Package: PSTR (via r-universe)

June 6, 2026

Type Package

Title Panel Smooth Transition Regression Modelling

Version 2.1.0

Description Implements the Panel Smooth Transition Regression (PSTR) framework for nonlinear panel data modelling. The modelling procedure consists of three stages: Specification, Estimation and Evaluation. The package provides tools for model specification testing, to do PSTR model estimation, and to do model evaluation. The implemented tests allow for cluster dependence and are heteroskedasticity-consistent. The wild bootstrap and wild cluster bootstrap tests are also implemented. Parallel computation (optional) is supported for computationally intensive routines such as bootstrap tests.

Depends R (>= 4.1.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports tibble, R6, knitr, cli, ggplot2, plotly, magrittr

Suggests rmarkdown, future, future.apply, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/yukai-yang/PSTR

BugReports https://github.com/yukai-yang/PSTR/issues

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev

Repository https://yukai-yang.r-universe.dev

Date/Publication 2026-03-08 22:33:13 UTC

RemoteUrl https://github.com/yukai-yang/pstr

RemoteRef HEAD

RemoteSha 4b9f5f018063448f4a69fe5ebf2a926e5ea2b241

Contents

`EstPSTR` *Estimate a PSTR model by nonlinear least squares*

Description

`EstPSTR` estimates either a nonlinear PSTR model (when `iq` is provided) or a linear fixed-effects panel regression (when `iq = NULL`).

Usage

```
EstPSTR(
  use,
  im = 1,
  iq = NULL,
  par = NULL,
  useDelta = FALSE,
  vLower = 2,
  vUpper = 2,
  method = "L-BFGS-B"
)
```

Arguments

`use`	An object of class `"PSTR"` created by `NewPSTR`.
`im`	Integer. Number of switches m in the transition function. Default is `1`.
`iq`	Either an integer index (column number in the transition-variable matrix) or a character string (transition-variable name) specifying which transition variable to use. If `NULL`, a linear fixed-effects panel regression is estimated.

`par`	Numeric vector of length `im + 1` giving initial values for the nonlinear parameters. The expected order is `c(delta, c_1, ..., c_m)` if `useDelta = TRUE`, or `c(gamma, c_1, ..., c_m)` if `useDelta = FALSE`. If `NULL`, defaults are constructed automatically and `useDelta` is ignored.
`useDelta`	Logical. If `TRUE`, the first element of `par` is interpreted as δ. If `FALSE`, it is interpreted as γ and internally converted to $\delta = \log(\gamma)$ before optimisation.
`vLower`	Numeric scalar or vector. Lower offsets defining the lower bounds in the optimiser. Bounds are applied to the internal parameter vector used in optimisation (with the first element being δ).
`vUpper`	Numeric scalar or vector. Upper offsets defining the upper bounds in the optimiser. Bounds are applied to the internal parameter vector used in optimisation (with the first element being δ).
`method`	Character. Optimisation method passed to `stats::optim`. Default is `"L-BFGS-B"` (bounded optimisation).

Details

Two equivalent interfaces are available:

1. Wrapper function: `EstPSTR(use = obj, ...)`.
2. R6 method: `obj$EstPSTR(...)`.

The wrapper calls the corresponding R6 method and returns `use` invisibly.

The transition function is logistic and depends on a transition variable q_{it} and nonlinear parameters $\gamma > 0$ and switching locations $c_1 < \cdots < c_m$:

$$g(q_{it}; \gamma, c_1, \ldots, c_m) = \left(1 + \exp\left[-\gamma \prod_{j=1}^{m} (q_{it} - c_j) \right] \right)^{-1}.$$

The smoothness parameter is internally reparametrised as $\gamma = \exp(\delta)$, where $\delta \in \mathbb{R}$. The optimisation is always carried out in δ and c.

If `par = NULL`, the function constructs default initial values from quantiles of the selected transition variable and treats the first element as δ.

