Package ‘vows’

August 21, 2016

Type Package
Title Voxelwise Semiparametrics
Version 0.5
Date 2016-08-21
Author Philip Reiss, Yin-Hsiu Chen, Lei Huang, Lan Huo, Ruixin Tan, and Rong Jiao
Maintainer Philip Reiss <phil.reiss@nyumc.org>
Depends R (>= 2.9.0), fda, gamm4
Imports mgcv, RLRsim, oro.nifti, shape, stringr
Description Parametric and semiparametric inference for massively parallel models, i.e., a large number of models with common design matrix, as often occurs with brain imaging data.
License GPL (>= 2)
LazyLoad yes
RoxygenNote 5.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2016-08-21 12:22:51

R topics documented:

vows-package ................................................................. 2
extract.fd ................................................................. 3
F.mp ................................................................. 4
Fdr.rlrt ................................................................. 5
funkmeans ................................................................. 6
funkmeans4d ............................................................. 7
lm.mp ................................................................. 8
lm4d ................................................................. 9
nii2R ................................................................. 10
permF.mp ............................................................. 11
Description

This package efficiently performs inference on a large set of parametric or semiparametric regressions that are "parallel" in the sense that they have a common design matrix. The functions are inspired by neuroimaging applications, where the parallel models pertain to a grid of brain locations known as voxels.

Details

Functions ending in ".mp" ("massively parallel") are designed for responses in the form of a (wide) matrix; functions ending in "4d" take four-dimensional response data (e.g., a set of images) and convert it to matrix form so that the corresponding ".mp" function can be applied. Examples include `lmNmp` and `lmN4d` for ordinary linear models, `rlrtNmp` and `rlrtN4d` for restricted likelihood ratio tests (RLRTs) of a parametric null hypothesis vs. a smooth alternative, and `semiparNmp` and `semiparN4d` for smoothing (see Reiss et al., 2014).

Author(s)

Philip Reiss <phil.reiss@nyumc.org>, Yin-Hsiu Chen <enjoychen0701@gmail.com>, Lei Huang <huanggracer@gmail.com>, Lan Huo, Ruixin Tan and Rong Jiao <jiaorong007@gmail.com>

Maintainer: Philip Reiss <phil.reiss@nyumc.org>
References


---

**extract.fd**

*Extract curve estimates to be clustered*

---

**Description**

Given a massively parallel smoothing object created by `semipar.mp`, this function extracts an object of class "fd" representing the curves that one wishes to cluster using `funkmeans`.

**Usage**

```r
eextract.fd(obj, term = 1, intercept = (term == 1))
```

**Arguments**

- `obj`: object created by `semipar.mp`
- `term`: which smooth term to extract (useful if the fitted model includes more than one term).
- `intercept`: logical; if TRUE, intercept will be added to all coefficients. For simple nonparametric regression this should be done to recover the fitted values.

**Value**

an object of class "fd" representing the fitted curves, which can be clustered by `funkmeans`.

**Author(s)**

Ruixin Tan

**See Also**

`semipar.mp`, `funkmeans`

**Examples**

```r
# see example for plot.funkmeans
```
F.mp

F-tests for massively parallel linear models

Description

Performs F-tests for removing one or more terms from each of a large number of models with
common design matrix.

Usage

F.mp(formula, which)

Arguments

formula a formula such as "Y ~ X", where Y is an \( n \times V \) response matrix and X is an
\( n \times p \) design matrix common to all \( V \) models.

which number or vector indicating which column(s) of the model matrix are to be tested
for removal from the model.

Value

F F-statistics for each of the models.

df1 numerator degrees of freedom.

df2 denominator degrees of freedom.

pvalue upper-tailed p-value.

X design matrix.

Author(s)

Philip Reiss <phil.reiss@nyumc.org> and Lei Huang <huangracer@gmail.com>

See Also

lm.mp, permF.mp

Examples

Y = matrix(rnorm(6000), nrow=20)
X = rnorm(20)
t2 = F.mp(Y~X, which=2)
Description

Given a set of RLRT results and a threshold, this function outputs an estimate of the FDR (in the empirical Bayes sense of Efron, 2010) when the given threshold is used to determine which null hypotheses to reject.

Usage

Fdr.rlrt(rlrt.obj, threshold)

Arguments

rlrt.obj an RLRT object obtained from rlrt.mp or rlrt4d.
threshold threshold at which the null hypothesis is rejected.

Value

A list with elements

MoM FDR based on method of moments estimator of RLRT parameters (Greven et al., 2008).
ML FDR based on maximum likelihood estimation of RLRT parameters, as described in Greven et al. (2008).

Author(s)

Philip Reiss <phil.reiss@nyumc.org>

References


See Also

rlrt.mp, rlrt4d

Examples

# See example for rlrt.mp
Description

This function performs k-means clustering for curve estimates corresponding to each of a 3D grid of points. For example, when scatterplot smoothing is performed at each of a grid of brain voxels as in Reiss et al. (2014), this function can be used to cluster the obtained smooths.

