Package 'princurve'

July 14, 2018

Version 2.1.0

Title Fits a Principal Curve in Arbitrary Dimension

Description Fitting a principal curve to a data matrix in arbitrary dimensions.

License GPL-2

Depends R (>= 3.0)

Imports stats, graphics, grDevices, Rcpp

Suggests devtools, microbenchmark, testthat

NeedsCompilation yes

RoxygenNote 6.0.1

URL https://github.com/dynverse/princurve

BugReports https://github.com/dynverse/princurve/issues

LinkingTo Rcpp

Collate 'RcppExports.R' 'bias_correct_curve.R' 'get.lam.R' 'package.R' 'periodic_lowess.R' 'principal.curve.R' 'smoother_functions.R' 'principal_curve.R' 'start_circle.R'

Author Trevor Hastie [aut],
Andreas Weingessel [aut],
Kurt Hornik [aut] (<https://orcid.org/0000-0003-4198-9911>),
Henrik Bengtsson [ctb] (HenrikBengtsson),
Robrecht Cannoodt [aut, cre] (<https://orcid.org/0000-0003-3641-729X>, rcannood)

Maintainer Robrecht Cannoodt <rcannood@gmail.com>

Repository CRAN

Date/Publication 2018-07-14 06:50:04 UTC

R topics documented:

`princurve-package` *Fits a Principal Curve in Arbitrary Dimension*

Description

Fits a Principal Curve in Arbitrary Dimension

References

Hastie, T. and Stuetzle, W., Principal Curves, JASA, Vol. 84, No. 406 (Jun., 1989), pp. 502-516, DOI: 10.2307/2289936 (PDF).

See also Banfield and Raftery (JASA, 1992).

See Also

`principal_curve`, `project_to_curve`

`get.lam` *Projection Index*

Description

This function will be deprecated on August 1st, 2018. See `project_to_curve` instead.

Usage

```
get.lam(x, s, tag = NULL, stretch = 2)
```

Arguments

`x`	a matrix of data points.
`s`	a parametrized curve, represented by a polygon.
`tag`	the order of the point in `s`. Default is the given order.
`stretch`	A stretch factor for the endpoints of the curve; a maximum of 2. it allows the curve to grow, if required, and helps avoid bunching at the end.

`principal.curve` *Fit a Principal Curve*

Description

This function will be deprecated on August 1st, 2018. Use `principal_curve` instead.

Usage

```
principal.curve(x, start = NULL, thresh = 0.001, plot.true = FALSE,
  maxit = 10, stretch = 2, smoother = c("smooth_spline", "lowess",
  "periodic_lowess"), trace = FALSE, ...)

## S3 method for class 'principal.curve'
lines(x, ...)

## S3 method for class 'principal.curve'
plot(x, ...)

## S3 method for class 'principal.curve'
points(x, ...)
```

Arguments

`x`	a matrix of points in arbitrary dimension.
`start`	either a previously fit principal curve, or else a matrix of points that in row order define a starting curve. If missing or NULL, then the first principal component is used. If the smoother is `"periodic_lowess"`, then a circle is used as the start.
`thresh`	convergence threshold on shortest distances to the curve.
`plot.true`	If `TRUE` the iterations are plotted.
`maxit`	maximum number of iterations.
`stretch`	a factor by which the curve can be extrapolated when points are projected. Default is 2 (times the last segment length). The default is 0 for `smoother` equal to `"periodic_lowess"`.
`smoother`	choice of smoother. The default is `"smooth_spline"`, and other choices are `"lowess"` and `"periodic_lowess"`. The latter allows one to fit closed curves. Beware, you may want to use `iter = 0` with `lowess()`.
`trace`	If `TRUE`, the iteration information is printed
`...`	additional arguments to the smoothers

principal_curve *Fit a Principal Curve*

Description

Fits a principal curve which describes a smooth curve that passes through the `middle` of the data `x` in an orthogonal sense. This curve is a nonparametric generalization of a linear principal component. If a closed curve is fit (using `smoother = "periodic_lowess"`) then the starting curve defaults to a circle, and each fit is followed by a bias correction suggested by Jeff Banfield.

