Package ‘tvspecAdapt’

August 29, 2018

Type Package

Title Data-Adaptive Estimation of Time-Varying Spectral Densities

Version 1.0

Date 2018-08-14

Author Anne van Delft and Michael Eichler

Maintainer Anne van Delft <anne.vandelft@rub.de>

Description Routine to data-adaptively estimate time-varying spectral densities as described in van Delft & Eichler (2018) <doi:10.1080/10618600.2018.1512866>. The package also contains an example of how to use the routine as well as some other functions useful in time-varying spectral analysis. The routine is efficient thanks to the compiler language C++ and Openmp support.

License GPL (>= 2)

Imports Rcpp (>= 0.12.18)

LinkingTo Rcpp

RoxygenNote 6.1.0

NeedsCompilation yes

Archs i386, x64

R topics documented:

tvspecAdapt-package *Data-Adaptive Estimation of Time-Varying Spectral Densities*

Description

Contains the routine as described in van Delft & Eichler (2018) to data-adaptively estimate time-varying spectral densities as well as examples of how to use the routine. The routine is implemented mainly in the compiler language `C++` and allows for `Openmp` support.

Details

The algorithm of aforementioned paper is the main function of this package `tvSpectralEstimate`. This function relies on several auxiliary `R`- and `C++`-functions. Those functions that can be used as stand-alone functions are included in this package. For example, the `R`-functions `PreCov` and `Preperiodogram` allow for the local analysis of the second order structure of a locally stationary time series in time- and frequency domain respectively, while the `C++`-function `finitC` computes non-adaptive time-varying spectral estimates based on global bandwidths. We refer to van Delft & Eichler (2018) for details.

Author(s)

Anne van Delft <anne.vandelft@rub.de> and Michael Eichler <m.eichler@maastrichtuniversity.nl>.

Maintainer: Anne van Delft<anne.vandelft@rub.de>

References

Anne van Delft and Michael Eichler (2018). Data-Adaptive estimation of time-varying spectral densities. https://doi.org/10.1080/10618600.2018.1512866. To appear in *Journal of Computational and Graphical Statistics*.

See Also

`tvSpectralEstimate`

Examples

```
## Not run:
#Simulate from examples provided
example =1
T=256
X = tsSimulate(T,example)
#specify point in the plane
point=list(t=0.5,f=0)
#compute the time-varying spectral density estimates
res=tvSpectralEstimate(X,point=point)
# plot some results with rgl
if( "rgl" %in% rownames(installed.packages()) )
{library(rgl);
#plot of the adaptively estimated time-varying spectral density
persp3d(res$adapt.est,col="yellow");
#plot of the final adaptive kernel of \code{point}
if(!is.null(point)){persp3d(res$K,col="yellow")}
```

```
}

## End(Not run)
```

`finitC` *Auxiliary `C++`-function. Constructs the initial non-adaptive spectral density estimates using global bandwidths (see equation (3) of van Delft & Eichler (2018)). Makes use of Openmp support if enabled.*

Description

Auxiliary `C++`-function. Constructs the initial non-adaptive spectral density estimates using global bandwidths (see equation (3) of van Delft & Eichler (2018)). Makes use of Openmp support if enabled.

Usage

```
finitC(bwt, bwf, J, T)
```

Arguments

`bwt`	real value in $[0, 1]$ that specifies the bandwidth in time direction.
`bwf`	real value in $[0, 1]$ that specifies the bandwidth in frequency direction.
`J`	the periodically extended symmetric prePeriodogram (see `prePeriodogram`) which is a matrix of dimension (2T x T).
`T`	length of the time series vector X.

Value

A matrix of dimension (T x T) containing the time-varying spectral density estimates evaluated at equally spaced points (λ_j, u_s) in $(-\pi, \pi]$ x $[0, 1]$.

Author(s)

Anne van Delft <anne.vandelft@rub.de> and Michael Eichler

References

Anne van Delft and Michael Eichler (2018). Data-Adaptive estimation of time-varying spectral densities. https://doi.org/10.1080/10618600.2018.1512866. To appear in *Journal of Computational and Graphical Statistics*.

Rainer Dahlhaus. Local inference for locally sationary time series based on the empirical spectral measure. *Journal of Econometrics* 151 (2009):101-112.

Examples

```
#Simulate from examples provided
example =2
T=256
X = tsSimulate(T,example)
#compute local auto-covariance estimator
C=preCov(X)
#construct periodically extended pre-periodogram
J=prePeriodogram(C)
#compute non-adaptive spectral density kernel estimates
finit=finitC(0.05,0.05,J,length(X))
```

`preCov` *Auxiliary function. Constructs the localized auto-covariance estimator of a demeaned time series X.*

Description

Auxiliary function. Constructs the localized auto-covariance estimator of a demeaned time series X.

Usage

```
preCov(X)
```

Arguments

`X`	demeaned time series vector of length T.

Value

A matrix of dimension 2T x (T+1) providing the local auto-covariance estimator of X.

