Package ‘SCBmeanfd’

February 19, 2015

Type Package
Title Simultaneous Confidence Bands for the Mean of Functional Data
Version 1.1
Date 2014-01-31
Author David Degras
Maintainer David Degras <ddegrasv@depaul.edu>
Description This package implements statistical methods for estimating and inferring the mean of functional data. The methods are based on simultaneous confidence bands and local polynomial fitting. They feature: bandwidth selection by plug-in and cross-validation, goodness-of-fit tests for parametric models, equality tests for two-sample problems, and plotting functions.
License GPL-3
Depends R (>= 2.10), stats, graphics
Imports boot, KernSmooth
URL http://www.R-package.org
NeedsCompilation no
Repository CRAN
Date/Publication 2014-02-01 17:35:26

R topics documented:

  SCBmeanfd-package .................................................. 2
cv.score ................................................................. 3
cv.select ................................................................. 4
phoneme ................................................................. 6
plasma ................................................................. 7
plot.SCBand ............................................................. 8
plrt.model ............................................................. 9
plugin.select .......................................................... 11
print.SCBand .......................................................... 12
scb.equal ............................................................... 13
Simultaneous Confidence Bands for Mean Functions

Description
This package implements statistical methods for the mean of functional data based on local polynomials and simultaneous confidence bands (SCB). Functions for kernel bandwidth selection, SCB estimation, goodness-of-fit tests, and two-sample tests are provided. Visualization tools are also available.

Details

Package: SCBmeanfd
Type: Package
Version: 1.1
Date: 2014-01-29
License: GPL-3

Author(s)

David Degras <ddegrasv@depaul.edu>

References


cv.score

Leave-One-Curve-out Cross-Validation Score

Description

Compute the cross-validation score of Rice and Silverman (1991) for the local polynomial estimation of a mean function.

Usage

cv.score(bandwidth, x, y, degree)

Arguments

bandwidth  the kernel bandwidth smoothing parameter.
x        numeric vector of x data. This observation grid must be uniform and missing values are not accepted.
y        matrix or data frame with functional observations (= curves) stored in rows. The number of columns of y must match the length of x. Missing values are not accepted.
degree  degree of local polynomial used.

Details

The cross-validation score is obtained by leaving out one curve at a time and computing the prediction error of the local polynomial smoother based on all other curves. For a bandwith value $h$, this score is

$$S(h) = \frac{1}{np} \sum_{i=1}^{n} \sum_{j=1}^{p} \left( Y_{ij} - \hat{\mu}^{-\langle i \rangle}(t_j; h) \right)^2,$$

where $Y_{ij}$ is the measurement of the $i$-th curve at time $t_j$ for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, and $\hat{\mu}^{-\langle i \rangle}(t_j; h)$ is the local polynomial estimator with bandwidth $h$ based on all curves except the $i$-th.

Value

the cross-validation score.

References


See Also

cv.select
Examples

```r
## Artificial example
x <- seq(0, 1, len = 100)
mu <- x + .2 * sin(2 * pi * x)
y <- matrix(mu + rnorm(2000, sd = .25), 20, 100, byrow = TRUE)
h <- c(.005, .01, .02, .05, .1, .15)
cv <- numeric()
for (i in 1:length(h)) cv[i] <- cv.score(h[i], x, y, 1)
plot(h, cv, type = "l")

## Plasma citrate data
## Compare cross-validation scores and bandwidths
## for local linear and local quadratic smoothing
## Not run:
data(plasma)
time <- 8:21
h1 <- seq(.5, 1.3, .05)
h2 <- seq(.75, 2, .05)
cv1 <- sapply(h1, cv.score, x = time, y = plasma, degree = 1)
cv2 <- sapply(h2, cv.score, x = time, y = plasma, degree = 2)
plot(h1, cv1, type = "l", xlab = "Bandwidth (hour)", ylab = "CV score",
     main = "Cross validation for local polynomial estimation")
lines(h2, cv2, col = 2)
legend("topleft", legend = c("Linear", "Quadratic"), lty = 1, col = 1:2, cex = .9)

## Note: using local linear (resp. quadratic) smoothing
## with a bandwidth smaller than .5 (resp. .75) can result
## in non-definiteness or numerical instability of the estimator.
## End(Not run)
```

---

### cv.select

**Cross-Validation Bandwidth Selection for Local Polynomial Estimation**

**Description**

Select the cross-validation bandwidth described in Rice and Silverman (1991) for the local polynomial estimation of a mean function based on functional data.

