Package ‘VBV’

January 20, 2025

Title The Generalized Berlin Method for Time Series Decomposition

Version 0.6.2

Description Time series decomposition for univariate time series using the ``Verallgemeinerte Berliner Verfahren'' (Generalized Berlin Method) as described in 'Kontinuierliche Messgrößen und Stichprobenstrategien in Raum und Zeit mit Anwendungen in den Natur-, Umwelt-, Wirtschafts- und Finanzwissenschaften', by Hebbel and Steuer, Springer Berlin Heidelberg, 2022 <doi:10.1007/978-3-662-65638-9>, or 'Decomposition of Time Series using the Generalised Berlin Method (VBV)' by Hebbel and Steuer, in Jan Beran, Yuanhua Feng, Hartmut Hebbel (Eds.): Empirical Economic and Financial Research - Theory, Methods and Practice, Festschrift in Honour of Prof. Siegfried Heiler. Series: Advanced Studies in Theoretical and Applied Econometrics. Springer 2014, p. 9-40.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Author Detlef Steuer [aut, cre] (<`https://orcid.org/0000-0003-2676-5290`>), Hartmut Hebbel [aut]

Maintainer Detlef Steuer <`steuer@hsu-hh.de`>

Repository CRAN

Date/Publication 2023-01-09 10:00:11 UTC

Contents

`decomposition` *decomposition - decompose a time series with VBV*

Description

decomposition - decompose a time series with VBV

Usage

```
decomposition(t.vec, p, q.vec, base.period, lambda1, lambda2)
```

Arguments

`t.vec`	vector of observation points.
`p`	maximum exponent in polynomial for trend
`q.vec`	vector containing frequencies to use for seasonal component, given as integers, i.e. c(1, 3, 5) for 1/2*pi, 3/2*pi, 5/2*pi (times length of base period)
`base.period`	base period in number of observations, i.e. 12 for monthly data with yearly oscillations
`lambda1`	penalty weight for smoothness of trend
`lambda2`	penalty weight for smoothness of seasonal component (lambda1 == lambda2 == Inf result in estimations of the original Berliner Verfahren)

Value

list with the following components:

- trendA function which returns the appropriate weights if applied to a point in time
- saisonA function which returns the appropriate weights if applied to a point in time
- A, G1, G2Some matrices that allow to calclate SSE etc. Exposed only to reuse their calculation. See the referenced paper for details.

Examples

```
### Usage of decomposition
t <- 1:121 # equidistant time points, i.e. 5 days
p <- 2     # maximally quadratic
q <- c(1, 3, 5)   # 'seasonal' components within the base period
base.period <- 24 # i.e. hourly data with daily cycles
l1 <- 1
l2 <- 10

dec <- decomposition( t, p, q, base.period, l1, l2)
### Note: decomosition is independent of data, only depends on time
```

estimation *estimation – estimate trend and seasonal components statically*

Description

estimation – estimate trend and seasonal components statically

Usage

```
estimation(t.vec, y.vec, p, q.vec, base.period, lambda1, lambda2)
```

Arguments

`t.vec`	vector of points in time as integers
`y.vec`	vector of data
`p`	maximum exponent in polynomial for trend
`q.vec`	vector containing frequencies to use for seasonal component, given as integers, i.e. c(1, 3, 5) for 1/2*pi, 3/2*pi, 5/2*pi (times length of base period)
`base.period`	base period in number of observations, i.e. 12 for monthly data with yearly oscillations
`lambda1`	penalty weight for smoothness of trend
`lambda2`	penalty weight for smoothness of seasonal component (lambda1 == lambda2 == Inf result in estimations of the original Berliner Verfahren)

Value

A dataframe with the following components:

- dataoriginal data y.vec
- trendvector of estimated trend of length length(y.vec)
- seasonvector of estimated season of length length(y.vec)

Examples

```
### using of estimation

t <- 1:121 # equidistant time points, i.e. 5 days
y <- 0.1*t + sin(t) + rnorm(length(t))

p <- 2     # maximally quadratic
q <- c(1, 3, 5)   # 'seasonal' components within the base period
base.period <- 24 # i.e. hourly data with daily cycles
l1 <- 1
l2 <- 10

est <- estimation( t, y, p, q, base.period, l1, l2)
plot(est$data)
lines(est$trend + est$season)
```

moving.decomposition *moving.decomposition – decompose a times series into locally estimated trend and season figures*

Description

moving.decomposition – decompose a times series into locally estimated trend and season figures

Usage

```
moving.decomposition(n, p, q.vec, m, base.period, lambda1, lambda2)
```

Arguments

`n`	number of observation points (must be odd!). Internally this will be transformed to seq(-(n-1)/2, (n-1)/2, 1)
`p`	maximum exponent in polynomial for trend
`q.vec`	vector containing frequencies to use for seasonal component, given as integers, i.e. c(1, 3, 5) for 1/2*pi, 3/2*pi, 5/2*pi (times length of base period)
`m`	width of moving window
`base.period`	base period in number of observations, i.e. 12 for monthly data with yearly oscillations
`lambda1`	penalty weight for smoothness of trend
`lambda2`	penalty weight for smoothness of seasonal component

Value

list with the following components:

- W1nxn matrix of weights. Trend is estimated as W1 %*% y, if y is the data vector* W2nxn matrix of weights. Season is estimated as W2 %% y, if y is the data vector

Note

lambda1 == lambda2 == Inf result in estimations of the original Berliner Verfahren

Examples

```
### Usage of moving.decomposition

t <- 1:121 # equidistant time points, i.e. 5 days

m <- 11

p <- 2     # maximally quadratic
q <- c(1, 3, 5)   # 'seasonal' components within the base period
base.period <- 24 # i.e. hourly data with daily cycles
```

```
l1 <- 1
l2 <- 1

m.dec <- moving.decomposition( length(t), p, q, m, base.period, l1, l2)
```

`moving.estimation` *moving.estimation – estimate locally optimized trend and season figures*

Description

moving.estimation – estimate locally optimized trend and season figures

Usage

```
moving.estimation(t.vec, y.vec, p, q.vec, m, base.period, lambda1, lambda2)
```

Arguments

`t.vec`	vector of points in time as integers
`y.vec`	vector of data
`p`	maximum exponent in polynomial for trend
`q.vec`	vector containing frequencies to use for seasonal component, given as integers, i.e. c(1, 3, 5) for 1/2*pi, 3/2*pi, 5/2*pi (times length of base period)
`m`	width of moving window
`base.period`	base period in number of observations, i.e. 12 for monthly data with yearly oscillations
`lambda1`	penalty weight for smoothness of trend
`lambda2`	penalty weight for smoothness of seasonal component

Value

A dataframe with the following components:

- dataoriginal data y.vec
- trendvector of estimated trend of length length(y.vec)
- seasonvector of estimated season of length length(y.vec)

Note

lambda1 == lambda2 == Inf result in estimations of the original Berliner Verfahren

Index