Author(s)

Anne van Delft <anne.vandelft@rub.de> and Michael Eichler

References

Anne van Delft and Michael Eichler (2018). Data-Adaptive estimation of time-varying spectral densities. https://doi.org/10.1080/10618600.2018.1512866. To appear in *Journal of Computational and Graphical Statistics*.

Examples

```
#Simulate from examples provided
example =2
T=256
X = tsSimulate(T,example)
#compute local auto-covariance estimator
C=preCov(X)
```

`prePeriodogram` *Auxiliary function. Computes (periodically extended) symmetric pre-periodogram.*

Description

Auxiliary function. Computes (periodically extended) symmetric pre-periodogram.

Usage

```
prePeriodogram(C, Pextend)
```

Arguments

`C`	matrix of dimension (2T x (T+1)) of local auto-covariances.
`Pextend`	(optional) boolean value. If `TRUE` then the pre-periodogram is periodically extended in frequency direction. If no value is provided, it defaults to `TRUE`.

Value

The symmetric pre-periodogram of dimension (2T x T), where the first dimension is frequency and the second localized time evaluated at equally spaced points (λ_j, u_s) in $(-2\pi, 2\pi]$ x $[0, 1]$. If `Pextend=FALSE` then it returns a matrix of dimension (T x T) where the values are equally spaced in $(-\pi, \pi]$ x $[0, 1]$.

Author(s)

Anne van Delft <anne.vandelft@rub.de> and Michael Eichler

References

Anne van Delft and Michael Eichler (2018). Data-Adaptive estimation of time-varying spectral densities. https://doi.org/10.1080/10618600.2018.1512866. To appear in *Journal of Computational and Graphical Statistics*.

Michael H. Neumann and Rainer von Sachs. Wavelet Thresholding in Anisotropic Function Classes and Application to Adaptive Estimation of Evolutionary Spectra. *The Annals of Statistics* 25.1 (1997):38-76.

Examples

```
#Simulate from examples provided
example =2
T=256
X = tsSimulate(T,example)
#compute local auto-covariance estimator
C=preCov(X)
#construct periodically extended pre-periodogram
Jper=prePeriodogram(C)
```

`trueSpectrum` *Example function. Computes the true time-varying spectrum from one of three example processes taken from van Delft & Eichler (2018).*

Description

Example function. Computes the true time-varying spectrum from one of three example processes taken from van Delft & Eichler (2018).

Usage

```
trueSpectrum(T, example = 1)
```

Arguments

`T`	positive integer. Length of the time series vector.
`example`	integer in $\{1, 2, 3\}$. Determines for which example process (See `tsSimulate`) the true time-varying spectrum is created.

Value

A matrix of dimension (T x T). The true time-varying spectrum of a process with data-generating process `example` evaluated at equally spaced points (λ_j, u_s) in $(-\pi, \pi]$ x $[0, 1]$.

Author(s)

Anne van Delft <anne.vandelft@rub.de> and Michael Eichler

References

Anne van Delft and Michael Eichler (2018). Data-Adaptive estimation of time-varying spectral densities. https://doi.org/10.1080/10618600.2018.1512866. To appear in *Journal of Computational and Graphical Statistics*.

`tsSimulate` *Example function. Simulates one of the three example processes from van Delft & Eichler (2018)*

Description

Example function. Simulates one of the three example processes from van Delft & Eichler (2018)

Usage

```
tsSimulate(T, example = 1)
```

Arguments

`T` positive integer. Length of the time series vector.

`example` integer in $\{1, 2, 3\}$. Determines from which of the three data generating processes is simulated.

- `example=1` creates white noise with break. See section 4.1 of van Delft & Eichler (2018).
- `example=2` creates time-varying moving average of order 1. See section 4.2 van Delft & Eichler (2018).
- `example=2` creates time-varying moving average of order 1 with break. See section 4.3 van Delft & Eichler (2018).

Value

A demeaned time series vector of length T generated from chosen example process.

Author(s)

Anne van Delft <anne.vandelft@rub.de> and Michael Eichler

References

Anne van Delft and Michael Eichler (2018). Data-Adaptive estimation of time-varying spectral densities. https://doi.org/10.1080/10618600.2018.1512866. To appear in *Journal of Computational and Graphical Statistics*.

`tvSpectralEstimate` *MAIN FUNCTION. computes for time series* `X` *the time-varying spectral estimates over* $(-\pi, \pi]$ *x* $[0, 1]$ *with data-adaptive kernels described in van Delft & Eichler (2018). Makes use of Openmp if enabled.*

Description

MAIN FUNCTION. computes for time series `X` the time-varying spectral estimates over $(-\pi, \pi]$ x $[0, 1]$ with data-adaptive kernels described in van Delft & Eichler (2018). Makes use of Openmp if enabled.

