Package ‘CPAT’

January 22, 2019

Title Change Point Analysis Tests

Version 0.2.0.9000

Date 2018-10-16

Maintainer Curtis Miller <cmiller@math.utah.edu>

Description Implements several statistical tests for structural change.

Depends R (>= 3.2)

Suggests cointReg (>= 0.2), foreach (>= 1.4), doRNG (>= 1.7), doParallel (>= 1.0), ggplot2 (>= 2.2), dplyr (>= 0.7), tikzDevice (>= 0.12), testthat (>= 2.0)

Imports stats (>= 3.2), utils (>= 3.2), grDevices (>= 3.2), Rdpack (>= 0.9), methods (>= 3.2), Rcpp (>= 0.12), purrr (>= 0.2),

RdMacros Rdpack

SystemRequirements GNU make

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LinkingTo Rcpp, RcppArmadillo, BH

RoxygenNote 6.1.0

NeedsCompilation yes

Author Curtis Miller [aut, cre]

R topics documented:

`.onAttach` *Package Attach Hook Function*

Description

Hook triggered when package attached

Usage

```
.onAttach(lib, pkg)
```

Arguments

lib	a character string giving the library directory where the package defining the namespace was found
pkg	a character string giving the name of the package

Examples

```
CPAT:::.onAttach(.libPaths()[1], "CPAT")
```

Andrews.test — *Andrews' Test for End-of-Sample Structural Change*

Description

Performs Andrews' test for end-of-sample structural change, as described in (Andrews 2003). This function works for both univariate and multivariate data depending on the nature of x and whether formula is specified. This function is thus an interface to andrews_test and andrews_test_reg; see the documentation of those functions for more details.

Usage

```
Andrews.test(x, M, formula = NULL)
```

Arguments

x	Data to test for change in mean (either a vector or data.frame)
M	Numeric index of the location of the first potential change point
formula	The regression formula, which will be passed to lm

Value

A htest-class object containing the results of the test

References

Andrews DWK (2003). "End-of-Sample Instability Tests." *Econometrica*, **71**(6), 1661–1694. ISSN 00129682, 14680262, https://www.jstor.org/stable/1555535.

Examples

```
Andrews.test(rnorm(1000), M = 900)
x <- rnorm(1000)
y <- 1 + 2 * x + rnorm(1000)
df <- data.frame(x, y)
Andrews.test(df, y ~ x, M = 900)
```

andrews_test — *Univariate Andrews Test for End-of-Sample Structural Change*

Description

This implements Andrews' test for end-of-sample change, as described by Andrews (2003). This test was derived for detecting a change in univariate data. See (Andrews 2003) for a description of the test.

Usage

```
andrews_test(x, M, pval = TRUE, stat = TRUE)
```

Arguments

`x`	Vector of the data to test
`M`	Numeric index of the location of the first potential change point
`pval`	If `TRUE`, return a p-value
`stat`	If `TRUE`, return a test statistic

Value

If both `pval` and `stat` are `TRUE`, a list containing both; otherwise, a number for one or the other, depending on which is `TRUE`

References

Andrews DWK (2003). "End-of-Sample Instability Tests." *Econometrica*, **71**(6), 1661–1694. ISSN 00129682, 14680262, https://www.jstor.org/stable/1555535.

Examples

```
CPAT:::andrews_test(rnorm(1000), M = 900)
```

andrews_test_reg — *Multivariate Andrews' Test for End-of-Sample Structural Change*

Description

This implements Andrews' test for end-of-sample change, as described by Andrews (2003). This test was derived for detecting a change in multivarate data, aso originally described. See (Andrews 2003) for a description of the test.

Usage

```
andrews_test_reg(formula, data, M, pval = TRUE, stat = TRUE)
```

Arguments

formula	The regression formula, which will be passed to lm
data	data.frame containing the data
M	Numeric index of the location of the first potential change point
pval	If TRUE, return a p-value
stat	If TRUE, return a test statistic

Value

If both pval and stat are TRUE, a list containing both; otherwise, a number for one or the other, depending on which is TRUE

References

Andrews DWK (2003). "End-of-Sample Instability Tests." *Econometrica*, **71**(6), 1661–1694. ISSN 00129682, 14680262, https://www.jstor.org/stable/1555535.

Examples

```
x <- rnorm(1000)
y <- 1 + 2 * x + rnorm(1000)
df <- data.frame(x, y)
CPAT:::andrews_test_reg(y ~ x, data = df, M = 900)
```

---

a_n *Sequence a_n of the Darling-Erdös Law*

Description

Computes $a_n(m) = \sqrt{b_n(m)/(2 \log \log n)}$, with $b_n(m)$ as described by b_n.

Usage

```
a_n(n, m)
```

Arguments

n	The parameter n
m	The parameter m

Value

The number $a_n(m)$

Examples

```
CPAT:::a_n(5, 2)
```

banks *Bank Portfolio Returns*

Description

Data set representing the returns of an industry portfolio representing the banking industry based on company four-digit SIC codes, obtained from the data library maintained by Kenneth French. Data ranges from July 1, 1926 to October 31, 2017.

Usage

```
banks
```

Format

A data frame with 24099 rows and 1 variable:

Banks The return of a portfolio representing the banking industry

Row names are dates in YYYY-MM-DD format.

Source

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

---

besselJ_zeros *Compute Zeros of the Bessel Function of the First Kind*

Description

Returns the zeros of the Bessel function of the first kind, J_ν.

Usage

```
besselJ_zeros(b, a = 1, nu = 1)
```

Arguments

b	The (one-based) index of the last zero to return
a	The (one-based) index of the first zero to return (so `a = 1` represents the first positive zero)
nu	The order of the Bessel function

Details

This function is an interface to the function `besselJ_zeros_cpp`, a function written in C++ and serves effectively as an interface to a Boost C++ function `cyl_bessel_j_zero`. Thus this function does nothing other than make the Boost function available to R.

See the references of `besselJ` for more about bessel functions.

Value

A vector containing the zeros of the Bessel function

Examples

```
CPAT:::besselJ_zeros(4)
CPAT:::besselJ_zeros(a = 3, b = 10, nu = 3.5)
```

---

b_n *Sequence b_n of the Darling-Erdös Law*

Description

Computes $b_n(m) = (2 \log \log(n) + (m \log \log \log n)/2 - \log(\Gamma(m/2)))^2/(2 \log \log n)$

Usage

```
b_n(n, m)
```

Arguments

n	The parameter n
m	The parameter m

Value

The number $b_n(m)$

Examples

```
CPAT:::b_n(5, 2)
```

CPAT_startup_message *Create Package Startup Message*

Description

Makes package startup message.

Usage

```
CPAT_startup_message()
```

Examples

```
CPAT:::CPAT_startup_message()
```

cpt_consistent_var *Variance Estimation Consistent Under Change*

Description

Estimate the variance (using the sum of squared errors) with an estimator that is consistent when the mean changes at a known point.

Usage

```
cpt_consistent_var(x, k)
```

Arguments

`x`	A numeric vector for the data set
`k`	The potential change point at which the data set is split

Details

This is the estimator

$$\hat{\sigma}^2_{T,t} = T^{-1}\left(\sum_{s=1}^{t}\left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^{T}\left(X_s - \tilde{X}_{T-t}\right)^2\right)$$

where $\bar{X}_t = t^{-1}\sum_{s=1}^{t} X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^{T} X_s$. In this implementation, T is computed automatically as `length(x)` and `k` corresponds to t, a potential change point.