Value

Invisibly returns `use` with estimation results added. In particular, for a nonlinear PSTR model (`iq` not `NULL`), the object contains (among others):

`delta` Estimate of δ.

`gamma` Estimate of $\gamma = \exp(\delta)$.

`c` Estimates of $c_1, \ldots, c_m$.

`vg` Estimated transition-function values g_{it}.

`beta` Estimated coefficients (named as `var_0` for linear-part coefficients and `var_1` for nonlinear-part coefficients).

`vU` Residuals.

`vM` Estimated individual effects.

`s2` Estimated residual variance.

`cov` Cluster-robust and heteroskedasticity-consistent covariance matrix of all estimates.

`se` Standard errors corresponding to `est`.

`est` Vector of all estimates (coefficients followed by nonlinear parameters).

`mbeta` Estimates of coefficients in the second extreme regime (when available).

`mse` Standard errors for `mbeta` (when available).

For a linear fixed-effects model (`iq = NULL`), the object contains `beta`, `vU`, `vM`, `s2`, `cov`, `se`, and `est`.

See Also

`NewPSTR`, `LinTest`, `WCB_LinTest`, `EvalTest`, `stats::optim`.

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala","debta","cfa","sales"),
                tvars = c("vala"), iT = 14)

# 1) Linear fixed-effects model
pstr <- EstPSTR(use = pstr)
print(pstr, mode = "estimates", digits = 6)

# 2) Nonlinear PSTR model
pstr <- EstPSTR(use = pstr, im = 1, iq = 1, useDelta = TRUE,
                par = c(.63, 0), vLower = 4, vUpper = 4)
print(pstr, mode = "estimates", digits = 6)

# R6 method interface (equivalent)
pstr$EstPSTR(im = 1, iq = 1, useDelta = TRUE, par = c(.63, 0), method = "CG")
```

`EvalTest` *Evaluate an estimated PSTR model*

Description

`EvalTest` provides post-estimation evaluation tests for an estimated PSTR model. It supports two null hypotheses:

Parameter constancy No time variation in parameters (labelled `"time-varying"`).

No remaining nonlinearity No remaining nonlinearity/heterogeneity given a candidate transition variable (labelled `"heterogeneity"`).

Usage

```
EvalTest(use, type = c("time-varying", "heterogeneity"), vq = NULL)

WCB_TVTest(use, iB = 100, parallel = FALSE, cpus = 4)