Usage

```r
funkmeans(fdobj, deriv = 1, lambda = 0, ncomp, centers, nstart = 10,
           store.fdobj = TRUE)
```

Arguments

- `fdobj`: a functional data object, of class "fd", defining the set of curves being clustered.
- `deriv`: which derivative of the curves should be clustered. If 0, the curves themselves are clustered; if 1 (the default), their first derivatives are clustered, a natural way to assign curves of similar shape to the same cluster.
- `lambda`: smoothing parameter for functional PCA as implemented by `pca.fd`.
- `ncomp`: number of functional principal components.
- `centers`: number of clusters.
- `nstart`: number of randomly chosen sets of initial centers used by the `kmeans` function.
- `store.fdobj`: logical: Should the input fd object be stored in the output? May wish to set to FALSE for large sets of smooths.

Details

The functional clustering algorithm consists of performing (i) functional principal component analysis of the curve estimates or their derivatives, followed by (ii) k-means clustering of the functional PC scores (Tarpey and Kinateder, 2003).

Value

An object of class "funkmeans", which is a list with elements:

- `cluster`, `centers`, `withinss`, `tots`, `tot.withinss`, `betweenness`, `size` see `kmeans`.
- `basis.coef`: basis object and coefficient matrix defining the functional data object (see `fd`) for the curves that are clustered.
- `fpca`: functional principal components object, output by `pca.fd`.
- `R2`: proportion of variance explained by the k clusters.
Author(s)

Philip Reiss <phil.reiss@nyumc.org>, Lei Huang <huangracer@gmail.com> and Lan Huo

References


See Also

*funkmeans4d*

Examples

```r
data(test)
d4 = test$d4
x = test$x
semi.obj = semipar4d(d4, -sf(x), -5:5, data.frame(x = x))
fdobj = extract.fd(semi.obj)
fkmobj = funkmeans4d(fdobj, d4, ncomp=6, centers=3)
```

---

**funkmeans4d**

*Functional k-means clustering for parallel smooths for 4-dimensional data*

Description

This is a wrapper function for *funkmeans* to handle 3D image responses.

Usage

*funkmeans4d(fdobj, arr4d, ...)*

Arguments

- `fdobj`: a functional data object, of class "fd", defining the set of curves being clustered.
- `arr4d`: a 4-dimensional array containing the raw data that were smoothed at each point. The first 3 dimensions refer to x, y, and z coordinates and the last dimension corresponds to different images.
- `...`: other arguments, passed to *funkmeans*. 
**Value**

An object of class "funkmeans4d", which is also of class "funkmeans" but has the additional component `arr.cluster`: an array, of dimension `dim(arr4d)[1:3]`, giving the cluster memberships.

**Author(s)**

Philip Reiss <phil.reiss@nyumc.org>, Lei Huang <huangracer@gmail.com> and Lan Huo

**See Also**

funkmeans

**Examples**

```r
# See example for funkmeans
```

---

**Description**

Efficiently fits $V$ linear models with a common design matrix, where $V$ may be very large, e.g., the number of voxels in a brain imaging application.

**Usage**

```r
lm.mp(Y, formula, store.fitted = FALSE)
```

**Arguments**

- `Y`: $n \times V$ outcome matrix.
- `formula`: a formula object such as "~ x1 + x2".
- `store.fitted`: logical: Should the fitted values be stored? For large $V$, setting this to TRUE may cause memory problems.

**Value**

- `coef`: $p \times V$ matrix of coefficient estimates.
- `sigma2`: $V$-dimensional vector of error variance estimates.
- `se.coef`: $p \times V$ matrix of coefficient standard error estimates.
- `X`: $n \times p$ common design matrix.
- `fitted`: $n \times V$ matrix of fitted values.
**lm4d**

**Author(s)**

Philip Reiss <phil.reiss@nyumc.org>, Lei Huang <huangracer@gmail.com>, and Yin-Hsiu Chen <enjoychen0701@gmail.com>

**See Also**

`lm4d`, `summary.lm.mp`

**Examples**

```r
# Please see example for lm4d
```

---

**Description**

This is a wrapper function for `lm.mp` to handle 3D image responses.

**Usage**

```r
lm4d(arr4d, formula, store.fitted = FALSE)
```

**Arguments**

- `arr4d`:
  a 4-dimensional response array, where the first 3 dimensions refer to spatial coordinates and the last dimension corresponds to different images.

- `formula, store.fitted`:
  see `lm.mp`.

**Value**

An object of class "`lm.mp`", with two changes. (1) If `store.fitted = TRUE`, the fitted values are given as a 4-dimensional array. (2) A call component is included.

**Author(s)**

Lei Huang <huangracer@gmail.com>, Yin-Hsiu Chen <enjoychen0701@gmail.com>, and Philip Reiss <phil.reiss@nyumc.org>

**See Also**

`lm.mp`
Examples

data(test)
d4 = test$d4
x = test$x
lmobj = lm4d(d4, ~x)

# Convert d4 to a matrix, and confirm that lm.mp() gives the same results as lm4d()
d4.2 = d4
dim(d4.2) = c(prod(dim(d4)[1:3]), dim(d4)[4])
Y = t(d4.2)
lmobj2 = lm.mp(Y, ~x)
all.equal(lmobj$coef, lmobj2$coef)

nii2R  NIfTI-to-R conversion

Description

Reads in a NIfTI (.nii) file and puts the data in a 4-dimensional array.