Value

The estimated change-consistent variance

Examples

```
CPAT:::cpt_consistent_var(c(rnorm(500, mean = 0), rnorm(500, mean = 1)), k = 500)
```

CUSUM.test *CUSUM Test*

Description

Performs the CUSUM test for change in mean, as described in (Rice et al.).

Usage

```
CUSUM.test(x, formula = NULL, use_kernel_var = FALSE,
  stat_plot = FALSE, kernel = "ba", bandwidth = "and")
```

Arguments

- `x` — Data to test for change in mean (either `numeric` or a `data.frame`)
- `formula` — Formula used for defining the regression model, if applicable
- `use_kernel_var` — Set to `TRUE` to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if `FALSE`, then the long-run variance is estimated using $\hat{\sigma}^2_{T,t} = T^{-1}\left(\sum_{s=1}^{t}\left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^{T}\left(X_s - \tilde{X}_{T-t}\right)^2\right)$, where $\bar{X}_t = t^{-1}\sum_{s=1}^{t} X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^{T} X_s$
- `stat_plot` — Whether to create a plot of the values of the statistic at all potential change points
- `kernel` — If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
- `bandwidth` — If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)

Details

This is effectively an interface to `stat_Vn`; see its documentation for more details.

When `x` is a (numeric) vector, the CUSUM test is perfomed directly on the data. When `x` is a `data.frame` and `formula` is not `NULL`, then a regression model is estimated first with `lm` and the test is performed on the residuals of the regression model (see (Ploberger and Krämer 1992)).

p-values are computed using `pkolmogorov`, which represents the limiting distribution of the statistic under the null hypothesis.

Value

A `htest`-class object containing the results of the test

References

Ploberger W, Krämer W (1992). "The CUSUM test with OLS residuals." *Econometrica*, **60**(2), 271–285.

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
CUSUM.test(rnorm(1000))
CUSUM.test(rnorm(1000), use_kernel_var = TRUE, kernel = "bo",
           bandwidth = "nw")
x <- rnorm(1000)
y <- 1 + 2 * x + rnorm(1000)
df <- data.frame(x, y)
CUSUM.test(df, formula = y ~ x, use_kernel_var = TRUE)
```

dBst *Density Function of the First Hitting Time of a Bessel Process*

Description

Density function of the distribution of the first time a Bessel process with parameter $\nu > 1$ hits $b > 0$.

Usage

```
dBst(x, b, nu = -1/2, summands = NULL)
```

Arguments

x	Points at which to evaluate the density function
b	Point in space Bessel process hits
nu	The parameter $\nu > -1$ of the Bessel process
summands	Number of summands to use in summation; default is to pick the number of summands with dBst_summand_solver (it could be slow, so for performance it may be best to pick a fixed number)

Details

Let $\tau_b^{(\nu)}$ be the first time a Bessel process with parameter ν hits $b > 0$. Let $J_\nu(x)$ be the Bessel function (of the first kind) with order ν, and let $j_{\nu,k}$ be the kth zero of $J_\nu(x)$. Let $\Gamma(x)$ be the gamma function. Then the density function of $\tau_b^{(\nu)}$ is

$$\frac{1}{2^\nu b^2 \Gamma(\nu+1)} \sum_{k=1}^{\infty} \frac{j_{\nu,k}^{\nu+1}}{J_{\nu+1}(j_{\nu,k})} e^{-\frac{j_{\nu,k}^2}{2b^2} t}$$

This was found by differentiating the CDF computed by pBst.

Value

The value of the density function at x

Examples

```
CPAT:::dBst(0.1, 1)
```

dBst_summand_solver — *Find Number of Summands Needed for Numerical Accuracy of* dBst

Description

Find the number of summands needed to achieve numerical accuracy of the sum involved in dBst.

Usage

```
dBst_summand_solver(x, b, nu = -1/2, error = .Machine$double.eps)
```

Arguments

x	Quantile input to PDF
b	Point in space Bessel process hits
nu	The parameter $\nu > -1$ of the Bessel process
error	The desired numerical error of the sum

Details

The number of summands needed is determined by using a loop that runs over the summands until it encounters a summand that is not greater than the specified level of numerical accuracy. The index of that last summand is then returned.

Value

Integer for number of summands

Examples

```
dBst_summand_solver(1, 1)
```

DE.test — *Darling-Erdös Test*

Description

Performs the (univariate) Darling-Erdös test for change in mean, as described in (Rice et al.).

Usage

```
DE.test(x, formula = NULL, a = log, b = log,
  use_kernel_var = FALSE, stat_plot = FALSE, kernel = "ba",
  bandwidth = "and")
```

Arguments

- `x` Data to test for change in mean (either a `numeric` vector or a `data.frame`)
- `formula` Formula used for defining the regression model, if applicable
- `a` The function that will be composed with $l(x) = (2 \log x)^{1/2}$
- `b` The function that will be composed with $u(x) = 2 \log x + \frac{1}{2} \log \log x - \frac{1}{2} \log \pi$
- `use_kernel_var` Set to `TRUE` to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if `FALSE`, then the long-run variance is estimated using $\hat{\sigma}^2_{T,t} = T^{-1} \left(\sum_{s=1}^{t} \left(X_s - \bar{X}_t \right)^2 + \sum_{s=t+1}^{T} \left(X_s - \tilde{X}_{T-t} \right)^2 \right)$, where $\bar{X}_t = t^{-1} \sum_{s=1}^{t} X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1} \sum_{s=t+1}^{T} X_s$
- `stat_plot` Whether to create a plot of the values of the statistic at all potential change points
- `kernel` If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
- `bandwidth` If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)

Details

This is effectively an interface to `stat_de`; see its documentation for more details.

When `x` is a (numeric) vector, the CUSUM test is perfomed directly on the data. When `x` is a `data.frame` and `formula` is not `NULL`, then a regression model is estimated first with `lm` and the test is performed on the residuals of the regression model.

p-values are computed using `pdarling_erdos`, which represents the limiting distribution of the test statistic under the null hypothesis when `a` and `b` are chosen appropriately. (Change those parameters at your own risk!)

Value

A `htest`-class object containing the results of the test

References

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
DE.test(rnorm(1000))
DE.test(rnorm(1000), use_kernel_var = TRUE, kernel = "bo", bandwidth = "nw")
x <- rnorm(1000)
y <- 1 + 2 * x + rnorm(1000)
df <- data.frame(x, y)
DE.test(df, formula = y ~ x, use_kernel_var = TRUE)
```

dZn — *Rényi-Type Statistic Limiting Distribution Density Function*

Description

Function for computing the value of the density function of the limiting distribution of the Rényi-type statistic.

Usage

```
dZn(x, d = 1, summands = NULL)
```

Arguments

x	Point at which to evaluate the density function (note that this parameter is not vectorized)
d	Dimension parameter
summands	Number of summands to use in summation (the default should be machine accurate)

Details

The density function was found by differentiating the CDF, as described by pZn.

Value

Value of the density function at x

Examples

```
CPAT:::dZn(1)
```

ff — *Fama-French Five Factors*

Description

Data set containing the five factors described by Fama and French (2015), from the data library maintained by Kenneth French. Data ranges from July 1, 1963 to October 31, 2017.