WCB_HETest(use, vq, iB = 100, parallel = FALSE, cpus = 4)
```

Arguments

use	An object of class "PSTR" returned by EstPSTR. The model must be estimated (nonlinear PSTR) before evaluation tests can be run.
type	Character vector. Which evaluation tests to run in EvalTest. Must be a subset of c("time-varying","heterogeneity"). Default is both.
vq	Numeric vector. Candidate transition variable used by the no remaining nonlinearity test. Required if "heterogeneity" is included in type, and required for WCB_HETest.
iB	Integer. Number of bootstrap replications. Default is 100.
parallel	Logical. Whether to use parallel computation (via the **future/future.apply** backend).
cpus	Integer. Number of CPU cores used if parallel = TRUE.

Details

Wild bootstrap (WB) and wild cluster bootstrap (WCB) versions are available via WCB_TVTest (parameter constancy) and WCB_HETest (no remaining nonlinearity).

Two equivalent interfaces are available for each test:

1. Wrapper function, for example EvalTest(use = obj, ...).
2. R6 method, for example obj$EvalTest(...).

Each wrapper calls the corresponding R6 method and returns use invisibly.

The bootstrap variants are computationally intensive. WB is robust to heteroskedasticity, while WCB is additionally robust to within-individual dependence (cluster dependence). Parallel execution can be enabled via parallel and cpus.

Value

Invisibly returns use with evaluation results added.

tv A list of parameter-constancy (time-varying) test results, one element per m.

ht A list of no remaining nonlinearity (heterogeneity) test results, one element per m.

wcb_tv A numeric matrix of WB/WCB p-values for parameter-constancy tests (one row per m).

wcb_ht A numeric matrix of WB/WCB p-values for no remaining nonlinearity tests (one row per m).

The individual list elements in tv and ht contain LM-type test statistics and p-values (including HAC variants), consistent with the output from LinTest.

See Also

NewPSTR, LinTest, WCB_LinTest, EstPSTR.

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala","debta","cfa","sales"),
                tvars = c("vala"), iT = 14)

# estimate first
pstr <- EstPSTR(use = pstr, im = 1, iq = 1, useDelta = TRUE, par = c(.63, 0), method = "CG")

# evaluation tests
pstr <- EvalTest(
  use = pstr,
  type = c("time-varying","heterogeneity"),
  vq = as.matrix(Hansen99['vala'])[,1]
)
print(pstr, mode = "evaluation")

# bootstrap variants (parallel via future/future.apply; can be slow)

# Optional: if parallel bootstrap exports a large object, temporarily
# increase the maximum size of exported globals.
old_max <- getOption("PSTR.future.globals.maxSize")
options(PSTR.future.globals.maxSize = 4 * 1024^3)  # 4 GB, if needed

pstr <- WCB_TVTest(use = pstr, iB = 4, parallel = TRUE, cpus = 2)
pstr <- WCB_HETest(
  use = pstr,
  vq = as.matrix(Hansen99[, "vala"])[, 1],
  iB = 4, parallel = TRUE, cpus = 2
)

# Reset to the previous option value.
options(PSTR.future.globals.maxSize = old_max)

print(pstr, mode = "evaluation")
```

Hansen99 *A balanced panel of 565 US firms observed for the years 1973–1987*

Description

A dataset containing a balanced panel data of annual observations over the period 1973-1987 (15 years) for 560 US firms for the variables described below.

Usage

```
Hansen99
```

Format

A tibble with 7840 rows and 20 variables:

cusip Committee on Uniform Security Identication Procedures firm code number, the first 6 digits (CNUM)

year 2-digit year of the data

inva investment to assets ratio

dt_75 dummy variable for 1975

dt_76 dummy variable for 1976

dt_77 dummy variable for 1977

dt_78 dummy variable for 1978

dt_79 dummy variable for 1979

dt_80 dummy variable for 1980

dt_81 dummy variable for 1981

dt_82 dummy variable for 1982

dt_83 dummy variable for 1983

dt_84 dummy variable for 1984

dt_85 dummy variable for 1985

dt_86 dummy variable for 1986

dt_87 dummy variable for 1987

vala lagged total market value to assets ratio ("Tobin's Q")

debta lagged long term debt to assets ratio

cfa lagged cash flow to assets ratio

sales lagged sales during the year (million USD)

Details

The structure of the dataset is such that the time index runs "fast", while the firm index runs "slow"; that is, first all 14 observations for the first firm are given, then the 14 observations for the second firm, etc.

Since we used one year lagged variables of "vala", "debta", "cfa" and "cfa" as regressors, the records in 1973 are skipped.

All values are nominal and millions of dollars except where otherwise noted. Stocks are end of year.

Source

https://www.ssc.wisc.edu/~bhansen/progs/joe_99.html

`LinTest` *Linearity (homogeneity) tests for PSTR models*

Description

These functions conduct linearity (homogeneity) tests against the alternative of a logistic smooth transition component in a Panel Smooth Transition Regression (PSTR) model.

Usage

```
LinTest(use)

WCB_LinTest(use, iB = 100, parallel = FALSE, cpus = 2)
```

Arguments

`use`	An object of class `"PSTR"` created by `NewPSTR`.
`iB`	Integer. Number of bootstrap repetitions. Default is `100`.
`parallel`	Logical. Whether to use parallel computation in bootstrap routines.
`cpus`	Integer. Number of CPU cores to use when `parallel = TRUE`. Ignored otherwise.