Usage

nii2R(niifilename, which.vols = NULL, savename = NULL, remove.zero = TRUE, maskname = NULL, ind = NULL, ind.auto = TRUE, coord = NULL)

Arguments

niifilename  the path for the .nii file.
which.vols  which volumes (images) to include. In terms of the 4D array, this refers to sub-setting in the fourth dimension. If NULL (the default), all volumes are included.
savename  if non-NULL, the name of the .RData file to which the 4D array will be saved.
remove.zero  optional when maskname is not provided. If TRUE, a binary array indicating the voxels with nonzero measures based on the first three dimension of the nii file will be provided. If FALSE, a 3D array with TRUE everywhere will be provided.
maskname  name of a .nii file providing a "mask", a 3D binary array indicating which voxels to include.
ind, ind.auto  ind is an optional list saying which indices (which slices of the image) to include in each of the three dimensions. If NULL, this will be all slices with nonzero data if ind.auto = TRUE, and all slices otherwise.
coord  coordinates of the first three dimensions of the 4D array created.

Value

a 4-dimensional array.
permF.mp

Author(s)
Lei Huang <huangracer@gmail.com>, Philip Reiss <phil.reiss@nyumc.org> and Rong Jiao <jiaorong007@gmail.com>

See Also
R2nii

permF.mp  Permutation F-tests for massively parallel linear models

Description
Performs permutation F-tests for parallel linear models with a common design matrix. Currently restricted to testing with the intercept-only model as the null hypothesis. The permutation method controls the familywise error rate (FWER) at a desired level; see Details.

Usage
permF.mp(formula, nperm = 499, alpha = 0.05, report.every = 50)

Arguments
formula a formula such as "Y ~ X", where Y is an n × V response matrix and X is an n × p design matrix common to all V models.
nperm number of permutations.
alpha level at which to control the FWER.
report.every parameter controlling how often to report the number of permutations performed; by default, every 50.

Details
The observed F-statistics are referred to a permutation distribution of the maximum F-statistic over all V tests. This is a standard approach to FWER control in neuroimaging (Nichols and Holmes, 2001).

Value
maxF.perm maximal F-statistics obtained from each of the permuted data sets.
F.obs the observed F-statistics.
threshold critical value obtained from the permutations.
pvalue adjusted (familywise error rate-controlling) p-values.

Author(s)
Philip Reiss <phil.reiss@nyumc.org> and Lei Huang <huangracer@gmail.com>
References

See Also
fNmp

Examples

```r
Y = matrix(rnorm(6000), nrow=20)
X = rnorm(20)
t3 = permF.mp(Y-X)
```

plot.funkmeans Plotting of k-means clustering results for massively parallel smooths

Description
Visualization of functional k-means clustering as implemented by `funkmeans`.

Usage
```r
## S3 method for class 'funkmeans'
plot(x, fdobj = NULL, deriv = 0,
     ncluster = nrow(x$centers), new.array = TRUE, mfrow = NULL,
     colvec = NULL, cex.mtext = 0.7, xlabs = "", ylabs = "", titles = "",
     ...)
```

Arguments

- `x`: a functional k-means clustering object obtained from `funkmeans`.
- `fdobj`: a functional data object, of class "fd", defining the set of curves being clustered. By default, this is taken to be `x$fdobj`; but if the latter is NULL, `fdobj` must be specified. See the two cases in the example.
- `deriv`: which derivative to display in the plots, which show 30 randomly selected curves, along with the cluster center, from each cluster. By default, the "0th derivative" is used (i.e., the curves themselves).
- `ncluster`: number of clusters to display. By default, all are displayed.
- `new.array`: logical: if TRUE, plots will be displayed in an array whose dimensions are set by the `mfrow` argument.
- `mfrow`: a vector of length 2 giving the numbers of rows and columns for the array of plots. By default, the number of rows will exceed the number of columns by 0 or 1, depending on `ncluster`. 

plot.funkmeans  
Plotting of k-means clustering results for massively parallel smooths
Display cross-sections of voxelwise RLRT results

Plots slices of the 3D array representing a set of voxelwise RLRT results.

Author(s)

Yin-Hsiu Chen <enjoychen0701@gmail.com>, Philip Reiss <phil.reiss@nyumc.org>, Lan Huo, and Ruixin Tan

See Also

funkmeans

Examples

data(test)
d4 = test$d4
x = test$x
semi.obj = semipar4d(d4, formula = -sf(x), data = data.frame(x = x), lsp=-5:5)
myfdobj = extract.fd(semi.obj)

# Case 1: fd object is stored in funkmeans object...
fkmobj = funkmeans(myfdobj, ncomp = 8, centers = 6)
plot(fkmobj)

# Case 2: fd object is not stored...
fkmobj = funkmeans(myfdobj, ncomp = 8, centers = 6, store.fdobj=FALSE)
plot(fkmobj, myfdobj)
Usage

```r
## S3 method for 'rlrt4d'
plot(x, array4d, disp = c("stat", "p", "fdr", "wdcf"),
   titl = NULL, slices = NULL, colbar = TRUE,
   col.image = shape::femmemiel(100)[100:1], neglog10 = FALSE,
   threshold = NULL, mar = c(2, 2, 2, 2), digit = 2, nrow = NULL, ...)
```

