Package 'corset'

July 22, 2025

Type Package

Title Arbitrary Bounding of Series and Time Series Objects

Version 0.1-5

Date 2023-02-12

Author Fran Urbano

Maintainer Fran Urbano <viraltux@gmail.com>

Depends R (>= 2.10)

Imports stats

Suggests forecast, hts, testthat

Description Set of methods to constrain numerical series and time series within arbitrary boundaries.

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-02-21 13:50:03 UTC

Contents

corset *Arbitrary Bounding of Series and Time Series Objects*

Description

When working with times series we often have boundary constraints that cannot be easily introduced in mathematical models like ARIMA or ETS, or simply cannot be introduced in already existing R packages returning series and time series objects.

The `corset` package intends to be a companion to series and time series analysis to easily enhance and tune already existing results in a seamless way. A typical use case would be to force declining financial time series forecasts to converge to zero instead having negative values.

The `corset` function allows to introduce boundary constraints in series, time series as well as in `forecast` & `gts/hts` objects.

In particular, when applied on a `forecast` object it does not only apply the boundaries to the forecast object but also to its confidence intervals.

Usage

```
corset(x, method = c('bezier', 'exp', 'naive'),
       min = 0, max = Inf, proximity = 0, centrality = FALSE)
```

Arguments

`x`	Numerical series or time series.
`method`	There are three different methodologies available to bound `x`: - Partial Bezier (default) - Exponential - Naive See the details section for more information.
`min`	vector with lower boundaries. If a smaller vector on single value is introduced it will be converted into a vector of the right size.
`max`	vector with upper boundaries. If a smaller vector on single value is introduced it will be converted into a vector of the right size.
`proximity`	measure of how close the bounded data is to the actual data in `x`. This parameter works for the Partial Bezier, it has no effect on the Exponential & Naive method. See the details section for more information.
`centrality`	when TRUE it forces the mean of a forecast object into the middle values of the smallest prediction interval.

Details

The three methods available are:

- Naive: This method simply removes values outside the min/max boundaries and replace them by the values in the boundary. - Exponential: The exponential method updates values outside the boundaries with the average pre and post values of the series. - Partial Bezier: The partial bezier method uses a uni-dimensional bezier curves to adjust, not only the values outside the boundaries,

but those values within the boundaries in order to offer a more smooth transition from bounded to unbounded values.

In general the Partial Bezier method should be preferred and tuned with the proximity parameter. However, the Partial Bezier method has precision problems when using a large number of points, in those cases, the Exponential method or the Naive method might help.

The `proximity` parameter is an positive or zero integer value (rounded otherwise) which expresses the similarity desired between the input and the bounded output. The higher `proximity` is the closer the bounded output will be to the output of the Naive method. If the value is larger than 100 the result is equivalent to use the Naive method.

Value

The value returned will be of the same class of the input `x` with the boundaries applied to its values.

Author(s)

Fran Urbano <fran.urbano@merck.com>

Examples

```
## Comparison of methods ##
###########################

x <- ts(sin(seq(-2*pi, pi, 0.3)))
x.b.0 <- corset(x,'bezier')
x.b.1 <- corset(x,'bezier', proximity = 1)
x.e <- corset(x,'exp')
x.n <- corset(x,'naive')

layout(matrix(c(1,2,1,3),ncol=2))
plot(x, type = 'o', lwd = 3, main = 'Partial Bezier [0,Inf)\nproximity = 0/1')
lines(x.b.0, col = 'blue', lwd = 3, lty = 1)
lines(x.b.1, col = 'blue', lwd = 3, lty = 1)
abline(h=0)

plot(x, type = 'o', lwd = 3, main = 'Exponential [0,Inf)')
lines(x.e, col = 'green', lwd = 3, lty = 1)
abline(h=0)

plot(x, type = 'o', lwd = 3, main = 'Naive [0,Inf)')
lines(x.n, col = 'red', lwd = 3, lty = 1)
abline(h=0)
layout(1)

## Linear Boundaries Example ##
###############################

x <- ts(sin(seq(-5*pi, 5*pi, length.out = 200)))
min <- seq(-0.1,-1, length.out = 200)
max <- seq(0.1,1, length.out = 200)
plot(x, main = 'Partial Bezier\nLinear Boundaries')
```

```
lines(min, col = 'red')
lines(max, col = 'green')
lines(corset(x, method = 'bezier',
             min = min, max = max, proximity = 2),
      lwd = 3, col = 'blue')

## Centrality Example for forecast object ##
############################################

if ('forecast' %in% installed.packages()){
  layout(matrix(c(1,2,3),ncol=1))
 f <- forecast::forecast(forecast::ets(ts(33:14 + rep(c(-8,8),10), frequency = 2), 'AAN'),50)
  plot(f, main = 'forecast object | f'); abline(h=0, lwd = 2)
  c <- corset(f, centrality = FALSE)
  plot(c, main = '"Corseted" forecast object | corset(f)')
  cc <- corset(f, centrality = TRUE)
  plot(cc, main = '"Corseted" forecast object | corset(f, centrality = TRUE)')
  layout(1)
}
```

Index