Package ‘RTransferEntropy’

January 20, 2025

Type Package

Title Measuring Information Flow Between Time Series with Shannon and Renyi Transfer Entropy

Version 0.2.21

Description Measuring information flow between time series with Shannon and Rényi transfer entropy. See also Dimpfl and Peter (2013) <doi:10.1515/snde-2012-0044> and Dimpfl and Peter (2014) <doi:10.1016/j.intfin.2014.03.004> for theory and applications to financial time series. Additional references can be found in the theory part of the vignette.

License GPL-3

URL `https://github.com/BZPaper/RTransferEntropy`

BugReports `https://github.com/BZPaper/RTransferEntropy/issues`

Encoding UTF-8

Depends R (>= 3.1.2)

Imports future (>= 1.19.0), future.apply, Rcpp

LazyData true

RoxygenNote 7.2.0

LinkingTo Rcpp

Suggests data.table, ggplot2, gridExtra, knitr, quantmod, rmarkdown, testthat, vars, xts, zoo

VignetteBuilder knitr

NeedsCompilation yes

Author David Zimmermann [aut, cre],
Simon Behrendt [aut],
Thomas Dimpfl [aut],
Franziska Peter [aut]

Maintainer David Zimmermann <`david_j_zimmermann@hotmail.com`>

Repository CRAN

Date/Publication 2023-02-01 17:30:05 UTC

Contents

calc_ete *Calculates the Effective Transfer Entropy for two time series*

Description

Calculates the Effective Transfer Entropy for two time series

Usage

```
calc_ete(
  x,
  y,
  lx = 1,
  ly = 1,
  q = 0.1,
  entropy = "Shannon",
  shuffles = 100,
  type = "quantiles",
  quantiles = c(5, 95),
  bins = NULL,
  limits = NULL,
  burn = 50,
  seed = NULL,
  na.rm = TRUE
)
```

Arguments

x	a vector of numeric values, ordered by time. Also allowed are xts, zoo, or ts objects.
y	a vector of numeric values, ordered by time. Also allowed are xts, zoo, or ts objects.
lx	Markov order of x, i.e. the number of lagged values affecting the current value of x. Default is lx = 1.

`ly` Markov order of y, i.e. the number of lagged values affecting the current value of y. Default is `ly = 1`.

`q` a weighting parameter used to estimate Renyi transfer entropy, parameter is between 0 and 1. For `q = 1`, Renyi transfer entropy converges to Shannon transfer entropy. Default is `q = 0.1`.

`entropy` specifies the transfer entropy measure that is estimated, either 'Shannon' or 'Renyi'. The first character can be used to specify the type of transfer entropy as well. Default is `entropy = 'Shannon'`.

`shuffles` the number of shuffles used to calculate the effective transfer entropy. Default is `shuffles = 100`.

`type` specifies the type of discretization applied to the observed time series:'quantiles', 'bins' or 'limits'. Default is `type = 'quantiles'`.

`quantiles` specifies the quantiles of the empirical distribution of the respective time series used for discretization. Default is `quantiles = c(5,95)`.

`bins` specifies the number of bins with equal width used for discretization. Default is `bins = NULL`.

`limits` specifies the limits on values used for discretization. Default is `limits = NULL`.

`burn` the number of observations that are dropped from the beginning of the bootstrapped Markov chain. Default is `burn = 50`.

`seed` a seed that seeds the PRNG (will internally just call set.seed), default is `seed = NULL`.

`na.rm` if missing values should be removed (will remove the values at the same point in the other series as well). Default is `TRUE`.