Arguments

- **x**: a voxelwise RLRT object as produced by `rlrt4d`.
- **array4d**: the 4D array on which the voxelwise RLRT was performed.
- **disp**: values from the RLRT object to be displayed: either RLRT statistics, p-values, or FDR values.
- **titl**: title of the panel.
- **slices**: indices of the slice(s) to be displayed.
- **colbar**: logical: Should a color bar be included?
- **col.image**: color scheme for the color bar, as generated by `rainbow`, `heat.colors`, etc.
- **neglog10**: logical; if TRUE, negative base 10 logarithm (of the quantity specified by `disp`) is displayed.
- **threshold**: the upper limit of the values to be plotted. All larger values will be replaced by the threshold value.
- **mar**: a numerical vector of the form c(bottom, left, top, right) specifying the number of lines of margin on the four sides of the plot.
- **digit**: number of significant digits in labels.
- **nrow**: number of rows on the plot.
- **...**: arguments passed to `plot`.

Author(s)

Lei Huang <huangracer@gmail.com>, Philip Reiss <phil.reiss@nyumc.org> and Lan Huo

See Also

`rlrt4d`

Examples

```
# Please see the example for rlrt4d
```
**Description**

Given a massively parallel smoothing object produced by `semipar.mp`, the function plots the fitted smooth(s) for a given point (e.g., at a given voxel).

**Usage**

```r
## S3 method for class 'semipar.mp'
plot(x, Y, arr.ind = NULL, which.vox = NULL,
     which.smooth = NULL, coverage = 0.95, length.new = 100, ylim = NULL,
     ylab = NULL, ...)
```

**Arguments**

- `x`: an object of class "semipar.mp".
- `Y`: an n x V outcome matrix.
- `arr.ind`: a 3-element vector specifying the element of the 3-dimensional array of locations (e.g., voxels) for which plotting is desired. If NULL, which.vox must be specified.
- `which.vox`: the index of the voxel to be plotted. If NULL, arr.ind must be specified.
- `which.smooth`: the index of the smooth term of which the confidence interval plot is to be displayed. The default value is NULL which refers to displaying the plots for all the smooth terms in the model.
- `coverage`: the confidence level of the pointwise confidence intervals in the plot.
- `length.new`: length of the vector of ordered variables with which to predict.
- `ylim`, `ylab`, `...`: arguments to be passed to `plot`.

**Author(s)**

Yin-Hsiu Chen <enjoychen0701@gmail.com>, Philip Reiss <phil.reiss@nyumc.org>, and Lan Huo

**Examples**

```r
n<-32
Ys <- matrix(0, n, 5)
for(i in 1:n) Ys[i,]<-~2:2+rnorm(5, i^2, i^0.5)+sin(i)
x1 <- rnorm(n,0,5)
x2 <- 1:n+runif(n, 1, 20)
semipar.obj <- semipar.mp(~x1+sf(x2,k=10), Y=Ys,lsp=seq(5,50,30))
plot(semipar.obj, Y=Ys, which.vox=2)
```
 DESCRIPTION

Fits a possibly very large number of models, with common design matrix, by quadratically penalized least squares, with identifiability constraints imposed. This function serves as the fitting engine for semipar.mp.

 USAGE

qplsc.mp(Y, modmat, penmat, constr.list = NULL, lsp, nulldim = NULL, store.reml = FALSE, store.fitted = FALSE)

 ARGUMENTS

Y an \( n \times V \) response matrix (\( V \) refers to number of models fitted in parallel, e.g., voxels in neuroimaging applications).
modmat model matrix, e.g., a matrix of B-spline basis functions.
penmat penalty matrix.
constr.list a list of length equal to number of constraints to be imposed, containing information for reparametization to an unconstrained optimization. Attribute 'C' is the constraint matrix, and 'start' and 'end' refer to the corresponding column positions of the model matrix.
lsp vector of candidate tuning parameters (\( \log(\lambda) \)).
nulldim null space dimension, ordinarily equal to the order of the derivative penalty.
store.reml logical: should the pointwise REML criterion at each grid point be included in the output? FALSE by default, as this output can be very large.
store.fitted logical: should the fitted values be included in the output? FALSE by default.

 VALUE

An object of class "qplsc.mp", which is a list with elements:
fitted fitted value matrix, if store.fitted = TRUE.
edf matrix giving the effective degrees of freedom per parameter, as in Wood (2004), for each model.
pwdf vector of point-wise degrees of freedom, equal to the column sums of edf.
pwlsp vector of point-wise log smoothing parameters.
coef matrix of coefficients.
reml matrix giving the point-wise REML criterion at each grid point, if store.reml = TRUE.
modmat model matrix.
penmat penalty matrix.
RinvU \quad R^{-1}U, \text{ as in Reiss et al. (2014); this and } \tau \text{ are used for plotting.}

\tau \quad \text{singular values of } R^{-T}PR^{-1}, \text{ as in Reiss et al. (2014).}

\sigma_R \quad \text{vector of variance estimates.}

ttu \quad \text{matrix for transformation to an unconstrained problem.}

Author(s)

Lei Huang <huangracer@gmail.com>, Yin-Hsiu Chen <enjoychen0701@gmail.com>, and Philip Reiss <phil.reiss@nyumc.org>

References


Examples

```r
## see semipar.mp
```

R2nii

Save data to a NIfTI file

Description

This function can be used to output the results of voxelwise RLRT or smoothing.

