Package 'LPCM'

August 30, 2019

Type Package

Title Local Principal Curve Methods

Version 0.46-3

Date 2019-08-30

Author Jochen Einbeck and Ludger Evers

Maintainer Jochen Einbeck <jochen.einbeck@durham.ac.uk>

Depends R (>= 2.10)

Suggests scatterplot3d, lattice, dr

Description Fitting multivariate data patterns with local principal curves, including tools for data compression (projection) and measuring goodness-of-fit; with some additional functions for mean shift clustering.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2019-08-30 11:20:02 UTC

R topics documented:

`LPCM-package` *Local principal curve methods*

Description

Fitting multivariate data patterns with local principal curves, including tools for data compression (projection) and measuring goodness-of-fit; with some additional functions for mean shift clustering.

This package implements the techniques introduced in Einbeck, Tutz & Evers (2005), and successive related papers.

The main functions to be called by the user are

- `lpc`, for the estimation of the local centers of mass which describe the principal curve;
- `lpc.spline`, which is a smooth and fully parametrized cubic spline respresentation of the latter;
- `lpc.project`, which enables to compress data by projecting them orthogonally onto the curve;
- `lpc.coverage` and `Rc` for assessing goodness-of-fit;
- `lpc.self.coverage` for bandwidth selection;
- the generic `plot` and `print` methods for objects of class `lpc` and `lpc.spline`.

This package also contains some code for density mode detection ('local principal points') and mean shift clustering (as well as bandwidth selection in this context), which implements the methods presented in Einbeck (2011). See the help file for `ms` for details. These functionalities have been substantially improved with version 0.46-0.

Details

Package:	LPCM
Type:	Package
License:	GPL (>=2)
LazyLoad:	yes

Acknowledgements

Contributions (in form of pieces of code, or useful suggestions for improvements) by Jo Dwyer, Mohammad Zayed, and Ben Oakley are gratefully acknowledged.

Author(s)

Jochen Einbeck and Ludger Evers

Maintainer: Jochen Einbeck <jochen.einbeck@durham.ac.uk>

References

Einbeck, J., Tutz, G., & Evers, L. (2005): Local principal curves, Statistics and Computing 15, 301-313.

Einbeck, J., Evers, L., & Powell, B. (2010): Data compression and regression through local principal curves and surfaces, International Journal of Neural Systems, 20, 177-192.

Einbeck, J. (2011): Bandwidth selection for nonparametric unsupervised learning techniques – a unified approach via self-coverage. Journal of Pattern Recognition Research 6, 175-192.

See Also

pcurve, princurve

---

calspeedflow *Speed-flow data from Calfornia.*

Description

A 'fundamental diagram' with observations of speed and flow recorded from 9th of July 2007, 9am, to 10th of July 2007, 10pm, on Line 5 of the Californian Freeway SR57-N, VDS number 1202263. The data were originally measured in intervals of thirty seconds, and then aggregated over intervals of 5 minutes length.

Usage

```
data(calspeedflow)
```

Format

A data frame with 444 observations on the following 4 variables.

`Date` a factor with levels `07/09/2007`... `07/10/2007`.

`Timestamp` a factor with a timestamps in intervals of five minutes.

`Lane5Flow` a numeric vector of vehicle flow in vehicles per 5 minutes.

`Lane5Speed` a numeric vector of vehicle speed in miles per hour.

Source

Retrieved from PeMS.

References

Einbeck, J., and Dwyer, J. (2011). Using principal curves to analyze traffic patterns on freeways. Transportmetrica 7, 229-246.

Examples

```
data(calspeedflow)
plot(calspeedflow[,3:4])
```

---

coverage *Coverage and self-coverage plots.*

Description

These functions compute coverages and self-coverages, and produce corresponding plots, for any principal curve object. The former may be used as goodness-of-fit measres, and the latter for for bandwidth selection.

Usage

```
coverage.raw(X, vec, tau, weights=1, plot.type="p", print=FALSE,
      label=NULL,...)

coverage(X, vec, taumin=0.02, taumax, gridsize=25, weights=1,
      plot.type="o", print=FALSE,...)

lpc.coverage(object, taumin=0.02, taumax, gridsize=25, quick=TRUE,
      plot.type="o", print=FALSE, ...)

lpc.self.coverage(X,  taumin=0.02, taumax=0.5,   gridsize=25, x0=1,
     way = "two", scaled=1,  weights=1, pen=2, depth=1,
     control=lpc.control(boundary=0, cross=FALSE),   quick=TRUE,
     plot.type="o", print=FALSE, ... )

ms.self.coverage(X, taumin=0.02, taumax=0.5, gridsize=25,
       thr=0.001, scaled=1, cluster=FALSE, plot.type="o",
       print=FALSE, ...)

select.self.coverage(self,  smin, plot.type="o", plot.segments=NULL)
```