Details

Two equivalent interfaces are available:

1. **Wrapper functions:** `LinTest(use = obj)` and `WCB_LinTest(use = obj, ...)`.
2. **R6 methods:** `obj$LinTest()` and `obj$WCB_LinTest(...)`.

The wrapper functions call the corresponding R6 methods and return the (mutated) object invisibly.

The tests are carried out for each potential transition variable specified in `tvars` when creating the model via `NewPSTR`. For each transition variable, tests are computed for the number of switches $m = 1, \ldots, im$, where im is the maximal number of switches.

The procedures produce two families of tests:

(i) Linearity tests for each m For a fixed m, the null hypothesis is

$$H_0^i : \beta_i = \beta_{i-1} = \cdots = \beta_1 = 0, \qquad i = 1, \ldots, m.$$

(ii) Sequence tests for selecting m These are conditional tests with null

$$H_0^i : \beta_i = 0 \mid \beta_{i+1} = \cdots = \beta_m = 0, \qquad i = 1, \ldots, m.$$

For each hypothesis, four asymptotic LM-type tests are reported:

- χ^2-version LM test.
- F-version LM test.

- χ^2-version HAC LM test (heteroskedasticity and autocorrelation consistent).
- F-version HAC LM test.

`WCB_LinTest` additionally reports wild bootstrap (WB) and wild cluster bootstrap (WCB) p-values. WB is robust to heteroskedasticity, while WCB is robust to both heteroskedasticity and within-individual dependence (cluster dependence). The bootstrap routines can be computationally expensive; parallel execution can be enabled via `parallel = TRUE`.

Results are stored in the returned object (see **Value**).

Value

Both functions return `use` invisibly, after adding the following components:

`test` List. Asymptotic linearity test results for each transition variable and m.

`sqtest` List. Asymptotic sequence test results for each transition variable and m.

`wcb_test` List (only for `WCB_LinTest`). WB and WCB p-values for the linearity tests.

`wcb_sqtest` List (only for `WCB_LinTest`). WB and WCB p-values for the sequence tests.

See Also

`NewPSTR`, `EstPSTR`, `EvalTest`.

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala","debta","cfa","sales"),
                tvars = c("vala"), iT = 14)

# R6 method interface
pstr$LinTest()

# Wrapper interface (equivalent)
pstr <- LinTest(pstr)

# Show results
print(pstr, mode = "tests")


# Bootstrap tests (can be slow).
# For parallel execution, the exported object may exceed the default
# size limit. If that happens, temporarily increase the limit below.
old_max <- getOption("PSTR.future.globals.maxSize")
options(PSTR.future.globals.maxSize = 4 * 1024^3)  # 4 GB, if needed

pstr$WCB_LinTest(iB = 200, parallel = TRUE, cpus = 2)
# or
pstr <- WCB_LinTest(use = pstr, iB = 200, parallel = TRUE, cpus = 2)

# Reset to the previous option value.
options(PSTR.future.globals.maxSize = old_max)
```

```
print(pstr, mode = "tests")
```

NewPSTR *Create a PSTR model object*

Description

Create an R6 object of class "PSTR" to be used as the main container for Panel Smooth Transition Regression (PSTR) modelling in this package. You typically call NewPSTR() once, and then pass the returned object to specification, estimation and evaluation functions.

Usage