**Usage**

```r
cv.select(x, y, degree, interval = NULL, ...)
```
cv.select

Arguments

- **x**: numeric vector of x data. x must be a uniform grid; missing values are not accepted.
- **y**: matrix or data frame with functional observations (= curves) stored in rows. The number of columns of y must match the length of x. Missing values are not accepted.
- **degree**: degree of local polynomial used.
- **interval**: numeric vector of length 2; the lower and upper bounds of the search interval.
- ...: additional arguments to pass to the optimization function *optimize*.

Details

The cross-validation score is obtained by leaving out one entire curve at a time and computing the prediction error of the local polynomial smoother based on all other curves. For a bandwidth value \( h \), this score is

\[
S(h) = \sum_{i=1}^{n} \sum_{j=1}^{p} (Y_{ij} - \hat{\mu}^{-i}(t_j; h))^2,
\]

where \( Y_{ij} \) is the measurement of the \( i \)-th curve at time \( t_j \), and \( \hat{\mu}^{-i}(t_j; h) \) is the local polynomial estimator with bandwidth \( h \) based on all curves except the \( i \)-th.

cv.select uses the standard R function *optimize* to optimize cv.score. If the argument interval is not specified, the lower bound of the search interval is by default \((x[2] - x[1])/2 \) if degree < 2 and \( x[2] - x[1] \) if degree \( >= 2 \). The default value of the upper bound is \((\max(x) - \min(x))/2 \). These values guarantee in most cases that the local polynomial estimator is well defined. It is often useful to plot the function to be optimized for a range of argument values (grid search) before applying a numerical optimizer. In this way, the search interval can be narrowed down and the optimizer is more likely to find a global solution.

Value

a bandwidth that minimizes the cross-validation score.

References


See Also

- cv.score, plugin.select

Examples

```r
## Not run:
## Plasma citrate data
## Compare cross-validation scores and bandwidths
## for local linear and local quadratic smoothing
```
data(plasma)
time <- 8:21

## Local linear smoothing
cv.select(time, plasma, 1) # local solution h = 3.76, S(h) = 463.08
cv.select(time, plasma, 1, interval = c(.5, 1)) # global solution = .75, S(h) = 439.54

## Local quadratic smoothing
cv.select(time, plasma, 2) # global solution h = 1.15, S(h) = 432.75
cv.select(time, plasma, 2, interval = c(1, 1.5)) # same

## End(Not run)

### Phoneme data

<table>
<thead>
<tr>
<th>Code</th>
<th>Phoneme</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;sh&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;iy&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&quot;dcl&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;aa&quot;</td>
</tr>
<tr>
<td>5</td>
<td>&quot;ao&quot;</td>
</tr>
</tbody>
</table>

## Description

Log-periodograms of five phonemes ("sh" as in "she", "dcl" as in "dark", "iy" as the vowel in "she", "aa" as the vowel in "dark", and "ao" as the first vowel in "water") spoken by a sample of ~50 male speakers. For each phoneme, 400 log-periodograms are observed on a uniform grid of 150 frequencies.

## Usage

data(phoneme)

## Format

A data frame with 2000 rows and 151 columns. Each row contains a discretized log-periodogram (150 first columns) followed by a phoneme indicator (last column) coded as follows:

## Source

This dataset was created by the STAPH group in Toulouse, France. [http://www.math.univ-toulouse.fr/staph/npfda/](http://www.math.univ-toulouse.fr/staph/npfda/).

## Examples

## Not run:
Plasma citrate data

Description

Plasma citrate concentrations measured on 10 human subjects on the same day. The measurements on an individual were taken each hour from 8:00AM to 9:00PM. A possible statistical analysis is to estimate the population mean plasma citrate concentration as a function of time of day.