Usage

```
ff
```

Format

A data frame with 13679 rows and 6 variables:

Mkt.RF Market excess returns

RF The risk-free rate of return

SMB The return on a diversified portfolio of small stocks minus return on a diversified portfolio of big stocks

HML The return of a portfolio of stocks with a high book-to-market (B/M) ratio minus the return of a portfolio of stocks with a low B/M ratio

RMW The return of a portfolio of stocks with robust profitability minus a portfolio of stocks with weak profitability

CMA The return of a portfolio of stocks with conservative investment minus the return of a portfolio of stocks with aggressive investment

Row names are dates in YYYYMMDD format.

Source

`http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`

`getLongRunWeights` *Weights for Long-Run Variance*

Description

Compute some weights for long-run variance. This code comes directly from the source code of **cointReg**; see `getLongRunWeights`.

Usage

```
getLongRunWeights(n, bandwidth, kernel = "ba")
```

Arguments

`n`	Length of weights' vector
`bandwidth`	A number for the bandwidth
`kernel`	The kernel function; see `getLongRunVar` for possible values

Value

List with components `w` containing the vector of weights and `upper`, the index of the largest non-zero entry in `w`

Examples

```
CPAT:::getLongRunWeights(10, 1)
```

get_lrv_vec *Long-Run Variance Estimation With Possible Change Points*

Description

Computes the estimates of the long-run variance in a change point context, as described in (Rice et al.). By default it uses kernel and bandwidth selection as used in the package **cointReg**, though changing the parameters `kernel` and `bandwidth` can change this behavior. If **cointReg** is not installed, the Bartlett internal (defined internally) will be used and the bandwidth will be the square root of the sample size.

Usage

```
get_lrv_vec(dat, kernel = "ba", bandwidth = "and")
```

Arguments

`dat`	The data vector
`kernel`	If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
`bandwidth`	If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)

Value

A vector of estimates of the long-run variance

References

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
x <- rnorm(1000)
CPAT:::get_lrv_vec(x)
CPAT:::get_lrv_vec(x, kernel = "pa", bandwidth = "nw")
```

HR.test *Rényi-Type Test*

Description

Performs the (univariate) Rényi-type test for change in mean, as described in (Rice et al.). This is effectively an interface to `stat_Zn`; see its documentation for more details. p-values are computed using `pZn`, which represents the limiting distribution of the test statistic under the null hypothesis, which represents the limiting distribution of the test statistic under the null hypothesis when kn represents a sequence t_T satisfying $t_T \to \infty$ and $t_T/T \to 0$ as $T \to \infty$. (`log` and `sqrt` should be good choices.)

Usage

```
HR.test(x, kn = log, use_kernel_var = FALSE, stat_plot = FALSE,
  kernel = "ba", bandwidth = "and")
```

Arguments

`x`	Data to test for change in mean
`kn`	A function corresponding to the trimming parameter t_T; by default, the square root function
`use_kernel_var`	Set to `TRUE` to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if `FALSE`, then the long-run variance is estimated using $\hat{\sigma}^2_{T,t} = T^{-1}\left(\sum_{s=1}^{t}\left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^{T}\left(X_s - \tilde{X}_{T-t}\right)^2\right)$, where $\bar{X}_t = t^{-1}\sum_{s=1}^{t} X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^{T} X_s$; if `custom_var` is not `NULL`, this argument is ignored
`stat_plot`	Whether to create a plot of the values of the statistic at all potential change points
`kernel`	If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
`bandwidth`	If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)

Value

A `htest`-class object containing the results of the test

References

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
HR.test(rnorm(1000))
HR.test(rnorm(1000), use_kernel_var = TRUE, kernel = "bo", bandwidth = "nw")
```

`HS.test` *Hidalgo-Seo Test*

Description

Performs the Hidalgo-Seo test for structural change, as proposed by Hidalgo and Seo (2013).

Usage

```
HS.test(x, formula = NULL, m = sqrt, corr = TRUE,
  stat_plot = FALSE)
```

Arguments

x	Data to test for change in mean (either a vector or data.frame)
formula	The formula defining the regression model, when applicable
m	Either numeric or a function that returns numeric; corresponds to m used in computing the estimate of the long-run variance
corr	If TRUE, the long-run variance will be computed under the assumption of correlated residuals; ignored if custom_var is not NULL or use_kernel_var is TRUE
stat_plot	Whether to create a plot of the values of the statistic at all potential change points

Details

This function can perform both univariate and regression versions of the test described by Hidalgo and Seo.

If formula is NULL and x is numeric, this function performs the (univariate) Hidalgo-Seo test for change in mean, as described in (Rice et al.). This is effectively an interface to stat_hs; see its documentation for more details.

Otherwise, the function tests for structural change in a linear regression model (estimated via least squares), and serves as an interface to stat_hs_reg; see its documentation for more details. In this mode the parameter corr is effectively ignored.

p-values are computed using phidalgo_seo, which represents the limiting distribution of the test statistic when the null hypothesis is true.

Value

A htest-class object containing the results of the test

References

Hidalgo J, Seo MH (2013). "Testing for structural stability in the whole sample." *Journal of Econometrics*, **175**(2), 84 - 93. ISSN 0304-4076, doi: 10.1016/j.jeconom.2013.02.008, http://www.sciencedirect.com/science/article/pii/S0304407613000626.

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
HS.test(rnorm(1000))
HS.test(rnorm(1000), corr = FALSE)
x <- rnorm(1000)
y <- 1 + 2 * x + rnorm(1000)
df <- data.frame(x, y)
HS.test(df, formula = y ~ x)
```

pBst *CDF of First Hitting Time of Bessel Process*

Description

CDF of the distribution of the first time a Bessel process with parameter $\nu > -1$ hits $b > 0$.

Usage

```
pBst(q, b, nu = -1/2, summands = NULL)
```

Arguments

q	Quantile input to CDF
b	Point in space Bessel process hits
nu	The parameter $\nu > -1$ of the Bessel process
summands	Number of summands to use in summation; default is to pick the number of summands with pBst_summand_solver (it could be slow, so for performance it may be best to pick a fixed number)

Details

Let $\tau_b^{(\nu)}$ be the first time a Bessel process with parameter ν hits $b > 0$. Let $J_\nu(x)$ be the Bessel function (of the first kind) with order ν, and let $j_{\nu,k}$ be the kth zero of $J_\nu(x)$. Let $\Gamma(x)$ be the gamma function. Then the CDF of $\tau_b^{(\nu)}$ is

$$1 - \frac{1}{2^{\nu-1}\Gamma(\nu+1)} \sum_{k=1}^{\infty} \frac{j_{\nu,k}^{\nu-1}}{J_{\nu+1}(j_{\nu,k})} e^{-\frac{j_{\nu,k}^2}{2b^2} t}$$

(This was obtained in (Kent 1980), but the formula above was given in (Hamana and Matsumoto 2013).)

Value

If T is the random variable as described, $P(T \leq q)$

References

Hamana Y, Matsumoto H (2013). "The probability distributions of the first hitting times of Bessel processes." *Transactions of the American Mathematical Society*, **365**(10), 5237–5257.

Kent JT (1980). "Eigenvalue expansions for diffusion hitting times." *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, **52**(3), 309–319. ISSN 1432-2064, doi: 10.1007/BF00538895, https://doi.org/10.1007/BF00538895.

Examples

```
CPAT:::pBst(1, 1)
```

pBst_summand_solver — *Find Number of Summands Needed for Numerical Accuracy of* `pBst`

Description

Find the number of summands needed to achieve numerical accuracy of the sum involved in `pBst`.

Usage

