Package 'PLRModels'

July 21, 2025

Type Package

Title Statistical Inference in Partial Linear Regression Models

Version 1.4

Date 2023-08-19

Author German Aneiros Perez and Ana Lopez-Cheda

Maintainer Ana Lopez-Cheda <ana.lopez.cheda@udc.es>

Description

Contains statistical inference tools applied to Partial Linear Regression (PLR) models. Specifically, point estimation, confidence intervals estimation, bandwidth selection, goodness-of-fit tests and analysis of covariance are considered. Kernel-based methods, combined with ordinary least squares estimation, are used and time series errors are allowed. In addition, these techniques are also implemented for both parametric (linear) and nonparametric regression models.

License GPL-3

Imports stats

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-19 10:42:44 UTC

Contents

`PLRModels-package` *Statistical inference in partial linear regression models*

Description

This package provides statistical inference tools applied to Partial Linear Regression (PLR) models. Specifically, point estimation, confidence intervals estimation, bandwidth selection, goodness-of-fit tests and analysis of covariance are considered. Kernel-based methods, combined with ordinary least squares estimation, are used and time series errors are allowed. In addition, these techniques are also implemented for both parametric (linear) and nonparametric regression models.

Details

The most important functions are those directly related with the PLR models; that is, `plrm.gcv`, `plrm.cv`, `plrm.beta`, `plrm.est`, `plrm.gof`, `plrm.ancova` and `plrm.ci`. Although the other functions included in the package are auxiliary ones, they can be used independiently.

Author(s)

Authors: German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

Maintainer: Ana Lopez Cheda <ana.lopez.cheda@udc.es>

`barnacles1` *Sales of barnacles in Cedeira*

Description

Information about sales and prices of barnacles in two galician towns for each month from 2004 to 2013. The data have been transformed using the logarithm function.

Usage

```
data(barnacles1)
```

Format

A matrix containing 3 columns:

`barnacles1[, 1]` contains the number of sales (in kg) of barnacles in Cedeira's fish market;

`barnacles1[, 2]` contains the prices (in euro/kg) of the barnacles in Cedeira's fish market;

`barnacles1[, 3]` contains the number of sales (in kg) of barnacles in Carino's fish market.

Source

http://dm.udc.es/modes/sites/default/files/barnacles1.rar http://dm.udc.es/modes/sites/default/files/barnacles1.zip

---

barnacles2 *Sales of barnacles in Cangas*

Description

Information about sales and prices of barnacles in two galician towns for each month from 2004 to 2013. The data have been transformed using the logarithm function.

Usage

```
data(barnacles2)
```

Format

A matrix containing 3 columns:

`barnacles1[, 1]` contains the number of sales (in kg) of barnacles in Cangas' fish market;

`barnacles1[, 2]` contains the prices (in euro/kg) of the barnacles in Cangas' fish market;

`barnacles1[, 3]` contains the number of sales (in kg) of barnacles in Baiona's fish market.

Source

http://dm.udc.es/modes/sites/default/files/barnacles2.rar http://dm.udc.es/modes/sites/default/files/barnacles2.zip

`np.ancova` *Nonparametric analysis of covariance*

Description

This routine tests the equality of L nonparametric regression curves $(m_1, ..., m_L)$ from samples $(Y_{ki}, t_i) : i = 1, ..., n, k = 1, ..., L$, where:

$$Y_{ki} = m_k(t_i) + \epsilon_{ki}.$$

The unknown functions m_k are smooth, fixed equally spaced design is considered, and the random errors, ϵ_{ki}, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : m_1 = ... = m_L$, derives from a Cramer-von-Mises-type functional based on different distances between nonparametric estimators of the regression functions.

Usage

```
np.ancova(data = data, h.seq = NULL, w = NULL, estimator = "NW",
kernel = "quadratic", time.series = FALSE, Tau.eps = NULL,
h0 = NULL, lag.max = 50, p.max = 3, q.max = 3, ic = "BIC",
num.lb = 10, alpha = 0.05)
```

Arguments

`data`	`data[, k]` contains the values of the response variable, Y_k, for each model k ($k = 1, ..., L$); `data[, L+1]` contains the values of the explanatory (common) variable, t, for each model k ($k = 1, ..., L$).
`h.seq`	the statistic test is performed using each bandwidth in the vector `h.seq` (the same bandwidth is used to estimate all the regression functions). If `NULL` (the default), 10 equidistant values between 0 and the first half of the range of t_i are considered.
`w`	support interval of the weigth function in the test statistic. If `NULL` (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
`time.series`	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
`Tau.eps`	`Tau.eps[k]` contains the sum of autocovariances associated to the random errors of the regression model k ($k = 1, ..., L$). If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted nonparametric regression model and, then, it obtains the sum of the autocovariances of such ARMA model.

h0	if Tau.eps=NULL, h0 contains the pilot bandwidth used for obtaining the residuals to construct the default for Tau.eps. If NULL (the default), a quarter of the range of t_i is considered.
lag.max	if Tau.eps=NULL, lag.max contains the maximum delay used to construct the default for Tau.eps. The default is 50.
p.max	if Tau.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Tau.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Tau.eps=NULL, ic contains the information criterion used to suggest the ARMA models. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Tau.eps=NULL, it checks the suitability of the selected ARMA models according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Tau.eps=NULL, alpha contains the significance level which the ARMA models are checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

If Tau.eps=NULL and the routine is not able to suggest an approximation for Tau.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Tau.eps, the procedures suggested in Muller and Stadmuller (1988) and Herrmann *et al.* (1992) can be followed.

For more details, see Vilar-Fernandez and Gonzalez-Manteiga (2004).

Value

A list with a dataframe containing:

h.seq	sequence of bandwidths used in the test statistic.
Q.m	values of the test statistic (one for each bandwidth in h.seq).
Q.m.normalised	normalised value of Q.m.
p.value	p-values of the corresponding statistic tests (one for each bandwidth in h.seq).

Moreover, if data is a time series and Tau.eps is not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Dette, H. and Neumeyer, N. (2001) Nonparametric analysis of covariance. *Ann. Statist.* **29**, no. 5, 1361-1400.

Herrmann, E., Gasser, T. and Kneip, A. (1992) Choice of bandwidth for kernel regression when residuals are correlated. *Biometrika* **79**, 783-795

Muller, H.G. and Stadmuller, U. (1988) Detecting dependencies in smooth regression models. *Biometrika* **75**, 639-650

Vilar-Fernandez, J.M. and Gonzalez-Manteiga, W. (2004) Nonparametric comparison of curves with dependent errors. *Statistics* **38**, 81-99.

See Also

Other related functions are `np.est`, `par.ancova` and `plrm.ancova`.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

data2 <- matrix(10,120,2)
data(barnacles2)
barnacles2 <- as.matrix(barnacles2)
data2[,1] <- barnacles2[,1]
data2 <- diff(data2, 12)
data2[,2] <- 1:nrow(data2)

data3 <- matrix(0, nrow(data),ncol(data)+1)
data3[,1] <- data[,1]
data3[,2:3] <- data2

np.ancova(data=data3)



# EXAMPLE 2: SIMULATED DATA
## Example 2.1: dependent data: true null hypothesis

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m1 <- function(t) {0.25*t*(1-t)}
f <- m1(t)

epsilon1 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
```

```
y1 <-  f + epsilon1

epsilon2 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y2 <- f + epsilon2

data_eq <- cbind(y1, y2, t)

# We apply the test
np.ancova(data_eq, time.series=TRUE)


## Example 2.2: dependent data: false null hypothesis
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m3 <- function(t) {0.25*t*(1-t)}
m4 <- function(t) {0.25*t*(1-t)*0.75}

f3 <- m3(t)
epsilon3 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y3 <-  f3 + epsilon3

f4 <- m4(t)
epsilon4 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y4 <-  f4 + epsilon4

data_neq<- cbind(y3, y4, t)