Arguments

X	a $N \times d$ data matrix.
object	An object of type lpc, lpc.spline or ms.
vec	A matrix with d columns. The rows contain the points which make up the fitted object.
tau	tube size.
taumin	Minimal tube size.
taumax	Maximal tube size.
weights	An optional vector of weights. If weights are specified, then the coverage is the weighted mean of the indicator functions for falling within the tube. The function lpc.coverage does not have a weights argument, as it extracts the weights from the $weights component of the fitted object.
label	Experimental option; don't use.
gridsize	The number of different tube sizes to consider.
quick	If TRUE, an approximate coverage curve is provided by computing distances between data points and the curve through the closest local centers or mass; whereas with FALSE we use the distances of the points when projected orthogonally onto the spline representation of the local principal curve. The latter takes considerably more computing time. The resulting coverage curves are generally very similar, but the quick version may deliver little spurious peaks occasionally.
thr	adjacent mean shift clusters are merged if their relative distance falls below this threshold.
cluster	if TRUE, distances are always measured to the cluster to which an observation is assigned, rather than to the nearest cluster.
self	An object of class self, or a matrix with two colums providing a self-coverage curve.
smin	Minimum coverage for bandwidth selection. Default: 1/3 for clustering, 2/3 for principal curves.
plot.type	If set to 0, no plotted output is given. Otherwise, an appropriate plot is provided, using the plotting type as specified.
plot.segments	A list with default list(lty=c(1,2,3),lwd=c(2,1,1),lcol=c(3,3,3)) which specifies how (and how many) bandwidth candidates, in order of decreasing negative second derivative of self-coverage, are to be highlighted.
print	If TRUE, coverage values are printed on the screen as soon as computed. This is quite helpful especially if gridsize is large.
x0, way, scaled, pen, depth, control	Auxiliary parameters as outlined in lpc, lpc.control, and ms.
...	Optional graphical parameters passed to the corresponding plotting functions.

Details

The function `coverage.raw` computes the coverage, i.e. the proportion of data points lying inside a circle or band with radius τ, for a fixed value `tau`. The whole coverage curve $C(\tau)$ is constructed through function `coverage`.

Functions `coverage.raw` and `coverage` can be used for any object fitted by an unsupervised learning technique (for instance, HS principal curves, or even clustering algorithms), while the functions prefixing with `lpc.` and `ms.` can only be used for the corresponding objects. The functions `lpc.coverage` and `ms.coverage` are wrappers around `coverage` which operate directly a fitted object, rather than a data matrix.

Function `select.self.coverage` extracts suitable bandwidths from the self-coverage curve, and produces a plot. The function is called from within `lpc.self.coverage` or `ms.self.coverage` but can also be called directly by the user (for instance, if the graphical output is to be reproduced, or if the minimum coverage `smin` is to be modified). The component `$select` contains the selected candidate bandwidths, in the order of strength of evidence provided by the self-coverage criterion (the best bandwidth comes first, etc.). A plot is produced as a by-product, which symbolizes the best bandwidth by a thick solid line, the second-best by a dashed line, and the third-best by a dotted line. It is recommended to run the self-coverage functions with fixed starting points, as in the examples below, and to scale by the range only.

See Einbeck (2011) for details. Note that the original publication by Einbeck, Tutz, and Evers (2005) uses 'quick' coverage curves.

Value

A list of items, and a plot (unless `plot.type=0`).

The functions `lpc.self.coverage` and `ms.self.coverage` produce an object of class `self`. The component `$select` recommends suitable bandwidths for the use in `lpc`, in the order of strength of evidence. These correspond to points of strong negative curvature (implemented via second differences) of the self-coverage curve.

Author(s)

J. Einbeck

References

Einbeck, J., Tutz, G., & Evers, L. (2005). Local principal curves. Statistics and Computing 15, 301-313.

Einbeck, J. (2011). Bandwidth selection for mean-shift based unsupervised learning techniques: a unified approach via self-coverage. Journal of Pattern Recognition Research 6, 175-192.

See Also

`lpc`, `ms`

Examples

```
data(faithful)
mfit <- ms(faithful)
coverage(mfit$data, mfit$cluster.center, gridsize=18)

## Not run:
f.self <- ms.self.coverage(faithful,gridsize= 50, taumin=0.1, taumax=0.5, plot.type="o")
h <- select.self.coverage(f.self)$select
mfit2 <- ms(faithful,h=h[2]) # using `second-best' suggested bandwidth

## End(Not run)

## Not run:
data(gvessel)
g.self <-lpc.self.coverage(gvessel[,c(2,4,5)], x0=c(35, 1870, 6.3), print=FALSE, plot.type=0)
h <- select.self.coverage(g.self)$select
g.lfit <- lpc(gvessel[,c(2,4,5)], h=h[1],  x0=c(35, 1870, 6.3))
lpc.coverage(g.lfit, gridsize=10, print=FALSE)

## End(Not run)
```

---

followx *Fit an individual branch of a local principal curve.*

Description

Internal function of package **LPCM** called by `lpc`. Do not use!

Usage

```
followx(Xi, x0, h, t0, iter, way, weights, pen = 2, phi = 1,
    lasteigenvector = 0, rho0 = 0.4, boundary=0.005,
    convergence.at= 0.000001, cross=TRUE)
```

Arguments

`Xi`

`x0`

`h`

`t0`

`iter`

`way`

`weights`

`pen`

`phi`
`lasteigenvector`

`rho0`
`boundary`
`convergence.at`
`cross`

Author(s)

JE

See Also

`lpc`

`gaia` *Gaia data*

Description

(Simulated) spectral decomposition of stellar objects, generated in the framework of the Gaia project.

Usage

```
data(gaia)
```

Format

A data frame with 8286 observations on the following 22 variables.

`ID` ID of the object

`metallicity` metallicity (abundance); that is proportion of matter other than hydrogen and helium relative to that of the sun.

`gravity` the surface gravity; that is acceleration due to gravity at the surface of the star.

`temperature` the 'effective' temperature (K); that is the temperature of the observable part of the stellar atmosphere.

