Package: SCBmeanfd (via r-universe)

August 22, 2024

Type Package

Title Simultaneous Confidence Bands for the Mean of Functional Data

Version 1.2.2

Date 2016-12-22

Author David Degras

Maintainer David Degras <ddegrasv@gmail.com>

Description Statistical methods for estimating and inferring the mean of functional data. The methods include simultaneous confidence bands, local polynomial fitting, 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 https://CRAN.R-project.org/package=SCBmeanfd

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-27 14:15:41

Contents

`SCBmeanfd-package` *Simultaneous Confidence Bands for Mean Functions*

Description

Statistical methods for estimating and inferring the mean of functional data. The methods include simultaneous confidence bands, local polynomial fitting, bandwidth selection by plug-in and cross-validation, goodness-of-fit tests for parametric models, equality tests for two-sample problems, and plotting functions.

Details

Package:	SCBmeanfd
Type:	Package
Version:	1.2.1
Date:	2016-09-24
License:	GPL-3

Author(s)

David Degras <ddegrasv@gmail.com>

References

Azzalini, A. and Bowman, A. (1993). On the use of nonparametric regression for checking linear relationships. *Journal of the Royal Statistical Society. Series B* **55**, 549-557.

Benhenni, K. and Degras, D. (2014). Local polynomial estimation of the mean function and its derivatives based on functional data and regular designs. *ESAIM: Probability and Statistics* **18**, 881-899.

Degras, D. (2011). Simultaneous confidence bands for nonparametric regression with functional data. *Statistica Sinica* **21**, 1735-1765.

Rice, J. A. and Silverman, B. W. (1991). Estimating the mean and covariance structure nonparametrically when the data are curves. *Journal of the Royal Statistical Society. Series B* **53**, 233-243.

`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 = 1, gridsize = length(x))
```

Arguments

`bandwidth`	kernel bandwidth.
`x`	observation points. 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 the local polynomial fit.
`gridsize`	size of evaluation grid for the smoothed data.

Details

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

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

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

If the `x` values are not equally spaced, the data are first smoothed and evaluated on a grid of length `gridsize` spanning the range of `x`. The smoothed data are then interpolated back to `x`.

Value

the cross-validation score.

References

Rice, J. A. and Silverman, B. W. (1991). Estimating the mean and covariance structure nonparametrically when the data are curves. *Journal of the Royal Statistical Society. Series B (Methodological)*, **53**, 233–243.

See Also

cv.select

Examples

```
## 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", xlim = range(c(h1,h2)), ylim = range(c(cv1, cv2)),
  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

```
cv.select(x, y, degree = 1, interval = NULL, gridsize = length(x), ...)
```

Arguments

x	observation points. 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 the local polynomial fit.
interval	lower and upper bounds of the search interval (numeric vector of length 2).
gridsize	size of evaluation grid for the smoothed data.
...	additional arguments to pass to the optimization function optimize.

Details

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

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

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

If the x values are not equally spaced, the data are first smoothed and evaluated on a grid of length gridsize spanning the range of x. The smoothed data are then interpolated back to x.

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

Rice, J. A. and Silverman, B. W. (1991). Estimating the mean and covariance structure nonparametrically when the data are curves. *Journal of the Royal Statistical Society. Series B (Methodological)*, **53**, 233–243.

See Also

cv.score, plugin.select

Examples

```
## 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 *Phoneme data*

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:

Code	Phoneme
1	"sh"
2	"iy"
3	"dcl"
4	"aa"
5	"ao"

Source

This dataset was created by the STAPH group in Toulouse, France
http://www.math.univ-toulouse.fr/staph/npfda/.
The larger original dataset can be found at
http://statweb.stanford.edu/~tibs/ElemStatLearn/.

Examples

```
## Not run:
data(phoneme)
freq <- 1:150
classes <- phoneme[,151]
phoneme <- phoneme[,-151]
classnames <- c("sh", "iy", "dcl", "aa", "ao")

## Local linear fit to the mean log-periodogram for each phoneme
llfit <- mapply(locpoly, y = by(phoneme, classes, colMeans),
          MoreArgs = list(x = freq, bandwidth = 2, degree = 1,
          gridsize = length(freq)))
llfit.y <- matrix(unlist(llfit["y",]), 150, 5)
matplot(freq, llfit.y, type = "l", lty = 1, xlab = "Frequency (scaled)",
 ylab = "Log-intensity",
 main = "Local linear estimation\nof the population mean log-periodogram")
legend("topright", legend = classnames, col = 1:5, lty = 1)

## End(Not run)
```

---

plasma *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

Andersen, A. H., Jensen, E. B. and Schou, G. (1981). Two-way analysis of variance with correlated errors. *International Statistical Review* **49**, 153–157.

Hart, T. D. and Wehrly, T. E. (1986). Kernel regression estimation using repeated measurements data. *Journal of the American Statistical Association* **81**, 1080–1088.

Examples

