Package ‘mvLSWimpute’

October 13, 2022

Type Package
Title Imputation Methods for Multivariate Locally Stationary Time Series
Version 0.1.1
Date 2022-08-15
Author Rebecca Wilson [aut],
      Matt Nunes [aut, cre],
      Idris Eckley [ctb, ths],
      Tim Park [ctb]
Maintainer Matt Nunes <nunesrpackages@gmail.com>
License GPL-2
Depends wavethresh, mvLSW
Imports binhf, xts, zoo, imputeTS, utils
NeedsCompilation no
Repository CRAN
Date/Publication 2022-08-16 10:20:02 UTC

R topics documented:

mvLSWimpute-package ............................................... 2
correct_per .......................................................... 3
form_lacv_forward .................................................. 4
haarWT ................................................................. 6
mv_impute ............................................................. 7
pdef ................................................................. 8
pred_eq_forward .................................................... 10
smooth_per .......................................................... 12
spec_estimation ....................................................... 13

Index 15
Description


Details

The DESCRIPTION file:

Package: mvLSWimpute
Type: Package
Title: Imputation Methods for Multivariate Locally Stationary Time Series
Version: 0.1.1
Date: 2022-08-15
Author: Rebecca Wilson [aut]. Matt Nunes [aut, cre]. Idris Eckley [ctb, ths]. Tim Park [ctb]
Authors@R: c(person("Rebecca", "Wilson", role = "aut"), person("Matt", "Nunes", role="aut","cre"), email="nunesrpackages@gmail.com")
Maintainer: Matt Nunes <nunesrpackages@gmail.com>
Description: Implementation of imputation techniques based on locally stationary wavelet time series forecasting methods for multivariate time series containing missing values.
License: GPL-2
Depends: wavethresh, mvLSW
Imports: binhf, xts, zoo, imputeTS, utils

Index of help topics:

- **correct_per**: Function to smooth the raw wavelet periodogram
- **form_lacv_forward**: Function to form the local autocovariance array for the forecasting / backcasting step.
- **haarWT**: Function to apply the (univariate) Haar wavelet transform
- **mvLSWimpute-package**: Imputation Methods for Multivariate Locally Stationary Time Series
- **mv_impute**: Function to apply the mvLSWimpute method and impute missing values in a multivariate locally stationary time series
- **pdef**: Function to regularise the LWS matrix.
- **pred_eq_forward**: Function to form the prediction equations for the forecasting / backcasting step.
- **smooth_per**: Function to smooth the raw wavelet periodogram using the default 'mvLSW' routine.
- **spec_estimation**: Function to estimate the Local Wavelet Spectral matrix for a multivariate locally stationary time series containing missing values
The main routine of the package is `mv_impute` which performs forward or forward and backward imputation of locally stationary multivariate time series, using one-step ahead forecasting (and backcasting).

Author(s)

Rebecca Wilson [aut], Matt Nunes [aut, cre], Idris Eckley [ctb, ths], Tim Park [ctb]

Maintainer: Matt Nunes <nunesrpackages@gmail.com>

References


---

**correct_per**

Function to smooth the raw wavelet periodogram

**Description**

This function corrects the raw wavelet periodogram, similar to the `mvEWS` function in the `mvLSW` package, except acting on the raw periodogram directly. See `mvEWS` for more details. Note: this function is not really intended to be used separately, but internally within the `spec_estimation` function.

**Usage**

`correct_per(RawPer)`

**Arguments**

- `RawPer`: Raw wavelet periodogram that is to be corrected, can be either a 4D array or a `mvLSW` object.

**Details**

The raw wavelet periodogram as an estimator for the local wavelet spectrum (LWS) is biased, and thus needs to be corrected. This is done using a correction (debiasing) matrix, formed from the inner product of autocorrelation wavelets, see Park et al. (2014), Taylor et al. (2019) for more details. This function performs this bias-correction.

