Package ‘PSTR’

June 3, 2019

Type Package
Title Panel Smooth Transition Regression Modelling
Version 1.2.4
Description Provides the Panel Smooth Transition Regression (PSTR) modelling. The modelling procedure consists of three stages: Specification, Estimation and Evaluation. The package offers sharp tools helping the package user(s) to conduct model specification tests, to do PSTR model estimation, and to do model evaluation. The tests implemented in the package allow for cluster-dependency and are heteroskedasticity-consistent. The wild bootstrap and wild cluster bootstrap tests are also implemented. Parallel computation (as an option) is implemented in some functions, especially the bootstrap tests. The package suits tasks running many cores on super-computation servers.

Depends R (>= 3.0.0)
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
Imports tibble, snowfall, ggplot2, plotly, stats, magrittr
Suggests knitr, rmarkdown
URL https://github.com/yukai-yang/PSTR
BugReports https://github.com/yukai-yang/PSTR/issues
VignetteBuilder knitr
NeedsCompilation no
Author Yukai Yang [aut, cre] (https://orcid.org/0000-0002-2623-8549)
Maintainer Yukai Yang <yukai.yang@statistik.uu.se>
Repository CRAN
Date/Publication 2019-06-03 19:20:04 UTC
EstPSTR

Description

This function implements the estimation of the PSTR model.

Usage

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

Arguments

- **use**: an object of the class PSTR, created by `NewPSTR` function.
- **im**: specifies the number of switches in the transition function. The default value is 1.
- **iq**: a column number (in `mq`) or variable name specifying the transition variable to use.
- **par**: initial values for the parameters $\gamma$ or $\delta$, and $c$ to be optimized over. It is a vector of length $im + 1$, where $im$ is the number of switches. When missing, the function will choose the initial values automatically, and `useDelta=TRUE`.
- **useDelta**: whether delta is used in par in the estimation. Note that if `par` is missing, this argument will be ignored.
- **vLower**: a vector or number of the lower offsets determining the lower bounds of the parameters. The lower bounds of the parameters are `par - vLower`.
- **vUpper**: a vector or number of the upper offsets determining the upper bounds of the parameters. The upper bounds of the parameters are `par + vUpper`.
- **method**: the method to be used in optimization. See the function `stats::optim`.
**Details**

The function needs the return value (an object of the class PSTR) from the `NewPSTR`. It copies the object, reuses its contents to estimate the corresponding PSTR model, and then returns a new object of the class PSTR containing the results from the estimation. The user can choose to save the return value to a new object or simply to overwrite the object returned from `NewPSTR`.

The PSTR model to be estimated takes the logistic form in nonlinearity. Remember the \( g \) function in the model. It takes the form

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

with \( \gamma > 0 \) and \( c_1 < c_2 < ... < c_m \). \( \gamma \) can be reparametrized as \( \gamma = \exp \delta \) where \( \delta \) is a real number.

The user should have obtained the information about which transition variable \( (q_{it}) \) to use (from `Lintest` and/or `WCB_Lintest`) in estimation before running the function to estimate the model.

The estimation function never change the existing values in the input PSTR object. It adds more values (attributes) into the input object and return.

**Value**

a new object of the class PSTR containing the results from the estimation.

The object is a list containing the components made in `NewPSTR` and the following new components:

- `iq`: specify which transition variable will be used in estimation. The default value `NULL` implies a linear panel regression model.
- `delta`: the estimate of \( \delta \).
- `c`: the estimates of \( c \).
- `vg`: the values of the transition function given the estimates of \( \delta \) and \( c \) and the transition variables \( q_{it} \).
- `beta`: the estimates of the coefficient parameters.
- `vu`: the residuals.
- `vM`: a vector of the estimated time-invariant individual effect.
- `s2`: the variance of the residuals.
- `cov`: the covariance matrix of the estimates which is cluster-dependency and heteroskedasticity consistent.
- `est`: a vector of all the estimates
- `se`: a vector of the standard errors of all the estimates which is cluster-dependency and heteroskedasticity consistent.
- `mbeta`: a vector of the estimates of the parameters in the second extreme regime.
- `mse`: a vector of the standard errors of the estimates of the parameters in the second extreme regime.
- `convergence`: an integer code showing the convergence, see `optim`.
- `par`: a vector of the initial values used in the optimization. Note that the first element is always delta, no matter whether gamma is used as input.
Author(s)
Yukai Yang, <yukai.yang@statistik.uu.se>