Usage

```
tvSpectralEstimate(X, b0t, b0f, control = list(), point = NULL)
```

Arguments

`X` time series vector of length T

`b0t` real value in $[0, 1]$ that specifies the bandwidth in time direction of the initial kernel spectral estimator. If `b0t` is missing then its value is set to a conservative value based on the asymptotic rate criterion `b0t` $= \sqrt{log(T)^{1.9}/2\pi T}$. See section 3.1 of van Delft & Eichler (2018).

`b0f` real value in $[0, 1]$ that specifies the bandwidth in frequency direction of the initial kernel spectral estimator. If `b0f` is missing then its value is set to a conservative value based on the asymptotic rate criterion `b0f` $= \sqrt{log(T)^{1.9}/2\pi T}$. See section 3.1 of van Delft & Eichler (2018).

`control` A list of control parameters consisting of the following components.

- `growt`: exponential growth rate in time direction (default is 1.2).
- `growf`: exponential growth rate in frequency direction (default value is 1.2).
- `penaltycut`: cut-off value for the penalty kernel. The default is $\chi^2_{1,0.9}$. This is based on the asymptotic distribution of the discrepancy statistic, which is χ^2_1-distributed under homogeneity (See the online supplement of van Delft & Eichler (2018)).
- `snrcut`: cut-off value for the the local signal-to-noise ratio in the relaxation parameter (see Section 3.1 (step (e),\,(h)\,(i)) of van Delft & Eichler (2018)). If the local signal to noise ratio computed is less than `snrcut` full weight is given to local smoothed version of the estimator. The lower the value the less smoothing is imposed. In order to distinguish signal from noise `snrcut` should be set considerably larger than 1. The default is set to 2.
- `improvebw`: boolean value that specifies whether the algorithm will look for a larger starting bandwidth. This option allows to reduce computational complexity and can improve stability. The default is set to `TRUE`. In case there is evidence signals are very localized (e.g., plot of the preperiodogram or prior knowledge of the problem) the user is advised to turn it to `FALSE` and to provide the starting bandwidths. The presence and magnitude of negative values in the initial estimates are used as an indication of stability and could imply undersmoothing. If set to `TRUE` the algorithm looks for a slightly larger starting bandwidth based upon the bandwidth specified on `b0t` and `growt`. More specifically, the initial estimates are recomputed with increasingly larger starting bandwidths $b0t_i$=$b0f_i$ where

$$b0t_i = b0f_i = b0t \cdot growt^{0.5i}$$

  for $i = \{1, 2, 3, 4\}$, until magnitude and percentage of negative cross terms is less than the control parameter `cross`.
- `cross`: real number in the interval $[0, 1]$. If `improvebw` is set to `TRUE`, this value specifies the percentage of negative cross terms and their magnitude that is acceptable for the user. The lower the value the smoother the intitial set of estimates. The default is set to 0.025.
- `kmax`: maximum number of iterations, defaults to `kmax`=$\lceil -\frac{log(b0t)}{log(growt)} \rceil$ which allows to search the entire plane for each design point (see Section 3.2.4 of van Delft & Eichler (2018)). The algorithm breaks earlier if further smoothing does not lead to improvement of the estimates due to violation of local homogeneity.

`point` (optional) list of two components `t` in $([0, 1])$ and `f` in $([0, \pi])$ that specify the time and frequency coordinate of a point for which the penalty and the smoothing kernel at the final iteration is returned. It defaults to `NULL` in which case no penalty or smoothing kernel are returned.

Details

NB: At the moment the parallelization does not work properly for large sample sizes without the use of 'Variable length arrays'. Therefore, `tvSpectralEstimate()` checks if the compiler is either g++ or clang, which are known compilers that support this c++-extension, and only allow for the use of openmp if either of the two is detected. Otherwise a single threaded version of the code is imposed.

Value

adapt.est	a matrix of dimension (T x T). The final adaptive estimates at equally spaced points (λ_j, u_s) in $(-\pi, \pi]$ x $[0, 1]$.
adapt.est.iter	array of intermediate adaptive estimates of dimension (T x T x kmax).
k	final iteration of final adaptive estimates.
K_p	penalty kernel at final iteration (returned only if point!=NULL).
K	smoothing kernel at final iteration (returned only if point!=NULL).

Author(s)

Anne van Delft <anne.vandelft@rub.de> and Michael Eichler

References

Anne van Delft and Michael Eichler (2018). Data-Adaptive estimation of time-varying spectral densities. https://doi.org/10.1080/10618600.2018.1512866. To appear in *Journal of Computational and Graphical Statistics*.

Examples

```
## Not run:
#Simulate from examples provided
example =1
T=256
X = tsSimulate(T,example)
#specify point in the plane
point=list(t=0.5,f=0)
#compute the time-varying spectral density estimates
res=tvSpectralEstimate(X,point=point)
# plot some results with rgl
if( "rgl" %in% rownames(installed.packages()) )
{library(rgl);
#plot of the adaptively estimated time-varying spectral density
persp3d(res$adapt.est,col="yellow");
#plot of the final adaptive kernel of \code{point}
if(!is.null(point)){persp3d(res$K,col="yellow")}
}

## End(Not run)
```

Index