**Value**

Returns a `mvLSW` object containing the smoothed EWS of a multivariate locally stationary time series.

**Author(s)**

Rebecca Wilson
References


See Also

mvEWS, spec_estimation

Examples

```r
## Sample bivariate locally stationary time series

set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[(-(1:2^7), 2] <- X[(-(1:2^7), 2] - 0.95 * X[(-(1:2^7), 1]
X[(-(1:2^7), 1] <- X[(-(1:2^7), 1] * 4
X <- as.ts(X)

# form periodogram, reshaping array as necessary

tmp = apply(X, 2, function(x){haarWT(x)$D})
D = array(t(tmp), dim = c(2, 2^8, 8))

RawPer = array(apply(D, c(2, 3), tcrossprod), dim = c(2, 2, 2^8, 8))
RawPer = aperm(RawPer, c(1, 2, 4, 3))

# now correct

correctedper = correct_per(RawPer)
```

Description

This function generates the local autocovariance (LACV) array that is used in the forecasting / backcasting step to form the prediction equations.
form_lacv_forward

Usage

form_lacv_forward(spectrum, index, p.len = 2)
form_lacv_backward(spectrum, index, p.len = 2)

Arguments

spectrum Local wavelet spectral matrix for which we wish to form the local autocovariance array.
index Time index of the missing data which we wish to impute.
p.len Number of terms to include in the clipped predictor when forecasting / back-casting.

Details

In order to form the one-step ahead predictor for use in the imputation algorithm of Wilson et al. (2021), one needs the local autocovariance (LACV). This is computed using the relationship between the LACV and the local wavelet spectrum (LWS). See equations (4) and (5) in Wilson et al. (2021) for more details.

Value

Returns the local autocovariance array that can be used as an input to the pred_eq_forward or pred_eq_backward function.

Author(s)

Rebecca Wilson

References


See Also

pred_eq_forward, pred_eq_backward
Examples

## Sample bivariate locally stationary time series

```r
set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

# create some missing values, taking care to provide some data at the start of the series
missing.index = sort(sample(10:2^8, 30))
X[missing.index, ] <-NA

# estimate the spectrum
spec = spec_estimation(X)
out <- form_lacv_forward(spec$spectrum, missing.index[1], p.len=2)
```

---

**haarWT**

*Function to apply the (univariate) Haar wavelet transform*

---

**Description**

This function applies the (univariate) Haar wavelet transform. For a time series containing missing values, the wavelet coefficients are generating and any NAs remain intact.

**Usage**

`haarWT(data)`

**Arguments**

- `data` Input univariate time series.

**Value**

Returns a list containing the following elements:

- `C` Matrix containing the smooth coefficients for the transform.
- `D` Matrix containing the detail coefficients for the transform.
Examples

```r
set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

# compute the haar wavelet coefficients of the first time series component:
Xwt1 = haarWT(X[, 1])
```

mv_impute

Function to apply the mvLSWimpute method and impute missing values in a multivariate locally stationary time series

Description

This function applies the mvLSWimpute method to impute missing values in a multivariate locally stationary time series. The imputation can be based on forecasts only or use information from both a forecasting and backcasting step.

Usage

```
mv_impute(data, p = 2, type = "forward", index = NULL)
```

Arguments

- **data**: Input multivariate time series, matrix of dimension TxP where P is the number of channels and T is the length of the series.
- **p**: The number of terms to include in the clipped predictor when carrying out one step ahead forecasting/backcasting.
- **type**: The type of imputation to carry out, either "forward" or "forward-backward"
- **index**: The set of time indices containing missing values, this is NULL by default and will be determined from the input series.

Value

Returns a list containing the following elements:

- **ImputedData**: Matrix containing the imputed time series.
- **missing.index**: Vector containing the set of time indices that have missing values.

