

# Package ‘RTransferEntropy’

September 21, 2018

**Type** Package

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

**Version** 0.2.7

**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, future.apply, Rcpp

**LazyData** true

**RoxygenNote** 6.1.0

**LinkingTo** Rcpp

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

**VignetteBuilder** knitr

**NeedsCompilation** yes

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

**Maintainer** Simon Behrendt <simon.behrendt@zu.de>

**Repository** CRAN

**Date/Publication** 2018-09-21 16:00:03 UTC

## R topics documented:

calc_ete . . . . .	2
calc_te . . . . .	4
coef.transfer_entropy . . . . .	5
is.transfer_entropy . . . . .	6
print.transfer_entropy . . . . .	7
set_quiet . . . . .	8
stocks . . . . .	8
summary.transfer_entropy . . . . .	9
transfer_entropy . . . . .	10

<b>Index</b>	<b>13</b>
--------------	-----------

---

calc_ete	<i>Calculates the Effective Transfer Entropy for two time series</i>
----------	--

---

### 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)
```

### Arguments

x	a vector of numeric values, ordered by time. Also allowed are <code>xts</code> , <code>zoo</code> , or <code>ts</code> objects.
y	a vector of numeric values, ordered by time. Also allowed are <code>xts</code> , <code>zoo</code> , or <code>ts</code> objects.
lx	Markov order of x, i.e. the number of lagged values affecting the current value of x. Default is <code>lx = 1</code> .
ly	Markov order of y, i.e. the number of lagged values affecting the current value of y. Default is <code>ly = 1</code> .
q	a weighting parameter used to estimate Renyi transfer entropy, parameter is between 0 and 1. For <code>q = 1</code> , Renyi transfer entropy converges to Shannon transfer entropy. Default is <code>q = 0.1</code> .
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 <code>entropy = 'Shannon'</code> .
shuffles	the number of shuffles used to calculate the effective transfer entropy. Default is <code>shuffles = 100</code> .

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.

**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)
```

### Arguments

x	a vector of numeric values, ordered by time. Also allowed are <code>xts</code> , <code>zoo</code> , or <code>ts</code> objects.
y	a vector of numeric values, ordered by time. Also allowed are <code>xts</code> , <code>zoo</code> , or <code>ts</code> objects.
lx	Markov order of x, i.e. the number of lagged values affecting the current value of x. Default is <code>lx = 1</code> .
ly	Markov order of y, i.e. the number of lagged values affecting the current value of y. Default is <code>ly = 1</code> .
q	a weighting parameter used to estimate Renyi transfer entropy, parameter is between 0 and 1. For <code>q = 1</code> , Renyi transfer entropy converges to Shannon transfer entropy. Default is <code>q = 0.1</code> .
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 <code>entropy = 'Shannon'</code> .
shuffles	the number of shuffles used to calculate the effective transfer entropy. Default is <code>shuffles = 100</code> .
type	specifies the type of discretization applied to the observed time series: 'quantiles', 'bins' or 'limits'. Default is <code>type = 'quantiles'</code> .
quantiles	specifies the quantiles of the empirical distribution of the respective time series used for discretization. Default is <code>quantiles = c(5,95)</code> .
bins	specifies the number of bins with equal width used for discretization. Default is <code>bins = NULL</code> .
limits	specifies the limits on values used for discretization. Default is <code>limits = NULL</code> .
burn	the number of observations that are dropped from the beginning of the bootstrapped Markov chain. Default is <code>burn = 50</code> .
seed	a seed that seeds the PRNG (will internally just call <code>set.seed</code> ), default is <code>seed = NULL</code> .

**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), ...)
```

## 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
...	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))
```

---

set_quiet	<i>Set the quiet-parameter for all RTransferEntropy Calls</i>
-----------	---

---

### 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	<i>Daily stock data for 10 stocks from 2000-2017</i>
--------	--

---

### 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 <a href="#">printCoefmat</a>

**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	<i>Function to estimate Shannon and Renyi transfer entropy between two time series x and y.</i>
------------------	---

---

### 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)

```

### Arguments

x	a vector of numeric values, ordered by time. Also allowed are <code>xts</code> , <code>zoo</code> , or <code>ts</code> objects.
y	a vector of numeric values, ordered by time. Also allowed are <code>xts</code> , <code>zoo</code> , or <code>ts</code> objects.
lx	Markov order of x, i.e. the number of lagged values affecting the current value of x. Default is <code>lx = 1</code> .
ly	Markov order of y, i.e. the number of lagged values affecting the current value of y. Default is <code>ly = 1</code> .
q	a weighting parameter used to estimate Renyi transfer entropy, parameter is between 0 and 1. For <code>q = 1</code> , Renyi transfer entropy converges to Shannon transfer entropy. Default is <code>q = 0.1</code> .
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 <code>entropy = 'Shannon'</code> .
shuffles	the number of shuffles used to calculate the effective transfer entropy. Default is <code>shuffles = 100</code> .

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.

### 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(multiprocess)

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)
```

# Index

## \*Topic **datasets**

stocks, 8

calc\_ete, 2, 5

calc\_te, 3, 4

coef, 11

coef.transfer\_entropy, 5

getSymbols, 9

is.transfer\_entropy, 6

print.transfer\_entropy, 7, 11

printCoefmat, 9

set\_quiet, 8

stocks, 8

summary.transfer\_entropy, 9

transfer\_entropy, 3, 5, 10

ts, 2, 4, 10

xts, 2, 4, 10

zoo, 2, 4, 10