`band1` photon counts in band 1

`band2` photon counts in band 2

`band3` photon counts in band 3

`band4` photon counts in band 4

`band5` photon counts in band 5

`band6` photon counts in band 6

band7 photon counts in band 7

band8 photon counts in band 8

band9 photon counts in band 9

band10 photon counts in band 10

band11 photon counts in band 11

band12 photon counts in band 12

band13 photon counts in band 13

band14 photon counts in band 14

band15 photon counts in band 15

band16 photon counts in band 16

Details

Gaia is an astrophysics mission of the European Space Agency (ESA) which will undertake a detailed survey of over 10^9 stars in our Galaxy and extragalactic objects. An important part of the scientific analysis of these data is the classification of all the objects as well as the estimation of stellar astrophysical parameters (effective stellar temperature, surface gravity, metallicity). This will be done on the basis of high-dimensional spectroscopic and astrometric data such as those ones given here.

More precisely, the spectral data come in form of photon counts ("fluxes") observed in (originally) 96 wavelength intervals ("bands"), see Bailer-Jones (2010) for more details. The data given here are a 16-dimensional subset created by binning/selecting from the 96 bands. The counts given here are standardized, i.e. they are divided by the total number of incoming photons over all filters (in other words, they add up to 1). Note that these data are simulated using computer models. The satellite which will collect the actual data will be launched in 2012.

The 16-d spectral data have been used in Einbeck, Evers and Bailer-Jones (2008) as well as Einbeck, Evers and Powell (2010) in order to predict the stellar temperature.

Source

Coryn Bailer-Jones (MPIA Heidelberg).

References

Bailer-Jones, C.A.L. (2010). The ILIUM forward modelling algorithm for multivariate parameter estimation and its application to derive stellar parameters from Gaia spectrophotometry, Monthly Notices of the Royal Astronomical Society, vol. 403, pp. 96-116.

Einbeck, J., Evers, L., and Bailer-Jones, C.A.L. (2008). Representing complex data using localized principal components with application to astronomical data. In: Gorban, A, Kegl, B, Wunsch, D, & Zinovyev, A: Principal Manifolds for Data Visualization and Dimension Reduction; Lecture Notes in Computational Science and Engineering 58, 180-204, ISSN/ISBN: 978-3-540-73749-0.

Einbeck, J., Evers, L., and Powell, B. (2010): Data compression and regression through local principal curves and surfaces, International Journal of Neural Systems, 20, 177-192.

Examples

```
data(gaia)
s <- sample(nrow(gaia),200)
library(lattice)
splom(gaia[s,5:20], cex=0.3,pscales=0)

gaia.pc <-  princomp(gaia[s,5:20])
temp <- gaia$temperature
tempcol     <- (temp[s]- min(temp[s]))/max(temp[s]- min(temp[s]))
library(scatterplot3d)
scatterplot3d(gaia.pc$scores[,c(2,1,3)], pch="+",
     color=rgb(sqrt(tempcol),0,1-sqrt(tempcol)))
     # This is a 3D scatterplot of the first three principal component scores;
     # with higher stellar temperatures shaded in red colour.
```

---

gvessel *North Atlantic Water Temperature Data.*

Description

These are observations taken over nine days in May 2000 by the German vessel Gauss in the North Atlantic.

Usage

```
data(gvessel)
```

Format

A data frame with 643 observations on the following 7 variables.

`day2g` an integer for the day at which the measurement was taken.

`salg` a numeric vector with measurements of salinity according to the PSS (Practical Salinity Scale).

`tempg` a numeric vector with measurements of water temperature in degrees Celsius.

`depthg` a numeric vector with the water depths (in meters) at which the measurements were taken.

`oxyg` a numeric vector with measurements of oxygen content (mm per litre of water)

`longg` longitude

`latg` latitude

Source

Retrieved by B. Powell from the World Ocean Database, http://www.nodc.noaa.gov/OC5/WOD/pr_wod.html.

References

Einbeck, J., Evers, L., and Powell, B. (2010): Data compression and regression through local principal curves and surfaces, International Journal of Neural Systems, 20, 177-192.

Examples

```
data(gvessel)
pairs(gvessel[,c(3,2,4,5)])
tcol <- (gvessel$tempg- min(gvessel$tempg))/(max(gvessel$tempg)- min(gvessel$tempg))
require(scatterplot3d)
scatterplot3d(gvessel[,2],gvessel[,4],gvessel[,5], color=rgb(tcol,0,1-tcol))
```

---

kernels.and.distances *Auxilary kernel and distance functions.*

Description

Internal **LPCM** functions which are normally not to be called by the user.

Usage

```
kern(y, x = 0, h = 1)
kernd(X, x, h)
kdex(X, x, h)
distancevector(X, y, d = "euclid", na.rm = TRUE)
vecdist(X,Y)
mindist(X,y)
enorm(x)
```

Arguments

x	a number or vector.
y	a vector.
h	a bandwidth.
X	a matrix.
Y	a matrix.
d	type of distance measure (only 'euclid').
na.rm	...

Details

kern specifies the base kernel (by default Gaussian) used in lpc ; kernd is the corresponding multivariate product kernel. kdex is a pointwise multivariate kernel density estimator.

distancevector makes use of function vdisseuclid from R package **hopach** (but that package does not need to be loaded). enorm is the Euclidean norm.

Author(s)

JE

References

Pollard, van der Laan, and Wall (2010). Hierarchical Ordered Partitioning and Collapsing Hybrid (HOPACH). R package **hopach** version 2.9.1.

