Package ‘stinepack’

February 20, 2015

Title Stineman, a consistently well behaved method of interpolation

Version 1.3

Author Tomas Johannesson & Halldor Bjornsson, Icelandic Met. Office;
Gabor Grothendieck

Depends

Description A consistently well behaved method of interpolation based
on piecewise rational functions using Stineman’s algorithm

Maintainer Tomas Johannesson <tj@vedur.is>

License GPL-2

Repository CRAN

Date/Publication 2012-10-29 08:59:51

NeedsCompilation no

R topics documented:

na.stinterp Replace NAs by Stineman interpolation
parabolaSlopes
stinemanSlopes
stinterp

Index

na.stinterp

Description

Generic functions for replacing each NA with Stineman interpolated values.
Usage

na.stinterp(object, ...)
## Default S3 method:
na.stinterp(object, along = time(object), na.rm = TRUE, ...)

Arguments

object  object in which NAs are to be replaced.
along   variable to use for interpolation. Has to be numeric, is otherwise coerced to numeric.
na.rm   logical. Should leading and trailing NAs be removed?
...     further arguments passed to methods.

Details

Missing values (NAs) are replaced by piecewise rational interpolation via stinterp.

By default the time index associated with object is used for interpolation. Note, that if this calls time.default this gives an equidistant spacing 1:NROW(object). If object is a matrix or data.frame, the interpolation is done separately for each column.

Value

An object in which each NA in the input object is replaced by interpolating the non-NA values before and after it. Leading and trailing NAs are omitted (if na.rm = TRUE) or not replaced (if na.rm = FALSE).

Author(s)

Gabor Grothendieck

See Also

stinterp.

Examples

na.stinterp(c(2,NA,1,4,5,2))
nasntinterp(ts(c(2,NA,1,4,5,2)))
## Not run:

#comparison of gap filling with na.stinterp/stinterp and splines,
#the solid lines show the continuous interpolation functions
#implicitly assumed by the gap-filling, they show how the Stineman
#interpolation restricts the range of the interpolant to the
#nearby range of the points and suppresses the well known oscillations
#characteristic of splines and other methods based on polynomials
x <- 1:6
y <- c(2,NA,1,4,5,2)
plot(x,y,ylim=c(-1,5))
### parabolaSlopes

Estimate the slope of an interpolating function using a parabola

#### Description

Returns estimates of the slope of an interpolating function that runs through a set of points in the xy-plane.

#### Usage

```r
parabolaSlopes(x, y)
```

#### Arguments

- `x, y` coordinates of points defining the interpolating function.

#### Value

Returns an estimate of the slope of the interpolant at (x,y).

#### Note

This function is used as part of the Stineman interpolation function `stinterp`. It is rarely called directly by the user, and checking of x and y must be performed by the calling function.

The parabola method provides a better approximation of the slope for smooth functions than the original method suggested by Stineman (1980), which is provided by the function `stinemanSlopes` (see the documentation of the function `stinterp` for further information), but it results in higher slopes near abrupt steps or spikes and can lead to some overshooting where Stineman’s method does not.
**Author(s)**
Norbert Nemec, Institute of Theoretical Physics, University of Regensburg. Translation from Python code by Halldor Bjornsson

**See Also**
`stinterp` and `stinemanSlopes`.

**Examples**
```r
x <- seq(0,2*pi,by=pi/6)
y <- sin(x)
## Not run: parabolaSlopes(x,y)
```

---

**stinemanSlopes**

*Estimate the slope of an interpolating function using an arc*

**Description**
Returns estimates of the slope of an interpolating function that runs through a set of points in the xy-plane. The slopes are calculated using to the algorithm of Stineman (1980), i.e. from the tangent of circles passing through every three consecutive points.

**Usage**
```
stinemanSlopes(x,y, scale=FALSE)
```

**Arguments**

- `x,y`
  coordinates of points defining the interpolating function.
- `scale`
  if true (default) then the x and y values are normalized prior to the slope calculation.

**Value**
Returns an estimate of the slope of the interpolant at (x,y).

**Note**
This function is used as part of the Stineman interpolation function `stinterp`. It is rarely called directly by the user, and checking of x and y must be performed by the calling function.

Stineman’s method provides a more robust interpolating function near abrupt steps or spikes in the point sequence than the alternative method based on a second degree interpolating polynomial, which is provided by the function `parabolaSlopes` (see the documentation of the function `stinterp` for further information), but it results in slightly less accuracy for smooth functions.
Author(s)

Tomas Johannesson

References


See Also

`stinterp` and `parabolaSlopes`.

Examples

```r
## Interpolate a smooth curve
x <- seq(0,2*pi,by=pi/6)
y <- sin(x)
stinemanSlopes(x,y,scale=TRUE)
stinemanSlopes(x,y,scale=FALSE)
```

Description

Returns the values of an interpolating function that runs through a set of points in the xy-plane according to the algorithm of Stineman (1980).

Usage

```r
stinterp(x,y,xout,yp,method=c("scaledstineman","stineman","parabola"))
```

Arguments

- `x,y` coordinates of points defining the interpolating function. NAs are not allowed, but see `na.stinterp` for gap-filling based on `stinterp`.
- `xout` the x-coordinate where the interpolant is to be found.
- `yp` slopes of the interpolating function at x. Optional: only given if they are known, else the argument is not used.
- `method` method for computing the slope at the given points if the slope is not known. With `method="stineman"`, Stineman’s original method based on an interpolating circle is used. Use `method="scaledstineman"` if scaling of x and y is to be carried out before Stineman’s method is applied, and use `method="parabola"` to calculate the slopes from a parabola through every three points. Partial argument matching is supported (for example `m="pa"` instead of `method="parabola"`).
Value

stinterp returns a list with components 'x' and 'y' with the coordinates of the interpolant at the points specified by xout.