See Also
NewPSTR, LinTest and WCB_LinTest

Examples

```r
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debta','cfa','sales'), tvars=c('vala'), iT=14) # create a new PSTR object

# estimate a linear panel regression model
pstr = EstPSTR(use=pstr)
print(pstr, "estimates", digits=6)

# "L-BFGS-B" is used by default
pstr = EstPSTR(use=pstr, im=1, iq=1, useDelta=TRUE, par=c(.63,0), vLower=4, vUpper=4)
# You can also choose the method yourself.
pstr = EstPSTR(use=pstr, im=1, iq=1, useDelta=TRUE, par=c(.63,0), method='CG')

print(pstr, "estimates", digits=6)

# The estimation of a linear panel regression model with fix effects is also implemented.
pstr0 = EstPSTR(use=pstr)

print(pstr0,"estimates")
```

---

### EvalTest

Conduct the evaluation tests.

#### Description

These functions conduct the evaluation tests against two alternatives: 1. the parameters are time-varying and 2. there is remaining nonlinearity (remaining heterogeneity).

#### Usage

- `EvalTest(use, type = c("time-varying", "heterogeneity"), vq = NULL)`
- `WCB_TVTest(use, IB = 100, parallel = F, cpus = 4)`
- `WCB_HETest(use, vq, IB = 100, parallel = F, cpus = 4)`
**EvalTest**

**Arguments**

- **use**: an object of the class PSTR, created by `EstPSTR` function.
- **type**: a character vector specifying the types of the evaluation tests to be conducted. The value can be taken either or both of "time-varying" "heterogeneity". By default, do both.
- **vq**: a vector of a new transition variable for the no remaining nonlinearity test.
- **ib**: specify the number of repetitions in the bootstrap procedure. By default, it is 100.
- **parallel**: a boolean value showing if the parallel computation is applied.
- **cpus**: number of cores used in the parallel computation. The value will be ignored if `parallel=F`.

**Details**

`EvalTest` implements the evaluation tests.

- `wcb_tvtest` implements the wild bootstrap (WB) and the wild cluster bootstrap (WCB) evaluation test of no time-varying parameters.
- `wcb_hetest` implements the wild bootstrap (WB) and the wild cluster bootstrap (WCB) evaluation test of no remaining nonlinearity (no remaining heterogeneity).

The functions need the return value (an object of the class PSTR) from the `EstPSTR`. Note that the PSTR model should be estimated before conducting the evaluation tests. They copy the object, reuse its contents, especially the estimates, to produce the evaluation test results, and then return a new object of the class PSTR. The user can choose to save the return value to a new object or simply to overwrite the object returned from `EstPSTR`. See the example below.

The functions conduct two kinds of evaluation tests. The first kind of tests does the time-varying evaluation tests. The second kind of tests does the no remaining nonlinearity (no remaining heterogeneity) evaluation tests based on the vector of a new transition variable that the user input in the arguments.

The results of the evaluation tests include four kinds of tests

- $\chi^2$-version LM test: the LM test with asymptotically $\chi^2$ distribution under the null hypothesis. The finite sample actual size is supposed to be improved.
- $F$-version LM test: the LM test with asymptotically $F$ distribution under the null hypothesis. The finite sample actual size is supposed to be improved.
- $\chi^2$-version HAC test: the HAC LM test with asymptotically $\chi^2$ distribution under the null hypothesis, which is heteroskedasticity and autocorrelation consistent.
- $F$-version HAC test: the HAC LM test with asymptotically $F$ distribution under the null hypothesis, which is heteroskedasticity and autocorrelation consistent. The finite sample actual size is supposed to be improved.