---

lpc *Local principal curves*

Description

This is the main function which computes the actual local principal curve, i.e. a sequence of local centers of mass.

Usage

```
lpc(X, h, t0 = mean(h),  x0,  way = "two",  scaled = 1,
      weights=1, pen = 2, depth = 1, control=lpc.control())
```

Arguments

X	data matrix with N rows (observations) and d columns (variables).
h	bandwidth. May be either specified as a single number, then the same bandwidth is used in all dimensions, or as a d-dimensional bandwidth vector. If the data are scaled, then the bandwidth has to be specified in fractions of the data range or standard deviation, respectively, e.g. `scaled=1` and `h= c(0.2,0.1)` gives 20 percent of the range of the first variable and 10 percent of the range of the second variable. If left unspecified, then default settings are invoked; see the 'Notes' section below.
t0	scalar step length. Default setting is `t0=h`, if `h` is a scalar, and `t0=mean(h)`, if `h` is a vector.
x0	specifies the choice of starting points. The default choice `x0=1` will select one suitable starting point automatically (in form of a local density mode). The second built-in option `x0=0` will use all local density modes as starting points, hence produce as many branches as modes. Optionally, one can also set one or more starting points manually here. This can be done in form of a matrix, where each row corresponds to a starting point, or in form of a vector, where starting points are read in consecutive order from the entries of the vector. The starting point has always to be specified on the original data scale, even if `scaled>0`. A fixed number of starting points can be enforced through option `mult` in `lpc.control`.
way	"one": go only in direction of the first local eigenvector, "back": go only in opposite direction, "two": go from starting point in both directions.

scaled	if 1 (or TRUE), scales each variable by dividing through its range. If scaled=2, scaling is performed by dividing through the standard deviation (see also the Notes section below).
weights	a vector of observation weights (can also be used to exclude individual observations from the computation by setting their weight to zero.)
pen	power used for angle penalization (see [1]). If set to 0, the angle penalization is switched off.
depth	maximum depth of branches (ϕ_{max} in [2]), restricted to the values 1,2 or 3 (The original LPC branch has depth 1. If, along this curve, a point features a high second local PC, this launches a new starting point, and the resulting branch has depth 2. If, along this branch, a point features a high second local PC, this launches a new starting point, and the resulting branch has depth 3.)
control	Additional parameters steering particularly the starting-, boundary-, and convergence behavior of the fitted curve. See lpc.control.

Value

A list of items:

LPC	The coordinates of the local centers of mass of the fitted principal curve.
Parametrization	Curve parameters and branch labels for each local center of mass.
h	The bandwidth used for the curve estimation.
to	The constant t_0 used for the curve estimation.
starting.points	The coordinates of the starting point(s) used.
data	The data frame used for curve estimation.
scaled	the user-supplied value, could be boolean or numerical
weights	The vector of weights used for curve estimation.
control	The settings used in lpc.control()
Misc	Miscellannea.

Note

All values provided in the output refer to the scaled data, unless scaled=0 or (equivalently) scaled=FALSE. Use unscale to convert the results back to the original data scale.

The default option scaled=1 or scaled=TRUE scales the data by dividing each variable through their range (differing from the scaling through the standard deviation as common e.g. for PCA). The setting scaled=2, and in fact all other settings scaled>0, will scale the data by their standard deviation.

If scaled=1 or if no scaling is applied, then the default bandwidth setting is 10 percent of the data range in each direction. If the data are scaled through the standard deviation, then the default setting is 40 percent of the standard deviation in each direction.

Author(s)

J. Einbeck and L. Evers. See LPCM-package for further acknowledgements.

References

[1] Einbeck, J., Tutz, G., & Evers, L. (2005). Local principal curves. Statistics and Computing 15, 301-313.

[2] Einbeck, J., Tutz, G., & Evers, L. (2005): Exploring Multivariate Data Structures with Local Principal Curves. In: Weihs, C. and Gaul, W. (Eds.): Classification - The Ubiquitous Challenge. Springer, Heidelberg, pages 256-263.

Examples

```
data(calspeedflow)
lpc1 <- lpc(calspeedflow[,3:4])
plot(lpc1)

data(mussels, package="dr")
 lpc2 <- lpc(mussels[,-3], x0=as.numeric(mussels[49,-3]),scaled=0)
 plot(lpc2, curvecol=2)

data(gaia)
s <- sample(nrow(gaia),200)
gaia.pc <-  princomp(gaia[s,5:20])
lpc3 <- lpc(gaia.pc$scores[,c(2,1,3)],scaled=0)
plot(lpc3, curvecol=2, type=c("curve","mass"))

# Simulated letter 'E' with branched LPC
ex<- c(rep(0,40), seq(0,1,length=20), seq(0,1,length=20), seq(0,1,length=20))
ey<- c(seq(0,2,length=40), rep(0,20), rep(1,20), rep(2,20))
sex<-rnorm(100,0,0.01); sey<-rnorm(100,0,0.01)
eex<-rnorm(100,0,0.1);  eey<-rnorm(100,0,0.1)
ex1<-ex+sex; ey1<-ey+sey
ex2<-ex+eex; ey2<-ey+eey
e1<-cbind(ex1,ey1); e2<-cbind(ex2,ey2)
lpc.e1 <- lpc(e1, h= c(0.1,0.1),  depth=2, scaled=0)
plot(lpc.e1, type=c("curve","mass", "start"))
```

`lpc.control` *Auxiliary parameters for controlling local principal curves.*

Description

This function bundles parameters controlling mainly the starting-, convergence-, boundary-, and stopping-behaviour of the local principal curve. It will be used only inside the `lpc()` function argument.

