Package 'princurve'

July 23, 2025

Version 2.1.6

Title Fit a Principal Curve in Arbitrary Dimension

Description Fitting a principal curve to a data matrix in arbitrary dimensions.
Hastie and Stuetzle (1989) <doi:10.2307/2289936>.

License GPL-2

Encoding UTF-8

Depends R (>= 3.0)

Imports stats, graphics, grDevices, Rcpp

Suggests devtools, testthat

LinkingTo Rcpp

NeedsCompilation yes

RoxygenNote 7.1.1

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

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

Collate 'RcppExports.R' 'bias_correct_curve.R' 'deprecated.R'
'package.R' 'periodic_lowess.R' 'smoother_functions.R'
'principal_curve.R' 'start_circle.R'

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

Maintainer Robrecht Cannoodt <rcannood@gmail.com>

Repository CRAN

Date/Publication 2021-01-18 16:40:06 UTC

Contents

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

Description

Fit 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 non-parametric 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.

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`

`principal.curve` *Deprecated functions*

Description

This function is deprecated, please use `principal_curve` and `project_to_curve` instead.

Usage

```
principal.curve(...)

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

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

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

get.lam(...)
```

Arguments

...	Catch-all for old parameters.

---

principal_curve *Fit a Principal Curve*

Description

Fit 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 non-parametric 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))
fit <- principal_curve(x)
plot(fit)
lines(fit)
points(fit)
whiskers(x, fit$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

Examples

```
t <- runif(100, -1, 1)
x <- cbind(t, t ^ 2) + rnorm(200, sd = 0.05)
s <- matrix(c(-1, 0, 1, 1, 0, 1), ncol = 2)

proj <- project_to_curve(x, s)

plot(x)
lines(s)
segments(x[, 1], x[, 2], proj$s[, 1], proj$s[, 2])
```

---

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