The wild bootstrap (WB) evaluation tests are heteroskedasticity robust, while the wild cluster bootstrap (WCB) ones are both cluster-dependency and heteroskedasticity robust. Cluster-dependency implies that there can be dependency (autocorrelation) within individual, but no correlation across individuals. The WB and WCB tests may take quite a long time to run which depends on the model specification and the number of repetitions `ib`. It is strongly recommended to use super-computation.
server with many cores to run the code instead of a personal computer. The user may first try a small number of repetitions and estimate the time consumed for a larger number of repetitions.

The functions never change the existing values in the input PSTR object. They add more values (attributes) into the input object and return.

**Value**

a new object of the class PSTR containing the results from the evaluation tests.

The return object from EvalTest contains the following new components:

- **tv**
  a list of the time-varying evaluation tests. The length of the list is the maximal number of switches. Each element of the list corresponds to the time-varying evaluation test results based on different number of switches.

- **ht**
  a list of the no remaining nonlinearity (no remaining heterogeneity) evaluation tests. The length of the list is the maximal number of switches. Each element of the list corresponds to the time-varying evaluation test results based on different number of switches. The input vector of a new transition variable is used to compute the tests.

The return object from WCB_TVTest contains the following new component:

- **wcb_tv**
  a matrix containing the results from the WB and WCB time-varying tests. Each row of the matrix contains the p-value of the WB and WCB tests.

The return object from WCB_HETest contains the following new component:

- **wcb_ht**
  a matrix containing the results from the WB and WCB no remaining nonlinearity (heterogeneity) tests. Each row of the matrix contains the p-value of the WB and WCB tests.

**Author(s)**

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

**See Also**

NewPSTR, LinTest, WCB_LinTest and EstPSTR

**Examples**

```r
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debta','cfa','sales'),
    tvars=c('vala'), it=14)  # create a new PSTR object

# Estimate the model first
pstr = EstPSTR(use=pstr, lm=1, iq=1, useDelta=TRUE, par=c(.63,0), method='CG')

# Then you can evaluate the model
pstr = EvalTest(use=pstr, vq=pstr$mQ[,1])

print(pstr, "eval")
```
# You can do the wild bootstrap and wild cluster bootstrap

```r
library(snowfall)

pstr = WCB_TVTest(use=pstr, iB=4, parallel=TRUE, cpus=2)

# pstr$mQ[,1] is the transition variable stored in the object
# You can also try other transition variables.

pstr = WCB_HETest(use=pstr, vq=pstr$mQ[,1], iB=4, parallel=TRUE, cpus=2)

print(pstr, "eval")

# Don't forget to change the values of iB and cpus during experiments.
```

---

**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 Identification 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


LinTest Conduct the linearity tests.

Description

These functions conduct the linearity tests against the alternative of a logistic smooth transition nonlinear component.

Usage

LinTest(use)

WCB_LinTest(use, iB = 100, parallel = F, cpus = 4)

Arguments

use an object of the class PSTR, created by NewPSTR function.
iB specify the number of repetitions in the bootstrap procedure. By default, it is 100.
parallel a boolean value showing if the parallel computation is applied.
cpus number of cores used in the parallel computation. The value will be ignored if parallel=F.
Details

LinTest implements the linearity tests.

wcb_lintest implements the wild bootstrap (WB) and the wild cluster bootstrap (WCB) linearity tests.

The functions need the return value (an object of the class PSTR) from the NewPSTR. They copy the object, reuse its contents to produce the linearity test results, and then return a new object of the class PSTR. The user can choose to save the return value to a new object or simply to overwrite the object returned from NewPSTR. See the example below.

The functions conduct two kinds of linearity tests.