Usage

```
lpc.control(iter =100, cross=TRUE,
            boundary = 0.005, convergence.at = 0.00001,
            mult=NULL, ms.h=NULL, ms.sub=30,
            pruning.thresh=0.0, rho0=0.4)
```

Arguments

iter	Maximum number of iterations on either side of the starting point within each branch.
cross	Logical parameter. If FALSE, a curve is stopped when it comes too close to an another part of itself. Note: Even when cross=FALSE, different branches of the curve (for higher depth or multiple starting points) are still allowed to cross. This option only avoids crossing of each particular branch with itself. Used in the self-coverage functions to avoid overfitting.
boundary	This boundary correction [2] reduces the bandwidth adaptively once the relative difference of parameter values between two centers of mass falls below the given threshold. This measure delays convergence and enables the curve to proceed further into the end points. If set to 0, this boundary correction is switched off.
convergence.at	This forces the curve to stop if the relative difference of parameter values between two centers of mass falls below the given threshold. If set to 0, then the curve will always stop after exactly iter iterations.
mult	numerical value which enforeces a fixed number of starting points. If the number given here is larger than the number of starting points provided at x0, then the missing points will be set at random (For example, if $d = 2$, mult=3, and x0=c(58.5,17.8,80,20), then one gets the starting points (58.5, 17.8), (80,20), and a randomly chosen third one. Another example for such a situation is x0=NULL with mult=1, in which one random starting point is chosen). If the number given here is smaller the number of starting points provided at x0, then only the first mult starting points will be used.
ms.h	sets the bandwidth (vector) for the initial mean shift procedure which finds the local density modes, and, hence, the starting points for the LPC. If unspecified, the bandwidth h used in function lpc is used here too.
ms.sub	proportion of data points (default=30) which are used to initialize mean shift trajectories for the mode finding. In fact, we use min(max(ms.sub,floor(ms.sub*N/100)),10*ms.sub) trajectories.
pruning.thresh	Prunes branches corresponding to higher-depth starting points if their density estimate falls below this threshold. Typically, a value between 0.0 and 1.0. The setting 0.0 means no pruning.
rho0	A numerical value which steers the birth process of higher-depth starting points. Usually, between 0.3 and 0.4 (see reference [1]).

Value

A list of the nine specified imput parameters, which can be read by the control argument of the lpc function.

Author(s)

JE

References

[1] Einbeck, J., Tutz, G. & Evers, L. (2005): Exploring Multivariate Data Structures with Local Principal Curves. In: Weihs, C. and Gaul, W. (Eds.): Classification - The Ubiquitous Challenge. Springer, Heidelberg, pages 256-263.

[2] Einbeck, J. and Zayed, M. (2014). Some asymptotics for localized principal components and curves. Communications in Statistics - Theory and Methods 43, 1736-1749.

Examples

```
data(calspeedflow)
fit1 <- lpc(calspeedflow[,c(3,4)], x0=c(50,60),scaled=1,
   control=lpc.control(iter=20, boundary=0))
plot(fit1, type=c("curve","start","mass"))
```

---

lpc.project *Projection onto LPC*

Description

Projects a new observation onto the spline representation of the local principal curve.

Usage

```
lpc.project(object, newdata, ...)
```

Arguments

object	Object of class lpc or lpc.spline.
newdata	A data frame containing the new data to be projected.
...	Additional arguments to be passed to lpc.project.spline.

Value

closest.pi	Projection index of projected point(s) (in cubic spline parametrization).
closest.or.pi	Projection index of projected point(s) (in terms of the original LPC parametrization).
closest.coords	Coordinates of projected data point(s)
closest.dist	Euclidean distance between data point(s) and their projected counterpart(s).
closest.branch	ID of branch onto which the data point was projected (the IDs get allocated in the output component $Parametrization of function lpc).

Note

The parametrization of the cubic spline function is not exactly the same as that of the original LPC. The reason is that the latter uses Euclidean distances between centers of masses, while the former uses the arc length along the cubic spline. The differences are normally quite small, though.

Author(s)

J. Einbeck and L. Evers

References

Einbeck, J., Evers, L. & Hinchliff, K. (2010): Data compression and regression based on local principal curves. In A. Fink, B. Lausen, W. Seidel, and A. Ultsch (Eds), Advances in Data Analysis, Data Handling, and Business Intelligence, Heidelberg, pp. 701–712, Springer.

See Also

lpc, lpc.spline

Examples

```
data(gvessel)
gvessel.lpc <- lpc(gvessel[,c(2,4,5)], scaled=TRUE,   h=0.11,  x0=c(35, 1870, 6.3))
lpc.project(gvessel.lpc, newdata=data.frame(salg=35,dephtg= 2000,oxyg=6))
```

---

lpc.spline *Representing local principal curves through a cubic spline.*

Description

Fis a natural cubic spline component-wise through the series of local centers of mass. This provides a continuous parametrization in terms of arc length distance, which can be used to compute a projection index for the original or new data points.

Usage

```
lpc.spline(lpcobject, optimize = TRUE, compute.Rc=FALSE,
     project=FALSE, ...)
```

Arguments

lpcobject	Object of class lpc.
optimize	Boolean. If TRUE, optimize is used to find the point on the curve with minimum distance. Otherwise, data points are only projected onto the closest knot.
compute.Rc	Boolean. If TRUE, the goodness-of-fit measure R_C suggested in [1] is computed and returned (using the scaled data, if scaled=TRUE in lpcobject).
project	Boolean. If TRUE, projections onto curve are computed.
...	Additional arguments to be passed to lpc.project.spline

Details

See reference [2].