# We apply the test
np.ancova(data_neq, time.series=TRUE)
```

`np.cv` *Cross-validation bandwidth selection in nonparametric regression models*

Description

From a sample $(Y_i, t_i) : i = 1, ..., n$, this routine computes, for each l_n considered, an optimal bandwidth for estimating m in the regression model

$$Y_i = m(t_i) + \epsilon_i.$$

The regression function, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series. The optimal bandwidth is selected by means of the leave-$(2l_n + 1)$-out cross-validation procedure. Kernel smoothing is used.

Usage

```
np.cv(data = data, h.seq = NULL, num.h = 50, w = NULL, num.ln = 1,
ln.0 = 0, step.ln = 2, estimator = "NW", kernel = "quadratic")
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2] contains the values of the explanatory variable, t.
h.seq	sequence of considered bandwidths in the CV function. If NULL (the default), num.h equidistant values between zero and a quarter of the range of t_i are considered.
num.h	number of values used to build the sequence of considered bandwidths. If h.seq is not NULL, num.h=length(h.seq). Otherwise, the default is 50.
w	support interval of the weigth function in the CV function. If NULL (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
num.ln	number of values for l_n: $2l_n+1$ observations around each point t_i are eliminated to estimate $m(t_i)$ in the CV function. The default is 1.
ln.0	minimum value for l_n. The default is 0.
step.ln	distance between two consecutives values of l_n. The default is 2.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the CV function to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

For more details, see Chu and Marron (1991).

Value

h.opt	dataframe containing, for each ln considered, the selected value for the bandwidth.
CV.opt	CV.opt[k] is the minimum value of the CV function when de k-th value of ln is considered.
CV	matrix containing the values of the CV function for each bandwidth and ln considered.
w	support interval of the weigth function in the CV function.
h.seq	sequence of considered bandwidths in the CV function.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Chu, C-K and Marron, J.S. (1991) Comparison of two bandwidth selectors with dependent errors. *The Annals of Statistics* **19**, 1906-1918.

See Also

Other related functions are: np.est, np.gcv, plrm.est, plrm.gcv and plrm.cv.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

aux <- np.cv(data, ln.0=1,step.ln=1, num.ln=2)
aux$h.opt
plot.ts(aux$CV)

par(mfrow=c(2,1))
plot(aux$h.seq,aux$CV[,1], xlab="h", ylab="CV", type="l", main="ln=1")
plot(aux$h.seq,aux$CV[,2], xlab="h", ylab="CV", type="l", main="ln=2")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

epsilon <- rnorm(n, 0, 0.01)
y <-  f + epsilon
data_ind <- matrix(c(y,t),nrow=100)

# We apply the function
a <-np.cv(data_ind)
a$CV.opt

CV <- a$CV
h <- a$h.seq
plot(h,CV,type="l")
```

---

np.est *Nonparametric estimate of the regression function*

Description

This routine computes estimates for $m(newt_j)$ $(j = 1, ..., J)$ from a sample $(Y_i, t_i) : i = 1, ..., n$, where:

$$Y_i = m(t_i) + \epsilon_i.$$

The regression function, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series. Kernel smoothing is used.

Usage

```
np.est(data = data, h.seq = NULL, newt = NULL,
estimator = "NW", kernel = "quadratic")
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2]` contains the values of the explanatory variable, t.
`h.seq`	the considered bandwidths. If `NULL` (the default), only one bandwidth, selected by means of the cross-validation procedure, is used.
`newt`	values of the explanatory variable where the estimates are obtained. If NULL (the default), the considered values will be the values of `data[,2]`.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

See Fan and Gijbels (1996) and Francisco-Fernandez and Vilar-Fernandez (2001).

Value

YHAT: a length(`newt`) x length(`h.seq`) matrix containing the estimates for $m(newt_j)$

$(j = 1, ...,$length(`newt`)) using the different bandwidths in `h.seq`.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Fan, J. and Gijbels, I. (1996) *Local Polynomial Modelling and its Applications*. Chapman and Hall, London.

Francisco-Fernandez, M. and Vilar-Fernandez, J. M. (2001) Local polynomial regression estimation with correlated errors. *Comm. Statist. Theory Methods* **30**, 1271-1293.

See Also

Other related functions are: np.gcv, np.cv, plrm.est, plrm.gcv and plrm.cv.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

aux <- np.gcv(data)
h <- aux$h.opt
ajuste <- np.est(data=data, h=h)
plot(data[,2], ajuste, type="l", xlab="t", ylab="m(t)")
plot(data[,1], ajuste, xlab="y", ylab="y.hat", main="y.hat vs y")
abline(0,1)
residuos <- data[,1] - ajuste
mean(residuos^2)/var(data[,1])



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

epsilon <- rnorm(n, 0, 0.01)
y <-  f + epsilon
data_ind <- matrix(c(y,t),nrow=100)

# We estimate the nonparametric component of the PLR model
# (CV bandwidth)
est <- np.est(data_ind)
plot(t, est, type="l", lty=2, ylab="")
points(t, 0.25*t*(1-t), type="l")
legend(x="topleft", legend = c("m", "m hat"), col=c("black", "black"), lty=c(1,2))
```

---

np.gcv *Generalized cross-validation bandwidth selection in nonparametric regression models*

Description

From a sample $(Y_i, t_i) : i = 1, ..., n$, this routine computes an optimal bandwidth for estimating m in the regression model

$$Y_i = m(t_i) + \epsilon_i.$$

The regression function, m, is a smooth but unknown function. The optimal bandwidth is selected by means of the generalized cross-validation procedure. Kernel smoothing is used.

Usage

```
np.gcv(data = data, h.seq=NULL, num.h = 50, estimator = "NW",
kernel = "quadratic")
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2]` contains the values of the explanatory variable, t.
`h.seq`	sequence of considered bandwidths in the GCV function. If `NULL` (the default), `num.h` equidistant values between zero and a quarter of the range of t_i are considered.
`num.h`	number of values used to build the sequence of considered bandwidths. If `h.seq` is not `NULL`, `num.h=length(h.seq)`. Otherwise, the default is 50.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

See Craven and Wahba (1979) and Rice (1984).

Value

`h.opt`	selected value for the bandwidth.
`GCV.opt`	minimum value of the GCV function.
`GCV`	vector containing the values of the GCV function for each considered bandwidth.
`h.seq`	sequence of considered bandwidths in the GCV function.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Craven, P. and Wahba, G. (1979) Smoothing noisy data with spline functions. *Numer. Math.* **31**, 377-403.

Rice, J. (1984) Bandwidth choice for nonparametric regression. *Ann. Statist.* **12**, 1215-1230.

See Also

Other related functions are: `np.est`, `np.cv`, `plrm.est`, `plrm.gcv` and `plrm.cv`.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

aux <- np.gcv(data)
aux$h.opt
plot(aux$h.seq, aux$GCV, xlab="h", ylab="GCV", type="l")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

epsilon <- rnorm(n, 0, 0.01)
y <-  f + epsilon
data_ind <- matrix(c(y,t),nrow=100)

# We apply the function
a <-np.gcv(data_ind)
a$GCV.opt

GCV <- a$GCV
h <- a$h.seq
plot(h, GCV, type="l")
```

`np.gof` *Goodness-of-Fit tests in nonparametric regression models*

Description

This routine tests the equality of a nonparametric regression curve, m, and a given function, m_0, from a sample $(Y_i, t_i) : i = 1, ..., n$, where:

$$Y_i = m(t_i) + \epsilon_i.$$

The unknown function m is smooth, fixed equally spaced design is considered, and the random errors, ϵ_i, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : m = m_0$, derives from a Cramer-von-Mises-type functional distance between a nonparametric estimator of m and m_0.

Usage