The first kind of tests does the linearity tests based on each potential transition variable specified in the argument tvars when the user calls the NewPSTR function. For each potential transition variable, the function conducts linearity tests for numbers of switches from 1 up to im. The linearity tests has the null hypothesis

\[ H_0^i : \beta_i = \beta_{i-1} = \beta_{i-2} = ... = \beta_1 = 0 \]

for \( i = 1, ..., m \), where \( m \) is the maximal number of switches im.

The second kind does the linearity tests for selecting the number of switches based on each potential transition variable. The linearity tests for selecting the number of switches has the null hypothesis

\[ H_0^i : \beta_i = 0 | \beta_{i+1} = \beta_{i+2} = ... = \beta_m = 0 \]

for \( i = 1, ..., m \), where \( m \) is the maximal number of switches im.

The results of the linearity tests include four kinds of tests

- \( \chi^2 \)-version Linearity test: the linearity LM test with asymptotically \( \chi^2 \) distribution under the null hypothesis of linearity.
- F-version Linearity test: the linearity LM test with asymptotically \( F \) distribution under the null hypothesis of linearity. The finite sample actual size is supposed to be improved.
- \( \chi^2 \)-version HAC Linearity test: the linearity LM test with asymptotically \( \chi^2 \) distribution under the null hypothesis of linearity, which is heteroskedasticity and autocorrelation consistent.
- F-version HAC Linearity test: the linearity LM test with asymptotically \( F \) distribution under the null hypothesis of linearity, which is heteroskedasticity and autocorrelation consistent. The finite sample actual size is supposed to be improved.

The wild bootstrap (WB) tests are heteroskedasticity robust, while the wild cluster bootstrap (WCB) ones are both cluster-dependency and heteroskedasticity robust. Cluster-dependency implies that there can be dependency (autocorrelation) within individual, but no correlation across individuals. The WB and WCB tests may take quite a long time to run which depends on the model specification and the number of repetitions ib. It is strongly recommended to use super-computation server with many cores to run the code instead of a personal computer. The user may first try a small number of repetitions ib and estimate the time consumed for a larger number of ib.

The two functions never change the existing values in the input PSTR object. They add more values (attributes) into the input object and return.
Value

a new object of the class PSTR containing the results from the linearity tests. The object is a list containing the components made in NewPSTR and the following new components:

- **test**: a list of the linearity test results. The length is the number of potential transition variables specified when creating the object of the class PSTR by calling NewPSTR. See argument tvars in NewPSTR. Each element of the list corresponds to the linearity test results based on the corresponding transition variable, and the element is also a list whose elements correspond to different numbers of switches.

- **sqtest**: a list of the linearity test results for selecting number of switches. It has the same length as test. Each element of the list corresponds to the linearity test results based on the corresponding transition variable, and the element is also a list whose elements correspond to different numbers of switches.

- **wcb_test**: a list of the linearity test results. The length is the number of potential transition variables specified when creating the object of the class PSTR by calling NewPSTR. See argument tvars in NewPSTR. Each element of the list is a matrix containing the linearity test results (p-values) based on the corresponding transition variable. The rows are different numbers of switches, and two columns from WB to WCB.

- **wcb_sqtest**: a list of the linearity test results for selecting number of switches. It has the same length as test. Each element of the list is a matrix containing the linearity test results based on the corresponding transition variable. The rows are different numbers of switches, and two columns from WB to WCB.

Author(s)

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

See Also

NewPSTR

Examples

```r
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debtz','cfa','sales'),
               tvars=c('vala'), iT=14) # create a new PSTR object

pstr = LinTest(pstr)

print(pstr, "tests")

# Don't forget to attach the package for the parallel computation.
library(snowfall)

# you should not run this on personal computer!
# pstr = WCB_LinTest(use=pstr, iB=5000, parallel=TRUE, cpus=50)
```
NewPSTR

# a light version for checking on your personal computer.
pstr = WCB_LinTest(use=pstr, IB=4, parallel=TRUE, cpus=2)

print(pstr, "tests")

NewPSTR

Create an object of the class PSTR.