Usage

`R2nii(arr, name.nii)`

Arguments

- `arr` a 3D or 4D array containing data to be saved.
- `name.nii` filename, excluding the .nii extension.

Value

None; a NIfTI file is created.

Author(s)

Lei Huang <huangracer@gmail.com>, Philip Reiss <phil.reiss@nyumc.org> and Rong Jiao <jiaorong007@gmail.com>
rlrt.mp  

**Massively parallel restricted likelihood ratio tests**

**Description**

Conducts a possibly very large number of restricted likelihood ratio tests (Crainiceanu and Ruppert, 2004), with common design matrix, for a polynomial null against a smooth alternative.

**Usage**

```r
rlrt.mp(Y, x = NULL, loginvsp, nbasis = 15, norder = 4, nulldim = NULL,
        evalarg = NULL, get.df = FALSE, B = NULL, P = NULL)
```

**Arguments**

- `Y`: ordinarily, an \( n \times V \) outcome matrix, where \( V \) is the number of hypotheses (in brain imaging applications, the number of voxels). Can also be given by an object of class "fd".
- `x`: a vector or matrix of covariates.
- `loginvsp`: a grid of candidate values of the log inverse smoothing parameter.
- `nbasis`: number of B-spline basis functions.
- `norder`: order of B-splines.
- `nulldim`: dimension of the null space of the penalty.
- `evalarg`: if `Y` is of class "fd", the argument values at which the functions are evaluated.
- `get.df`: logical: Should the effective df of the smooth at each point be obtained?
- `B`: evaluation matrix of the B-spline basis functions.
- `P`: penalty matrix.

**Details**

The **RLRsim** package of Scheipl et al. (2008) is used to simulate the common null distribution of the RLRT statistics.

**Value**

A list with components

- `table`: matrix of log restricted likelihood ratio values at each grid point, for each test.
- `stat`: RLRT statistics, i.e., the supremum of the values in `table` for each test.
- `logsp`: log smoothing parameter at which the supremum of the restricted likelihood ratio is attained for each test.
if get.df = TRUE, the effective degrees of freedom corresponding to the log smoothing parameter values in logsp.

sim values simulated from the null distribution of the restricted likelihood ratio statistic.

pvalue p-values for the RLRT statistics.

fdr Benjamini-Hochberg false discovery rates corresponding to the above p-values.

call the call to the function.

Author(s)

Lei Huang <huangracer@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

References


See Also

rlrt4d, and Fdr.rlrt for a more sophisticated false discovery rate procedure.

Examples

```r
Y = matrix(rnorm(6000), nrow=20)
x = rnorm(20)
t4 = rlrt.mp(Y, x, loginvsp = -22:0)
f4 = Fdr.rlrt(t4, 6)
```

Description

Conducts a possibly very large number of restricted likelihood ratio tests (Crainiceanu and Ruppert, 2004), with specified random-effects design matrix and fixed-effects design matrix, for a polynomial null against a smooth alternative.
**Usage**

```r
rlrt.mp.fit(Y, X, Z, loginvsp, evalarg = NULL, get.df = FALSE)
```

**Arguments**

- **Y** ordinarily, an \( n \times V \) outcome matrix, where \( V \) is the number of hypotheses (in brain imaging applications, the number of voxels)
- **X** the fixed-effects design matrix.
- **Z** the random-effects design matrix.
- **loginvsp** a grid of candidate values of the log inverse smoothing parameter.
- **evalarg** if \( Y \) is of class "fd", the argument values at which the functions are evaluated.
- **get.df** logical: Should the effective df of the smooth at each point be obtained?

**Details**

The **RLRsim** package of Scheipl et al. (2008) is used to simulate the common null distribution of the RLRT statistics.

**Value**

A list with components

- **table** matrix of log restricted likelihood ratio values at each grid point, for each test.
- **stat** RLRT statistics, i.e., the supremum of the values in **table** for each test.
- **logsp** log smoothing parameter at which the supremum of the restricted likelihood ratio is attained for each test.
- **df** if \( \text{get.df} = \text{TRUE} \), the effective degrees of freedom corresponding to the log smoothing parameter values in **logsp**.
- **sim** values simulated from the null distribution of the restricted likelihood ratio statistic.
- **pvalue** p-values for the RLRT statistics.
- **fdr** Benjamini-Hochberg false discovery rates corresponding to the above p-values.
- **call** the call to the function.

