Package ‘nse’

September 13, 2018

Version 1.19
Date 2018-08-31
Title Numerical Standard Errors Computation in R
Author David Ardia [aut],
Keven Bluteau [aut, cre]
Maintainer Keven Bluteau <keven.bluteau@unine.ch>
License GPL (>= 2)
BugReports https://github.com/keblu/nse/issues
URL https://github.com/keblu/nse
Imports Rcpp (>= 0.12.0), coda, mcmc, mcmcse, np, sandwich
LinkingTo Rcpp
Encoding UTF-8
NeedsCompilation yes
RoxygenNote 6.1.0
Suggests testthat
Repository CRAN
Date/Publication 2018-09-13 14:50:10 UTC

R topics documented:

nse ................................................................. 2
nse.andrews .................................................... 3
nse.boot ......................................................... 5
nse.cos .......................................................... 6
nse.geyer ........................................................ 7
nse.hiruk ........................................................ 8
nse.nw ............................................................ 9
nse.spec0 ......................................................... 10
Description

*nse* (Ardia and Bluteau, 2017) is an R package for computing the numerical standard error (NSE), an estimate of the standard deviation of a simulation result, if the simulation experiment were to be repeated many times. The package provides a set of wrappers around several R packages, which give access to more than thirty NSE estimators, including batch means estimators (Geyer, 1992, Section 3.2), initial sequence estimators Geyer (1992, Equation 3.3), spectrum at zero estimators (Heidelberger and Welch, 1981), heteroskedasticity and autocorrelation consistent (HAC) kernel estimators (Newey and West, 1987; Andrews, 1991; Andrews and Monahan, 1992; Newey and West, 1994; Hirukawa, 2010), and bootstrap estimators Politis and Romano (1992, 1994); Politis and White (2004). The full set of estimators is described in Ardia et al. (2018).

Functions

- **nse.geyer**: Geyer NSE estimator.
- **nse.spec0**: Spectral density at zero NSE estimator.
- **nse.nw**: Newey-West NSE estimator.
- **nse.andrews**: Andrews NSE estimator.
- **nse.hiruk**: Hirukawa NSE estimator.
- **nse.boot**: Bootstrap NSE estimator.

Note

Functions rely on the packages *coda*, *mcmc*, *mcmcse*, *np*, and *sandwich*.

Please cite the package in publications. Use `citation("nse")`.

Author(s)

David Ardia and Keven Bluteau

References


**See Also**

Useful links:
- [https://github.com/keblu/nse](https://github.com/keblu/nse)

---

### nse.andrews

**Andrews estimator**

**Description**

Function which calculates the numerical standard error with the kernel based variance estimator by Andrews (1991).

**Usage**

```r
nse.andrews(x, type = c("bartlett", "parzen", "tukey", "qs", "trunc"),
lag.prewhite = 0, approx = c("AR(1)", "ARMA(1,1)"))
```

**Arguments**

- `x` A numeric vector.
- `type` The type of kernel used among which "bartlett", "parzen", "qs", "trunc" and "tukey". Default is type = "bartlett".
- `lag.prewhite` Prewhite the series before analysis (integer or NULL). When lag.prewhite = NULL this performs automatic lag selection. Default is lag.prewhite = 0 that is no prewhitening.
- `approx` Andrews approximation, either "AR(1)" or "ARMA(1,1)". Default is approx = "AR(1)".
Details
This kernel based variance estimation apply weight to the auto-covariance function with a kernel and sums up the value.

Value
The NSE estimator.

Note
nse.andrews is a wrapper around lrvar from the sandwich package and uses Andrews (1991) automatic bandwidth estimator. See the documentation of sandwich for details.

Author(s)
David Ardia and Keven Bluteau

References


Examples
```r
n = 1000
ar = 0.9
mean = 1
sd = 1

set.seed(1234)
x = as.vector(arima.ssim(n = n, list(ar = ar), sd = sd) + mean)

nse.andrews(x = x, type = "parzen", lag.prewhite = 0)
nse.andrews(x = x, type = "tukey", lag.prewhite = 1)
nse.andrews(x = x, type = "qs", lag.prewhite = NULL)
```
**Description**

Function which calculates the numerical standard error with bootstrap estimator.