```
NewPSTR(data, dep, indep, indep_k = NULL, tvars, im = 1, iT)
```

Arguments

data	A tibble containing the panel in long format. The number of rows must be iT * N for some integer N. Rows are assumed to be ordered by time within individual, consistently with the package conventions.
dep	A single column index or a single column name specifying the dependent variable.
indep	A vector of column indices or column names specifying the regressors in the linear part.
indep_k	Optional. A vector of column indices or column names specifying the regressors in the non-linear part. If NULL, the non-linear part is set equal to the linear part.
tvars	A vector of column indices or column names specifying the candidate transition variables.
im	Integer. The maximal number of switches used in linearity-related tests. Default is 1.
iT	Integer. The time dimension (number of time observations per individual).

Details

The candidate transition variables in tvars will be stored in the object and can be tested one by one by functions such as LinTest.

Missing values in the dependent variable, linear regressors, non-linear regressors, or transition variables are removed internally (row-wise). The number of individuals N is inferred from nrow(data) and iT after removing missing values.

Value

An R6 object of class "PSTR".

See Also

LinTest, WCB_LinTest, EstPSTR, EvalTest, WCB_TVTest, WCB_HETest.

Examples

```
pstr <- NewPSTR(
  Hansen99,
  dep = "inva",
  indep = 4:20,
  indep_k = c("vala", "debta", "cfa", "sales"),
  tvars = c("vala", "debta"),
  iT = 14
)

# print summary (your R6 print method)
pstr
print(pstr, mode = "summary")

# after running tests/estimation, you can print other sections
# print(pstr, mode = "tests")
# print(pstr, mode = "estimates")
# print(pstr, mode = "evaluation")
```

---

plot_coefficients *Plot coefficients, standard errors, and p-values against the transition variable*

Description

This function plots three curves against the transition variable: the coefficient function, its standard error, and the corresponding p-value.

Usage

```
plot_coefficients(obj, vars, length.out = 100, color = "blue", size = 1.5)
```

Arguments

obj	An object of class "PSTR".
vars	A vector of column indices or variable names from the nonlinear part.
length.out	Number of grid points over the transition variable.
color	Line colour.
size	Line width.

Details

For each selected variable j, the curves are

$$f_1(x) = \beta_{0j} + \beta_{1j} g(x; \gamma, c)$$

$$f_2(x) = se\{f_1(x)\}$$

$$f_3(x) = 1 - \Pr\left\{X < [f_1(x)/f_2(x)]^2\right\}$$

where X follows a chi-square distribution with one degree of freedom.

In addition to the exported function `plot_coefficients(obj = ...)`, the same functionality is available as an R6 method via `obj$plot_coefficients(...)`.

Value

A named list of `ggplot2` objects.

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala","debta","cfa","sales"),
                tvars = c("vala","debta","cfa","sales"), iT = 14)

pstr <- EstPSTR(use = pstr, im = 1, iq = 1,
                useDelta = TRUE, par = c(.63,0), method = "CG")

# Exported function
ret <- plot_coefficients(pstr, vars = 1:4)

# R6 method
ret2 <- pstr$plot_coefficients(vars = 1:4)
```

---

plot_response *Plot the expected response against selected variables*

Description

This function plots the effect-adjusted expected response for selected nonlinear variables in a PSTR model.

Usage

```
plot_response(
  obj,
  vars,
  log_scale = FALSE,
  length.out = 20,
  color = "blue",
  size = 1.5
)
```

Arguments

obj	An object of class "PSTR".
vars	Integer vector of column indices from the nonlinear part.
log_scale	Logical scalar or length-2 vector indicating whether to use log scale for the regressor and/or transition variable.
length.out	Scalar or length-2 numeric vector controlling grid size.
color	Line colour.
size	Line width.

Details

If the selected variable differs from the transition variable, a 3-D surface of

$$(\beta_{k,0} + \beta_{k,1} g(q; \gamma, c)) z_k$$

is plotted against z_k and the transition variable.

If the selected variable coincides with the transition variable, a curve is plotted instead.

In addition to the exported function `plot_response(obj = ...)`, the same functionality is available as an R6 method via `obj$plot_response(...)`.

Value

A named list of `ggplot2` (curve) and/or `plotly` (surface) objects.