**Author(s)**

Lei Huang <huangracer@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

**References**


Examples

Y = matrix(rnorm(6000), nrow=20)
x = rnorm(20)
z = rep(1:5, each = 4)
t4. = rlrt.mp.fit(Y, x, z, loginvsp = -22:0)

Description

A wrapper function for rlrt.mp to handle 3D image responses.

Usage

rlrt4d(arr4d, x = NULL, nbasis = 15, norder = 4, nulldim = NULL,
loginvsp, get.df = FALSE, B = NULL, P = NULL)

Arguments

arr4d      a 4-dimensional response array, where the first 3 dimensions refer to spatial
coordinates and the last dimension corresponds to different images.
x, nbasis, norder, nulldim, loginvsp, get.df, B, P

see rlrt.mp.

Value

A massively parallel RLRT object, as produced by rlrt.mp.

Author(s)

Lei Huang <huangracer@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

See Also

plot.rlrt4d, rlrt.mp

Examples

data(test)
d4 = test$d4
x = test$x
rlrtobj = rlrt4d(d4, x, loginvsp = -5:5)
plot(rlrtobj, d4, slice=5)
Description

Inputs a voxelwise smoothing object as produced by `semipar4d`, and outputs an object containing the results for a subset of the voxels.

Usage

```
screen.vox(semi.obj, arr4d, include)
```

Arguments

- `semi.obj`: an object of class `semipar.mp`.
- `arr4d`: the 4-dimensional array used to generate the object.
- `include`: a logical matrix indicating which points (or voxels) should be included.

Value

A modified version of `semipar.obj`, with pointwise coefficients (`coef` component), pointwise degrees of freedom (`pwdf`), pointwise log smoothing parameter (`pwlsp`), and pointwise variance estimate (`sigma2`) for the points specified by `include` only.

Author(s)

Lei Huang <huangracer@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

See Also

`semipar.mp`

Examples

```r
data(test)
d4 = test$d4
x = test$x
vw.obj = semipar4d(d4, formula = ~sf(x), data = data.frame(x = x), lsp=-5:5)

# Include only the first 600 voxels
sv = screen.vox(vw.obj, d4, rep(1:0, c(600,400)))
```
Massively parallel semiparametric mixed models

Description

Fits a set of semiparametric mixed models, with a common design matrix, by repeated calls to \texttt{gamm4}. Only a single smooth term is permitted.

Usage

\texttt{semipar.mix.mp(y, x, param = NULL, random, data.ran, k = 10, norder = 4, pen.order = 2, knots = "quantile", store.gamm = FALSE)}

Arguments

- \texttt{y} \hspace{1cm} n \times V \hspace{1cm} \text{response matrix.}
- \texttt{x} \hspace{1cm} \text{a vector giving the predictor upon which each column of } y \text{ is regressed.}
- \texttt{param} \hspace{1cm} \text{a matrix or vector for the parametric terms in the model.}
- \texttt{random} \hspace{1cm} \text{a formula, passed to } \texttt{gamm4}, \text{ specifying the random effects structure in } \texttt{lmer} \text{ style. See the example.}
- \texttt{data.ran} \hspace{1cm} \text{a required data frame containing the factors used for random effects.}
- \texttt{k} \hspace{1cm} \text{number of knots.}
- \texttt{norder} \hspace{1cm} \text{order of B-splines: the default, 4, gives cubic B-splines.}
- \texttt{pen.order} \hspace{1cm} \text{order of the derivative penalty.}
- \texttt{knots} \hspace{1cm} \text{knot placement for the B-spline bases. The default, "quantile", gives knots at equally spaced quantiles of the data. The alternative, "equispaced", gives equally spaced knots.}
- \texttt{store.gamm} \hspace{1cm} \text{logical: should the } \texttt{gamm4} \text{ objects to be stored in the output? FALSE by default.}

Details

Unlike \texttt{semipar.mp}, this function does not use large matrix multiplications to avoid looping through model fits. Instead it performs a separate call to \texttt{gamm4} to fit a semiparametric mixed model for each column of \texttt{Y}.

Value

- \texttt{coef} \hspace{1cm} \text{matrix of the coefficients obtained from } \texttt{gamm4} \text{ looping (including both parametric and nonparametric parts).}
- \texttt{bsplinecoef} \hspace{1cm} \text{matrix of B-spline coefficients.}
- \texttt{pwdf} \hspace{1cm} \text{vector of pointwise effective degrees of freedom.}
- \texttt{pwlsp} \hspace{1cm} \text{vector of pointwise log smoothing parameters: grid values maximizing the restricted likelihood at each point.}
matrix of basis function values.
constrains.
transformation matrix to impose constraints.
B-spline basis object, of the type created by the fdas package; the coefficient estimates are with respect to this basis.

Author(s)
Yin-Hsiu Chen <enjoychen0701@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

Examples
Y = matrix(rnorm(3000),3)
x1 = rnorm(1000)
x2 = matrix(rnorm(2000),2)
family.fac <- factor(rep(1:20,rep(50,20)))
person.fac <- factor(rep(rep(1:25,rep(2,25)),rep(20,50)))
semimix = semipar.mp(Y = Y, x = x1, param = x2, random = ~ (1|a/b),
data.ran = data.frame(a = family.fac, b = person.fac))

Description
Fits a possibly very large number of semiparametric models by quadratically penalized least squares. The model may include a combination of parametric terms, smooth terms, varying-coefficient terms, and simple random effect structures.