```
np.gof(data = data, m0 = NULL, h.seq = NULL, w = NULL,
estimator = "NW", kernel = "quadratic", time.series = FALSE,
Tau.eps = NULL, h0 = NULL, lag.max = 50, p.max = 3,
q.max = 3, ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2]` contains the values of the explanatory variable, t.
`m0`	the considered function in the null hypothesis. If `NULL` (the default), the zero function is considered.
`h.seq`	the statistic test is performed using each bandwidth in the vector `h.seq`. If `NULL` (the default), 10 equidistant values between zero and a quarter of the range of t_i are considered.
`w`	support interval of the weigth function in the test statistic. If `NULL` (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
`time.series`	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
`Tau.eps`	it contains the sum of autocovariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted nonparametric regression model and, then, it obtains the sum of the autocovariances of such ARMA model.

h0	if Tau.eps=NULL, h0 contains the pilot bandwidth used for obtaining the residuals to construct the default for Tau.eps. If NULL (the default), a quarter of the range of t_i is considered.
lag.max	if Tau.eps=NULL, lag.max contains the maximum delay used to construct the default for Tau.eps. The default is 50.
p.max	if Tau.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Tau.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Tau.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Tau.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Tau.eps=NULL, alpha contains the significance level which the ARMA model is checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

If Tau.eps=NULL and the routine is not able to suggest an approximation for Tau.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Tau.eps, the procedures suggested in Muller and Stadmuller (1988) and Herrmann *et al.* (1992) can be followed.

The implemented statistic test particularizes that one in Gonzalez Manteiga and Vilar Fernandez (1995) to the case where the considered class in the null hypothesis has only one element.

Value

A list with a dataframe containing:

h.seq	sequence of bandwidths used in the test statistic.
Q.m	values of the test statistic (one for each bandwidth in h.seq).
Q.m.normalised	normalised value of Q.m.
p.value	p-values of the corresponding statistic tests (one for each bandwidth in h.seq).

Moreover, if data is a time series and Tau.eps is not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Biedermann, S. and Dette, H. (2000) Testing linearity of regression models with dependent errors by kernel based methods. *Test* **9**, 417-438.

Gonzalez-Manteiga, W. and Aneiros-Perez, G. (2003) Testing in partial linear regression models with dependent errors. *J. Nonparametr. Statist.* **15**, 93-111.

Gonzalez-Manteiga, W. and Cao, R. (1993) Testing the hypothesis of a general linear model using nonparametric regression estimation. *Test* **2**, 161-188.

Gonzalez Manteiga, W. and Vilar Fernandez, J. M. (1995) Testing linear regression models using non-parametric regression estimators when errors are non-independent. *Comput. Statist. Data Anal.* **20**, 521-541.

Herrmann, E., Gasser, T. and Kneip, A. (1992) Choice of bandwidth for kernel regression when residuals are correlated. *Biometrika* **79**, 783-795

Muller, H.G. and Stadmuller, U. (1988) Detecting dependencies in smooth regression models. *Biometrika* **75**, 639-650

See Also

Other related functions are `np.est`, `par.gof` and `plrm.gof`.

Examples

```
# EXAMPLE 1: REAL DATA
data <- matrix(10,120,2)
data(barnacles1)
barnacles1 <- as.matrix(barnacles1)
data[,1] <- barnacles1[,1]
data <- diff(data, 12)
data[,2] <- 1:nrow(data)

np.gof(data)



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
f.function <- function(u) {0.25*u*(1-u)}

epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  f + epsilon
data <- cbind(y,t)

## Example 2a.1: true null hypothesis
```

```
np.gof(data, m0=f.function, time.series=TRUE)

## Example 2a.2: false null hypothesis
np.gof(data, time.series=TRUE)
```

---

par.ancova *Parametric analysis of covariance (based on linear models)*

Description

This routine tests the equality of L vector coefficients, $(\beta_1, ..., \beta_L)$, from samples $(Y_{ki}, X_{ki1}, ..., X_{kip})$: $i = 1, ..., n$, $k = 1, ..., L$, where:

$$\beta_k = (\beta_{k1}, ..., \beta_{kp})$$

is an unknown vector parameter and

$$Y_{ki} = X_{ki1} * \beta_{k1} + ... + X_{kip} * \beta_{kp} + \epsilon_{ki}.$$

The random errors, ϵ_{ki}, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : \beta_1 = ... = \beta_L$, derives from the asymptotic normality of the ordinary least squares estimator of β_k $(k = 1, ..., L)$, this result giving a χ^2-test.

Usage

```
par.ancova(data = data, time.series = FALSE, Var.Cov.eps = NULL,
p.max = 3, q.max = 3, ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

`data`	`data[, 1, k]` contains the values of the response variable, Y_k, for each model k $(k = 1, ..., L)$; `data[, 2:(p+1), k]` contains the values of the explanatory variables, $X_{k1}, ..., X_{kp}$, for each model k $(k = 1, ..., L)$.
`time.series`	it denotes whether the data is independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
`Var.Cov.eps`	`Var.Cov.eps[, , k]` contains the `n` x `n` matrix of variances-covariances associated to the random errors of the regression model k $(k = 1, ..., L)$. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted linear regression model and, then, it obtains the var-cov matrix of such ARMA model.
`p.max`	if `Var.Cov.eps=NULL`, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
`q.max`	if `Var.Cov.eps=NULL`, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.

ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA models. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
num.lb	if Var.Cov.eps=NULL, it checks the suitability of the ARMA models according to the Ljung-Box and the t.test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Var.Cov.eps=NULL, alpha contains the significance level (default is 0.05) which the ARMA models are checked.

Details

If Var.Cov.eps=NULL and the routine is not able to suggest an approximation for Var.Cov.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Var.Cov.eps, the procedure suggested in Domowitz (1982) can be followed.

The implemented procedure particularizes the parametric test in the routine plrm.ancova to the case where is known that the nonparametric components in the corresponding PLR models are null.

Value

A list with a dataframe containing:

Q.beta	value of the test statistic.
p.value	p-value of the corresponding statistic test.

Moreover, if data is a time series and Var.Cov.eps is not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Domowitz, J. (1982) The linear model with stochastic regressors and heteroscedastic dependent errors. Discussion paper No 543, Center for Mathematical studies in Economic and Management Science, Northwestern University, Evanston, Illinois.

Judge, G.G., Griffiths, W.E., Carter Hill, R., Lutkepohl, H. and Lee, T-C. (1980) *The Theory and Practice of Econometrics*. Wiley.

Seber, G.A.F. (1977) *Linear Regression Analysis*. Wiley.

See Also

Other related functions are np.ancova and plrm.ancova.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

data(barnacles2)
data2 <- as.matrix(barnacles2)
data2 <- diff(data2, 12)
data2 <- cbind(data2[,1],1,data2[,-1])

data3 <- array(0, c(nrow(data),ncol(data),2))
data3[,,1] <- data
data3[,,2] <- data2

par.ancova(data=data3)



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data - true null hypothesis

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)

x1 <- matrix(rnorm(200,0,1), nrow=n)
sum1 <- x1%*%beta
epsilon1 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y1 <-  sum1 + epsilon1
data1 <- cbind(y1,x1)

x2 <- matrix(rnorm(200,1,2), nrow=n)
sum2 <- x2%*%beta
epsilon2 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y2 <- sum2 + epsilon2
data2 <- cbind(y2,x2)

data_eq <- array(cbind(data1,data2),c(100,3,2))

# We apply the test
par.ancova(data_eq, time.series=TRUE)


## Example 2a: dependent data - false null hypothesis
# We generate the data
n <- 100
beta3 <- c(0.05, 0.01)
beta4 <- c(0.05, 0.02)
```

```
x3 <- matrix(rnorm(200,0,1), nrow=n)
sum3 <- x3%*%beta3
epsilon3 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y3 <-  sum3 + epsilon3
data3 <- cbind(y3,x3)

x4 <- matrix(rnorm(200,1,2), nrow=n)
sum4 <- x4%*%beta4
epsilon4 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y4 <-  sum4 + epsilon4
data4 <- cbind(y4,x4)

data_neq <- array(cbind(data3,data4),c(100,3,2))

# We apply the test
par.ancova(data_neq, time.series=TRUE)
```

`par.ci` *Confidence intervals estimation in linear regression models*

Description

This routine obtains a confidence interval for the value $a^T * \beta$, by asymptotic distribution and bootstrap, from a sample $(Y_i, X_{i1}, ..., X_{ip}) : i = 1, ..., n$, where:

$$a = (a_1, ..., a_p)^T$$

is an unknown vector,

$$\beta = (\beta_1, ..., \beta_p)^T$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series.