```
pBst_summand_solver(q, b, nu = -1/2, error = .Machine$double.eps)
```

Arguments

`q`	Quantile input to CDF
`b`	Point in space Bessel process hits
`nu`	The parameter $\nu > -1$ of the Bessel process
`error`	The desired numerical error of the sum

Details

The number of summands needed is determined by using a loop that runs over the summands until it encounters a summand that is not greater than the specified level of numerical accuracy. The index of that last summand is then returned.

Value

Integer for number of summands

Examples

```
pBst_summand_solver(1, 1)
```

pdarling_erdos — *Darling-Erdös Statistic CDF*

Description

CDF for the limiting distribution of the Darling-Erdös statistic.

Usage

```
pdarling_erdos(q)
```

Arguments

`q`	Quantile input to CDF

Value

If Z is the random variable with this distribution, the quantity $P(Z \leq q)$

Examples

```
CPAT:::pdarling_erdos(0.1)
```

---

phidalgo_seo *Hidalgo-Seo Statistic CDF*

Description

CDF of the limiting distribution of the Hidalgo-Seo statistic

Usage

```
phidalgo_seo(q)
```

Arguments

q	Quantile input to CDF

Value

If Z is the random variable following the limiting distribution, the quantity $P(Z \leq q)$

Examples

```
CPAT:::phidalgo_seo(0.1)
```

---

pkolmogorov *Kolmogorov CDF*

Description

CDF of the Kolmogorov distribution.

Usage

```
pkolmogorov(q, summands = ceiling(q * sqrt(72) + 3/2))
```

Arguments

q	Quantile input to CDF
summands	Number of summands for infinite sum (the default should have machine accuracy)

Value

If Z is the random variable following the Kolmogorov distribution, the quantity $P(Z \leq q)$

Examples

```
CPAT:::pkolmogorov(0.1)
```

pZn *Rènyi-Type Statistic CDF*

Description

CDF for the limiting distribution of the Rènyi-type statistic.

Usage

```
pZn(q, d = 1, summands = NULL)
```

Arguments

q	Quantile input to CDF
d	Dimension parameter
summands	Number of summands for infinite sum; if NULL, automatically determined using pBst_summand_solver (which isn't necessarily fast, so consider picking a fixed number if speed is important)

Details

If $G_{\nu,b}(x)$ is the CDF of the first time a Bessel process with parameter ν hits $b > 0$ (as described by pBst) then the CDF of the Rényi-type statistic when the null hypothesis is true is $F(x) = (1 - G_{d/2-1,x}(1))^2$, where d is the dimensionality parameter of the statistic. (This comes from combining the limiting distribution of the statistic described in (Rice et al.) with the expression for the CDF of the hitting time of the Bessel process described in (Hamana and Matsumoto 2013).)

Value

If Z is the random variable following the limiting distribution, the quantity $P(Z \leq q)$

References

Hamana Y, Matsumoto H (2013). "The probability distributions of the first hitting times of Bessel processes." *Transactions of the American Mathematical Society*, **365**(10), 5237–5257.

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
CPAT:::pZn(0.1)
```

qBst *Bessel Process First Hitting Time Quantile Function*

Description

Quantile function of the distribution of the first time a Bessel process with parameter $\nu > -1$ hits $b > 0$.

Usage

```
qBst(p, b, nu = -1/2, summands = NULL, interval = c(0, 100),
  tol = .Machine$double.eps, ...)
```

Arguments

`p`	The probability associated with the desired quantile
`b`	Point in space Bessel process hits
`nu`	The parameter $\nu > -1$ of the Bessel process
`summands`	Number of summands to use in summation; default is to pick the number of summands with `pBst_summand_solver` (it could be slow, so for performance it may be best to pick a fixed number)
`interval, tol, ...`	Arguments to be passed to `uniroot`

Details

This function uses `uniroot` for finding this quantity, and many of the the accepted parameters are arguments for that function; see its documentation for more details.

Value

The quantile associated with `p`

Examples

```
CPAT:::qBst(0.5, b = 1)
```

qdarling_erdos *Darling-Erdös Statistic Limiting Distribution Quantile Function*

Description

Quantile function for the limiting distribution of the Darling-Erdös statistic.

Usage

```
qdarling_erdos(p)
```

Arguments

p	The probability associated with the desired quantile

Value

The quantile associated with p

Examples

```
CPAT:::qdarling_erdos(0.5)
```

---

qhidalgo_seo — *Hidalgo-Seo Statistic Limiting Distribution Quantile Function*

Description

Quantile function for the limiting distribution of the Hidalgo-Seo statistic

Usage

```
qhidalgo_seo(p)
```

Arguments

p	The probability associated with the desired quantile

Value

A The quantile associated with p

Examples

```
CPAT:::qhidalgo_seo(0.5)
```

---

qkolmogorov — *Kolmogorov Distribution Quantile Function*

Description

Quantile function for the Kolmogorov distribution.

Usage

```
qkolmogorov(p, summands = 500, interval = c(0, 100),
  tol = .Machine$double.eps, ...)
```

Arguments

p	Value of the CDF at the quantile
summands	Number of summands for infinite sum
interval, tol, ...	Arguments to be passed to uniroot

Details

This function uses uniroot for finding this quantity, and many of the the accepted parameters are arguments for that function; see its documentation for more details.

Value

The quantile associated with p

Examples

```
CPAT:::qkolmogorov(0.5)
```

---

qZn *Rènyi-Type Statistic Quantile Function*

Description

Quantile function for the limiting distribution of the Rènyi-type statistic.

Usage

```
qZn(p, d = 1, summands = 500, interval = c(0, 100),
  tol = .Machine$double.eps, ...)
```

Arguments

p	Value of the CDF at the quantile
d	Dimension parameter
summands	Number of summands for infinite sum
interval, tol, ...	Arguments to be passed to uniroot

Details

This function uses uniroot for finding this quantity, and many of the the accepted parameters are arguments for that function; see its documentation for more details.

Value

The quantile associated with p

Examples

```
CPAT:::qZn(0.5)
```

rchangepoint — *Simulate Univariate Data With a Single Change Point*

Description

This function simulates univariate data with a structural change.

Usage

```
rchangepoint(n, changepoint = NULL, mean1 = 0, mean2 = 0,
  dist = rnorm, meanparam = "mean", ...)
```

Arguments

`n`	An integer for the data set's sample size
`changepoint`	An integer for where the change point occurs
`mean1`	The mean prior to the change point
`mean2`	The mean after the change point
`dist`	The function with which random data will be generated
`meanparam`	A string for the parameter in `dist` representing the mean
`...`	Other arguments to be passed to dist

Details

This function generates artificial change point data, where up to the specified change point the data has one mean, and after the point it has a different mean. By default, the function simulates standard Normal data with no change. If `changepoint` is `NULL`, then by default the change point will be at about the middle of the data.

Value

A vector of the simulated data

Examples

```
CPAT:::rchangepoint(500)
CPAT:::rchangepoint(500, changepoint = 10, mean2 = 2, sd = 2)
CPAT:::rchangepoint(500, changepoint = 250, dist = rexp, meanparam = "rate",
                    mean1 = 1, mean2 = 2)
```

sim_de_stat *Darling-Erdös Statistic Simulation*

Description

Simulates multiple realizations of the Darling-Erdös statistic.