```
## 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")
lines(8:21, colMeans(plasma), col = 2, lwd = 1.5)
legend("topright", col = 2, lty = 1, lwd = 1.5, legend = "Mean")
## End(Not run)
```

`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`	SCBand object, typically result of a call to `scb.mean`, `scb.model`, or `scb.equal`.
`y`	optional y data.
`xlim`	x limits of the plot (numeric vector of length 2).
`ylim`	y limits of the plot (numeric vector of length 2).
`main`	title of the plot.
`xlab`	label of the x axis.
`ylab`	label of the y axis.
`col`	colors for lines and points.
`cex`	scale of plotting characters and symbols relative to default (numerical vector).
`pch`	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`	legend location: `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` or `"center"`.
`text`	text of the legend (character vector).

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	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 col, cex, pch, lty, and 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, cex and pch must be specified as vectors of length equal to the number of y data sets to be plotted (0, 1, or 2); lty and lwd must specified as numeric vectors of length equal to the total number of estimates and SCB components; 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

```
## 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

```
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

Azzalini, A. and Bowman, A. (1993). On the use of nonparametric regression for checking linear relationships. *Journal of the Royal Statistical Society. Series B* **55**, 549-557.

See Also

scb.model

Examples

```
## 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
```

`plugin.select` *Plug-in Bandwidth Selection for Local Polynomial Estimation*

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

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

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`.
`gridsize`	the size of the grid over which the mean function is to be estimated. Defaults to `length(x)`.
`...`	additional arguments to pass to the optimizer of the CV score.

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

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

where $\mu(x)$ is the mean function and $\hat{\mu}(t;h)$ 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

Benhenni, K. and Degras, D. (2011). Local polynomial estimation of the average growth curve with functional data. http://arxiv.org/abs/1107.4058

See Also

cv.select

Examples

```
## Not run:
## Phoneme data
data(phoneme)
classes <- phoneme[,151]
phoneme <- phoneme[,-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

```
## 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.

See Also

`plot.SCBand`, `summary.SCBand`

Examples

```
## 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` *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

```
scb.equal(x, y, bandwidth, level = .05, degree = 1,
 scbtype = c("normal","bootstrap","both","no"), gridsize = NULL,
 keep.y = TRUE, nrep = 2e4, nboot = 1e4, parallel = c("no","multicore","snow"),
 ncpus = getOption("boot.ncpus",1L), cl = NULL)
```

Arguments

`x`	a list of length 2 or matrix with 2 columns containing the `x` values of each sample. If the two samples are observed on the same grid, `x` can be specified as a single vector. The range of `x` must be the same for each sample; 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 bandwidths (numeric vector of length 1 or 2).
level	the significance level of the test (default = .05).
degree	the degree of the local polynomial fit.
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 the smallest x grid.
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 boostrap-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

Degras, D. (2011). Simultaneous confidence bands for nonparametric regression with functional data. *Statistica Sinica*, **21**, 1735–1765.

See Also

scb.mean, scb.model

Examples

```
## 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)

## End(Not run)
```

---

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

```
scb.mean(x, y, bandwidth, level = .95, degree = 1,
 scbtype = c("normal","bootstrap","both","no"), gridsize = length(x),
 keep.y = TRUE, nrep = 2e4, nboot = 5e3, parallel = c("no", "multicore", "snow"),
 ncpus = getOption("boot.ncpus",1L), cl = NULL)
```

Arguments

x	a numeric vector of x data. 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.
degree	the degree of the local polynomial fit.
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.
ncpus	number of cores to use for parallel computing when parallel = "multicore".
cl	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 typically require more computation time than normal SCB.

Value

An object of class "SCBand". To accommodate the different functions creating objects of this class (scb.mean, scb.model, and scb.equal), 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`.
`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` 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

Degras, D. (2011). Simultaneous confidence bands for nonparametric regression with functional data. *Statistica Sinica*, **21**, 1735–1765.

See Also

`scb.equal`, `scb.model`

Examples

```
## 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)
scbplasma
plot(scbplasma, cex = .2, legend.cex = .85, xlab = "Time", ylab = "Concentration",
  main = "Plasma citrate data")

## End(Not run)
```

scb.model *Goodness-Of-Fit of a Model for the Mean Function*

Description

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 (curvilinear 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

```
scb.model(x, y, model, bandwidth, level = .05, degree = 1,
 scbtype = c("normal","bootstrap","both","no"), gridsize = length(x),
 keep.y = TRUE, nrep = 2e4, nboot = 5e3, 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).
degree	the degree of the local polynomial fit.
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 length(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.equal), 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 boostrap-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

Degras, D. (2011). Simultaneous confidence bands for nonparametric regression with functional data.
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.
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

```
## 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, normscb, 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)
scbplasma <- scb.mean(time, plasma, bandwidth = h, scbtype = "both", gridsize = 100)
summary(scbplasma)

## End(Not run)
```

Index