Usage

```
par.ci(data=data, seed=123, CI="AD", B=1000, N=50, a=NULL,
p.arima=NULL, q.arima=NULL, p.max=3, q.max=3, alpha=0.05,
alpha2=0.05, num.lb=10, ic="BIC", Var.Cov.eps=NULL)
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the explanatory variables, $X_1, ..., X_p$.
`seed`	the considered seed.

CI	method to obtain the confidence interval. It allows us to choose between: "AD" (asymptotic distribution), "B" (bootstrap) or "all" (both). The default is "AD".
B	number of bootstrap replications. The default is 1000.
N	Truncation parameter used in the finite approximation of the MA(infinite) expression of ϵ.
a	Vector which, multiplied by beta, is used for obtaining the confidence interval of this result.
p.arima	the considered p to fit the model ARMA(p,q).
q.arima	the considered q to fit the model ARMA(p,q).
p.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
alpha	1 - alpha is the confidence level of the confidence interval. The default is 0.05.
alpha2	significance level used to check (if needed) the ARMA model fitted to the residuals. The default is 0.05.
num.lb	if Var.Cov.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).
Var.Cov.eps	n x n matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.

Value

A list containing:

Bootstrap	a dataframe containing ci_inf and ci_sup, the confidence intervals using bootstrap and p_opt and q_opt (the orders for the ARMA model fitted to the residuals).
AD	a dataframe containing ci_inf and ci_sup, the confidence intervals using the asymptotic distribution and p_opt and q_opt (the orders for the ARMA model fitted to the residuals).
pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Liang, H., Hardle, W., Sommerfeld, V. (2000) Bootstrap approximation in a partially linear regression model. *Journal of Statistical Planning and Inference* **91**, 413-426.

You, J., Zhou, X. (2005) Bootstrap of a semiparametric partially linear model with autoregressive errors. *Statistica Sinica* **15**, 117-133.

See Also

A related function is `plrm.ci`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

## Not run: par.ci(data, a=c(1,0,0), CI="all")
## Not run: par.ci(data, a=c(0,1,0), CI="all")
## Not run: par.ci(data, a=c(0,0,1), CI="all")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(123)
# We generate the data
n <- 100
beta <- c(0.5, 2)

x <- matrix(rnorm(200,0,3), nrow=n)
sum <- x%*%beta
sum <- as.matrix(sum)
eps <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.1, n = n)
eps <- as.matrix(eps)
y <-  sum + eps
data_parci <- cbind(y,x)

# We estimate the confidence interval of a^T * beta in the PLR model
## Not run: par.ci(data, a=c(1,0), CI="all")
## Not run: par.ci(data, a=c(0,1), CI="all")
```

`par.est` *Estimation in linear regression models*

Description

This routine computes the ordinary least squares estimate for β from a sample $(Y_i, X_{i1}, ..., X_{ip})$, $i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series.

Usage

```
par.est(data = data)
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the explanatory variables, $X_1, ..., X_p$.

Details

See Seber (1977) and Judge *et al.* (1980).

Value

A vector containing the corresponding estimate.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Judge, G.G., Griffiths, W.E., Carter Hill, R., Lutkepohl, H. and Lee, T-C. (1980) *The Theory and Practice of Econometrics*. Wiley.

Seber, G.A.F. (1977) *Linear Regression Analysis*. Wiley.

See Also

Other related functions are `plrm.beta` and `plrm.est`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

beta <- par.est(data=data)
beta
residuos <- data[,1] - data[,-1]%*%beta
mean(residuos^2)/var(data[,1])

fitted.values <- data[,-1]%*%beta
plot(data[,1], fitted.values, xlab="y", ylab="y.hat", main="y.hat vs y")
abline(0,1)



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
beta <- c(0.05, 0.01)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + epsilon
data_ind <- matrix(c(y,x),nrow=100)

# We estimate the parametric component of the PLR model
par.est(data_ind)


## Example 2b: dependent data

set.seed(1234)
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + epsilon
data_dep <- matrix(c(y,x),nrow=100)

# We estimate the parametric component of the PLR model
par.est(data_dep)
```

par.gof *Goodness-of-Fit tests in linear regression models*

Description

This routine tests the equality of the vector of coefficients, β, in a linear regression model and a given parameter vector, β_0, from a sample $(Y_i, X_{i1}, ..., X_{ip}) : i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series. The test statistic used for testing the null hypothesis, $H0 : \beta = \beta_0$, derives from the asymptotic normality of the ordinary least squares estimator of β, this result giving a χ^2-test.

Usage

```
par.gof(data = data, beta0 = NULL, time.series = FALSE,
Var.Cov.eps = NULL, p.max = 3, q.max = 3, ic = "BIC",
num.lb = 10, alpha = 0.05)
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the explanatory variables, $X_1, ..., X_p$.
beta0	the considered parameter vector in the null hypothesis. If NULL (the default), the zero vector is considered.
time.series	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
Var.Cov.eps	n x n matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted linear regression model and, then, it obtains the var-cov matrix of such ARMA model.
p.max	if Var.Cov.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).

num.lb	if Var.Cov.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
alpha	if Var.Cov.eps=NULL, alpha contains the significance level which the ARMA model is checked. The default is 0.05.

Details

If Var.Cov.eps=NULL and the routine is not able to suggest an approximation for Var.Cov.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Var.Cov.eps, the procedure suggested in Domowitz (1982) can be followed.

The implemented procedure particularizes the parametric test in the routine plrm.gof to the case where is known that the nonparametric component in the corresponding PLR model is null.

Value

A list with a dataframe containing:

Q.beta	value of the test statistic.
p.value	p-value of the corresponding statistic test.

Moreover, if data is a time series and Var.Cov.eps is not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Domowitz, J. (1982) The linear model with stochastic regressors and heteroscedastic dependent errors. Discussion paper No 543, Center for Mathematical studies in Economic and Management Science, Northwestern University, Evanston, Illinois.

Judge, G.G., Griffiths, W.E., Carter Hill, R., Lutkepohl, H. and Lee, T-C. (1980) *The Theory and Practice of Econometrics*. Wiley.

Seber, G.A.F. (1977) *Linear Regression Analysis*. Wiley.

See Also

Other related functions are np.gof and plrm.gof.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data[,1],1,data[,-1])

## Example 1.1: false null hypothesis
par.gof(data)
## Example 1.2: true null hypothesis
par.gof(data, beta0=c(0,0.15,0.4))



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(1234)
# We generate the data
n <- 100
beta <- c(0.05, 0.01)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + epsilon
data <- cbind(y,x)

## Example 2a.1: true null hypothesis
par.gof(data, beta0=c(0.05, 0.01))

## Example 2a.2: false null hypothesis
par.gof(data)
```

---

plrm.ancova *Semiparametric analysis of covariance (based on PLR models)*

Description

From samples $(Y_{ki}, X_{ki1}, ..., X_{kip}, t_i) : i = 1, ..., n,\ k = 1, ..., L$, this routine tests the null hypotheses $H0 : \beta_1 = ... = \beta_L$ and $H0 : m_1 = ... = m_L$, where:

$$\beta_k = (\beta_{k1}, ..., \beta_{kp})$$

is an unknown vector parameter;

$$m_k(.)$$

is a smooth but unknown function and

$$Y_{ki} = X_{ki1} * \beta_{k1} + ... + X_{kip} * \beta_{kp} + m(t_i) + \epsilon_{ki}.$$

Fixed equally spaced design is considered for the "nonparametric" explanatory variable, t, and the random errors, ϵ_{ki}, are allowed to be time series. The test statistic used for testing $H0 : \beta_1 = ... = \beta_L$ derives from the asymptotic normality of an estimator of β_k ($k = 1, ..., L$) based on both ordinary least squares and kernel smoothing (this result giving a χ^2-test). The test statistic used for testing $H0 : m_1 = ... = m_L$ derives from a Cramer-von-Mises-type functional based on different distances between nonparametric estimators of m_k ($k = 1, ..., L$).