Usage

data(plasma)

Format

A matrix with 10 rows (corresponding to subjects) and 14 columns (corresponding to hours).

Source


Examples

```r
## Not run:
data(plasma)
matplot(x = 8:21, y = t(plasma), cex = .75, type = "l", col = 1, lty = 1,
lwd = .5, xlab = "Time of day (hour)", ylab = "Concentration",
main = "Plasma citrate data")
```
plot.SCBand

Plot a SCBand Object

Description

plot method for class "SCBand".

Usage

## S3 method for class 'SCBand'
plot(x, y = NULL, xlim = NULL, ylim = NULL, main = NULL, xlab = NULL,
ylab = NULL, col = NULL, cex = NULL, pch = NULL, lty = NULL, lwd = NULL,
legend = TRUE, where = NULL, text = NULL, legend.cex = NULL, horiz = TRUE,
bty = "n", ...)  

Arguments

x a SCBand object, typically result of a call to scb.mean, scb.model, or scb.equal.
y optional y data.
xlim the x limits (x1, x2) of the plot.
ylim the y limits of the plot.
main an overall title for the plot.
xlab a title for the x axis.
ylab a title for the y axis.
col the colors for lines and points.
cex a numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default.
pch a vector of plotting characters or symbols: see points.
lty a vector of line types, see par.
lwd a vector of line widths, see par.
legend logical; if TRUE, a legend is added to the plot.
where a character vector indicating the legend location: "bottomright", "bottom",
"bottomleft", "left", "topleft", "top", "topright", "right" or "center".
text a character vector to appear in the legend.
legend.cex character expansion factor relative to current par("cex") for the legend.
horiz logical; if TRUE, the legend is displayed horizontally rather than vertically.
bty the type of box to be drawn around the legend. The allowed values are "n" (the default) and "o".
... additional arguments passed to the function matplot for displaying the y data.
Details

The argument \( y \) can be used to plot subsets of the \( y \) data. If non null, this argument has priority over the component \( x \$ y \) for plotting.

The graphical parameters \( \texttt{col}, \texttt{cex}, \texttt{pch}, \texttt{lty}, \texttt{and} \texttt{lwd} \) apply to the following components to be plotted: data, parametric estimate, nonparametric estimate(s), normal simultaneous confidence bands (SCB), and bootstrap SCB. More precisely, \( \texttt{cex} \) and \( \texttt{pch} \) must be specified as vectors of length equal to the number of \( y \) data sets to be plotted (0, 1, or 2); \( \texttt{lty} \) and \( \texttt{lwd} \) must specified as numeric vectors of length equal to the total number of estimates and SCB components; \( \texttt{col} \) applies to all components and should be specified accordingly. If necessary, graphical parameters are recycled to match the required length.

By default a legend is plotted horizontally at the bottom of the graph.

Examples

```r
## Not run:
## Plasma citrate data
time <- 8:21
data(plasma)
h <- cv.select(time, plasma, degree = 1, interval = c(.5, 1))
scbplasma <- scb.mean(time, plasma, bandwidth = h, scbtype = "both",
                      gridsize = 100)
plot(scbplasma, cex = .2, legend.cex = .85, xlab = "Time of day",
ylab = "Concentration", main = "Plasma citrate data")
## End(Not run)
```

### plrt.model

**Pseudo-Likelihood Ratio Test for Models of the Mean Function**

**Description**

Implement the Pseudo-Likelihood Ratio Test (PLRT) of Azzalini and Bowman (1993) with functional data. The test is used to assess whether a mean function belongs to a given finite-dimensional function space.

**Usage**

```r
plrt.model(x, y, model, verbose = FALSE)
```

**Arguments**

- \( x \):
  - A numeric vector of \( x \) data. \( x \) must be a uniform grid; missing values are not accepted.

