Package 'IOLS'

July 21, 2025

Title Iterated Ordinary Least Squares Regression

Version 0.1.4

Description Addresses the 'log of zero' by developing a new family of estimators called iterated Ordinary Least Squares.
This family nests standard approaches such as log-linear and Poisson regressions, offers several computational advantages, and corresponds to the correct way to perform the popular log(Y + 1) transformation. For more details about how to use it, see the notebook at: <https://www.davidbenatia.com/>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.2

Imports stats, utils, sandwich, matlib, boot, randomcoloR, stringr

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Nassim Zbalah [cre],
David Benatia [aut]

Maintainer Nassim Zbalah <nas66.nz@gmail.com>

Repository CRAN

Date/Publication 2023-04-07 20:50:02 UTC

Contents

DATASET *Sample Data for Analysis*

Description

A simple `data_frame` obtained by a Data Generating Process that is for testing and running of examples

y outcome variable

x two bivariate normal variables x1 and x2

Usage

```
DATASET
```

Format

An object of class `data.frame` with 1000 rows and 3 columns.

iOLS *iOLS*

Description

`iOLS` regression is used to fit log-linear model/log-log model, adressing the "log of zero" problem, based on the theoretical results developed in the following paper : https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3444996.

Usage

```
iOLS(y, X, VX, tX, d, epsi = 10^-5, b_init, error_type = "HC0")
```

Arguments

`y`	the dependent variable, a vector.
`X`	the regressors matrix x with a column of ones added.
`VX`	a matrix that MUST be equal to (X'X)^-1.
`tX`	a matrix that MUST be equal to X^t (the transpose of X).
`d`	the value of the hyper-parameter delta, numeric.
`epsi`	since the estimated parameters are obtained by converging, we need a convergence criterion epsi (supposed to be small, usually around 10^-5), to make the program stop once the estimations are near their limits. A numeric.

b_init	the point from which the solution starts its converging trajectory. A vector that has the same number of elements as there are parameters estimated in the model.
error_type	a character string specifying the estimation type of the covariance matrix. Argument of the vcovHC function, then click this link for details. "HC0" is the default value, this the White's estimator.

Value

an iOLS fitted model object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
tX = t(X)
library(matlib) ; VX = inv(tX %*% X)
f = iOLS(y, X, VX, tX, 20, b_init = lm_coef)
```

---

iOLS_path *iOLS_path*

Description

iOLS regression repeated for several values of the hyper-parameter delta.

Usage

```
iOLS_path(
  y,
  X,
  deltainf = 10^-5,
  deltasup = 10^4,
  nbre_delta = 20,
  epsi = 10^-3,
  b_init,
  error_type = "HC0"
)
```

Arguments

y	the dependent variable, a vector.
X	the regressors matrix x with a column of ones added.

deltainf	numeric, the lowest hyper-parameter delta we want to apply iOLS with. The default value is 10^-5.
deltasup	numeric, the highest hyper-parameter delta we want to apply iOLS with. The default value is 10000.
nbre_delta	integer, the number of hyper-parameters delta we want between deltainf and deltasup.
epsi	since the estimated parameters are obtained by converging, we need a convergence criterion epsi (supposed to be small, usually around 10^-5), to make the program stop once the estimations are near their limits. A numeric.
b_init	the point from which the solution starts its converging trajectory. A vector that has the same number of elements as there are parameters estimated in the model.
error_type	a character string specifying the estimation type of the covariance matrix. Argument of the vcovHC function, then click this link for details. "HC0" is the default value, this the White's estimator.

Value

an `iOLS_path` fitted model object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
k = iOLS_path(y, X, b_init = lm_coef,
deltainf = 10^-5, deltasup = 10^4, nbre_delta = 20,
epsi = 10^-3, error_type = "HC0")
```

`iOLS_path_plot` *iOLS_path_plot*

Description

Function that plots an `iOLS_path` fitted model object.

Usage

```
iOLS_path_plot(m, delta_rank = NULL, plot_beta = "", ...)
```

Arguments

`m`	An `iOLS_path` fitted model object.
`delta_rank`	Among all the hyper-parameters delta, we can choose to plot the "iOLS_path" fitted model object corresponding to the chosen delta_rank. If a value is not precised, the default value is NULL and the function will only display the estimated parameter(s) in function of log(delta).
`plot_beta`	If you want to see the trajectory of one estimated parameter beta only, just precise plot_beta = k (k=0 if you want to see the intercept's trajectory for example). Otherwise, write plot_beta = "" (the default value), and you will see all parameters' trajectory. In this case, the colors of each curve is assigned randomly, but by precising which parameters' trajectory you want to see, it will be drawn in black.
`...`	other parameters.