Description
Create an object of the S3 class PSTR for later usage. This function should be run prior to the other functions in the package. It will return an object which you will use as an input for the other functions. It builds up the basic settings for the Panel Smooth Transition Regression (PSTR) Modelling.

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>a tibble of data. The number of rows of data must be the sample size iT times individuals number N.</td>
</tr>
<tr>
<td>dep</td>
<td>column number or name of the dependent variable. Note that this must be specified.</td>
</tr>
<tr>
<td>indep</td>
<td>a vector of column numbers of names of the independent variables. Note that this must be specified.</td>
</tr>
<tr>
<td>indep_k</td>
<td>a vector of column numbers of names of the independent variables in the nonlinear part. If indep_k is not given (= NULL), the nonlinear part will be the same as the linear part.</td>
</tr>
<tr>
<td>tvars</td>
<td>a vector of column numbers or names of the potential transition variables to be tested.</td>
</tr>
<tr>
<td>im</td>
<td>maximal number of switches in the transition function used in the linearity evaluation tests, by default im=1.</td>
</tr>
<tr>
<td>iT</td>
<td>sample size.</td>
</tr>
</tbody>
</table>

Details
Potential transition variables in tvars will be tested one by one in, for example, LinTest function. There is no need to specify the number of individuals, as it will be obtained automatically inside the function given the number of rows and the sample size iT.

NAs in data are removed automatically inside the function.
plot_coefficients

Value
An object of the class PSTR for later usage.
The object is a list containing the following components:

- iT: the time length of the panel
- iN: the number of individuals
- vY: the vector of the dependent variable
- mX: the matrix of the explanatory variables in the linear part
- mK: the matrix of the explanatory variables in the nonlinear part
- mQ: the matrix of the potential transition variables
- im: the maximal number of switches used in the linearity test

Author(s)
Yukai Yang. <yukai.yang@statistik.uu.se>

See Also
LinTest

Examples
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debt','cfa','sales'),
  tvars=c('vala','debt'), iT=14)

pstr
print(pstr,"summary")

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

Description
This function plots the curves of the coefficients, the standard errors and the p-values against the transition variable.

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

- **obj**: an object of the class PSTR returned from some functions in the package. Note that the corresponding PSTR model must be estimated first.
- **vars**: a vector of column numbers or names (character strings) specifying which variables in the nonlinear part to use.
- **length.out**: a scalar of desired length (number of points) for building the x-axis. 100 by default.
- **color**: the color of the lines.
- **size**: the size of the lines.

Details

The curves of the coefficients, the standard errors and the p-values against the transition variable are functions

\[ f_1(x) = \beta_0 + \beta_1 g(x; \gamma, c) \]
\[ f_2(x) = se(f_1(x)) \]
\[ f_3(x) = 1 - \text{Prob}\{X < [f_1(x)/f_2(x)]^2\} \]

where \(x\) is a variable taking the position of the transition variable, \(se\) stands for the cluster-robust and heteroskedasticity-consistent standard error of the estimate \(f_1(x)\) at \(x\), \(X\) is a random variable following chi-square distribution with degrees of freedom one.

More than one variable can be put in **vars**.

The return value is a list of the same length as **vars**, whose elements are plottable objects.

Value

A list of plottable objects from the *ggplot2* package.

Author(s)

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

See Also

Functions which return an object of the class PSTR can be input into this function

*EstPSTR*

Examples

```
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debta','cfa','sales'),
               tvars=c('vala','debta','cfa','sales'), iT=14) # create a new PSTR object

# estimate the PSTR model first
pstr = EstPSTR(use=pstr, im=1, iq=1, useDelta=TRUE, par=c(.63,0), method='CG')

# plot the curve and surfaces
```
plot_response

Curve or surfaces of the expected response against the corresponding variable.

Description

This function plots the curve or the surfaces of the expected response against the corresponding variable (and the transition variable if surface).