Usage

```
plrm.ancova(data = data, t = t, b.seq = NULL, h.seq = NULL,
w = NULL, estimator = "NW", kernel = "quadratic",
time.series = FALSE, Var.Cov.eps = NULL, Tau.eps = NULL,
b0 = NULL, h0 = NULL, lag.max = 50, p.max = 3, q.max = 3,
ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

`data`	`data[, 1, k]` contains the values of the response variable, Y_k, for each model k ($k = 1, ..., L$); `data[, 2:(p+1), k]` contains the values of the "linear" explanatory variables, $X_{k1}, ..., X_{kp}$, for each model k ($k = 1, ..., L$).
`t`	contains the values of the "nonparametric" explanatory (common) variable, t, for each model k ($k = 1, ..., L$).
`b.seq`	the statistic test for $H0 : \beta_1 = ... = \beta_L$ is performed using each bandwidth in the vector `b.seq`. If `NULL` (the default) but `h.seq` is not `NULL`, it takes `b.seq=h.seq`. If both `b.seq` and `h.seq` are `NULL`, 10 equidistant values between zero and a quarter of the range of t_i are considered.
`h.seq`	the statistic test for $H0 : m_1 = ... = m_L$ is performed using each pair of bandwidths (`b.seq[j], h.seq[j]`). If `NULL` (the default) but `b.seq` is not `NULL`, it takes `h.seq=b.seq`. If both `b.seq` and `h.seq` are `NULL`, 10 equidistant values between zero and a quarter of the range of t_i are considered for both `b.seq` and `h.seq`.
`w`	support interval of the weigth function in the test statistic for $H0 : m_1 = ... = m_L$. If `NULL` (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
`time.series`	it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.
`Var.Cov.eps`	`Var.Cov.eps[, , k]` contains the `n` x `n` matrix of variances-covariances associated to the random errors of the regression model k ($k = 1, ..., L$). If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.

`Tau.eps` `Tau.eps[k]` contains the sum of autocovariances associated to the random errors of the regression model k ($k = 1, ..., L$). If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the sum of the autocovariances of such ARMA model.

`b0` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `b0` contains the pilot bandwidth for the estimator of β_k ($k = 1, ..., L$) used for obtaining the residuals to construct the default for `Var.Cov.eps` and/or `Tau.eps`. If `NULL` (the default) but `h0` is not `NULL`, it takes `b0=h0`. If both `b0` and `h0` are `NULL`, a quarter of the range of t_i is considered.

`h0` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, (`b0, h0`) contains the pair of pilot bandwidths for the estimator of m_k ($k = 1, ..., L$) used for obtaining the residuals to construct the default for `Var.Cov.eps` and/or `Tau.eps`. If `NULL` (the default) but `b0` is not `NULL`, it takes `h0=b0`. If both `b0` and `h0` are `NULL`, a quarter of the range of t_i is considered for both `b0` and `h0`.

`lag.max` if `Tau.eps=NULL`, `lag.max` contains the maximum delay used to construct the default for `Tau.eps`. The default is 50.

`p.max` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=`p.max` and 0<=q<=`q.max`. The default is 3.

`q.max` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=`p.max` and 0<=q<=`q.max`. The default is 3.

`ic` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `ic` contains the information criterion used to suggest the ARMA models. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).

`num.lb` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, it checks the suitability of the selected ARMA models according to the Ljung-Box test and the t-test. It uses up to `num.lb` delays in the Ljung-Box test. The default is 10.

`alpha` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `alpha` contains the significance level which the ARMA models are checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic for testing $H0 : m_1 = ... = m_L$ to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m_k(t_i)$.

If `Var.Cov.eps=NULL` and the routine is not able to suggest an approximation for `Var.Cov.eps`, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct `Var.Cov.eps`, the procedure suggested in Aneiros-Perez and Vieu (2013) can be followed.

If `Tau.eps=NULL` and the routine is not able to suggest an approximation for `Tau.eps`, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct `Tau.eps`, the procedures suggested in Aneiros-Perez (2008) can be followed.

Expressions for the implemented statistic tests can be seen in (15) and (16) in Aneiros-Perez (2008).

Value

A list with two dataframes:

`parametric.test`
: a dataframe containing the bandwidths, the statistics and the p-values when one tests $H0 : \beta_1 = ... = \beta_L$.

`nonparametric.test`
: a dataframe containing the bandwidths b and h, the statistics, the normalised statistics and the p-values when one tests $H0 : m_1 = ... = m_L$.

Moreover, if `data` is a time series and `Tau.eps` or `Var.Cov.eps` are not especified:

`pv.Box.test`	p-values of the Ljung-Box test for the model fitted to the residuals.
`pv.t.test`	p-values of the t.test for the model fitted to the residuals.
`ar.ma`	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G. (2008) Semi-parametric analysis of covariance under dependence conditions within each group. *Aust. N. Z. J. Stat.* **50**, 97-123.

Aneiros-Perez, G. and Vieu, P. (2013) Testing linearity in semi-parametric functional data analysis. *Comput. Stat.* **28**, 413-434.

See Also

Other related functions are `plrm.est`, `par.ancova` and `np.ancova`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

data(barnacles2)
data2 <- as.matrix(barnacles2)
data2 <- diff(data2, 12)
data2 <- cbind(data2,1:nrow(data2))

data3 <- array(0, c(nrow(data),ncol(data)-1,2))
data3[,,1] <- data[,-4]
data3[,,2] <- data2[,-4]
t <- data[,4]
```

```
plrm.ancova(data=data3, t=t)



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data - true null hypotheses

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)

m1 <- function(t) {0.25*t*(1-t)}
f <- m1(t)
x1 <- matrix(rnorm(200,0,1), nrow=n)
sum1 <- x1%*%beta
epsilon1 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y1 <-  sum1 + f + epsilon1
data1 <- cbind(y1,x1)

x2 <- matrix(rnorm(200,1,2), nrow=n)
sum2 <- x2%*%beta
epsilon2 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y2 <- sum2 + f + epsilon2
data2 <- cbind(y2,x2)

data_eq <- array(c(data1,data2), c(n,3,2))

# We apply the tests
plrm.ancova(data=data_eq, t=t, time.series=TRUE)


## Example 2b: dependent data - false null hypotheses

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m3 <- function(t) {0.25*t*(1-t)}
m4 <- function(t) {0.25*t*(1-t)*0.75}
beta3 <- c(0.05, 0.01)
beta4 <- c(0.05, 0.02)

x3 <- matrix(rnorm(200,0,1), nrow=n)
sum3 <- x3%*%beta3
f3 <- m3(t)
epsilon3 <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y3 <-  sum3 + f3 + epsilon3
data3 <- cbind(y3,x3)

x4 <- matrix(rnorm(200,1,2), nrow=n)
sum4 <- x4%*%beta4
```

```
f4 <- m4(t)
epsilon4 <- arima.sim(list(order = c(0,0,1), ma=0.5), sd = 0.02, n = n)
y4 <-  sum4 + f4 + epsilon4
data4 <- cbind(y4,x4)

data_neq <- array(c(data3,data4), c(n,3,2))

# We apply the tests
plrm.ancova(data=data_neq, t=t, time.series=TRUE)
```

`plrm.beta` *Semiparametric estimate for the parametric component of the regression function in PLR models*

Description

This routine computes estimates for β from a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

The nonparametric component, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series. Ordinary least squares estimation, combined with kernel smoothing, is used.