Note

The interpolation method is described in an article by Russell W. Stineman in the July 1980 issue of Creative Computing with a note from the editor stating that while they were "not an academic journal but once in a while something serious and original comes in" adding that this was "apparently a real solution" to a well known problem.

According to Stineman, the interpolation procedure has "the following properties:

1. If values of the ordinates of the specified points change monotonically, and the slopes of the line segments joining the points change monotonically, then the interpolating curve and its slope will change monotonically.
2. If the slopes of the line segments joining the specified points change monotonically, then the slopes of the interpolating curve will change monotonically.
3. Suppose that the conditions in (1) or (2) are satisfied by a set of points, but a small change in the ordinate or slope at one of the points will result conditions (1) or (2) being not longer satisfied. Then making this small change in the ordinate or slope at a point will cause no more than a small change in the interpolating curve."

The method is based on rational interpolation with specially chosen rational functions to satisfy the above three conditions.

Slopes computed at the given points with the methods provided by the ‘stinterp’ function satisfy Stineman’s requirements. The original method suggested by Stineman (method="scaledstineman", the default, and "stineman") result in lower slopes near abrupt steps or spikes in the point sequence, and therefore a smaller tendency for overshooting. The method based on a second degree polynomial (method="parabola") provides better approximation to smooth functions, but it results in higher slopes near abrupt steps or spikes and can lead to some overshooting where Stineman’s method does not. Both methods lead to much less tendency for ‘spurious’ oscillations than traditional interpolation methods based on polynomials, such as splines (see the examples section).

Stineman states that "The complete assurance that the procedure will never generate ‘wild’ points makes it attractive as a general purpose procedure".

This interpolation method has been implemented in Matlab and Python in addition to R.

Author(s)

Tomas Johannesson, Halldor Bjornsson

References


See Also

na.stinterp, stinemanSlopes and parabolaSlopes.
Examples

## Interpolation with rational functions
This example shows how the rational interpolating functions used in
'stinterp' have a monotonic slope over an interval with widely varying
slopes specified at the end points, consistent with Stineman's
requirements above. A third degree interpolating function (commonly
used in spline interpolation), on the other hand, leads to
spurious oscillations, which are a well known problem in
interpolation with a single polynomial function and also
in piecewise polynomial interpolation.

```r
## Not run:
xo <- seq(0,1,by=1/50)
plot(c(0,1),c(0,1),xlim=c(0,1),ylim=c(-1,1),xlab="",ylab="")
for (s in 2:10) {
  lines(stinterp(c(0,1),c(0,1),xo,yp=c(0,s)))
  lines(xo,xo^2*((s-2)*xo-s+3),col=2)
}
```

## End(Not run)

Note that the two interpolation functions almost coincide for the
lowest value (s=2) of the slope at the right end point.
The user may verify that the rational interpolating functions continue
to provide "reasonable" results for much higher values of the slope at the
right end point (for example s=15, s=25 or s=100), for which the third degree
polynomial leads to absurd results (for most practical purposes).

## Interpolate a smooth curve
This example illustrates that the interpolation procedure
reproduces a smooth function with known slopes at the specified
points very well. If the slopes are not known, both methods for
estimating the slopes at the specified points (the default method
and method="parabola") lead to good interpolating functions, but
the "parabola" method is slightly more accurate. The traditional
spline interpolation method leads to a similar result as Stineman's
method with slopes computed with method="parabola".

```r
x <- seq(0,2*pi,by=pi/6)
y <- sin(x)
yp <- cos(x)
xo <- seq(0,2*pi,by=pi/150)
y1 <- stinterp(x,y,xo,yp)$y
y2 <- stinterp(x,y,xo)$y
y3 <- stinterp(x,y,xo,m="pa")$y
## Not run:
plot(x,y)
lines(xo,sin(xo))
points(stinterp(x,y,xo,yp),cex=1/5,col=2)
points(stinterp(x,y,xo),cex=1/5,col=3)
points(stinterp(x,y,xo,method="parabola"),cex=1/5,col=4)
points(spline(x,y,n=length(xo)),cex=1/5,col=5)
```

## End(Not run)

## Interpolate through a sharp oscillation
# This example shows that Stineman's interpolation, with the default method for estimating slopes at the given points, results in no oscillations in the neighbourhood of a spike. If the slopes at the given points are computed with method="parabola", some overshooting can be seen and spline interpolation leads to repeated oscillations near the spike.

```r
## not run:
yy <- y
yy[3] <- -1.5
plot(x,yy ylim=c(-1.5,1.5))
points(stinterp(x,yy,xo),cex=1/5,col=3)
points(stinterp(x,yy,xo,method="parabola"),cex=1/5,col=4)
points(spline(x,yy,n=length(xo)),cex=1/5,col=5)

## End(Not run)
```
Index

*Topic math
  parabolaSlopes, 3
  stinemanSlopes, 4
  stinterp, 5
*Topic smooth
  parabolaSlopes, 3
  stinemanSlopes, 4
  stinterp, 5
*Topic ts
  na.stinterp, 1
  parabolaSlopes, 3
  stinemanSlopes, 4
  stinterp, 5

na.stinterp, 1, 5, 6
parabolaSlopes, 3, 4–6
stinemanSlopes, 3, 4, 6
stinterp, 2–5, 5