- \( y \):
  - A matrix or data frame with functional observations (= curves) stored in rows. The number of columns of \( y \) must match the length of \( x \). Missing values are not accepted.
model an integer specifying the degree of a polynomial basis, or a data frame/matrix containing the basis functions stored in columns. In the latter case, the basis functions must be evaluated at \( x \).

verbose logical; if TRUE, information on the candidate model, kernel bandwidth, test statistic, and p value is printed; otherwise no print.

Value

the p value of the PLRT.

References


See Also

*scb.model*

Examples

```r
## Example: Gaussian process with mean = linear function + bump
## and Onstein-Uhlenbeck covariance. The bump is high in the y
## direction and narrow in the x direction. The SCB and PLRT
## tests are compared.

# The departure from linearity in the mean function is strong
# in the supremum norm (SCB test) but mild in the euclidean norm
# (PLRT). With either \( n = 20 \) or \( n = 100 \) curves, the SCB test
# strongly rejects the incorrect linear model for the mean
# function while the PLRT retains it.

p <- 100      # number of observation points
x <- seq(0, 1, len = p)
mu <- -1 + 1.5 * x + 0.2 * dnorm(x, .6, .02)
plot(x, mu, type = "l")
R <- (.25)^2 * exp(20 * log(.9) * abs(outer(x,x,"-")))  # covariance
eigR <- eigen(R, symmetric = TRUE)
simR <- eigR$vectors %*% diag(sqrt(eigR$values))

n <- 20
set.seed(100)
y <- mu + simR %*% matrix(rnorm(n*p), p, n)
y <- t(y)
points(x, colMeans(y))
h <- cv.select(x, y, 1)
scb.model(x, y, 1, bandwidth = h)  # p value: <1e-16
plrt.model(x, y, 1, verbose = TRUE)  # p value: .442
n <- 100
y <- mu + simR %*% matrix(rnorm(n*p), p, n)
y <- t(y)
h <- cv.select(x, y, 1)
```
scb.model(x, y, 1, bandwidth = h)  # p value: <1e-16
plrt.model(x, y, 1, verbose = TRUE)  # p value: .456

---

**Description**

Select the plug-in bandwidth described in Benhenni and Degras (2011) for the local polynomial estimation of a mean function and its first derivative based on functional data.

**Usage**

```r
plugin.select(x, y, drv = 0L, degree = drv + 1L)
```

**Arguments**

- `x` numeric vector of x data. This observation grid must be uniform and missing values are not accepted.
- `y` matrix or data frame with functional observations (= curves) stored in rows. The number of columns of `y` must match the length of `x`. Missing values are not accepted.
- `drv` order of the derivative to estimate. Must be 0 or 1.
- `degree` degree of local polynomial used. Must equal `drv` or `drv+1`.

**Details**

The plug-in method should not be used with small data sets, since it is based on asymptotic considerations and requires reasonably accurate estimates of derivatives of the mean and covariance functions. Both the number of observed curves and observation points should be moderate to large. The plug-in bandwidth is designed to minimize the asymptotic mean integrated squared estimation error

$$AMISE(h) = \int (\mu(x) - \hat{\mu}(x; h))^2 dx,$$

where $\mu(x)$ is the mean function and $\hat{\mu}(x; h)$ is a local polynomial estimator with kernel bandwidth $h$. The expression of the plug-in bandwidth can be found in Benhenni and Degras (2011).

**Value**

the plug-in bandwidth.

**References**

See Also

`cv.select`

Examples

```r
## Not run:
## Phoneme data
data(phoneme)
classes <- phoneme[,151]
phoneme <- phoneme[,1-151]
freq <- 1:150
plugin.bandwidth <- numeric(5)
cv.bandwidth <- numeric(5)  # compare with cross-validation
for (i in 1:5) {
  plugin.bandwidth[i] <- plugin.select(x = freq, y = phoneme[classes == i,],
                      drv = 0, degree = 1)
  cv.bandwidth[i] <- cv.select(x = freq, y = phoneme[classes == i,],
                      degree = 1)
}
round(cbind(plugin.bandwidth, cv.bandwidth), 4)

## End(Not run)
```

---

**print.SCBand**  
*Print a SCBand Object*

**Description**

`print` method for class "SCBand".