Value

a single numerical value for the effective transfer entropy

See Also

`calc_te` and`transfer_entropy`

Examples

```
# construct two time-series
set.seed(1234567890)
n <- 1000
x <- rep(0, n + 1)
y <- rep(0, n + 1)

for (i in seq(n)) {
  x[i + 1] <- 0.2 * x[i] + rnorm(1, 0, 2)
  y[i + 1] <- x[i] + rnorm(1, 0, 2)
}

x <- x[-1]
y <- y[-1]
```

```
# calculate the X->Y transfer entropy value
calc_ete(x, y)

# calculate the Y->X transfer entropy value
calc_ete(y, x)


  # Compare the results
  # even with the same seed, transfer_entropy might return slightly different
  # results from calc_ete
  calc_ete(x, y, seed = 123)
  calc_ete(y, x, seed = 123)
  transfer_entropy(x, y, nboot = 0, seed = 123)
```

calc_te *Calculates the Transfer Entropy for two time series*

Description

Calculates the Transfer Entropy for two time series

Usage

```
calc_te(
  x,
  y,
  lx = 1,
  ly = 1,
  q = 0.1,
  entropy = "Shannon",
  shuffles = 100,
  type = "quantiles",
  quantiles = c(5, 95),
  bins = NULL,
  limits = NULL,
  burn = 50,
  seed = NULL,
  na.rm = TRUE
)
```

Arguments

`x` a vector of numeric values, ordered by time. Also allowed are `xts`, `zoo`, or `ts` objects.

`y` a vector of numeric values, ordered by time. Also allowed are `xts`, `zoo`, or `ts` objects.

`lx` Markov order of x, i.e. the number of lagged values affecting the current value of x. Default is `lx = 1`.

`ly` Markov order of y, i.e. the number of lagged values affecting the current value of y. Default is `ly = 1`.

`q` a weighting parameter used to estimate Renyi transfer entropy, parameter is between 0 and 1. For `q = 1`, Renyi transfer entropy converges to Shannon transfer entropy. Default is `q = 0.1`.

`entropy` specifies the transfer entropy measure that is estimated, either 'Shannon' or 'Renyi'. The first character can be used to specify the type of transfer entropy as well. Default is `entropy = 'Shannon'`.

`shuffles` the number of shuffles used to calculate the effective transfer entropy. Default is `shuffles = 100`.

`type` specifies the type of discretization applied to the observed time series:'quantiles', 'bins' or 'limits'. Default is `type = 'quantiles'`.

`quantiles` specifies the quantiles of the empirical distribution of the respective time series used for discretization. Default is `quantiles = c(5,95)`.

`bins` specifies the number of bins with equal width used for discretization. Default is `bins = NULL`.

`limits` specifies the limits on values used for discretization. Default is `limits = NULL`.

`burn` the number of observations that are dropped from the beginning of the bootstrapped Markov chain. Default is `burn = 50`.

`seed` a seed that seeds the PRNG (will internally just call set.seed), default is `seed = NULL`.

`na.rm` if missing values should be removed (will remove the values at the same point in the other series as well). Default is `TRUE`.

Value

a single numerical value for the transfer entropy

See Also

`calc_ete` and`transfer_entropy`

Examples

```
# construct two time-series
set.seed(1234567890)
n <- 1000
x <- rep(0, n + 1)
y <- rep(0, n + 1)

for (i in seq(n)) {
  x[i + 1] <- 0.2 * x[i] + rnorm(1, 0, 2)
  y[i + 1] <- x[i] + rnorm(1, 0, 2)
}
```

```
x <- x[-1]
y <- y[-1]

# calculate the X->Y transfer entropy value
calc_te(x, y)

# calculate the Y->X transfer entropy value
calc_te(y, x)


  # Compare the results
  calc_te(x, y, seed = 123)
  calc_te(y, x, seed = 123)
  transfer_entropy(x, y, nboot = 0, seed = 123)
```

coef.transfer_entropy *Extract the Coefficient Matrix from a transfer_entropy*

Description

Extract the Coefficient Matrix from a transfer_entropy

Usage

```
## S3 method for class 'transfer_entropy'
coef(object, ...)
```

Arguments

object	a transfer_entropy
...	additional arguments, currently not in use

Value

a Matrix containing the coefficients

Examples

```
set.seed(1234567890)
n <- 500
x <- rep(0, n + 1)
y <- rep(0, n + 1)

for (i in seq(n)) {
  x[i + 1] <- 0.2 * x[i] + rnorm(1, 0, 2)
  y[i + 1] <- x[i] + rnorm(1, 0, 2)
}
```

```
x <- x[-1]
y <- y[-1]

te_result <- transfer_entropy(x, y, nboot = 100)
coef(te_result)
```