Note

As with other time series imputation methods, mv_impute requires some data values at the start of the series. In this case, we need 5 time points.
Author(s)

Rebecca Wilson

References


Examples

```r
set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

# create some fake missing data, taking care not to have missingness hear the start of the series
missing.index = sort(sample(10:2^8, 30))
X[missing.index, ] <- NA

newdata = mv_impute(X)
```

---

**pdef**

*Function to regularise the LWS matrix.*

**Description**

This function regularises each EWS matrix to ensure that they are strictly positive definite, similar to the `mvEWS` function in the `mvLSW` package, except acting on a (bias-corrected) periodogram directly. See `mvEWS` for more details. Note: this function is not really intended to be used separately, but internally within the `spec_estimation` function.

**Usage**

```r
pdef(spec, W = 1e-10)
```

**Arguments**

- `spec`  
  EWS matrix that is to be regularised, can be either a 4D array or a `mvLSW` object.

- `W`  
  Tolerance in applying matrix regularisation to ensure each EWS matrix to be strictly positive definite. This is `1e-10` by default.
Value

Returns a mvLSW object containing the regularised EWS of a multivariate locally stationary time series.

Author(s)

Rebecca Wilson

References


See Also

mvEWS, spec_estimation

Examples

```r
set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

# form periodogram
tmp = apply(X, 2, function(x){haarWT(x)$D})
D = array(t(tmp), dim = c(2, 2^8, 8))
RawPer = array(apply(D, c(2, 3), tcrossprod), dim = c(2, 2, 2^8, 8))
RawPer = aperm(RawPer, c(1, 2, 4, 3))

# now correct
correctedper = correct_per(RawPer)

# now regularize
newper = pdef(correctedper)
```
Function to form the prediction equations for the forecasting / backcasting step.

Description

This function generates the prediction equations (B matrix and RHS matrix) for one step ahead prediction.

Usage

pred_eq_forward(lacv.array, p = 2, index)

Arguments

lacv.array The local autocovariance array from which we want to form the prediction equations, can be obtained as the output of the `form_lacv_forward` or `form_lacv_backward` functions.

p Number of terms to include in the clipped predictor when forecasting / backcasting.

index Time index of the missing data which we wish to impute.

Details

The one-step ahead predictor is formed as a linear combination of the time series. The coefficients involved in optimal predictor (in the sense of minimising the mean square prediction error) are obtained by solving a matrix equation formed using parts of the (estimated) local autocovariance array. This function forms the matrices involved in the equation used to find the optimal linear predictor. See equation (6) in Wilson et al. (2021) or Section 3.3 in Fryzlewicz et al. (2003) for more details.

Value

Returns a list containing the following elements:

- **B** The left-hand side of the matrix equation to compute the optimal one-step ahead predictor, which is essentially used to approximate the MSPE for a particular set of coefficients used in a predictor.

- **RHS** The right hand side of the matrix equation used to compute the optimal one-step ahead predictor.

Author(s)

Rebecca Wilson
References


See Also

form_lacv_forward, pred_eq_backward

Examples

```r
## Sample bivariate locally stationary time series

set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

# create some missing values, taking care to provide some data at the start of the series
missing.index = sort(sample(10:2^8, 30))
X[missing.index, ] <-NA

# estimate the spectrum
spec = spec_estimation(X)

# obtain the LACV
lacvfor <- form_lacv_forward(spec$spectrum, missing.index[1], p.len = 2)

# form matrix equation terms
mspeterms = pred_eq_forward(lacvfor, p = 2, missing.index[1])

# compute the optimal coefficients in the linear predictor:
predcoeffs = solve(mspeterms$B, mspeterms$RHS)
```
smooth_per

Function to smooth the raw wavelet periodogram using the default mvLSW routine.

Description

This function smooths the raw wavelet periodogram, similar to the mvEWS function in the mvLSW package, except acting on the raw periodogram directly. See mvEWS for more details. Note: this function is not really intended to be used separately, but internally within the spec_estimation function.