**Usage**

```r
## S3 method for class 'SCBand'
print(x,...)
```

**Arguments**

- `x`  
  an object of class "SCBand".

- `...`  
  for compatibility with the generic `print` method; argument not currently used.

**Details**

The function `print.SCBand` concisely displays the information of an object of class "SCBand". More precisely it shows the data range, bandwidth used in local polynomial estimation, and key information on SCB and statistical tests.
**scb.equal**

**Compare Two Mean Functions**

**Description**

This two-sample test builds simultaneous confidence bands (SCB) for the difference between two population mean functions and retains the equality assumption if the null function is contained in the bands. Equivalently, SCB are built around one of the local linear estimates (the one for say, population 1), and the equality hypothesis is accepted if the other estimate (the one for population 2) lies within the bands.

**Usage**

```r
scb.equal(x, y, bandwidth, level = 0.05, scbtype = c("normal", "bootstrap", "both", "no"), gridsize, keep.y = TRUE, nrep = NULL, nboot = NULL, parallel = c("no", "multicore", "snow"), ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

**Arguments**

- **x**: a list of length 2 containing numeric vectors of x data; if the two samples are observed on the same grid, x can be specified as a single vector. x must be a uniform grid; missing values are not accepted.
- **y**: a list of length 2 containing matrices or data frames with functional observations (= curves) stored in rows. The number of columns of each component of y must match the length of the corresponding x. Missing values are not accepted.
- **bandwidth**: the kernel bandwidth smoothing parameters (numeric vector of length 2).
- **level**: the significance level of the test (default = .05).
- **scbtype**: the type of simultaneous confidence bands to build: "normal", "bootstrap", "both", or "no".

**Examples**

```r
## Not run:
# Plasma citrate data
data(plasma)
time <- 8:21
h <- cv.select(time, plasma, 1)
scbplasma <- scb.mean(time, plasma, bandwidth = h, scbtype = "both", gridsize = 100)
scbplasma
## End(Not run)
```
**scb.equal**

- `gridsize`: the size of the grid over which the mean function is to be estimated. Defaults to the length of `x`.
- `keep.y`: logical; if TRUE, keep `y` in the result.
- `nrep`: the number of replicates for the normal SCB method (default = 20,000).
- `nboot`: the number of replicates for the bootstrap SCB method (default = 5,000).
- `parallel`: the computation method for the bootstrap SCB. By default, computations are sequential ("no"). The function `boot` is used and can be run in parallel using the package `parallel`. Both options "multicore" and "snow" are available for parallel computing.
- `ncpus`: the number of cores to use for parallel computing if parallel = "multicore".
- `cl`: the name of the cluster to use for parallel computing if parallel = "snow".

**Value**

A list object of class "SCBand". Depending on the function used to create the object (scb.mean, scb.model, or scb.equal), some of its components are set to NULL. For scb.mean, the object has components:

- `x`: the argument `x`.
- `y`: if keep.y is TRUE, the argument `y`, else NULL.
- `call`: the function call.
- `model`: NULL.
- `par`: NULL.
- `nonpar`: a list of two local linear estimates, one for each population.
- `bandwidth`: the argument `bandwidth`.
- `degree`: the degree of local polynomial used. Currently, only local linear estimation is supported.
- `level`: the argument `level`.
- `scbtype`: the argument `scbtype`.
- `teststat`: the test statistic.
- `pnorm`: the p value for the normal-based statistical test.
- `pboot`: the p value for the bootstrap-based statistical test.
- `qnorm`: the quantile used to build the normal SCB.
- `qboot`: the quantile used to build the bootstrap SCB.
- `normscb`: a matrix containing the normal SCB stored in columns.
- `bootscb`: a matrix containing the bootstrap SCB stored in columns.
- `gridsize`: the argument `gridsize`, or `length(x)` if no argument was specified.
- `nrep`: the argument `nrep`.
- `nboot`: the argument `nboot`.

Depending on the value of `scbtype`, some or all of the fields `pnorm`, `qnorm`, `normscb`, `nrep`, `pboot`, `qboot`, `normboot` and `nboot` may be NULL.
References


See Also

`scb.mean`, `scb.model`

Examples