Usage

```
plrm.beta(data = data, b.seq = NULL, estimator = "NW", kernel = "quadratic")
```

Arguments

`data`	`data[, 1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; `data[, p+2]` contains the values of the "nonparametric" explanatory variable, t.
`b.seq`	vector of bandwidths for estimating β. If `NULL` (the default), only one estimate of β is computed, the corresponding bandwidth being selected by means of the cross-validation procedure.
`estimator`	allows us the choice between “NW” (Nadaraya-Watson) or “LLP” (Local Linear Polynomial). The default is “NW”.
`kernel`	allows us the choice between “gaussian”, “quadratic” (Epanechnikov kernel), “triweight” or “uniform” kernel. The default is “quadratic”.

Details

The expression for the estimator of β can be seen in page 52 in Aneiros-Perez *et al.* (2004).

Value

A list containing:

`BETA`	`p x length(b.seq)` matrix containing the estimate of β for each bandwidth in `h.seq`.
`G`	`n x p x length(b.seq)` array containing the nonparametric estimate of `E(X_{ij} | t_i)` ($i = 1, ..., n; j = 1, ..., p$) for each bandwidth in `b.seq`.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression model under long memory dependence. *Bernoulli* **10**, 49-78.

Hardle, W., Liang, H. and Gao, J. (2000) *Partially Linear Models*. Physica-Verlag.

Speckman, P. (1988) Kernel smoothing in partial linear models. *J. R. Statist. Soc. B* **50**, 413-436.

See Also

Other related functions are: `plrm.est`, `plrm.gcv`, `plrm.cv`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

b.h <- plrm.gcv(data)$bh.opt
ajuste <- plrm.beta(data=data, b=b.h[1])
ajuste$BETA



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
```

```
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)

# We estimate the parametric component of the PLR model
# (GCV bandwidth)
a <- plrm.beta(data_ind)

a$BETA


## Example 2b: dependent data

set.seed(1234)
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data_dep <- matrix(c(y,x,t),nrow=100)


# We estimate the parametric component of the PLR model
# (CV bandwidth)
b <- plrm.cv(data_dep, ln.0=2)$bh.opt[2,1]
a <-plrm.beta(data_dep, b=b)

a$BETA
```

---

plrm.ci *Confidence intervals estimation in partial linear regression models*

Description

This routine obtains a confidence interval for the value $a^T * \beta$, by asymptotic distribution and bootstrap, $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, where:

$$a = (a_1, ..., a_p)^T$$

is an unknown vector,

$$\beta = (\beta_1, ..., \beta_p)^T$$

is an unknown vector parameter and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

The nonparametric component, m, is a smooth but unknown function, and the random errors, ϵ_i, are allowed to be time series.

Usage

```
plrm.ci(data=data, seed=123, CI="AD", B=1000, N=50, a=NULL,
        b1=NULL, b2=NULL, estimator="NW",
        kernel="quadratic", p.arima=NULL, q.arima=NULL,
        p.max=3, q.max=3, alpha=0.05, alpha2=0.05, num.lb=10,
        ic="BIC", Var.Cov.eps=NULL)
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; data[, p+2] contains the values of the "nonparametric" explanatory variable, t.
seed	the considered seed.
CI	method to obtain the confidence interval. It allows us to choose between: "AD" (asymptotic distribution), "B" (bootstrap) or "all" (both). The default is "AD".
B	number of bootstrap replications. The default is 1000.
N	Truncation parameter used in the finite approximation of the MA(infinite) expression of ϵ.
a	Vector which, multiplied by beta, is used for obtaining the confidence interval of this result.
b1	the considered bandwidth to estimate the confidence interval by asymptotic distribution. If NULL (the default), it is obtained using cross-validation.
b2	the considered bandwidth to estimate the confidence interval by bootstrap. If NULL (the default), it is obtained using cross-validation.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".
p.arima	the considered p to fit the model ARMA(p,q).
q.arima	the considered q to fit the model ARMA(p,q).
p.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
q.max	if Var.Cov.eps=NULL, the ARMA models are selected between the models ARMA(p,q) with 0<=p<=p.max and 0<=q<=q.max. The default is 3.
alpha	1 - alpha is the confidence level of the confidence interval. The default is 0.05.
alpha2	significance level used to check (if needed) the ARMA model fitted to the residuals. The default is 0.05.
num.lb	if Var.Cov.eps=NULL, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to num.lb delays in the Ljung-Box test. The default is 10.
ic	if Var.Cov.eps=NULL, ic contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).

`Var.Cov.eps` n x n matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.

Value

A list containing:

`Bootstrap` a dataframe containing `ci_inf` and `ci_sup`, the confidence intervals using bootstrap; `p_opt` and `q_opt` (the orders for the ARMA model fitted to the residuals) and `b1` and `b2`, the considered bandwidths.

`AD` a dataframe containing `ci_inf` and `ci_sup`, the confidence intervals using the asymptotic distribution; `p_opt` and `q_opt` (the orders for the ARMA model fitted to the residuals) and `b1`, the considered bandwidth.

`pv.Box.test` p-values of the Ljung-Box test for the model fitted to the residuals.

`pv.t.test` p-values of the t.test for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Liang, H., Hardle, W., Sommerfeld, V. (2000) Bootstrap approximation in a partially linear regression model. *Journal of Statistical Planning and Inference* **91**, 413-426.

You, J., Zhou, X. (2005) Bootstrap of a semiparametric partially linear model with autoregressive errors. *Statistica Sinica* **15**, 117-133.

See Also

A related functions is `par.ci`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

b.h <- plrm.gcv(data)$bh.opt
b1 <- b.h[1]

## Not run: plrm.ci(data, b1=b1, b2=b1, a=c(1,0), CI="all")
## Not run: plrm.ci(data, b1=b1, b2=b1, a=c(0,1), CI="all")
```

```
# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(123)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
m <- function(t) {t+0.5}
f <- m(t)

beta <- c(0.5, 2)
x <- matrix(rnorm(200,0,3), nrow=n)
sum <- x%*%beta
sum <- as.matrix(sum)
eps <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.1, n = n)
eps <- as.matrix(eps)

y <-  sum + f + eps
data_plrmci <- cbind(y,x,t)

## Not run: plrm.ci(data, a=c(1,0), CI="all")
## Not run: plrm.ci(data, a=c(0,1), CI="all")
```

---

plrm.cv *Cross-validation bandwidth selection in PLR models*

Description

From a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, this routine computes, for each l_n considered, an optimal pair of bandwidths for estimating the regression function of the model

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i,$$

where

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$m(.)$$

is a smooth but unknown function. The random errors, ϵ_i, are allowed to be time series. The optimal pair of bandwidths, (`b.opt, h.opt`), is selected by means of the leave-$(2l_n+1)$-out cross-validation procedure. The bandwidth `b.opt` is used in the estimate of β, while the pair of bandwidths `(b.opt, h.opt)` is considered in the estimate of m. Kernel smoothing, combined with ordinary least squares estimation, is used.

Usage