Usage
semipar.mp(formula, Y, lsp, data = NULL, range.basis = NULL,
knots = "quantile", rm.constr = FALSE, random = NULL,
store.reml = FALSE, store.fitted = FALSE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>a formula object such as &quot;~ x1 + sf(x2) + sf(x2, effect = x3)&quot; where x1 is</td>
</tr>
<tr>
<td></td>
<td>a linear (parametric) predictor, x2 is a predictor on which the responses</td>
</tr>
<tr>
<td></td>
<td>depend smoothly, and x3 is a predictor whose effect is linear but varies</td>
</tr>
<tr>
<td></td>
<td>smoothly with x2 (i.e., a varying-coefficient predictor).</td>
</tr>
<tr>
<td>Y</td>
<td>an n × V response matrix, where V is the number of models fitted in parallel,</td>
</tr>
<tr>
<td></td>
<td>e.g., voxels in neuroimaging applications.</td>
</tr>
<tr>
<td>lsp</td>
<td>vector of candidate log tuning parameters (log(λ)).</td>
</tr>
<tr>
<td>data</td>
<td>an optional data frame containing the variables in the model.</td>
</tr>
</tbody>
</table>
range.basis a numeric vector of length 2 defining the interval over which the B-spline basis is created. If NULL, it will be set as the range of the variable to be evaluated by the basis.
knots knot placement for the B-spline bases. The default, "quantile", gives knots at equally spaced quantiles of the data. The alternative, "equispaced", gives equally spaced knots.
rm.constr logical: should the constraints be removed for varying-coefficient models?
random a formula or a matrix for random effects.
store.reml logical: should the pointwise REML criterion at each grid point be included in the output? FALSE by default, as this output can be very large.
store.fitted logical: should the fitted values be included in the output? FALSE by default.

Details
The basic approach to massively parallel smoothing is described in Reiss et al. (2014). Although simple mixed-effect models are available, semipar.mix.mp is generally preferable for mixed models with a single smooth term.

Each element of list.all corresponding to a nonparametric term of the model is a list with components modmat, penmat, pen.order, start, and end. For each parametric term, the same five components are included, plus basis, argvals, effect, k, and norder.

Value
An object of class "semipar.mp", which is also of class "qplsc.mp" but includes the following additional elements:

where.sf, where.nsf vectors or scalars identifying where the smooth and non-smooth terms, respectively, appear in the model formula.
list.all a list of lists, one for each term of the model; see Details.
formula model formula.
Y response matrix.
1sp candidate values for the log smoothing parameter.
data the supplied data frame, if any.

Author(s)
Yin-Hsiu Chen <enjoychen0701@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

References
semipar4d

Massively parallel semiparametric regression for 4-dimensional data

Description

This is a wrapper function for `semipar.mp` to handle 3D image responses.

Usage

```r
semipar4d(arr4d, formula, lsp, data, range.basis = NULL, knots = "quantile",
          rm.constr = FALSE, random = NULL, store.reml = FALSE,
          store.fitted = FALSE)
```

Arguments

- `arr4d`: a 4-dimensional response array, where the first 3 dimensions refer to spatial coordinates and the last dimension corresponds to different images.
- `formula`, `lsp`, `data`, `range.basis`, `knots`, `rm.constr`, `random`, `store.reml`, `store.fitted` see `semipar.mp`.

Value

An object of class "semipar.mp", with two changes. (1) If `store.fitted = TRUE`, the fitted values are given as a 4-dimensional array. (2) A call component is included.

Author(s)

Yin-Hsiu Chen <enjoychen0701@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

See Also

`semipar.mp`

Examples

```r
data(test)
d4 = test$d4
x = test$x
semi.obj = semipar4d(d4, ~sf(x), lsp=-5:5, data=data.frame(x = x))
plot(semi.obj, which.vox = 4)
```
Defining smooth functions in semiparametric model formulae

Description
This function is called by `semipar.mp` to define B-spline smooths.

Usage
```
sf(argvals, effect = NULL, k = 10, norder = 4, pen.order = 2, range.basis = NULL, knots = "quantile")
```

Arguments
- `argvals`: a vector or matrix of covariates.
- `effect`: predictor whose effect varies with respect to `argvals`. E.g., if the effect of diagnosis varies with age, use `sf(age, effect = diagnosis)`. Similar to argument by `s`.
- `k`: number of B-spline basis functions.
- `norder`: order of B-splines: the default, 4, gives cubic B-splines.
- `pen.order`: order of the penalty, i.e., of the derivative defining the penalty.
- `range.basis`: a numeric vector of length 2 defining the interval over which the B-spline basis is created. If NULL, set to the range of the variable.
- `knots`: knots placement method for B-spline smoothing. The default, "quantile", places the knots at equally spaced quantiles of the data; "equispaced" gives equally spaced knots.

Author(s)
Yin-Hsiu Chen <enjoychen0701@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

Summary model fits

Description
summary method for class "lm.mp".