Usage

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

Arguments

- **obj**: an object of the class PSTR returned from some functions in the package. Note that the corresponding PSTR model must be estimated first.
- **vars**: a vector of column numbers or names (character strings) specifying which variables in the nonlinear part to use.
- **log_scale**: a 2-dim vector or scalar specifying whether to take log scale for the variables and the transition variable.
- **length.out**: a 2-dim vector or scalar of desired length (number of points) for the parameters. 20 by default.
- **color**: the color of the line.
- **size**: the size of the line.

Details

The expected response is the expected value of the dependent variable minus the individual effect and all the other variables times their estimated coefficients. That is, if the variable is $z_{k,it}$ in both $x_{it}$ and $z_{it}$, then the function plots the surface of

$$y_{it} - \mu_t - \beta_{-k,0} \cdot x_{-k,it} + \beta_{-k,1} \cdot z_{-k,1} \cdot g_{it} - u_{it}$$

or simply

$$(\beta_{k,0} + \beta_{k,1} g_{it}) \cdot z_{k,it}$$

where $-k$ means with the $k$th element removed, against $z_{k,it}$ and $q_{it}$ if $z_{k,it} \neq q_{it}$.

If $z_{k,it} = q_{it}$, then the function plots the curve of the expected response defined above against $z_{k,it}$.
More than one variable can be put in \texttt{vars}. If \texttt{vars} contains the transition variable and the transition variable belongs to the nonlinear part, the function will plot a curve of the effect-adjusted expected response and the transition variable, otherwise, the function will plot a 3-D surface of the effect-adjusted expected response against a chosen variable in the nonlinear part and the transition variable. \texttt{length.out} takes a vector or a scalar. The vector must be two dimensional specifying numbers of points in the grid built for the surface. The first element of the vector corresponds to the variables, and the second to the transition variable. If it is a scalar, then grid has the same number of points for the variables and the transition variable.

The return value is a list of the same length as \texttt{vars}, whose elements are plottable objects.

\textbf{Value}

A list of plottable objects from the \texttt{ggplot2} (for curve) and/or \texttt{plotly} (for surface) package.

\textbf{Author(s)}

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

\textbf{See Also}

Functions which return an object of the class PSTR and can be input into this function

\texttt{EstPSTR}

\textbf{Examples}

```r
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debt','cfa','sales'),
                 tvars=c('vala','debt','cfa','sales'), iF=14) # create a new PSTR object

# estimate the PSTR model first
pstr = EstPSTR(use=pstr, im=1, iq=1, useDelta=TRUE, par=c(.63,0), method='CG')

# plot the curve and surfaces
ret = plot_response(obj=pstr, vars=1:4, log_scale = c(FALSE,TRUE), length.out=40)
attributes(ret)
ret$vala
ret$debt
```

\begin{verbatim}
plot_target
\end{verbatim}

\textit{Plot the surface of the target function for the nonlinear least square estimation.}

\textbf{Description}

This function plots the surface of the target function for the nonlinear least square estimation. It is useful for finding the suitable initial value for the estimation.
Usage

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

Arguments

obj  an object of the class PSTR returned from some functions in the package.
im  specifies the number of switches in the transition function. The default value is 1.
iq  a column number (in $mQ$) or variable name specifying the transition variable to use.
par  a vector of the values of the parameters. NULL by default, then it will be made automatically.
basedon  an integer vector of length 2 specify which two parameters to use to build the grid.
from  a vector of length 2 of the starting (minimal) values of the parameters.
to  a vector of length 2 of the end (maximal) values of the parameters.
length.out  a 2-dim vector or scalar of desired length (number of points) for the parameters. 40 by default.

Details

The function uses the plotly package to plot the 3-D surface of the target function for the nonlinear least square estimation.

The function takes the PSTR object as one of the inputs. The user needs to give the number of switches $im$, and the transition variable $iq$, such that the target function values can be computed.