Value

`knots.pi`	LPC parameters (in cubic spline parametrization) at position of the knots of the spline function (these are not identical to the LPC mass points!)
`knots.coords`	Coordinates of the spline knots.
`closest.pi`	Parameter of the projected data points.
`closest.coords`	Coordinates of projected data points.
`closest.dist`	Euclidean distance between original and projected data point.
`closest.branch`	ID Number of the branch on which the data point was projected (the IDs are given in the output of function `lpc`).
`Rc`	Value of R_C.
`project`	repeats the input value of `project`.
`lpcobject`	returns the provided object `lpcobject`.
`splinefun`	returns the cubic spline function (generated by `lpc.splinefun`).

Warning

Careful with options `project` and `compute.Rc` - they can take rather long if the data set is large!

Note

The parametrization of the cubic spline function is not exactly the same as that of the original LPC. The reason is that the latter uses Euclidean distances between centers of masses, while the former uses the arc length along the cubic spline. However, the differences are normally quite small.

Author(s)

J. Einbeck and L. Evers

References

[1] Einbeck, J., Tutz, G., and Evers, L. (2005). Local principal curves. Statistics and Computing 15, 301-313.

[2] Einbeck, J., Evers, L. & Hinchliff, K. (2010): Data compression and regression based on local principal curves. In A. Fink, B. Lausen, W. Seidel, and A. Ultsch (Eds), Advances in Data Analysis, Data Handling, and Business Intelligence, Heidelberg, pp. 701–712, Springer.

See Also

`lpc`

Examples

```
data(gvessel)
gvessel.lpc <- lpc(gvessel[,c(2,4,5)],   h=0.11,  x0=c(35, 1870, 6.3))
gvessel.spline  <- lpc.spline(gvessel.lpc)
plot(gvessel.spline, lwd=2)
```

---

lpc.spline.auxiliary.functions

Auxiliary functions for spline fitting and projection.

Description

Internal functions of package **LPCM** called by `lpc.spline` and others. These will rarely be called directly by the user.

Usage

```
lpc.splinefun(lpcobject)

lpc.fit.spline(lpcsl, num.knots = 100)

lpc.spline.eval(lpcsl, or.pi, branch = 0)

lpc.project.spline(lpcsl, newdata, num.knots = 100, optimize = TRUE)

lpc.curve.length(lpcsl, or.pi, branch = 0, total.subdivisions = 10000,
      min.subdivisions = 100)
```

Arguments

`lpcobject`	Object of type `lpc`.
`lpcsl`	Object generated by `lpc.splinefun`.
`num.knots`	number of spline knots
`or.pi`	original projection index
`branch`	branch ID
`newdata`	new data frame
`optimize`	Boolean.
`total.subdivisions`	total number of subdivisions for arc length computation.
`min.subdivisions`	minimum number of subdivisions for arc length computation.

Author(s)

L. Evers and J. Einbeck

See Also

lpc.spline

---

ms *Mean shift clustering.*

Description

Functions for mean shift, iterative mean shift, and mean shift clustering. The main function is ms which, for a given bandwidth, detects the local modes ('local principal points') and performs the clustering.

Usage

```
meanshift(X, x, h)
ms.rep(X, x, h, thresh= 0.0001, iter=200)
ms(X, h, subset,  thr=0.01, scaled= 1, iter=200, plot=TRUE, ...)
```

Arguments

X	data matrix or vector.
h	scalar or vector-valued bandwidth (by default, 5 percent of the data range, or 20 percent of the standard deviation, respectively, in each direction). If set manually and scaled>0, this bandwidth needs to be set on the scaled scale; for instance setting scaled=1 and h=0.10 will use a bandwidth of 10 percent of the data range in either direction.
x	point from which we wish to shift to the local mean.
subset	vector specifying a subset of 1:n, where n is the sample size. This allows to run the iterative mean shift procedure only from a subset of points (if unspecified, 1:n is used here, i.e. each data point serves as a starting point).
scaled	if equal to 1 (default), each variable is divided by its range, and if equal to 2 (or any other positive value other than 1), each variable is divided by its standard deviation. If equal to 0, then no scaling is applied.
thresh, iter	mean shift iterations are stopped when the mean shift length (relative to the distance of of x to the overall mean; see Note section) falls below thresh, or after iter iterations (whatever event happens first).
thr	adjacent mean shift clusters are merged if their relative distance falls below this threshold (see Note section).
plot	if equal to 0, then no plotted output. For bivariate data, plot=1 gives by default a dynamically created color plot showing the mean shift trajectories and the resulting clustering.
...	further graphical parameters.

Details

The methods implemented here can be used for density mode estimation, clustering, and the selection of starting points for the LPC algorithm.

Chen (1995) showed that, if the mean shift is computed iteratively, the resulting sequence of local means converges to a mode of the estimated density function. By assigning each data point to the mode to which it has converged, this turns into a clustering technique.

The concepts of coverage and self-coverage, which were originally introduced in the principal curve context, adapt straightforwardly to this setting (Einbeck, 2011).

The goodness-of-fit messure `Rc` can also be applied in this context. For instance, a value of $R_C = 0.8$ means that, after the clustering, the mean absolute residual length has been reduced by 80% (compared to the distances to the overall mean).