```
plrm.cv(data = data, b.equal.h = TRUE, b.seq=NULL, h.seq=NULL,
num.b = NULL, num.h = NULL, w = NULL, num.ln = 1, ln.0 = 0,
step.ln = 2, estimator = "NW", kernel = "quadratic")
```

Arguments

data	data[,1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; data[, p+2] contains the values of the "nonparametric" explanatory variable, t.
b.equal.h	if TRUE (the default), the same bandwidth is used for estimating both β and m.
b.seq	sequence of considered bandwidths, b, in the CV function for estimating β. If NULL (the default), num.b equidistant values between zero and a quarter of the range of t_i are considered.
h.seq	sequence of considered bandwidths, h, in the pair of bandwidths (b, h) used in the CV function for estimating m. If NULL (the default), num.h equidistant values between zero and a quarter of the range of t_i are considered.
num.b	number of values used to build the sequence of considered bandwidths for estimating β. If b.seq is not NULL, num.b=length(b.seq). Otherwise, if both num.b and num.h are NULL (the default), num.b=50 is considered; if num.b is NULL (the default) but num.h is not NULL, then num.b=num.h is considered; if b.equal.h=TRUE (the default) and both num.b and num.h are not NULL and different, the maximum value of num.b and num.h is considered for both.
num.h	pairs of bandwidths (b, h) are used for estimating m, num.h being the number of values considered for h. If h.seq is not NULL, num.h=length(h.seq). Otherwise, if both num.b and num.h are NULL (the default), num.h=50 is considered; if num.h is NULL (the default) but num.b is not NULL, num.h=num.b is considered; if b.equal.h=TRUE (the default) and both num.b and num.h are not NULL and different, the maximum value of num.b and num.h is considered for both.
w	support interval of the weigth function in the CV function. If NULL (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
num.ln	number of values for l_n: after estimating β, $2l_n + 1$ observations around each point t_i are eliminated to estimate $m(t_i)$ in the CV function. The default is 1.
ln.0	minimum value for l_n. The default is 0.
step.ln	distance between two consecutives values of l_n. The default is 2.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the CV function to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

As noted in the definition of num.ln, the estimate of β in the CV function is obtained from all data while, once β is estimated, $2l_n + 1$ observations around each t_i are eliminated to estimate $m(t_i)$ in the CV function. Actually, the estimate of β to be used in time i in the CV function could be done eliminating such $2l_n + 1$ observations too; that possibility was not implemented because both their

computational cost and the known fact that the estimate of β is quite insensitive to the bandwidth selection.

The implemented procedure generalizes that one in expression (8) in Aneiros-Perez and Quintela-del-Rio (2001) by including a weight function (see above) and allowing two smoothing parameters instead of only one (see Aneiros-Perez *et al.*, 2004).

Value

`bh.opt`	dataframe containing, for each `ln` considered, the selected value for `(b,h)`.
`CV.opt`	`CV.opt[k]` is the minimum value of the CV function when de k-th value of `ln` is considered.
`CV`	an array containing the values of the CV function for each pair of bandwidths and `ln` considered.
`b.seq`	sequence of considered bandwidths, `b`, in the CV function for estimating β.
`h.seq`	sequence of considered bandwidths, `h`, in the pair of bandwidths `(b, h)` used in the CV function for estimating m.
`w`	support interval of the weigth function in the CV function.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Aneiros-Perez, G. and Quintela-del-Rio, A. (2001) Modified cross-validation in semiparametric regression models with dependent errors. *Comm. Statist. Theory Methods* **30**, 289-307.

Chu, C-K and Marron, J.S. (1991) Comparison of two bandwidth selectors with dependent errors. *The Annals of Statistics* **19**, 1906-1918.

See Also

Other related functions are: `plrm.beta`, `plrm.est`, `plrm.gcv`, `np.est`, `np.gcv` and `np.cv`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

aux <- plrm.cv(data, step.ln=1, num.ln=2)
aux$bh.opt
plot.ts(aux$CV[,-2,])
```

```
par(mfrow=c(2,1))
plot(aux$b.seq,aux$CV[,-2,1], xlab="h", ylab="CV", type="l", main="ln=0")
plot(aux$b.seq,aux$CV[,-2,2], xlab="h", ylab="CV", type="l", main="ln=1")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)

# We apply the function
a <-plrm.cv(data_ind)
a$CV.opt

CV <- a$CV
h <- a$h.seq
plot(h, CV,type="l")


## Example 2b: dependent data and ln.0 > 0

set.seed(1234)
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data_dep <- matrix(c(y,x,t),nrow=100)

# We apply the function
a <-plrm.cv(data_dep, ln.0=2)
a$CV.opt

CV <- a$CV
h <- a$h.seq
plot(h, CV,type="l")
```

plrm.est *Semiparametric estimates for the unknown components of the regression function in PLR models*

Description

This routine computes estimates for β and $m(newt_j)$ $(j = 1, ..., J)$ from a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i)$: $i = 1, ..., n$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter,

$$m(.)$$

is a smooth but unknown function and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

The random errors, ϵ_i, are allowed to be time series. Kernel smoothing, combined with ordinary least squares estimation, is used.

Usage

```
plrm.est(data = data, b = NULL, h = NULL, newt = NULL, estimator = "NW",
kernel = "quadratic")
```

Arguments

data	data[, 1] contains the values of the response variable, Y; data[, 2:(p+1)] contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; data[, p+2] contains the values of the "nonparametric" explanatory variable, t.
b	bandwidth for estimating the parametric part of the model. If both b and h are NULL (the default), it is selected by means of the cross-validation procedure (fixing b=h); if b is NULL (the default) but h is not NULL, b=h is considered.
h	(b,h) is the pair of bandwidths for estimating the nonparametric part of the model. If both b and h are NULL (the default), it is selected by means of the cross-validation procedure (fixing b=h); if b is NULL (the default) but h is not NULL, b=h is considered; if h is NULL (the default) but b is not NULL, h=b is considered.
newt	values of the "nonparametric" explanatory variable where the estimator of m is evaluated. If NULL (the default), the considered values will be the values of data[,p+2].
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

Expressions for the estimators of β and m can be seen in page 52 in Aneiros-Perez *et al.* (2004).

Value

A list containing:

`beta`	a vector containing the estimate of β.
`m.t`	a vector containing the estimator of the non-parametric part, m, evaluated in the design points.
`m.newt`	a vector containing the estimator of the non-parametric part, m, evaluated in `newt`.
`residuals`	a vector containing the residuals: `Y - X*beta - m.t`.
`fitted.values`	the values obtained from the expression: `X*beta + m.t`
`b`	the considered bandwidth for estimating β.
`h`	`(b,h)` is the pair of bandwidths considered for estimating m.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Hardle, W., Liang, H. and Gao, J. (2000) *Partially Linear Models*. Physica-Verlag.

Speckman, P. (1988) Kernel smoothing in partial linear models. *J. R. Statist. Soc. B* **50**, 413-436.

See Also

Other related functions are: `plrm.beta`, `plrm.gcv`, `plrm.cv`, `np.est`, `np.gcv` and `np.cv`.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

b.h <- plrm.gcv(data)$bh.opt
ajuste <- plrm.est(data=data, b=b.h[1], h=b.h[2])
ajuste$beta
plot(data[,4], ajuste$m, type="l", xlab="t", ylab="m(t)")

plot(data[,1], ajuste$fitted.values, xlab="y", ylab="y.hat", main="y.hat vs y")
abline(0,1)
```

```
mean(ajuste$residuals^2)/var(data[,1])



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)

# We estimate the components of the PLR model
# (CV bandwidth)
a <- plrm.est(data_ind)

a$beta

est <- a$m.t
plot(t, est, type="l", lty=2, ylab="")
points(t, 0.25*t*(1-t), type="l")
legend(x="topleft", legend = c("m", "m hat"), col=c("black", "black"), lty=c(1,2))


## Example 2b: dependent data
# We generate the data
x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data_dep <- matrix(c(y,x,t),nrow=100)

# We estimate the components of the PLR model
# (CV bandwidth)
h <- plrm.cv(data_dep, ln.0=2)$bh.opt[3,1]
a <- plrm.est(data_dep, h=h)

a$beta

est <- a$m.t
plot(t, est, type="l", lty=2, ylab="")
points(t, 0.25*t*(1-t), type="l")
legend(x="topleft", legend = c("m", "m hat"), col=c("black", "black"), lty=c(1,2))
```

plrm.gcv *Generalized cross-validation bandwidth selection in PLR models*

Description

From a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, this routine computes an optimal pair of bandwidths for estimating the regression function of the model

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i,$$

where

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter and

$$m(.)$$

is a smooth but unknown function. The optimal pair of bandwidths, `(b.opt, h.opt)`, is selected by means of the generalized cross-validation procedure. The bandwidth `b.opt` is used in the estimate of β, while the pair of bandwidths `(b.opt, h.opt)` is considered in the estimate of m. Kernel smoothing, combined with ordinary least squares estimation, is used.

Usage