**Usage**

```r
nse.boot(x, nb, type = c("stationary", "circular"), b = NULL, 
lag.prewhite = 0)
```

**Arguments**

- `x`: A numeric vector.
- `nb`: The number of bootstrap replications.
- `type`: The bootstrap scheme used, among "stationary" and "circular". Default is `type = "stationary"`. 
- `b`: The block length for the block bootstrap. If NULL automatic block length selection. Default is `b = NULL`. 
- `lag.prewhite`: Prewhite the series before analysis (integer or NULL). When `lag.prewhite = NULL` this performs automatic lag selection. Default is `lag.prewhite = 0` that is no prewhitening.

**Value**

The NSE estimator.

**Note**

`nse.boot` uses `b.star` of the `np` package for the optimal block length selection.

**Author(s)**

David Ardia and Keven Bluteau

**References**


Examples

```r
n = 1000
ar = 0.9
mean = 1
sd = 1

set.seed(1234)
x = as.vector(arima.sim(n = n, list(ar = ar), sd = sd) + mean)

set.seed(1234)
nse.boot(x = x, nb = 1000, type = "stationary", b = NULL, lag.prewhite = 0)
nse.boot(x = x, nb = 1000, type = "circular", b = NULL, lag.prewhite = NULL)
nse.boot(x = x, nb = 1000, type = "circular", b = 10, lag.prewhite = NULL)
```

---

**nse.cos**

Long-run variance estimation using low-frequency cosine series.

**Description**

Function which calculates the numerical standard error with low-frequency cosine weighted averages of the original serie.

**Usage**

```r
nse.cos(x, q = 12, lag.prewhite = 0)
```

**Arguments**

- `x`: A numeric vector.
- `q`: Number of cosine series.
- `lag.prewhite`: Prewhite the series before analysis (integer or NULL). When `lag.prewhite = NULL` this performs automatic lag selection. Default is `lag.prewhite = 0` that is no prewhitening.

**Details**

The method estimate the series with a linear regression using cosine low frequency series. It than derived the NSE from the coefficient of the cosine series (Ulrich and Watson, 2017).

**Value**

The NSE estimator.

**Author(s)**

David Ardia and Keven Bluteau
nse.geyer

References

Examples
n = 1000
ar = 0.9
mean = 1
sd = 1
set.seed(1234)
x = as.vector(arima.sim(n = n, list(ar = ar), sd = sd) + mean)
nse.cos(x = x, q = 12, lag.prewhite = 0)
#’ nse.cos(x = x, q = 12, lag.prewhite = NULL)

nse.geyer  Geyer estimator

Description
Function which calculates the numerical standard error with the method of Geyer (1992).

Usage
nse.geyer(x, type = c("iseq", "bm", "obm", "iseq.bm"), nbatch = 30,
iseq.type = c("pos", "dec", "con"))

Arguments

x  A numeric vector.
type  The type which can be either "iseq", "bm", "obm" or "iseq.bm". See *Details*. Default is type = "iseq".
nbatch  Number of batches when type = "bm" and type = "iseq.bm". Default is nbatch = 30.
iseq.type  Constraints on function: "pos" for nonnegative, "dec" for nonnegative and non-increasing, and "con" for nonnegative, nonincreasing, and convex. Default is iseq.type = "pos".

Details
The type "iseq" gives the positive initial sequence estimator, "bm" is the batch mean estimator, "obm" is the overlapping batch mean estimator and "iseq.bm" is a combination of "iseq" and "bm".

Value
The NSE estimator.
Note

nse.geyer relies on the packages mcmc and mcmcse; see the documentation of these packages for more details.

Author(s)

David Ardia and Keven Bluteau

References


Examples

```r
n = 1000
ar = 0.9
mean = 1
sd = 1
set.seed(1234)
x = as.vector(arima.sim(n = n, list(ar = ar), sd = sd) + mean)
nse.geyer(x = x, type = "bm", nbatch = 30)
nse.geyer(x = x, type = "obm", nbatch = 30)
nse.geyer(x = x, type = "iseq", iseq.type = "pos")
nse.geyer(x = x, type = "iseq.bm", iseq.type = "con")
```

nse.hiruk  

Hirukawa estimator

Description


Usage

```r
nse.hiruk(x, type = c("bartlett", "parzen"), lag.prewhite = 0)
```

Arguments

- **x**: A numeric vector.
- **type**: The type of kernel used among "bartlett" and "parzen". Default is type = "Bartlett".
- **lag.prewhite**: Prewhite the series before analysis (integer or NULL). When lag.prewhite = NULL this performs automatic lag selection. Default is lag.prewhite = 0 that is no prewhitening.