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala","debta","cfa","sales"),
                tvars = c("vala","debta","cfa","sales"), iT = 14)

pstr <- EstPSTR(use = pstr, im = 1, iq = 1,
                useDelta = TRUE, par = c(.63,0), method = "CG")

# Exported interface
ret <- plot_response(pstr, vars = 1:4)

# R6 method
```

```
ret2 <- pstr$plot_response(vars = 1:4)
```

`plot_target` *Plot the surface of the target function for nonlinear least squares estimation*

Description

This function plots a 3-D surface of the nonlinear least squares (NLS) target function used in estimating a PSTR model. It is mainly intended as a diagnostic tool for choosing reasonable initial values for the nonlinear parameters.

Usage

```
plot_target(
  obj,
  im = 1,
  iq = NULL,
  par = NULL,
  basedon = c(1, 2),
  from,
  to,
  length.out = 40
)
```

Arguments

`obj`	An object of class `"PSTR"`.
`im`	Integer. The number of switches m in the transition function used to construct the target function surface. Default is `1`.
`iq`	Either an integer index (a column number in the transition-variable matrix) or a character string (a transition-variable name) specifying which transition variable is used when computing the target function.
`par`	Numeric vector of length `1 + im` giving fixed values of the nonlinear parameters in the order `c(delta, c_1, ..., c_m)`. If `NULL`, the function constructs a default initial vector from quantiles of the chosen transition variable.
`basedon`	Integer vector of length `2`. Positions in `par` indicating the two parameters to vary on the grid. The values at these positions in `par` are ignored and replaced by grid values. Use `1` for δ, and `2:(im+1)` for $c_1, \ldots, c_m$.
`from`	Numeric vector of length `2`. Lower bounds (start values) for the two grid parameters specified by `basedon`.
`to`	Numeric vector of length `2`. Upper bounds (end values) for the two grid parameters specified by `basedon`.

`length.out` Either a scalar or a numeric vector of length 2. If scalar, the same number of grid points is used for both dimensions. If length 2, the first element controls the grid resolution for the first parameter in basedon, and the second for the second parameter.

Details

The target function is evaluated on a two-dimensional grid over two selected parameters, while all other nonlinear parameters are held fixed at values provided by par. The nonlinear parameter vector is always ordered as $\delta, c_1, \ldots, c_m$, where $\gamma = \exp(\delta)$ and $m = im$.

In addition to the exported function `plot_target(obj = ...)`, the same functionality is available as an R6 method via `obj$plot_target(...)`.

Value

A plotly object representing a 3-D surface plot of the target function values evaluated on the specified parameter grid.

See Also

NewPSTR, LinTest, WCB_LinTest, EstPSTR, EvalTest, WCB_TVTest, WCB_HETest

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala", "debta", "cfa", "sales"),
                tvars = c("vala"), iT = 14)

# 1) Exported function interface
ret <- plot_target(obj = pstr, iq = 1, basedon = c(1, 2),
                   from = c(log(1), 6), to = c(log(18), 10),
                   length.out = c(40, 40))

# 2) R6 method interface
ret2 <- pstr$plot_target(iq = 1, basedon = c(1, 2),
                         from = c(log(1), 6), to = c(log(18), 10),
                         length.out = c(40, 40))
```

---

plot_transition *Plot the transition function of an estimated PSTR model*

Description

This function plots the estimated transition function $g(q; \gamma, c)$ of a fitted PSTR model.

Usage

```
plot_transition(
  obj,
  size = 1.5,
  color = "blue",
  xlim = NULL,
  ylim = NULL,
  fill = NULL,
  alpha = NULL
)
```

Arguments

obj	An object of class "PSTR".
size	Point size.
color	Point colour.
xlim	Optional numeric vector of length 2 specifying x-axis limits.
ylim	Optional numeric vector of length 2 specifying y-axis limits.
fill	Optional colour for highlighting the support of observed q.
alpha	Transparency level for points and shading.