Usage

```
sim_de_stat(size, a = log, b = log, use_kernel_var = FALSE,
  kernel = "ba", bandwidth = "and", n = 500, gen_func = rnorm,
  args = NULL, parallel = FALSE)
```

Arguments

`size`	Number of realizations to simulate
`a`	The function that will be composed wit $l(x) = (2\log(x))^{1/2}$
`b`	The function that will be composed with $u(x) = 2\log(x) + \frac{1}{2}\log(\log(x)) - \frac{1}{2}\log(pi)$
`use_kernel_var`	Set to `TRUE` to use kernel-based long-run variance estimation (`FALSE` means this is not employed)
`kernel`	If character, the identifier of the kernel function as used in the **cointReg** (see documentation for `cointReg::getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**); this parameter has no effect if `use_kernel_var` is `FALSE`
`bandwidth`	If character, the identifier of how to compute the bandwidth as defined in the **cointReg** package (see documentation for `cointReg::getLongRunVar`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth to use (the default behavior is to use the Andrews (1991) method, as used in **cointReg**); this parameter has no effect if `use_kernel_var` is `FALSE`
`n`	The sample size for each realization
`gen_func`	The function generating the random sample from which the statistic is computed
`args`	A list of arguments to be passed to `gen_func`
`parallel`	Whether to use the **foreach** and **doParallel** packages to parallelize simulation (which needs to be initialized in the global namespace before use)

Details

If `use_kernel_var` is set to `TRUE`, long-run variance estimation using kernel-based techniques will be employed; otherwise, a technique resembling standard variance estimation will be employed. Any technique employed, though, will account for the potential break points, as described in Rice et al. (). See the documentation for `stat_de` for more details.

The parameters `kernel` and `bandwidth` control parameters for long-run variance estimation using kernel methods. These parameters will be passed directly to `stat_de`.

Value

A vector of simulated realizations of the Darling-Erdös statistic

References

Andrews DWK (1991). "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica*, **59**(3), 817-858.

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
CPAT:::sim_de_stat(100)
CPAT:::sim_de_stat(100, use_kernel_var = TRUE,
                  gen_func = CPAT:::rchangepoint,
                  args = list(changepoint = 250, mean2 = 1))
```

---

sim_hs_stat *Hidalgo-Seo Statistic Simulation*

Description

Simulates multiple realizations of the Hidalgo-Seo statistic.

Usage

```
sim_hs_stat(size, corr = TRUE, gen_func = rnorm, args = NULL,
  n = 500, parallel = FALSE, use_kernel_var = FALSE, kernel = "ba",
  bandwidth = "and")
```

Arguments

size	Number of realizations to simulate
corr	Whether long-run variance should be computed under the assumption of correlated residuals
gen_func	The function generating the random sample from which the statistic is computed
args	A list of arguments to be passed to gen_func
n	The sample size for each realization
parallel	Whether to use the **foreach** and **doParallel** packages to parallelize simulation (which needs to be initialized in the global namespace before use)
use_kernel_var	Set to TRUE to use kernel-based long-run variance estimation (FALSE means this is not employed); *TODO: NOT CURRENTLY IMPLEMENTED*
kernel	If character, the identifier of the kernel function as used in the **cointReg** (see documentation for cointReg::getLongRunVar); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**); this parameter has no effect if use_kernel_var is FALSE; *TODO: NOT CURRENTLY IMPLEMENTED*
bandwidth	If character, the identifier of how to compute the bandwidth as defined in the **cointReg** package (see documentation for cointReg::getLongRunVar); if function, a function to use for computing the bandwidth; if numeric, the bandwidth to use (the default behavior is to use the Andrews (1991) method, as used in **cointReg**); this parameter has no effect if use_kernel_var is FALSE; *TODO: NOT CURRENTLY IMPLEMENTED*

Details

If corr is TRUE, then the residuals of the data-generating process are assumed to be correlated and the test accounts for this in long-run variance estimation; see the documentation for stat_hs for more details. Otherwise, the sample variance is the estimate for the long-run variance, as described in Hidalgo and Seo (2013).

Value

A vector of simulated realizations of the Hidalgo-Seo statistic

References

Andrews DWK (1991). "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica*, **59**(3), 817-858.

Hidalgo J, Seo MH (2013). "Testing for structural stability in the whole sample." *Journal of Econometrics*, **175**(2), 84 - 93. ISSN 0304-4076, doi: 10.1016/j.jeconom.2013.02.008, http://www.sciencedirect.com/science/article/pii/S0304407613000626.

Examples

```
CPAT:::sim_hs_stat(100)
CPAT:::sim_hs_stat(100, gen_func = CPAT:::rchangepoint,
                   args = list(changepoint = 250, mean2 = 1))
```

---

sim_Vn *CUSUM Statistic Simulation (Assuming Variance)*

Description

Simulates multiple realizations of the CUSUM statistic when the long-run variance of the data is known.

Usage

```
sim_Vn(size, n = 500, gen_func = rnorm, sd = 1, args = NULL)
```

Arguments

size	Number of realizations to simulate
n	The sample size for each realization
gen_func	The function generating the random sample from which the statistic is computed
sd	The square root of the second moment of the data
args	A list of arguments to be passed to gen_func

Value

A vector of simulated realizations of the CUSUM statistic

Examples

```
CPAT:::sim_Vn(100)
CPAT:::sim_Vn(100, gen_func = CPAT:::rchangepoint,
              args = list(changepoint = 250, mean2 = 1))
```

---

sim_Vn_stat *CUSUM Statistic Simulation*

Description

Simulates multiple realizations of the CUSUM statistic.

Usage

```
sim_Vn_stat(size, kn = function(n) {     1 }, tau = 0,
  use_kernel_var = FALSE, kernel = "ba", bandwidth = "and",
  n = 500, gen_func = rnorm, args = NULL, parallel = FALSE)
```

Arguments

size	Number of realizations to simulate
kn	A function returning a positive integer that is used in the definition of the trimmed CUSUSM statistic effectively setting the bounds over which the maximum is taken
tau	The weighting parameter for the weighted CUSUM statistic (defaults to zero for no weighting)
use_kernel_var	Set to TRUE to use kernel-based long-run variance estimation (FALSE means this is not employed)
kernel	If character, the identifier of the kernel function as used in the **cointReg** (see documentation for cointReg::getLongRunVar); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**); this parameter has no effect if use_kernel_var is FALSE
bandwidth	If character, the identifier of how to compute the bandwidth as defined in the **cointReg** package (see documentation for cointReg::getLongRunVar); if function, a function to use for computing the bandwidth; if numeric, the bandwidth to use (the default behavior is to use the method described in (Andrews 1991), as used in **cointReg**); this parameter has no effect if use_kernel_var is FALSE
n	The sample size for each realization
gen_func	The function generating the random sample from which the statistic is computed
args	A list of arguments to be passed to gen_func
parallel	Whether to use the **foreach** and **doParallel** packages to parallelize simulation (which needs to be initialized in the global namespace before use)

Details

This differs from `sim_Vn()` in that the long-run variance is estimated with this function, while `sim_Vn()` assumes the long-run variance is known. Estimation can be done in a variety of ways. If `use_kernel_var` is set to `TRUE`, long-run variance estimation using kernel-based techniques will be employed; otherwise, a technique resembling standard variance estimation will be employed. Any technique employed, though, will account for the potential break points, as described in Rice et al. (). See the documentation for `stat_Vn` for more details.

The parameters `kernel` and `bandwidth` control parameters for long-run variance estimation using kernel methods. These parameters will be passed directly to `stat_Vn`.

Versions of the CUSUM statistic, such as the weighted or trimmed statistics, can be simulated with the function by passing values to `kn` and `tau`; again, see the documentation for `stat_Vn`.

Value

A vector of simulated realizations of the CUSUM statistic

References

Andrews DWK (1991). "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica*, **59**(3), 817-858.

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
CPAT:::sim_Vn_stat(100)
CPAT:::sim_Vn_stat(100, kn = function(n) {floor(0.1 * n)}, tau = 1/3,
                   use_kernel_var = TRUE, gen_func = CPAT:::rchangepoint,
                   args = list(changepoint = 250, mean2 = 1))
```

`sim_Zn` *Rènyi-Type Statistic Simulation (Assuming Variance)*

Description

Simulates multiple realizations of the Rènyi-type statistic when the long-run variance of the data is known.

