet documented
\item \texttt{verbose} tell something on the progress?
\item \ldots additional arguments for plot
\end{itemize}

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 \texttt{spm} is \texttt{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 \texttt{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 \texttt{pvalues} for statistical evaluation. If \texttt{spm} is \texttt{false} the estimated BOLD response together with a confidence interval corresponding to \texttt{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 \texttt{tkdestroy(object)})

Note

3 dimensional plotting requires the \texttt{tkrplot} package.

Author(s)

Karsten Tabelow <tabelow@wias-berlin.de>

References


See Also

\texttt{fmri.pvalue}

Examples

\begin{verbatim}
## Not run: plot(pvalue)
\end{verbatim}
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)
**Description**

Read HEAD/BRIK file.

**Usage**

```r
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:

- `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


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, 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)
```
read.DICOM

**I/O function**

**Description**

Read DICOM file.

**Usage**

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

**Arguments**

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

**Details**

The function reads a DICOM file.

**Value**

Object with the following list entries:

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

**Note**

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

**Author(s)**

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

http://medical.nema.org

See Also

read.AFNI, read.ANALYZE

Examples

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

read.NIFTI

I/O Functions

Description

Read fMRI data from NIFTI file(s).

Usage

```R
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:

- `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

See Also
read.ANbeiten, read.AFNI

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

summary.fmridata

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.ANheiten.
...

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 filename I/O functions

Description

Write BRIK/HEAD files.

Usage

```r
def 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)
```
Description

Write a 4 dimensional datacube in ANALYZE file format.

Usage

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
[5] regular – character
[6] hkey – character
[9] datatype – integer, datatype usually "4"
[10] bitpix – 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.
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")
```

write.NIFTI   I/O Functions

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.
write.NIFTI

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

References

See Also
read.ANALYZE, 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.AFN1, 33
  read.ANALYZE, 34
  read.DICOM, 36
  read.NIFTI, 37
  write.AFN1, 39
  write.ANALYZE, 41
  write.NIFTI, 42
*Topic datasets
  hvred, 28
*Topic design
  fmri.design, 5
  fmri.designG, 6
  fmri.stimulus, 25
*Topic hplot
  plot.fmridata, 30
*Topic htest
  fmri.pvalue, 21
*Topic iplot
  plot.fmridata, 30
*Topic misc
  Convert Between fmridata and oro.nifti, 2
  *Topic multivariate
    fmriica, 27
    ngca, 28
  *Topic nonparametric
    fmriica, 27
    ngca, 28
  *Topic print
    plot.fmridata, 30
    summary.fmridata, 38
*Topic regression
  fmri.design, 5
  fmri.designG, 6
  fmri.detrend, 8
  fmri.lm, 9
  fmri.lmePar, 12
  fmri.metaPar, 16
  fmri.stimulus, 25
*Topic smooth
  fmri.smooth, 23
*Topic utilities
  cutroi, 3
  extract.data, 4
  fmri.gui, 9
  read.AFN1, 33
  read.ANALYZE, 34
  read.DICOM, 36
  read.NIFTI, 37
  write.AFN1, 39
  write.ANALYZE, 41
  write.NIFTI, 42
  apply, 13, 18
Convert Between fmridata and oro.nifti, 2
  cutroi, 3
  extract.data, 4
  fmri.design, 5, 6, 7, 9, 11, 15, 26
  fmri.designG, 6, 12, 15
  fmri.detrend, 8
  fmri.gui, 9
  fmri.lm, 5, 6, 8, 9, 9, 13, 17, 20, 22, 26
  fmri.lmePar, 7, 12, 18–20
  fmri.metaPar, 15, 16
  fmri.pvalue, 9, 21, 31
  fmri.smooth, 9, 22, 23
  fmri.stimulus, 6, 7, 9, 11, 15, 25
  fmri2oro(Convert Between fmridata and oro.nifti), 2
  fmriica, 27
  hvred, 28
  lme, 12, 15
ngca, 28, 28

oro2fmri (Convert Between fmridata and oronifti), 2

parApply, 13, 18
plot, 22
plot.fmridata, 9, 22, 30
plot.fmrisegment (plot.fmridata), 30
print.fmridata, 9, 32, 39

read.AFNI, 4, 9, 33, 35, 37, 38, 40
read.ANALYZE, 4, 9, 34, 37, 38, 42, 43
read.DICOM, 9, 36
read.NIFTI, 3, 4, 37
rma.uni, 16, 17, 20

summary.fmridata, 32, 38

write.AFNI, 9, 34, 39, 42, 43
write.ANALYZE, 9, 35, 40, 41
write.NIFTI, 9, 42