Value

The main function `ms` produces an object of class `ms`, with components:

`cluster.center`	a matrix which gives the coordinates of the estimated density modes (i.e., of the mean-shift based cluster centers).
`cluster.label`	assigns each data point to the cluster center to which its mean shift trajectory has converged.
`closest.label`	assigns each data point to the closest cluster center in terms of Euclidean distance.
`data`	the data frame (scaled if `scaled=TRUE`).
`scaled`	the user-supplied value, could be boolean or numerical.
`scaled.by`	the data were scaled by dividing each variable through the values provided in this vector.

For all other functions, use `names()`.

Note

All values provided in the output refer to the scaled data, unless `scaled=0` or (equivalently) `scaled=FALSE`.

The default option `scaled=1` or `scaled=TRUE` scales the data by dividing each variable through their range (differing from the scaling through the standard deviation as common e.g. for PCA). All other settings `scaled>0` will scale the data by their standard deviation.

If `scaled=1` or if no scaling is applied, then the default bandwidth setting is 5 percent of the data range in each direction. If the data are scaled through the standard deviation, then the default setting is 20 percent of the standard deviation in each direction.

The threshold `thresh` for stopping mean shift iterations works as follows. At each iteration, we compare the length of the mean shift, that is the Euclidean distance between the point `x` and its local mean `m`, to the Euclidean distance between the point `x` and the overall data mean `M`. If this distance falls below `thresh`, the mean shift procedure is stopped.

The threshold `thr` for merging cluster centers works as follows: After identification of a new cluster center, we compute the Euclidean distance of the new center to (each) existing center, relative to the Euclidean distance of the existing center to the overall mean. If this distance falls below `thr`, then the new center is deemed identical to the old one. The default setting for the relation of the two thresholds is `thresh = thr^2`.

Author(s)

J. Einbeck. See LPCM-package for further acknowledgements.

References

Chen, Y. (1995). Mean Shift, Mode Seeking, and Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17, 790-799.

Einbeck, J. (2011). Bandwidth selection for mean-shift based unsupervised learning techniques: a unified approach via self-coverage. Journal of Pattern Recognition Research 6, 175-192.

See Also

Rc, plot.ms

Examples

```
data(faithful)
# Mean shift clustering with default bandwidth (5 percent of data range)
ms(faithful)
```

---

plot.lpc *Plotting local principal curves and mean shift trajectories*

Description

Takes an object of class lpc, lpc.spline or ms. In the case of principal curves, it plots any subset of the following components of the local principal curve: Centers of mass; the curve connecting the local centers of mass; the cubic spline representation of the curve; the projections onto the curve; the starting points. For the mean shift procedure, it produces a plot of mean shift trajectories and cluster centers.

Usage

```
## S3 method for class 'lpc'
plot(x, type, unscale = TRUE, lwd = 1, datcol = "grey60",
    datpch = 21, masscol = NULL, masspch = 15, curvecol = 1, splinecol = 3,
    projectcol = 4, startcol = NULL,  startpch=NULL,...)
## S3 method for class 'lpc.spline'
plot(x, type, unscale = TRUE, lwd = 1, datcol = "grey60",
    datpch = 21, masscol = NULL, masspch = 15, curvecol = 1, splinecol = 3,
    projectcol = 4, startcol = NULL,  startpch=NULL,...)
## S3 method for class 'ms'
plot(x, unscale=FALSE, lwd=1, datcol="grey70",   datpch=21, masscol=NULL,
    masspch=15, curvecol=NULL, ...)
```

Arguments

x	an object of class lpc or lpc.spline.
type	a vector of type c("mass","spline",...) with possible entries mass,curve, spline,project,start.
unscale	if TRUE, then data (and all fitted componens) are scaled back to their original scale; otherwise the scaled data are plotted (only relevant if scaled=TRUE in the fitted object). For ms, this is currently unimplemented.
lwd	width of principal curves or trajectories.
datcol	color of data points.
datpch	plotting symbol for data points.
masscol	color of centers of mass (see below) or cluster centers.
masspch	plotting symbol for centers of mass or cluster centers.
curvecol	color of the curve interpolating the local centers of mass (this is the "local principal curve"!).
splinecol	color of the spline representation of the local principal curve.
projectcol	color of projections onto the spline representation of the local principal curve.
startcol	color of the plotted starting points.
startpch	plotting symbol for starting points; needs to be either a single symbol, or a vector of symbols of the same length as the number of starting points.
...	further arguments passed to plot or scatterplot3d.

Value

A 2D plot, 3D plot, or a pairs plot (depending on the type of object and the data dimension d.).

The most flexible plotting option is masscol. Depending on the length of the specified vector, this will be interpreted differently. If a scalar is provided, the corresponding color will be given to all centers of mass (or cluster centers). For LPCs, if the length of the vector is larger than 1, then this option will assign different colours to different depths, or different branch numbers, or to individual data points, depending on the length. The default setting is assigning colours according to depth, in the order red, blue, black.

With increasing dimension d, less plotting options tend to be supported. The nicest plots are obtained for $d = 2$ and $d = 3$.

Warning

This function computes all missing information (if posssible), so computation will take the longer the less informative the given object is, and the more advanced aspects are asked to plot!

Author(s)

JE

References

Einbeck, J., Tutz, G., and Evers, L. (2005). Local principal curves. Statistics and Computing 15, 301-313.