The number of parameters to estimate in the nonlinear least square estimation is $1 + im$, that is, one smoothness parameter and the $im$ switching locations. However, the 3-D plot is based on only two changing parameters with the others (if more than two parameters) constant. Thus, the user needs to input a vector $par$, which gives the values of the other parameters. Note that $par$ should still be of length $1 + im$ with the order $\delta$ (always use delta in this function), $c_1, ..., c_m$.

The user should give the vector $basedon$ of length two, that shows which two parameters will be used to build the grid. $basedon$ gives the positions of the two parameters in $par$. Thus, the values in the positions $basedon$ in $par$ will not be used.

$from$, $to$ and $length.out$ serve to build the grid for the two parameters. These arguments must be of length two for the two parameters, respectively. See the seq function for the details.

Value

A plottable object from the plotly package.

Author(s)

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

See Also

Functions which return an object of the class PSTR and can be input into this function

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

Examples

```
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debt','cfa','sales'),
              tvars=c('vala'), iT=14) # create a new PSTR object

# build the grid based on the first two parameters
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))
```

plot_transition

Plot the transition function of the estimated PSTR model.

Description

This function plots the transition function of the estimated PSTR model.

Usage

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

Arguments

- **obj**
  - an object of the class PSTR returned from some functions in the package. Note that the corresponding PSTR model must be estimated first.
- **size**
  - the size of the circle.
- **color**
  - the color of the circle.
- **xlim**
  - a numeric vector of dimension 2 specifying the limits of x-axis.
- **ylim**
  - a numeric vector of dimension 2 specifying the limits of y-axis.
- **fill**
  - the color used to fill the area on the transition curve with observations, NULL for not fill, see ggplot2.
- **alpha**
  - a number controlling the transparency of the points and filled area, NULL for default, see ggplot2.

Details

The function uses some functions in the ggplot2 package and aims to give a quick plot of the transition function. The user can customize the title, subtitle, caption, x and y labels, for details, read the help file for the `labs` function in ggplot2.
Value

A ggplot object. The user can plot it simply by print the object.

Author(s)

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

See Also

Functions which return an object of the class PSTR and can be input into this function

EstPSTR

Examples

```r
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debt','fca','sales'),
tvars=c('vala'), id=14) # create a new PSTR object

# estimate the PSTR model
pstr = EstPSTR(use=pstr, im=1, iq=1, useDelta=TRUE, par=c(.63,0), method='CG')

# plot the transition function
ret = plot_transition(pstr)
# plot by running
ret
ret = plot_transition(pstr, fill='blue', xlim=c(-2,20), color = "dodgerblue4", size = 2, alpha=.3)
ret
```

---

**print.PSTR**

*Print the object of the class PSTR.*

Description

This function prints the object of the class PSTR.

Usage

```r
## S3 method for class 'PSTR'
print(x, mode = c("su", "e"), digits = 4, ...)
```
Arguments

- **x**: an object of the class PSTR returned from some functions in the package. See below "See Also" for a list of these functions.

- **mode**: a vector of character strings specifying which results to print. It takes the values `c('summary', 'tests', 'estimates', 'evaluation')`. By default 'su' and 'e' which means all.

- **digits**: integer indicating the number of decimal places (for the `round` function inside) to be used. Negative values are allowed (see `round`).

- **...**: further arguments passed to or from other methods. Ignored here.

Author(s)

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

See Also

Functions which return an object of the class PSTR:

- `NewPSTR`, `LinTest`, `WCB_LinTest`, `EstPSTR`, `EvalTest`, `WCB_TVTest` and `WCB_HETest`

Examples

```r
pstr = NewPSTR(Hansen99, dep='inva', indep=4:20, indep_k=c('vala','debta','cfa','sales'),
    tvars=c('vala','debta','cfa','sales'), if=14)
print(pstr)
print(pstr, mode='summary', digits=2)
```

PSTR: A package implementing the Panel Smooth Transition Regression (PSTR) modelling.