`is.transfer_entropy` *Checks if an object is a transfer_entropy*

Description

Checks if an object is a transfer_entropy

Usage

```
is.transfer_entropy(x)
```

Arguments

`x`	an object

Value

a boolean value if x is a transfer_entropy

Examples

```
# see ?transfer_entropy
```

`print.transfer_entropy` *Prints a transfer-entropy result*

Description

Prints a transfer-entropy result

Usage

```
## S3 method for class 'transfer_entropy'
print(
  x,
  digits = 4,
  boot = TRUE,
  probs = c(0, 0.25, 0.5, 0.75, 1),
  tex = FALSE,
  ref = NA,
  file = NA,
  table = TRUE,
  ...
)
```

Arguments

`x`	a transfer_entropy
`digits`	the number of digits to display, defaults to 4
`boot`	if the bootstrapped results should be printed, defaults to TRUE
`probs`	numeric vector of quantiles for the bootstraps
`tex`	if the data should be outputted as a TeX-string
`ref`	the reference string of the LaTeX table (label) applies only if table = TRUE and tex = TRUE, defaults to FALSE
`file`	a file where the results are printed to
`table`	if the table environment should be printed as well (only applies if tex = TRUE), defaults to TRUE
`...`	additional arguments, currently not in use

Value

invisible the text

Examples

```
# construct two time-series
set.seed(1234567890)
n <- 500
x <- rep(0, n + 1)
y <- rep(0, n + 1)

for (i in seq(n)) {
  x[i + 1] <- 0.2 * x[i] + rnorm(1, 0, 2)
  y[i + 1] <- x[i] + rnorm(1, 0, 2)
}

x <- x[-1]
y <- y[-1]
```

```
# Calculate Shannon's Transfer Entropy
te_result <- transfer_entropy(x, y, nboot = 100)

print(te_result)

# change the number of digits
print(te_result, digits = 10)

# disable boot-print
print(te_result, boot = FALSE)

# specify the quantiles of the bootstraps
print(te_result, probs = c(0, 0.1, 0.4, 0.5, 0.6, 0.9, 1))

# get LaTeX output:
print(te_result, tex = TRUE)

# set the reference label for LaTeX table
print(te_result, tex = TRUE, ref = "tab:te_result")

## Not run:
# file output
print(te_result, file = "te_result_file.txt")
print(te_result, tex = TRUE, file = "te_result_file.tex")

## End(Not run)
```

set_quiet *Set the quiet-parameter for all RTransferEntropy Calls*

Description

Set the quiet-parameter for all RTransferEntropy Calls

Usage

```
set_quiet(quiet)
```

Arguments

quiet	if FALSE, the functions will give feedback on the progress

Value

nothing

Examples

```
# see ?transfer_entropy
```

stocks *Daily stock data for 10 stocks from 2000-2017*

Description

A dataset containing the daily stock returns for 10 stocks and the S&P 500 market returns for the time-period 2000-01-04 until 2017-12-29

Usage

```
stocks
```

Format

A data frame (or data.table if loaded) with 46940 rows and 4 variables:

date date of the observation

ticker ticker of the stock

ret Return of the stock

sp500 Return of the S&P 500 stock market index

Source

yahoo finance using `getSymbols`

summary.transfer_entropy *Prints a summary of a transfer-entropy result*

Description

Prints a summary of a transfer-entropy result

Usage

```
## S3 method for class 'transfer_entropy'
summary(object, digits = 4, probs = c(0, 0.25, 0.5, 0.75, 1), ...)
```

Arguments

`object`	a transfer_entropy
`digits`	the number of digits to display, defaults to 4
`probs`	numeric vector of quantiles for the bootstraps
`...`	additional arguments, passed to `printCoefmat`

Value

invisible the object

Examples

```
# construct two time-series
set.seed(1234567890)
n <- 500
x <- rep(0, n + 1)
y <- rep(0, n + 1)

for (i in seq(n)) {
  x[i + 1] <- 0.2 * x[i] + rnorm(1, 0, 2)
  y[i + 1] <- x[i] + rnorm(1, 0, 2)
}

x <- x[-1]
y <- y[-1]

# Calculate Shannon's Transfer Entropy
te_result <- transfer_entropy(x, y, nboot = 100)

summary(te_result)
```

`transfer_entropy` *Function to estimate Shannon and Renyi transfer entropy between two time series x and y.*

Description

Function to estimate Shannon and Renyi transfer entropy between two time series x and y.