Usage

```
sim_Zn(size, kn, n = 500, gen_func = rnorm, args = NULL, sd = 1)
```

Arguments

`size`	Number of realizations to simulate
`kn`	A function returning a positive integer that is used in the definition of the Rènyi-type statistic effectively setting the bounds over which the maximum is taken
`n`	The sample size for each realization
`gen_func`	The function generating the random sample from which the statistic is computed
`args`	A list of arguments to be passed to `gen_func`
`sd`	The square root of the second moment of the data

Value

A vector of simulated realizations of the Rènyi-type statistic

Examples

```
CPAT:::sim_Zn(100, kn = function(n) {floor(log(n))})
CPAT:::sim_Zn(100, kn = function(n) {floor(log(n))},
             gen_func = CPAT:::rchangepoint, args = list(changepoint = 250,
                                                         mean2 = 1))
```

---

sim_Zn_stat   *Rènyi-Type Statistic Simulation*

Description

Simulates multiple realizations of the Rènyi-type statistic.

Usage

```
sim_Zn_stat(size, kn = function(n) {     floor(sqrt(n)) },
  use_kernel_var = FALSE, kernel = "ba", bandwidth = "and",
  n = 500, gen_func = rnorm, args = NULL, parallel = FALSE)
```

Arguments

size	Number of realizations to simulate
kn	A function returning a positive integer that is used in the definition of the Rènyi-type statistic effectively setting the bounds over which the maximum is taken
use_kernel_var	Set to TRUE to use kernel-based long-run variance estimation (FALSE means this is not employed)
kernel	If character, the identifier of the kernel function as used in the **cointReg** (see documentation for cointReg::getLongRunVar); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**); this parameter has no effect if use_kernel_var is FALSE
bandwidth	If character, the identifier of how to compute the bandwidth as defined in the **cointReg** package (see documentation for cointReg::getLongRunVar); if function, a function to use for computing the bandwidth; if numeric, the bandwidth to use (the default behavior is to use the Andrews (1991) method, as used in **cointReg**); this parameter has no effect if use_kernel_var is FALSE
n	The sample size for each realization
gen_func	The function generating the random sample from which the statistic is computed
args	A list of arguments to be passed to gen_func
parallel	Whether to use the **foreach** and **doParallel** packages to parallelize simulation (which needs to be initialized in the global namespace before use)

Details

This differs from sim_Zn() in that the long-run variance is estimated with this function, while sim_Zn() assumes the long-run variance is known. Estimation can be done in a variety of ways. If use_kernel_var is set to TRUE, long-run variance estimation using kernel-based techniques will be employed; otherwise, a technique resembling standard variance estimation will be employed. Any technique employed, though, will account for the potential break points, as described in Rice et al. (). See the documentation for stat_Zn for more details.

The parameters kernel and bandwidth control parameters for long-run variance estimation using kernel methods. These parameters will be passed directly to stat_Zn.

Value

A vector of simulated realizations of the Rènyi-type statistic

References

Andrews DWK (1991). "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation." *Econometrica*, **59**(3), 817-858.

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
CPAT:::sim_Zn_stat(100)
CPAT:::sim_Zn_stat(100, kn = function(n) {floor(log(n))},
            use_kernel_var = TRUE, gen_func = CPAT:::rchangepoint,
            args = list(changepoint = 250, mean2 = 1))
```

---

stat_de *Compute the Darling-Erdös Statistic*

Description

This function computes the Darling-Erdös statistic.

Usage

```
stat_de(dat, a = log, b = log, estimate = FALSE,
  use_kernel_var = FALSE, custom_var = NULL, kernel = "ba",
  bandwidth = "and", get_all_vals = FALSE)
```

Arguments

dat	The data vector
a	The function that will be composed with $l(x) = (2 \log x)^{1/2}$
b	The function that will be composed with $u(x) = 2 \log x + \frac{1}{2} \log \log x - \frac{1}{2} \log \pi$
estimate	Set to TRUE to return the estimated location of the change point
use_kernel_var	Set to TRUE to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if FALSE, then the long-run variance is estimated using $\hat{\sigma}^2_{T,t} = T^{-1}\left(\sum_{s=1}^t \left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^T \left(X_s - \tilde{X}_{T-t}\right)^2\right)$, where $\bar{X}_t = t^{-1}\sum_{s=1}^t X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^T X_s$

`custom_var`	Can be a vector the same length as `dat` consisting of variance-like numbers at each potential change point (so each entry of the vector would be the "best estimate" of the long-run variance if that location were where the change point occured) or a function taking two parameters `x` and `k` that can be used to generate this vector, with `x` representing the data vector and `k` the position of a potential change point; if `NULL`, this argument is ignored
`kernel`	If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
`bandwidth`	If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)
`get_all_vals`	If `TRUE`, return all values for the statistic at every tested point in the data set

Details

If $\bar{A}_T(\tau, t_T)$ is the weighted and trimmed CUSUM statistic with weighting parameter τ and trimming parameter t_T (see `stat_Vn`), then the Darling-Erdös statistic is

$$l(a_T)\bar{A}_T(1/2, 1) - u(b_T)$$

with $l(x) = \sqrt{2 \log x}$ and $u(x) = 2 \log x + \frac{1}{2} \log \log x - \frac{1}{2} \log \pi$ ($\log x$ is the natural logarithm of x). The parameter `a` corresponds to a_T and `b` to b_T; these are both `log` by default.

See (Rice et al.) to learn more.

Value

If both `estimate` and `get_all_vals` are `FALSE`, the value of the test statistic; otherwise, a list that contains the test statistic and the other values requested (if both are `TRUE`, the test statistic is in the first position and the estimated changg point in the second)

References

Rice G, Miller C, Horváth L (????). "A new class of change point test of Rényi type." in-press.

Examples

```
CPAT:::stat_de(rnorm(1000))
CPAT:::stat_de(rnorm(1000), use_kernel_var = TRUE, bandwidth = "nw", kernel = "bo")
```

`stat_hs` *Compute the Univariate Hidalgo-Seo Statistic*

Description

This function computes the Hidalgo-Seo statistic for a change in mean model.

Usage