Description

The package implements the Panel Smooth Transition Regression (PSTR) modelling.

Details

The modelling procedure consists of three stages: Specification, Estimation and Evaluation. The package offers tools helping the package users to conduct model specification tests, to do PSTR model estimation, and to do model evaluation.

The cluster-dependency and heteroskedasticity-consistent tests are implemented in the package.

The wild bootstrap and cluster wild bootstrap tests are also implemented.

Parallel computation (as an option) is implemented in some functions, especially the bootstrap tests. Therefore, the package suits tasks running many cores on super-computation servers.

The Panel Smooth Transition Regression (PSTR) model is defined to be

\[
y_{it} = \mu_i + \beta_0 x_{it} + \beta_1 z_{it} g_{it} + u_{it}
\]
where $g_{it}$ is the transition function taking the logistic form with the transition variable for individual $i$, $x_{it}$ contains the explanatory variables in the linear part, and $z_{it}$ contains the explanatory variables in the nonlinear part, and they can be different.

The transition function $g_{it}$ takes the logistic form

$$g(q_{it}; \gamma, c) = \left(1 + \exp\left(-\gamma \prod_{j=1}^{m}(q_{it} - c_j)\right)\right)^{-1}$$

with $\gamma > 0$ and $c_1 < c_2 < ... < c_m$. $\gamma$ can be reparametrized as $\gamma = \exp \delta$ where $\delta$ is a real number.

Author and Maintainer

Yukai Yang
Department of Statistics, Uppsala University
<yukai.yang@statistik.uu.se>

References


Function for Initialization

NewPSTR initialize the modelling by creating an object of the class PSTR.

Functions for Model Specification

LinTest implements the linearity tests.
WCB_LinTest implements the wild bootstrap (WB) and the wild cluster bootstrap (WCB) linearity tests.

Function for Model Estimation

EstPSTR implements the estimation of the PSTR model.

Functions for Model Evaluation

EvalTest implements the evaluation tests.
WCB_TVTest implements the wild bootstrap (WB) and the wild cluster bootstrap (WCB) evaluation test of no time-varying parameters.
WCB_HETest implements the wild bootstrap (WB) and the wild cluster bootstrap (WCB) evaluation test of no remaining nonlinearity (no remaining heterogeneity).
**sunspot**

### Other Functions

- `version` shows the version number and some information of the package.
- `print.PSTR` prints the object of the class PSTR.
- `plot_transition` plots the transition function of an estimated PSTR model.
- `plot_response` plots curve or surfaces of the expected response against the corresponding variable.
- `plot_target` plots the surface of the target function for the nonlinear least square estimation.

### Data

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

---

<table>
<thead>
<tr>
<th>sunspot</th>
<th>Transformed Wolf annual sunspot numbers for the years 1710-1979</th>
</tr>
</thead>
</table>

### 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


Source


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

*Topic **datasets**
  - Hansen99, 7
  - sunspot, 21

*Topic **estimation**
  - EstPSTR, 2

*Topic **evaluation**
  - EvalTest, 4

*Topic **initialization**
  - NewPSTR, 11

*Topic **specification**
  - LinTest, 8

*Topic **utils**
  - plot_coefficients, 12
  - plot_response, 14
  - plot_target, 15
  - plot_transition, 17
  - print.PSTR, 18
  - version, 22

EstPSTR, 2, 5, 6, 13, 15, 17–20
EvalTest, 4, 17, 19, 20

Hansen99, 7, 21

LinTest, 3, 4, 6, 8, 12, 17, 19, 20

NewPSTR, 2–4, 6, 8–10, 11, 17, 19, 20

plot_coefficients, 12
plot_response, 14, 21
plot_target, 15, 21
plot_transition, 17, 21
print.PSTR, 18, 21
PSTR, 2, 19
PSTR=package (PSTR), 19

sunspot, 21, 21

version, 21, 22

WCB_HETest, 17, 19, 20