Usage
```
## S3 method for class 'lm.mp'
summary(object, ...)
```
Arguments

object an object of class `lm.mp`, ordinarily created by the function of that name or by `lm4d`.

... not currently used.

Value

tstat matrix of pointwise t-statistics for each coefficient in the linear model
pvalue matrix of the pointwise p-values for each coefficient in the linear model
aicc vector of pointwise corrected AIC

Author(s)

Philip Reiss <phil.reiss@nyumc.org> and Lei Huang <huangracer@gmail.com>

See Also

`lm.mp`

Examples

```r
Y = matrix(rnorm(6000), nrow=20)
X = rnorm(20)
t1 = lm.mp(Y, ~X)
st1 = summary(t1)
```

Description

A randomly generated data set consisting of 50 "response" images and 50 scalar "predictors".

Format

A list with two components:

- `list("d4")` a $10 \times 10 \times 10 \times 50$ array of responses
- `list("x")` a vector of 50 predictor values

---

test Toy data set

---
Utility functions related to the mgcv package

Description

These internal functions are used by `semipar.mix.mp` (but can also be used more generally) to customize the implementation of B-spline smoothing by `gam`. Specifically, a B-spline smooth with equispaced knots can be incorporated in a call to `gam` using a term of the form `s(x,bs="be")`, whereas knots at equally spaced quantiles of the data can be specified by `s(x,bs="bq")`.

Usage

```r
## S3 method for class 'bq.smooth.spec'
smooth.construct(object, data, knots)

## S3 method for class 'be.smooth.spec'
smooth.construct(object, data, knots)

## S3 method for class 'bspline.smooth'
Predict.matrix(object, data)
```

Arguments

- **object**: a `gam` smooth specification object generated by a term such as `s(x,bs="be")` or `s(x,bs="bq")`.

- **data**: For `smooth.construct.be.smooth.spec` and `smooth.construct.bq.smooth.spec`, a list containing just the data (including any by variable) required by the given term, with names corresponding to `object$term` (and `object$by`). The by variable is the last element. For `Predict.matrix.bspline.smooth`, a data frame containing the values of the (named) covariates at which the smooth term is to be evaluated. Exact requirements are as for `smooth.construct` and `smooth.construct2`.

- **knots**: a list containing any knots supplied for basis setup, in the same order and with the same names as `data`. If `NULL`, a default set of knots is used.

Details

These functions are not normally called directly. For further details, please see `smooth.construct.ps.smooth.spec` and `Predict.matrix.cr.smooth`.

Value

Either `smooth.construct.be.smooth.spec` or `smooth.construct.bq.smooth.spec` produces an object of class "bspline.smooth"; see `smooth.construct` for the elements that this object will contain. `Predict.matrix.bspline.smooth` produces a matrix mapping the coefficients for the smooth term to its values at the supplied data values.
Author(s)

Yin-Hsiu Chen <enjoychen0701@gmail.com> and Philip Reiss <phil.reiss@nyumc.org>

Examples

```r
x. = rnorm(20)
smoo.be <- smooth.construct.be.smooth.spec(s(x), data.frame(x = x.), NULL)
smoo.bq <- smooth.construct.bq.smooth.spec(s(x), data.frame(x = x.), NULL)
Predict.matrix.bspline.smooth(smoo.bq, data.frame(x = seq(min(x.), max(x.), 100)))
```
Index

*Topic **datasets**
  test, 28
*Topic **package**
  vows-package, 2

  extract.fd, 3

  F.mp, 4, 12
  fd, 3, 6, 7, 12, 18
  Fdr.rlrt, 5, 19
  funkmeans, 3, 6, 7, 8, 12, 13
  funkmeans4d, 7, 7

  gam, 29
  gamm4, 23

  heat.colors, 14

  kmeans, 6

  lm.mp, 2, 4, 8, 9, 28
  lm4d, 2, 9, 9, 28
  lmer, 23

  nii2R, 10, 18

  pca.fd, 6
  permF.mp, 4, 11
  plot, 13–15
  plot.funkmeans, 12
  plot.rlrt4d, 13, 21
  plot.sempar.mp, 15
  Predict.matrix.bspine.smooth
    (vows-mgcv), 29
  Predict.matrix.cr.smooth, 29

  qplsc.mp, 16, 25

  R2nii, 11, 17
  rainbow, 14
  rlrt.mp, 2, 5, 18, 21

  rlrt.mp.fit, 19
  rlrt4d, 2, 5, 14, 19, 21

  s, 27
  screen.vox, 22
  sempar.mix.mp, 23, 25, 29
  sempar.mp, 2, 3, 15, 16, 22, 23, 24, 26, 27
  sempar4d, 2, 22, 26
  sf, 27
  smooth.construct, 29
  smooth.construct.be.smooth.spec
    (vows-mgcv), 29
  smooth.construct.bq.smooth.spec
    (vows-mgcv), 29
  smooth.construct.ps.smooth.spec, 29
  smooth.construct2, 29
  summary.lm.mp, 9, 27

  test, 28

  vows(vows-package), 2
  vows-mgcv, 29
  vows-package, 2