Usage

smooth_per(RawPer, type = "all", kernel.name="daniell", optimize = FALSE, kernel.param = NULL, smooth.Jset = NULL)

Arguments

RawPer Raw wavelet periodogram that is to be smoothed, can be either a 4D array or a mvLSW object.

type Determines the type of smoothing to be performed, if "all" then the same smoothing kernel is applied to all levels, if "by.level" then a different smoothing kernel is applied to each level.

kernel.name Name of the smoothing kernel to be applied.

optimize Should the smoothing be optimized. If FALSE then smoothing is carried out with kernel.name and kernel.param.

kernel.param Value of the smoothing kernel parameter to be applied.

smooth.Jset Vector indicating which levels should be used in the calculation of the optimal kernel parameter. By default all levels are used.

Value

Returns a mvLSW object containing the smoothed EWS of a multivariate locally stationary time series.

Author(s)

Rebecca Wilson

References


See Also

`mvEWS`, `spec_estimation`

Examples

```r
## Sample bivariate locally stationary time series

set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

# form periodogram
tmp = apply(X, 2, function(x){haarWT(x)$D})
D = array(t(tmp), dim = c(2, 2^8, 8))

#sqrv <- function(d) return( d %*% t(d) )

#RawPer = array(apply(D, c(2, 3), sqrv), dim = c(2, 2, 2^8, 8))
RawPer = array(apply(D, c(2, 3), tcrossprod), dim = c(2, 2, 2^8, 8))
RawPer = aperm(RawPer, c(1, 2, 4, 3))

# now smooth

smoothper = smooth_per(RawPer)
```

---

`spec_estimation`  
*Function to estimate the Local Wavelet Spectral matrix for a multivariate locally stationary time series containing missing values*

**Description**

This function estimates the LWS matrix for a multivariate locally stationary time series containing missing values. If the input time series does not contain missing values then spectral estimation is carried out using routines from the `mvLSW` package.

**Usage**

```r
spec_estimation(data, interp = "linear")
```
Arguments

data Input multivariate time series, matrix of dimension TxP where P is the number of channels and T is the length of the series.

interp Method of interpolation of NAs in spectrum. Can be "linear" or "spline"; see na_interpolation for more details. See also note below.

Value

Returns a mvLSW object containing the estimated LWS matrix.

Note

For some series with a lot of missing values, the linear interpolation will result in zero periodogram values (due to the form of the Haar filters). This may not be desirable, so a higher order (spline) interpolation function may be better.

See Also

correct_per, smooth_per, mvEWS, na_interpolation

Examples

```r
## Sample bivariate locally stationary time series

set.seed(1)
X <- matrix(rnorm(2 * 2^8), ncol = 2)
X[1:2^7, 2] <- 3 * (X[1:2^7, 2] + 0.95 * X[1:2^7, 1])
X[-(1:2^7), 2] <- X[-(1:2^7), 2] - 0.95 * X[-(1:2^7), 1]
X[-(1:2^7), 1] <- X[-(1:2^7), 1] * 4
X <- as.ts(X)

# create some missing values, taking care to provide some data at the start of the series

missing.index = sort(sample(10:2^8, 30))
X[missing.index, ] <- NA

# estimate the spectrum

spec = spec_estimation(X)
```
Index

* package
  mvLSWimpute-package, 2

  correct_per, 3, 14

  form_lacv_backward(form_lacv_forward), 4

  form_lacv_forward, 4, 11

  haarWT, 6

  mv_impute, 3, 7

  mvEWS, 3, 4, 8, 9, 12–14

  mvLSWimpute (mvLSWimpute-package), 2

  mvLSWimpute-package, 2

  na_interpolation, 14

  pdef, 8

  pred_eq_backward, 5, 11

  pred_eq_backward (pred_eq_forward), 10

  pred_eq_forward, 5, 10

  smooth_per, 12, 14

  spec_estimation, 4, 9, 13, 13