Value

The NSE estimator.
Note
nse.hiruk is a wrapper around lrvar from the sandwich package and uses Hirukawa (2010) bandwidth estimator. See the documentation of sandwich for details.

Author(s)
David Ardia and Keven Bluteau

References

Examples
n = 1000
ar = 0.9
mean = 1
sd = 1

set.seed(1234)
x = as.vector(arima.sim(n = n, list(ar = ar), sd = sd) + mean)
nse.hiruk(x = x, type = "parzen", lag.prewhite = 0)
nse.hiruk(x = x, type = "bartlett", lag.prewhite = NULL)

Description
Function which calculates the numerical standard error with the Newey West (1987, 1994) HAC estimator.

Usage
nse.nw(x, lag.prewhite = 0)

Arguments
x A numeric vector
lag.prewhite Prewhiten the series before analysis (integer or NULL). When lag.prewhite = NULL this performs automatic lag selection. Default is lag.prewhite = 0 that is no prewhitening.

Value
The NSE estimator.
Note

nse.nw is a wrapper around lrvar from the sandwich package. See the documentation of sandwich for details.

Author(s)

David Ardia and Keven Bluteau

References


Examples

```r
n = 1000
ar = 0.9
mean = 1
sd = 1

set.seed(1234)
x = as.vector(arima.sim(n = n, list(ar = ar), sd = sd) + mean)

nse.nw(x, lag.prewhite = 0)
nse.nw(x, lag.prewhite = 1)
nse.nw(x, lag.prewhite = NULL)
```

---

**nse.spec0**  
*Spectral density at zero estimator*

**Description**

Function which calculates the numerical standard error with the spectrum at zero estimator.

**Usage**

```r
nse.spec0(x, type = c("ar", "glm", "daniell", "modified.daniell", "tukey-hanning", "parzen", "triweight", "bartlett-priestley", "triangular", "qs"), lag.prewhite = 0, welch = FALSE, steep = FALSE)
```
Arguments

x A numeric vector.

type Method to use in estimating the spectral density function, among "ar", "glm", "daniell", "modified.daniell", "tukey-hanning", "parzen", "triweight", "bartlett-priestley", "triangular", and "qs". See *Details*. Default is type = "ar".

lag.prewhite Prewhite the series before analysis (integer or NULL). When lag.prewhite = NULL this performs automatic lag selection. Default is lag.prewhite = 0 that is no prewhitening.

welch Use Welch’s method (Welsh, 1967) to estimate the spectral density.

steep Use steep or sharp version of the kernel (Phillips et al., 2006) (only available for type: "qs","triangular", and "parzen"). lag.prewhite must be set to 0 to use steep version.

Details

Welsh’s method use 50% overlap and 8 sub-samples. The method "ar" estimates the spectral density using an autoregressive model, "glm" using a generalized linear model Heidelberger & Welch (1981), "daniell" uses daniell window from the R kernel function, "modified.daniell" uses daniell window the R kernel function, "tukey-hanning" uses the tukey-hanning window, "parzen" uses the parzen window, "triweight" uses the triweight window, "bartlett-priestley" uses the Bartlett-Priestley window, "triangular" uses the triangular window, and "qs" uses the quadratic-spectral window.

This kernel based variance estimator apply weights to smooth out the spectral density using a kernel and takes the spectral density at frequency zero which is equivalent to the variance of the serie. Bandwidth for the kernel is automatically selected using cross-validatory methods (Hurvich, 1985).

Value

The NSE estimator.

Note

nse.spec0 relies on the packages coda; see the documentation of this package for more details.

Author(s)

David Ardia and Keven Bluteau

References


**Examples**

```r
n = 1000
ar = 0.9
mean = 1
sd = 1
set.seed(1234)
x = as.vector(arima.sim(n = n, list(ar = ar), sd = sd) + mean)
nse.spec0(x = x, type = "parzen", lag.prewhte = 0, welch = TRUE, steep = TRUE)
```
Index

b.star, 5

lrvar, 4, 9, 10

mcmc, 8
mcmcse, 8

np, 5
nse, 2
nse-package (nse), 2
nse.andrews, 2, 3
nse.boot, 2, 5
nse.cos, 6
nse.geyer, 2, 7
nse.hiruk, 2, 8
nse.nw, 2, 9
nse.spec0, 2, 10

sandwich, 4, 9, 10