```r
## Not run:
# Phoneme data: compare the mean log-periodograms
# for phonemes "aa" as the vowel in "dark" and "ao"
# as the first vowel in "water"
data(phoneme)
n <- nrow(phoneme)
N <- ncol(phoneme)
classes <- split(1:n, phoneme[,N])
names(classes) <- c("sh", "iy", "dcl", "aa", "ao")
freq <- 1:150
compare.aa.ao <- scb.equal(freq, list(phoneme[classes$aa,-N],
                                 phoneme[classes$ao,-N]), bandwidth = c(.75, .75),
                              scbtype = "both", nboot = 2e3)
summary(compare.aa.ao)
```

---

`scb.mean`  
*Build Simultaneous Confidence Bands for Mean Functions*

### Description

Fit a local linear estimator and build simultaneous confidence bands (SCB) for the mean of functional data.

### Usage

```r
scb.mean(x, y, bandwidth, level = 0.95, scbtype = c("normal", "bootstrap",
                      "both", "no"), gridsize, keep.y = TRUE, nrep = NULL, nboot = NULL,
       parallel = c("no", "multicore", "snow"), ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

### Arguments

- **x**: a numeric vector of x data. x must be a uniform grid; missing values are not accepted.
- **y**: a matrix or data frame with functional observations (= curves) stored in rows. The number of columns of y must match the length of x. Missing values are not accepted.
bandwidth the kernel bandwidth smoothing parameter.
level the level of the simultaneous confidence bands.
scbtype the type of simultaneous confidence bands to build: "normal", "bootstrap", "both", or "no".
gridsize the size of the grid used to evaluate the mean function estimates and SCB. Defaults to length(x).
keep.y logical; if TRUE, keep y in the result.
nrep number of replicates for the Gaussian SCB method (20,000 by default).
nboot number of replicates for the bootstrap SCB method (5,000 by default).
parallel the computation method for the SCB. By default, computations are sequential ("no"). The bootstrap method uses function boot and can be run in parallel using the package parallel. In this case both options "multicore" and "snow" are available.
cpus number of cores to use for parallel computing when parallel = "multicore".
c1 name of the cluster to use for parallel computing when parallel = "snow".

Details

The local polynomial fitting uses a standard normal kernel and is implemented via the locpoly function. Bootstrap SCB are implemented with the boot function and the typically require more computation time than normal SCB. If parallel computing is used while running R from the console, errors may occur. It may be safer to perform parallel computations using R from the command line.

Value

An object of class "SCBand". To accommodate the different functions creating objects of this class (scb.mean, scb.model, and scb.equality), some components of the object are set to NULL. The component list is:

x the x data.
y the y data if keep.y is TRUE, else NULL.
call the function call.
model NULL.
par NULL.
nonpar a nonparametric estimate.
bandwidth the argument bandwidth.
degree the degree of local polynomial used. Currently, only local linear estimation is supported.
level the argument level.
scbtype the argument type.
teststat NULL.
pnorm NULL.
pboot NULL.
the quantile used to build the normal SCB.
qboot the quantile used to build the bootstrap SCB.
normscb a matrix containing the normal SCB stored in columns.
bootscb a matrix containing the bootstrap SCB stored in columns.
gridsize the argument gridsize if nonnull, else length(x).
nrep the argument nrep.
nboot the argument nboot.

Depending on the value of scbtype, some of the fields qnorm, normscb, nrep, qboot, normboot and nboot may be NULL.