```
stat_hs(dat, estimate = FALSE, corr = TRUE, m = sqrt,
  get_all_vals = FALSE, custom_var = NULL, use_kernel_var = FALSE,
  kernel = "ba", bandwidth = "and")
```

Arguments

`dat`	The data vector
`estimate`	Set to `TRUE` to return the estimated location of the change point
`corr`	If `TRUE`, the long-run variance will be computed under the assumption of correlated residuals; ignored if `custom_var` is not `NULL` or `use_kernel_var` is `TRUE`
`m`	Either numeric or a function that returns numeric; corresponds to m used in computing the estimate of the long-run variance
`get_all_vals`	If `TRUE`, return all values for the statistic at every tested point in the data set
`custom_var`	Can be a vector the same length as `dat` consisting of variance-like numbers at each potential change point (so each entry of the vector would be the "best estimate" of the long-run variance if that location were where the change point occured) or a function taking two parameters `x` and `k` that can be used to generate this vector, with `x` representing the data vector and `k` the position of a potential change point; if `NULL`, this argument is ignored
`use_kernel_var`	Set to `TRUE` to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if `FALSE`, then the long-run variance is estimated using $\hat{\sigma}^2_{T,t} = T^{-1}\left(\sum_{s=1}^t \left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^T \left(X_s - \tilde{X}_{T-t}\right)^2\right)$, where $\bar{X}_t = t^{-1}\sum_{s=1}^t X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^T X_s$; if `custom_var` is not `NULL`, this argument is ignored
`kernel`	If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
`bandwidth`	If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)

Details

For a data set x_t with n observations, the test statistic is

$$\max_{1 \leq s \leq n-1} (\mathcal{LM}(s) - B_n)/A_n$$

where $\hat{u}_t = x_t - \bar{x}$ ($\bar{x}$ is the sample mean), $a_n = (2\log\log n)^{1/2}$, $b_n = a_n^2 - \frac{1}{2}\log\log\log n - \log\Gamma(1/2)$, $A_n = b_n/a_n^2$, $B_n = b_n^2/a_n^2$, $\hat{\Delta} = \hat{\sigma}^2 = n^{-1}\sum_{t=1}^n \hat{u}_t^2$, and $\mathcal{LM}(s) = n(n-s)^{-1}s^{-1}\hat{\Delta}^{-1}\left(\sum_{t=1}^s \hat{u}_t\right)^2$.

If `corr` is `FALSE`, then the residuals are assumed to be uncorrelated. Otherwise, the residuals are assumed to be correlated and $\hat{\Delta} = \hat{\gamma}(0) + 2\sum_{j=1}^{\lfloor m \rfloor}(1 - \frac{j}{\sqrt{n}})\hat{\gamma}(j)$ with $\hat{\gamma}(j) = \frac{1}{n}\sum_{t=1}^{n-j}\hat{u}_t\hat{u}_{t+j}$. m is controlled by the parameter `m`.

This statistic was presented in (Hidalgo and Seo 2013).

Value

If both `estimate` and `get_all_vals` are `FALSE`, the value of the test statistic; otherwise, a list that contains the test statistic and the other values requested (if both are `TRUE`, the test statistic is in the first position and the estimated change point in the second)

References

Hidalgo J, Seo MH (2013). "Testing for structural stability in the whole sample." *Journal of Econometrics*, **175**(2), 84 - 93. ISSN 0304-4076, doi: 10.1016/j.jeconom.2013.02.008, http://www.sciencedirect.com/science/article/pii/S0304407613000626.

Examples

```
CPAT:::stat_hs(rnorm(1000))
CPAT:::stat_hs(rnorm(1000), corr = FALSE)
```

---

stat_hs_reg — *Regression Model Hidalgo-Seo Statistic*

Description

Compute the Hidalgo-Seo statistic indended for detecting change in linear models (estimated via least squares regression).

Usage

```
stat_hs_reg(formula, data, m = sqrt, estimate = FALSE,
  get_all_vals = FALSE)
```

Arguments

`formula`	A `link[base]{formula}` that describes the regression model
`data`	A `data.frame`-like object containing the data set; should be able to be passed to the `data` argument of `lm`
`m`	If numeric, the number of terms of the periodogram to sum; if a function, how to compute the number of terms to sum (will be passed the number of rows of `data`)
`estimate`	Set to `TRUE` to return the estimated location of the change point
`get_all_vals`	If `TRUE`, return all values for the statistic at every tested point in the data set

Details

For a data set (y_t, x_t) with n observations, $y_t \in \mathbf{R}$, and $x_t \in \mathbf{R}^d$, the test statistic is

$$\max_{d<s\leq n-d} (\mathcal{LM}(s) - B_n)/A_n$$

where $a_n = \sqrt{2\log\log n}$; $b_n = a_n^2 + d\log\log\log n/2 - \log\Gamma(d/2)$; $A_n = b_n/a_n^2$; $B_n = b_n^2/a_n^2$; $\hat{\beta}$ is the least-squares estimate of the linear regression model coefficients; $\hat{u}_t = y_t - \hat{\beta}^T x_t$ are the residuals of the model;

$$\mathcal{LM}(s) = \left(\frac{n}{\hat{\Delta}(\hat{\beta})s(n-s)}\right)\left(\sum_{t=1}^{s} x_t \hat{u}_t\right)^T \left(\sum_{t=1}^{s} x_t \hat{u}_t\right)^T$$

and $\hat{\Delta}(\hat{\beta})$ is the long-run variance estimator

$$\hat{\Delta}(\beta) = \frac{1}{m}\sum_{j=1}^{m} I\left(\frac{2\pi j}{n}; \beta\right)$$

where $I(\cdot;\beta)$ is the periodogram estimated from the residuals when the regression model coefficients are given by β. This is the test statistic suggested by the procedure introduced in (Hidalgo and Seo 2013).

The parameter m described above can be controlled via the function parameter `m`, which can be either numeric or a function that returns numeric values.

Value

If both `estimate` and `get_all_vals` are `FALSE`, the value of the test statistic; otherwise, a list that contains the test statistic and the other values requested (if both are `TRUE`, the test statistic is in the first position and the estimated change point in the second)

References

Hidalgo J, Seo MH (2013). "Testing for structural stability in the whole sample." *Journal of Econometrics*, **175**(2), 84 - 93. ISSN 0304-4076, doi: 10.1016/j.jeconom.2013.02.008, http://www.sciencedirect.com/science/article/pii/S0304407613000626.

Examples

```
x <- rnorm(100)
y <- 1 + 2 * x + rnorm(100)
df <- data.frame("x" = x, "y" = y)
CPAT:::stat_hs_reg(y ~ x, data = df)
```

---

stat_Vn *Compute the CUSUM Statistic*

Description

This function computes the CUSUM statistic (and can compute weighted/trimmed variants, depending on the values of `kn` and `tau`).

Usage

```
stat_Vn(dat, kn = function(n) {     1 }, tau = 0, estimate = FALSE,
  use_kernel_var = FALSE, custom_var = NULL, kernel = "ba",
  bandwidth = "and", get_all_vals = FALSE)
```

Arguments

`dat`	The data vector
`kn`	A function corresponding to the trimming parameter t_T in the trimmed CUSUM variant; by default, is a function returning 1 (for no trimming)
`tau`	The weighting parameter τ for the weighted CUSUM statistic; by default, is 0 (for no weighting)
`estimate`	Set to `TRUE` to return the estimated location of the change point
`use_kernel_var`	Set to `TRUE` to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if `FALSE`, then the long-run variance is estimated using $\hat{\sigma}^2_{T,t} = T^{-1}\left(\sum_{s=1}^{t}\left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^{T}\left(X_s - \tilde{X}_{T-t}\right)^2\right)$, where $\bar{X}_t = t^{-1}\sum_{s=1}^{t} X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^{T} X_s$
`custom_var`	Can be a vector the same length as `dat` consisting of variance-like numbers at each potential change point (so each entry of the vector would be the "best estimate" of the long-run variance if that location were where the change point occured) or a function taking two parameters `x` and `k` that can be used to generate this vector, with `x` representing the data vector and `k` the position of a potential change point; if `NULL`, this argument is ignored
`kernel`	If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
`bandwidth`	If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)
`get_all_vals`	If `TRUE`, return all values for the statistic at every tested point in the data set

Details

The definition of the statistic is

$$T^{-1/2} \max_{1 \le t \le T} \hat{\sigma}_{t,T}^{-1} \left| \sum_{s=1}^{t} X_s - \frac{t}{T} \sum_{s=1}^{T} \right|$$

A more general version is

$$T^{-1/2} \max_{t_T \le t \le T - t_T} \hat{\sigma}_{t,T}^{-1} \left(\frac{t}{T} \left(\frac{T-t}{T} \right) \right)^{\tau} \left| \sum_{s=1}^{t} X_s - \frac{t}{T} \sum_{s=1}^{T} \right|$$

The parameter `kn` corresponds to the trimming parameter t_T and the parameter `tau` corresponds to τ.

See (Rice et al.) for more details.

Value

If both `estimate` and `get_all_vals` are `FALSE`, the value of the test statistic; otherwise, a list that contains the test statistic and the other values requested (if both are `TRUE`, the test statistic is in the first position and the estimated change point in the second)

References

Rice G, Miller C, Horváth L (????). “A new class of change point test of Rényi type.” in-press.

Examples