Einbeck, J., Evers, L. & Hinchliff, K. (2010): Data compression and regression based on local principal curves. In A. Fink, B. Lausen, W. Seidel, and A. Ultsch (Eds), Advances in Data Analysis, Data Handling, and Business Intelligence, Heidelberg, pp. 701–712, Springer.

See Also

lpc, lpc.spline , ms

Examples

```
data(calspeedflow)
lpc1 <- lpc(calspeedflow[,3:4])
plot(lpc1, type=c("spline","project"), lwd=2)
ms1<- ms(calspeedflow[,3:4], subset=sample.int(444,100), plot=FALSE)
    # starts trajectories from 100 random obs'n
plot(ms1, masscol=1)
plot(ms1, curvecol="grey30")

data(mussels, package="dr")
ms2 <- ms(mussels[,-3], scaled=1, h=0.1, plot=FALSE)
plot(ms2, datpch=20, masspch=24)
```

---

print.lpc *Printing output for lpc, lpc.spline, and ms objects*

Description

Takes an object of class lpc, lpc.spline, ms and displays some standard output.

Usage

```
## S3 method for class 'lpc'
print( x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'lpc.spline'
print( x, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'ms'
print( x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	an object of class lpc, lpc.spline, or ms.
digits	not yet in use.
...	further arguments.

Value

Some short text.

Author(s)

JE

See Also

lpc, ms

Examples

```
data(calspeedflow)
lpc1 <- lpc(calspeedflow[,3:4])
print(lpc1)
lpc2 <- lpc.spline(lpc1)
print(lpc2)
## Not run:
ms1<- ms(calspeedflow[,3:4], plot=FALSE)
print(ms1)
## End(Not run)
```

Rc *Measuring goodness-of-fit for principal objects.*

Description

These functions compute the 'coverage coefficient' R_C for local principal curves, local principal points (i.e., kernel density estimates obtained through iterated mean shift), and other principal objects.

Usage

```
Rc(x,...)

## S3 method for class 'lpc'
Rc(x,...)
## S3 method for class 'lpc.spline'
Rc(x,...)
## S3 method for class 'ms'
Rc(x,...)

base.Rc(data,  closest.coords, type="curve")
```

Arguments

x	an object used to select a method.
...	Further arguments passed to or from other methods (not needed yet).
data	A data matrix.
closest.coords	A matrix of coordinates of the projected data.
type	For principal curves, don't modify. For principal points, set "points".

Details

Rc computes the coverage coefficient R_C, a quantity which estimates the goodness-of-fit of a fitted principal object. This quantity can be interpreted similar to the coeffient of determination in regression analysis: Values close to 1 indicate a good fit, while values close to 0 indicate a 'bad' fit (corresponding to linear PCA).

For objects of type lpc, lpc.spline, and ms, S3 methods are available which use the generic function Rc. This, in turn, calls the base function base.Rc, which can also be used manually if the fitted object is of another class. In principle, function base.Rc can be used for assessing goodness-of-fit of any principal object provided that the coordinates (closest.coords) of the projected data are available. For instance, for HS principal curves fitted via princurve, this information is contained in component $s, and for a a k-means object, say fitk, this information can be obtained via fitk$centers[fitk$cluster,]. Set type="points" in the latter case.

The function Rc attempts to compute all missing information, so computation will take the longer the less informative the given object x is. Note also, Rc looks up the option scaled in the fitted object, and accounts for the scaling automatically. Important: If the data were scaled, then do NOT unscale the results by hand in order to feed the unscaled version into base.Rc, this will give a wrong result.

In terms of methodology, these functions compute R_C directly through the mean reduction of absolute residual length, rather than through the area above the coverage curve.

These functions do currently not account for observation weights, i.e. R_C is computed through the unweighted mean reduction in absolute residual length (even if weights have been used for the curve fitting).

Author(s)

J. Einbeck.

References

Einbeck, Tutz, and Evers (2005). Local principal curves. Statistics and Computing 15, 301-313.

Einbeck (2011). Bandwidth selection for nonparametric unsupervised learning techniques – a unified approach via self-coverage. Journal of Pattern Recognition Research 6, 175-192.

See Also

lpc.spline, ms, coverage.

Examples

```
data(calspeedflow)
lpc1 <- lpc.spline(lpc(calspeedflow[,3:4]), project=TRUE)
Rc(lpc1)
# is the same as:
base.Rc(lpc1$lpcobject$data, lpc1$closest.coords)

## Not run:
ms1 <- ms(calspeedflow[,3:4], plot=FALSE)
Rc(ms1)
# is the same as:
base.Rc(ms1$data, ms1$cluster.center[ms1$closest.label,], type="points")

## End(Not run)
```

---

unscale *Unscaling local principal objects.*

Description

unscale takes an object of type lpc, lpc.spline, or ms, which had been fitted using option scaled=TRUE, and transforms the scaled components back to the original data scale.

Usage

```
unscale(x, ...)

## S3 method for class 'lpc'
unscale(x,...)
## S3 method for class 'lpc.spline'
unscale(x,...)
## S3 method for class 'ms'
unscale(x,...)
```

Arguments

x	an object used to select a method.
...	Further arguments passed to or from other methods (not needed yet).

Value

A list of relevant items, such as LPC, start, cluster.centers, etc., which gives the unscaled versions of these quantities (some of them may carry the value NULL, if the corresponding information was not available from x).

Author(s)

JE

See Also

lpc, lpc.spline, ms

Examples

```
data(gvessel)
unscale(lpc(gvessel[,c(2,4,5)],  h=0.11,  x0=c(35, 1870, 6.3)) )
```

Index