References


See Also

`scb.equal, scb.model`

Examples

```r
## Not run:
## Plasma citrate data
data(plasma)
time <- 8:21  
h <- cv.select(time, plasma, 1, c(.5, 1))
scbplasma <- scb.mean(time, plasma, bandwidth = h, scbtype = "both", gridsize = 100)
plot(scbplasma, cex = .2, legend.cex = .85, xlab = "Time", ylab = "Concentration", 
     main = "Plasma citrate data")
```

---

### Goodness-Of-Fit of a Model for the Mean Function

This is the goodness-of-fit test for parametric models of the mean function described in Degras (2011). The candidate model must be a finite-dimensional function space (multiple linear regression). The test is based on the sup-norm distance between a smoothed parametric estimate and a local linear estimate. Graphically, the candidate model is retained whenever one of the estimates lies within the SCB built around the other.
Usage

```r
scb.model(x, y, model, bandwidth, level = 0.05, scbtype = c("normal", "bootstrap", "both", "no"), gridsize, keep.y = TRUE, nrep = NULL, nboot = NULL, parallel = c("no", "multicore", "snow"), ncpus = getOption("boot.ncpus", 1L), cl = NULL)
```

Arguments

- **x**: a numeric vector of x data. x must be a uniform grid; missing values are not accepted.
- **y**: a matrix or data frame with functional observations (= curves) stored in rows. The number of columns of y must match the length of x. Missing values are not accepted.
- **model**: an integer specifying the degree of a polynomial basis, or a data frame/matrix containing the basis functions stored in columns. In the latter case, the basis functions must be evaluated on a uniform grid of size gridsize spanning the range of x.
- **bandwidth**: the kernel bandwidth smoothing parameter.
- **level**: the significance level of the test (default = .05).
- **scbtype**: the type of simultaneous confidence bands to build: "normal", "bootstrap", "both", or "no".
- **gridsize**: the size of the grid over which the mean function is to be estimated. Defaults to the length of x.
- **keep.y**: logical; if TRUE, keep y in the result.
- **nrep**: the number of replicates for the normal SCB method (default = 20,000).
- **nboot**: the number of replicates for the bootstrap SCB method (default = 5,000).
- **parallel**: the computation method for the bootstrap SCB. By default, computations are sequential ("no"). The function `boot` is used and can be run in parallel using the package `parallel`. Both options "multicore" and "snow" are available for parallel computing.
- **ncpus**: the number of cores to use for parallel computing when parallel = "multicore".
- **cl**: the name of the cluster to use for parallel computing when parallel = "snow".

Value

An object of class "SCBand". To accommodate the different functions creating objects of this class (scb.mean, scb.model, and scb.equality), some components of the object are set to NULL. The component list is:

- **x**: the argument x.
- **y**: the argument y if keep.y is TRUE, else NULL.
- **call**: the function call.
- **model**: the argument model.
- **par**: a smoothed parametric estimate.
nonpar a local linear estimate.
bandwidth the argument bandwidth.
degree the degree of the local polynomial. Currently, only local linear estimation is supported.
level the argument level.
scbtype the argument type.
teststat the test statistic.
pnorm the p value for the normal-based statistical test.
pboot the p value for the bootstrap-based statistical test.
qnorm the quantile used to build the normal SCB.
qboot the quantile used to build the bootstrap SCB.
normscb a matrix containing the normal SCB stored in columns.
bootscb a matrix containing the bootstrap SCB stored in columns.
gridsize the argument gridsize, or length(x) if no argument was specified.
nrep the argument nrep.
nboot the argument nboot.

Depending on the value of scbtype, some or all of the fields pnorm, qnorm, normscb, nrep, pboot, qboot, normboot and nboot may be NULL.

References

Statistica Sinica, 21, 1735–1765.

See Also

scb.equal, scb.mean

Examples

## Example from Degras (2011)
## Gaussian process with polynomial mean function
## and Ornstein-Uhlenbeck covariance function
## The SCB and PLRT tests are compared

set.seed(100)
p <- 100  # number of observation points
x <- seq(0, 1, len = p)
mu <- 10 * x^3 - 15 * x^4 + 6 * x^5  # mean
R <- (.25)^2 * exp(20 * log(.9) * abs(outer(x,x,"-")))  # covariance
eigR <- eigen(R, symmetric = TRUE)
simR <- eigR$vectors %*% diag(sqrt(eigR$values))