Value

a plot of an `iOLS_path` fitted model object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
k = iOLS_path(y, X, b_init = lm_coef, deltainf = 10^-5,
deltasup = 10^4, nbre_delta = 20,
epsi = 10^-3, error_type = "HC0")

#All the parameters, as a function of log(delta) (ie. each triplet from an iOLS regression) :
iOLS_path_plot(k)

#All the parameters from the 6th iOLS regression :
iOLS_path_plot(k, delta_rank = 6)

#Intercept from the 6th iOLS regression :
iOLS_path_plot(k, delta_rank = 6, plot_beta = 0)
```

`iOLS_plot` *iOLS_plot*

Description

Function that plots an `iOLS` fitted model object.

Usage

```
iOLS_plot(m, ..., plot_beta = "")
```

Arguments

`m`	An `iOLS` fitted model object.
`...`	other parameters.
`plot_beta`	If you want to see the trajectory of one estimated parameter beta only, just precise plot_beta = k (k=0 if you want to see the intercept's trajectory for example). Otherwise, write plot_beta = "" (the default value), and you will see all parameters' trajectory. In this case, the colors of each curve is assigned randomly, but by precising which parameters' trajectory you want to see, it will be drawn in black.

Value

a plot of an `iOLS` fitted model object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
tX = t(X)
library(matlib) ; VX = inv(tX %*% X)
f = iOLS(y, X, VX, tX, 20, b_init = lm_coef)

iOLS_plot(f)

#Only one of the estimated parameters, for example k=0 (the intercept):
iOLS_plot(f, plot_beta = 0)
```

`lambda_test` *lambda_test*

Description

Printing and plotting of the lambda-test.

Usage

```
lambda_test(f, nB)
```

Arguments

`f`	An `iOLS_path` fitted model object that you want to apply this test on.
`nB`	The number of iteration that you want to be done in the bootstrap process used in the function.

Value

a lambda_test object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
k = iOLS_path(y, X, b_init = lm_coef, deltainf = 10^-5,
deltasup = 10^4, nbre_delta = 20,
epsi = 10^-3, error_type = "HC0")

L = lambda_test(k, nB = 5)
```

`print` *print.iOLS*

Description

Function that prints an `iOLS` fitted model object.

Usage

```
print(m, ...)
```

Arguments

`m`	An `iOLS` fitted model object.
`...`	other parameters.

Value

a display of an `iOLS` fitted model object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
tX = t(X)
library(matlib) ; VX = inv(tX %*% X)
f = iOLS(y, X, VX, tX, 20, b_init = lm_coef)
print(f)
```

---

print.iOLS_path *print.iOLS_path*

Description

Function that prints an `iOLS_path` fitted model object.

Usage

```
## S3 method for class 'iOLS_path'
print(m, delta_rank = NULL, ...)
```

Arguments

`m`	An `iOLS_path` fitted model object.
`delta_rank`	Among all the hyper-parameters delta, we can choose to plot the "iOLS_path" fitted model object corresponding to the chosen delta_rank. If a value is not precised, the default value is NULL and the function will only display the estimated parameter(s) in function of log(delta).
`...`	other parameters.

Value

a display of a `iOLS_path` fitted model object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
k = iOLS_path(y, X, b_init = lm_coef, deltainf = 10^-5,
deltasup = 10^4, nbre_delta = 20,
```

```
epsi = 10^-3, error_type = "HC0")

#Printing of all the iOLS regression:
print(k)

#Printing of the 6th iOLS regression :
print(k, delta_rank = 6)
```

---

print.lambda_test *print.lambda_test*

Description

Function that prints a `lambda_test` object.

Usage

```
## S3 method for class 'lambda_test'
print(m, ...)
```

Arguments

m	A `lambda_test` object.
...	other parameters.

Value

a display and a plot of a `lambda_test` object.

Examples

```
data(DATASET)
y = DATASET$y
x = as.matrix(DATASET[,c("X1","X2")])
lm = lm(log(y+1) ~ x)
lm_coef = c(coef(lm))
X = cbind(rep(1, nrow(x)), x)
k = iOLS_path(y, X, b_init = lm_coef, deltainf = 10^-5,
deltasup = 10^4, nbre_delta = 20,
epsi = 10^-3, error_type = "HC0")

L = lambda_test(k, nB = 5)

print(L)
```

Index