Usage

```
principal_curve(x, start = NULL, thresh = 0.001, maxit = 10,
  stretch = 2, smoother = c("smooth_spline", "lowess", "periodic_lowess"),
  approx_points = FALSE, trace = FALSE, plot_iterations = FALSE, ...)

## S3 method for class 'principal_curve'
lines(x, ...)

## S3 method for class 'principal_curve'
plot(x, ...)

## S3 method for class 'principal_curve'
points(x, ...)

whiskers(x, s, ...)
```

Arguments

`x`	a matrix of points in arbitrary dimension.
`start`	either a previously fit principal curve, or else a matrix of points that in row order define a starting curve. If missing or NULL, then the first principal component is used. If the smoother is `"periodic_lowess"`, then a circle is used as the start.
`thresh`	convergence threshold on shortest distances to the curve.
`maxit`	maximum number of iterations.
`stretch`	A stretch factor for the endpoints of the curve, allowing the curve to grow to avoid bunching at the end. Must be a numeric value between 0 and 2.
`smoother`	choice of smoother. The default is `"smooth_spline"`, and other choices are `"lowess"` and `"periodic_lowess"`. The latter allows one to fit closed curves. Beware, you may want to use `iter = 0` with `lowess()`.
`approx_points`	Approximate curve after smoothing to reduce computational time. If `FALSE`, no approximation of the curve occurs. Otherwise, `approx_points` must be equal to the number of points the curve gets approximated to; preferably about 100.
`trace`	If `TRUE`, the iteration information is printed

plot_iterations
: If TRUE the iterations are plotted.

...
: additional arguments to the smoothers

s
: a parametrized curve, represented by a polygon.

Value

An object of class "principal_curve" is returned. For this object the following generic methods a currently available: plot, points, lines.

It has components:

s
: a matrix corresponding to x, giving their projections onto the curve.

ord
: an index, such that s[order,] is smooth.

lambda
: for each point, its arc-length from the beginning of the curve. The curve is parametrized approximately by arc-length, and hence is unit-speed.

dist
: the sum-of-squared distances from the points to their projections.

converged
: A logical indicating whether the algorithm converged or not.

num_iterations
: Number of iterations completed before returning.

call
: the call that created this object; allows it to be updated().

References

Hastie, T. and Stuetzle, W., Principal Curves, JASA, Vol. 84, No. 406 (Jun., 1989), pp. 502-516, DOI: 10.2307/2289936 (PDF).

See Also

project_to_curve

Examples

```
x <- runif(100,-1,1)
x <- cbind(x, x ^ 2 + rnorm(100, sd = 0.1))
fit1 <- principal_curve(x, plot_iterations = TRUE)
fit2 <- principal_curve(x, plot_iterations = TRUE, smoother = "lowess")
lines(fit1)
points(fit1)
plot(fit1)
whiskers(x, fit1$s)
```

project_to_curve *Project a set of points to the closest point on a curve*

Description

Finds the projection index for a matrix of points x, when projected onto a curve s. The curve need not be of the same length as the number of points.

Usage

```
project_to_curve(x, s, stretch = 2)
```

Arguments

x	a matrix of data points.
s	a parametrized curve, represented by a polygon.
stretch	A stretch factor for the endpoints of the curve, allowing the curve to grow to avoid bunching at the end. Must be a numeric value between 0 and 2.

Value

A structure is returned which represents a fitted curve. It has components

s	The fitted points on the curve corresponding to each point x
ord	the order of the fitted points
lambda	The projection index for each point
dist	The total squared distance from the curve
dist_ind	The squared distances from the curve to each of the respective points

See Also

principal_curve

smoother_functions *Smoother functions*

Description

Each of these functions have an interface function(lambda, xj, ...), and return smoothed values for xj. The output is expected to be ordered along an ordered lambda. This means that the following is true:

```
x <- runif(100)
y <- runif(100)
ord <- sample.int(100)
sfun <- smoother_functions[[1]]
all(sfun(x, y) == sfun(x[ord], y[ord]))
```

Usage

```
smoother_functions
```

Format

An object of class `list` of length 3.

`start_circle` *Generate circle as initial curve*

Description

The starting circle is defined in the first two dimensions, and has zero values in all other dimensions.

Usage

```
start_circle(x)
```

Arguments

`x`	The data for which to generate the initial circle

Examples

```
## Not run:
x <- cbind(
  rnorm(100, 1, .2),
  rnorm(100, -5, .2),
  runif(100, 1.9, 2.1),
  runif(100, 2.9, 3.1)
)
circ <- start_circle(x)
plot(x)
lines(circ)

## End(Not run)
```

Index