Usage

```
transfer_entropy(
  x,
  y,
  lx = 1,
  ly = 1,
  q = 0.1,
  entropy = "Shannon",
  shuffles = 100,
  type = "quantiles",
  quantiles = c(5, 95),
  bins = NULL,
  limits = NULL,
  nboot = 300,
  burn = 50,
  quiet = NULL,
  seed = NULL,
  na.rm = TRUE
)
```

Arguments

`x` a vector of numeric values, ordered by time. Also allowed are `xts`, `zoo`, or `ts` objects.

`y` a vector of numeric values, ordered by time. Also allowed are `xts`, `zoo`, or `ts` objects.

`lx` Markov order of x, i.e. the number of lagged values affecting the current value of x. Default is `lx = 1`.

`ly` Markov order of y, i.e. the number of lagged values affecting the current value of y. Default is `ly = 1`.

`q` a weighting parameter used to estimate Renyi transfer entropy, parameter is between 0 and 1. For `q = 1`, Renyi transfer entropy converges to Shannon transfer entropy. Default is `q = 0.1`.

`entropy` specifies the transfer entropy measure that is estimated, either 'Shannon' or 'Renyi'. The first character can be used to specify the type of transfer entropy as well. Default is `entropy = 'Shannon'`.

`shuffles` the number of shuffles used to calculate the effective transfer entropy. Default is `shuffles = 100`.

`type` specifies the type of discretization applied to the observed time series:'quantiles', 'bins' or 'limits'. Default is `type = 'quantiles'`.

`quantiles` specifies the quantiles of the empirical distribution of the respective time series used for discretization. Default is `quantiles = c(5,95)`.

`bins` specifies the number of bins with equal width used for discretization. Default is `bins = NULL`.

`limits` specifies the limits on values used for discretization. Default is `limits = NULL`.

`nboot` the number of bootstrap replications for each direction of the estimated transfer entropy. Default is `nboot = 300`.

`burn` the number of observations that are dropped from the beginning of the bootstrapped Markov chain. Default is `burn = 50`.

`quiet` if FALSE (default), the function gives feedback.

`seed` a seed that seeds the PRNG (will internally just call set.seed), default is `seed = NULL`.

`na.rm` if missing values should be removed (will remove the values at the same point in the other series as well). Default is `TRUE`.

Value

an object of class transfer_entropy, containing the transfer entropy estimates in both directions, the effective transfer entropy estimates in both directions, standard errors and p-values based on bootstrap replications of the Markov chains under the null hypothesis of statistical independence, an indication of statistical significance, and quantiles of the bootstrap samples (if `nboot > 0`).

See Also

coef, print.transfer_entropy

Examples

```
# construct two time-series
set.seed(1234567890)
n <- 500
x <- rep(0, n + 1)
y <- rep(0, n + 1)

for (i in seq(n)) {
  x[i + 1] <- 0.2 * x[i] + rnorm(1, 0, 2)
  y[i + 1] <- x[i] + rnorm(1, 0, 2)
}

x <- x[-1]
y <- y[-1]

# Calculate Shannon's Transfer Entropy
te_result <- transfer_entropy(x, y, nboot = 100)
te_result

summary(te_result)


  # Parallel Processing using the future-package
  library(future)
  plan(multisession)

  te_result2 <- transfer_entropy(x, y, nboot = 100)
  te_result2

  # revert back to sequential execution
  plan(sequential)

  te_result2 <- transfer_entropy(x, y, nboot = 100)
  te_result2

  # General set of quiet
  set_quiet(TRUE)
  a <- transfer_entropy(x, y, nboot = 0)

  set_quiet(FALSE)
  a <- transfer_entropy(x, y, nboot = 0)

  # close multisession, see also ?plan
  plan(sequential)
```

Index