# Candidate model for mu: polynomial of degree <= 3
# This model, although incorrect, closely approximates mu.
# With n = 50 curves, the SCB and PLRT incorrectly retain the model.
# With n = 70 curves, both tests reject it.

```r
n <- 50
y <- mu + simR %*% matrix(rnorm(n*p), p, n)  # simulate data
y <- t(y)  # arrange the trajectories in rows
h <- cv.select(x, y, 1)
scb.model(x, y, 3, bandwidth = h)  # p value: .652
plrt.model(x, y, 3, verbose = TRUE) # p value: .450

n <- 70
y <- mu + simR %*% matrix(rnorm(n*p), p, n)
y <- t(y)
h <- cv.select(x, y, 1)
scb.model(x, y, 3, bandwidth = h)  # p value: .004
plrt.model(x, y, 3, verbose = TRUE) # p value: .001

# Correct model: polynomials of degree <= 5
scb.model(x, y, 5, bandwidth = h)  # p value: .696
plrt.model(x, y, 5, verbose = TRUE) # p value: .628
```

---

**summary.SCBand**

*Summarize a SCBand Object*

**Description**

summary method for class "SCBand"

**Usage**

```r
## S3 method for class 'SCBand'
summary(object, ...)
```

**Arguments**

- `object` an object of class "SCBand"
- `...` additional arguments; not currently used.

**Details**

The function `summary.SCBand` displays all fields of a SCBand object at the exception of `x`, `y`, `par`, `nonpar`, `normsclb`, and `bootscb` which are potentially big. It provides information on the function call, data, local polynomial fit, SCB, and statistical tests.

**See Also**

`plot.SCBand`, `print.SCBand`
Examples

## Not run:

```
## Plasma citrate data
data(plasma)
time <- 8:21
h <- cv.select(time, plasma, 1)
cbplasma <- cb.mean(time, plasma, bandwidth = h, cbtype = "both", gridsize = 100)
summary(cbplasma)
```

## End(Not run)
## Index

**Topic datasets**
- phoneme, 6
- plasma, 7

**Topic methods**
- plot.SCBand, 8
- print.SCBand, 12
- summary.SCBand, 20

**Topic package**
- SCBmeanfd-package, 2

**Topic print**
- print.SCBand, 12

<table>
<thead>
<tr>
<th>Function</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot</td>
<td>14, 16, 18</td>
</tr>
<tr>
<td>class</td>
<td>14, 16, 18</td>
</tr>
<tr>
<td>cv.score</td>
<td>3, 5</td>
</tr>
<tr>
<td>cv.select</td>
<td>3, 4, 12</td>
</tr>
<tr>
<td>locpoly</td>
<td>16</td>
</tr>
<tr>
<td>matplot</td>
<td>8</td>
</tr>
<tr>
<td>optimize</td>
<td>5</td>
</tr>
<tr>
<td>par</td>
<td>8</td>
</tr>
<tr>
<td>parallel</td>
<td>14, 16, 18</td>
</tr>
<tr>
<td>phoneme</td>
<td>6</td>
</tr>
<tr>
<td>plasma</td>
<td>7</td>
</tr>
<tr>
<td>plot.SCBand</td>
<td>8, 13, 20</td>
</tr>
<tr>
<td>print.model</td>
<td>9</td>
</tr>
<tr>
<td>plugin.select</td>
<td>5, 11</td>
</tr>
<tr>
<td>points</td>
<td>8</td>
</tr>
<tr>
<td>print.SCBand</td>
<td>12, 20</td>
</tr>
<tr>
<td>scb.equal</td>
<td>13, 17, 19</td>
</tr>
<tr>
<td>scb.mean</td>
<td>15, 15, 19</td>
</tr>
<tr>
<td>scb.model</td>
<td>10, 15, 17, 17</td>
</tr>
<tr>
<td>SCBmeanfd</td>
<td>2</td>
</tr>
<tr>
<td>SCBmeanfd-package</td>
<td>2</td>
</tr>
<tr>
<td>summary.SCBand</td>
<td>13, 20</td>
</tr>
</tbody>
</table>