Details

Observed transition values are displayed together with the fitted transition curve. For models with multiple switches, multiple curves are shown.

In addition to the exported function `plot_transition(obj = ...)`, the same functionality is available as an R6 method via `obj$plot_transition(...)`.

Value

A ggplot2 object.

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala","debta","cfa","sales"),
                tvars = c("vala"), iT = 14)

pstr <- EstPSTR(use = pstr, im = 1, iq = 1,
                useDelta = TRUE, par = c(.63,0), method = "CG")

# Exported function
plot_transition(pstr)

# R6 method
pstr$plot_transition()
```

print.PSTR *Print a PSTR model object*

Description

Print method for objects of class "PSTR".

Arguments

x	An object of class "PSTR".
...	Further arguments passed to the underlying print routine. See **Arguments** below.

Details

The print output is organised into four sections:

"summary" Data summary: panel dimensions, dependent variable, linear/non-linear regressors, transition variables.

"tests" Specification tests: linearity (homogeneity) tests and the sequence of homogeneity tests (optionally with WB/WCB p-values if available).

"estimates" Estimation results: coefficient estimates with standard errors and t-ratios, printed in chunks to fit the console width.

"evaluation" Evaluation tests: parameter constancy and no-remaining-nonlinearity tests (optionally with WB/WCB p-values if available).

In addition to calling print(x, ...), the same functionality is available as an R6 method via x$print(...).

Value

Invisibly returns x.

Arguments

The following arguments are supported (they are forwarded to the R6 method x$print()):

format Character. Output format passed to knitr::kable() (for example "simple", "pipe", "latex"). Default is "simple".

mode Character vector specifying which sections to print. It is matched (partially) against c("summary","tests","estimat
Default is "summary".

digits Integer. Number of significant digits used in printed tables. Default is 4.

See Also

NewPSTR, LinTest, WCB_LinTest, EstPSTR, EvalTest, WCB_TVTest, WCB_HETest.

Examples

```
pstr <- NewPSTR(Hansen99, dep = "inva", indep = 4:20,
                indep_k = c("vala","debta","cfa","sales"),
                tvars = c("vala","debta","cfa","sales"), iT = 14)

# default: summary only
pstr

# specification tests
print(pstr, mode = "tests", format = "simple")
print(pstr, mode = "tests", format = "pipe", caption = "The test results")

# estimates
print(pstr, mode = "estimates")

# evaluation
print(pstr, mode = "evaluation")

# R6 method interface (same output)
pstr$print(mode = c("summary","tests"))
```

`sunspot` *Transformed Wolf annual sunspot numbers for the years 1710-1979*

Description

A dataset containing the transformed Wolf annual sunspot numbers for the years 1710-1979.

Usage

```
sunspot
```

Format

A tibble with 270 rows and 11 variables:

spot_0 transformed sunspot

spot_1 transformed sunspot, lag one

spot_2 transformed sunspot, lag two

spot_3 transformed sunspot, lag three

spot_4 transformed sunspot, lag four

spot_5 transformed sunspot, lag five

spot_6 transformed sunspot, lag six

spot_7 transformed sunspot, lag seven

spot_8 transformed sunspot, lag eight

spot_9 transformed sunspot, lag nine

spot_10 transformed sunspot, lag ten

Details

Each column of the data matrix is a lagged transformed sunspot observations from lag order 0 to 10.

The data were transformed by using the formula

$$y_t = 2\left\{(1 + x_t)^{1/2} - 1\right\}$$

see Ghaddar and Tong (1981)

References

Ghaddar, D. K. and Tong, H. (1981) Data transformation and self-exciting threshold autoregression, Applied Statistics, 30, 238–48.

Source

https://www.sidc.be/html/sunspot.html

`version` *Show the version number of some information.*

Description

This function shows the version number and some information of the package.

Usage

```
version()
```

Author(s)

Yukai Yang, <yukai.yang@statistik.uu.se>

Index