```
CPAT:::stat_Vn(rnorm(1000))
CPAT:::stat_Vn(rnorm(1000), kn = function(n) {0.1 * n}, tau = 1/2)
CPAT:::stat_Vn(rnorm(1000), use_kernel_var = TRUE, bandwidth = "nw", kernel = "bo")
```

---

stat_Zn *Compute the Rényi-Type Statistic*

Description

This function computes the Rényi-type statistic.

Usage

```
stat_Zn(dat, kn = function(n) {     floor(sqrt(n)) }, estimate = FALSE,
  use_kernel_var = FALSE, custom_var = NULL, kernel = "ba",
  bandwidth = "and", get_all_vals = FALSE)
```

Arguments

`dat`	The data vector
`kn`	A function corresponding to the trimming parameter t_T; by default, the square root function
`estimate`	Set to `TRUE` to return the estimated location of the change point
`use_kernel_var`	Set to `TRUE` to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if `FALSE`, then the long-run variance is estimated using $\hat{\sigma}^2_{T,t} = T^{-1}\left(\sum_{s=1}^{t}\left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^{T}\left(X_s - \tilde{X}_{T-t}\right)^2\right)$, where $\bar{X}_t = t^{-1}\sum_{s=1}^{t} X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^{T} X_s$; if `custom_var` is not `NULL`, this argument is ignored
`custom_var`	Can be a vector the same length as `dat` consisting of variance-like numbers at each potential change point (so each entry of the vector would be the "best estimate" of the long-run variance if that location were where the change point occured) or a function taking two parameters `x` and `k` that can be used to generate this vector, with `x` representing the data vector and `k` the position of a potential change point; if `NULL`, this argument is ignored
`kernel`	If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
`bandwidth`	If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)
`get_all_vals`	If `TRUE`, return all values for the statistic at every tested point in the data set

Details

The definition of the statistic is

$$\max_{t_T \leq t \leq T - t_T} \hat{\sigma}_{t,T}^{-1} \left| t^{-1} \sum_{s=1}^{t} X_s - (T-t)^{-1} \sum_{s=t+1}^{T} X_s \right|$$

The parameter kn corresponds to the trimming parameter t_T.

Value

If both `estimate` and `get_all_vals` are `FALSE`, the value of the test statistic; otherwise, a list that contains the test statistic and the other values requested (if both are `TRUE`, the test statistic is in the first position and the estimated change point in the second)

Examples

```
CPAT:::stat_Zn(rnorm(1000))
CPAT:::stat_Zn(rnorm(1000), kn = function(n) {floor(log(n))})
CPAT:::stat_Zn(rnorm(1000), use_kernel_var = TRUE, bandwidth = "nw",
               kernel = "bo")
```

`stat_Zn_reg` *Compute the Rényi-Type Statistic for Stability in Linear Regression Models*

Description

This function computes the Rényi-type statistic for detecting structural change in linear regression models.

Usage

```
stat_Zn_reg(formula, data, kn = function(n) {     floor(sqrt(n)) },
  estimate = FALSE, use_kernel_var = FALSE, custom_var = NULL,
  kernel = "ba", bandwidth = "and", get_all_vals = FALSE,
  fast = FALSE)
```

Arguments

`formula`	The regression formula, which will be passed to `lm`
`data`	`data.frame` containing the data
`kn`	A function corresponding to the trimming parameter t_T; by default, the square root function
`estimate`	Set to `TRUE` to return the estimated location of the change point
`use_kernel_var`	Set to `TRUE` to use kernel methods for long-run variance estimation (typically used when the data is believed to be correlated); if `FALSE`, then the long-run variance is estimated using $\hat{\sigma}_{T,t}^2 = T^{-1}\left(\sum_{s=1}^t \left(X_s - \bar{X}_t\right)^2 + \sum_{s=t+1}^T \left(X_s - \tilde{X}_{T-t}\right)^2\right)$, where $\bar{X}_t = t^{-1}\sum_{s=1}^t X_s$ and $\tilde{X}_{T-t} = (T-t)^{-1}\sum_{s=t+1}^T X_s$; if `custom_var` is not `NULL`, this argument is ignored

`custom_var`	Can be a vector the same length as `dat` consisting of variance-like numbers at each potential change point (so each entry of the vector would be the "best estimate" of the long-run variance if that location were where the change point occured) or a function taking two parameters `x` and `k` that can be used to generate this vector, with `x` representing the data vector and `k` the position of a potential change point; if `NULL`, this argument is ignored
`kernel`	If character, the identifier of the kernel function as used in **cointReg** (see `getLongRunVar`); if function, the kernel function to be used for long-run variance estimation (default is the Bartlett kernel in **cointReg**)
`bandwidth`	If character, the identifier for how to compute the bandwidth as defined in **cointReg** (see `getBandwidth`); if function, a function to use for computing the bandwidth; if numeric, the bandwidth value to use (the default is to use Andrews' method, as used in **cointReg**)
`get_all_vals`	If `TRUE`, return all values for the statistic at every tested point in the data set
`fast`	If `TRUE`, the test statistic is computed quickly but at a potential loss of numerical accuracy (by solving the normal equations); otherwise, use slower but more numerically stable solution techniques

Details

TODO: EXTENDED DESCRIPTION

TODO: THIS FUNCTION DOES NOT WORK AS MARKETED BECAUSE WE'RE STILL WORKING ON THE THEORY; `use_kernel_var`, `kernel`, AND `bandwidth` ARE IGNORED AND `custom_var` SHOULD NOT BE NULL, BUT CREATE A MATRIX THE SAME DIMENSION AS THE REGRESSION MODEL.

Value

TODO: RETURN VALUE DESCRIPTION

Examples

```
CPAT:::stat_Zn_reg()  # TODO: EXAMPLE
```

`%s%` *Concatenate (With Space)*

Description

Concatenate and form strings (with space separation)

Usage

```
x %s% y
```

Arguments

`x`	One object
`y`	Another object

Value

A string combining x and y with a space separating them

Examples

```
`%s%` <- CPAT:::`%s%`
"Hello" %s% "world"
```

%s0% *Concatenate (Without Space)*

Description

Concatenate and form strings (no space separation)

Usage

```
x %s0% y
```

Arguments

x	One object
y	Another object

Value

A string combining x and y

Examples

```
`%s0%` <- CPAT:::`%s0%`
"Hello" %s0% "world"
```

Index