```
plrm.gcv(data = data, b.equal.h = TRUE, b.seq=NULL, h.seq=NULL,
num.b = NULL, num.h = NULL, estimator = "NW", kernel = "quadratic")
```

Arguments

`data`	`data[,1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the "linear" explanatory variables, $X_1, ..., X_p$; `data[, p+2]` contains the values of the "nonparametric" explanatory variable, t.
`b.equal.h`	if TRUE (the default), the same bandwidth is used for estimating both β and m.
`b.seq`	sequence of considered bandwidths, `b`, in the GCV function for estimating β.
`h.seq`	sequence of considered bandwidths, `h`, in the pair of bandwidths `(b, h)` used in the GCV function for estimating m.
`num.b`	number of values used to build the sequence of considered bandwidths for estimating β. If `b.seq` is not `NULL`, `num.b=length(b.seq)`. Otherwise, if both `num.b` and `num.h` are `NULL` (the default), `num.b=50` is considered; if `num.b` is `NULL` (the default) but `num.h` is not `NULL`, then `num.b=num.h` is considered; if `b.equal.h=TRUE` (the default) and both `num.b` and `num.h` are not `NULL` and different, the maximum value of `num.b` and `num.h` is considered for both.

num.h	pairs of bandwidths (b, h) are used for estimating m, num.h being the number of values considered for h. If h.seq is not NULL, num.h=length(h.seq). Otherwise, if both num.b and num.h are NULL (the default), num.h=50 is considered; if num.h is NULL (the default) but num.b is not NULL, num.h=num.b is considered; if b.equal.h=TRUE (the default) and both num.b and num.h are not NULL and different, the maximum value of num.b and num.h is considered for both.
estimator	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
kernel	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

Details

The implemented procedure generalizes that one in page 423 in Speckman (1988) by allowing two smoothing parameters instead of only one (see Aneiros-Perez *et al.*, 2004).

Value

bh.opt	selected value for (b,h).
GCV.opt	minimum value of the GCV function.
GCV	matrix containing the values of the GCV function for each pair of bandwidths considered.
b.seq	sequence of considered bandwidths, b, in the GCV function for estimating β. If b.seq was not input by the user, it is composed by num.b equidistant values between zero and a quarter of the range of t_i.
h.seq	sequence of considered bandwidths, h, in the pair of bandwidths (b, h) used in the GCV function for estimating m. If h.seq was not input by the user, it is composed by num.h equidistant values between zero and a quarter of the range of t_i.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Green, P. (1985) Linear models for field trials, smoothing and cross-validation. *Biometrika* **72**, 527-537.

Speckman, P. (1988) Kernel smoothing in partial linear models *J. R. Statist. Soc. B* **50**, 413-436.

See Also

Other related functions are: plrm.beta, plrm.est, plrm.cv, np.est, np.gcv and np.cv.

Examples

```
# EXAMPLE 1: REAL DATA

data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

aux <- plrm.gcv(data)
aux$bh.opt
plot(aux$b.seq, aux$GCV, xlab="h", ylab="GCV", type="l")



# EXAMPLE 2: SIMULATED DATA
## Example 2a: independent data

set.seed(1234)

# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- rnorm(n, 0, 0.01)
y <-  sum + f + epsilon
data_ind <- matrix(c(y,x,t),nrow=100)

# We obtain the optimal bandwidths
a <-plrm.gcv(data_ind)
a$GCV.opt

GCV <- a$GCV
h <- a$h.seq
plot(h, GCV,type="l")
```

`plrm.gof` *Goodness-of-Fit tests in PLR models*

Description

From a sample $(Y_i, X_{i1}, ..., X_{ip}, t_i) : i = 1, ..., n$, this routine tests the null hypotheses $H_0 : \beta = \beta_0$ and $H_0 : m = m_0$, where:

$$\beta = (\beta_1, ..., \beta_p)$$

is an unknown vector parameter,

$$m(.)$$

is a smooth but unknown function and

$$Y_i = X_{i1} * \beta_1 + ... + X_{ip} * \beta_p + m(t_i) + \epsilon_i.$$

Fixed equally spaced design is considered for the "nonparametric" explanatory variable, t, and the random errors, ϵ_i, are allowed to be time series. The test statistic used for testing $H0 : \beta = \beta_0$ derives from the asymptotic normality of an estimator of β based on both ordinary least squares and kernel smoothing (this result giving a χ^2-test). The test statistic used for testing $H0 : m = m_0$ derives from a Cramer-von-Mises-type functional distance between a nonparametric estimator of m and m_0.

Usage

```
plrm.gof(data = data, beta0 = NULL, m0 = NULL, b.seq = NULL,
h.seq = NULL, w = NULL, estimator = "NW", kernel = "quadratic",
time.series = FALSE, Var.Cov.eps = NULL, Tau.eps = NULL,
b0 = NULL, h0 = NULL, lag.max = 50, p.max = 3, q.max = 3,
ic = "BIC", num.lb = 10, alpha = 0.05)
```

Arguments

`data`	`data[,1]` contains the values of the response variable, Y; `data[, 2:(p+1)]` contains the values of the "linear" explanatory variables $X_1, ...X_p$; `data[, p+2]` contains the values of the "nonparametric" explanatory variable, t.
`beta0`	the considered parameter vector in the parametric null hypothesis. If `NULL` (the default), the zero vector is considered.
`m0`	the considered function in the nonparametric null hypothesis. If `NULL` (the default), the zero function is considered.
`b.seq`	the statistic test for $H0 : \beta = \beta_0$ is performed using each bandwidth in the vector `b.seq`. If `NULL` (the default) but `h.seq` is not `NULL`, it takes `b.seq=h.seq`. If both `b.seq` and `h.seq` are `NULL`, 10 equidistant values between zero and a quarter of the range of t_i are considered.
`h.seq`	the statistic test for $H0 : m = m_0$ is performed using each pair of bandwidths (`b.seq[j], h.seq[j]`). If `NULL` (the default) but `b.seq` is not `NULL`, it takes `h.seq=b.seq`. If both `b.seq` and `h.seq` are `NULL`, 10 equidistant values between zero and a quarter of the range of t_i are considered for both `b.seq` and `h.seq`.
`w`	support interval of the weigth function in the test statistic for $H0 : m = m_0$. If `NULL` (the default), $(q_{0.1}, q_{0.9})$ is considered, where q_p denotes the quantile of order p of t_i.
`estimator`	allows us the choice between "NW" (Nadaraya-Watson) or "LLP" (Local Linear Polynomial). The default is "NW".
`kernel`	allows us the choice between "gaussian", "quadratic" (Epanechnikov kernel), "triweight" or "uniform" kernel. The default is "quadratic".

`time.series` it denotes whether the data are independent (FALSE) or if data is a time series (TRUE). The default is FALSE.

`Var.Cov.eps` `n` x `n` matrix of variances-covariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the var-cov matrix of such ARMA model.

`Tau.eps` it contains the sum of autocovariances associated to the random errors of the regression model. If NULL (the default), the function tries to estimate it: it fits an ARMA model (selected according to an information criterium) to the residuals from the fitted regression model and, then, it obtains the sum of the autocovariances of such ARMA model.

`b0` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `b0` contains the pilot bandwidth for the estimator of β used for obtaining the residuals to construct the default for `Var.Cov.eps` and/or `Tau.eps`. If `NULL` (the default) but `h0` is not `NULL`, it takes `b0=h0`. If both `b0` and `h0` are `NULL`, a quarter of the range of t_i is considered.

`h0` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, (`b0, h0`) contains the pair of pilot bandwidths for the estimator of m used for obtaining the residuals to construct the default for `Var.Cov.eps` and/or `Tau.eps`. If `NULL` (the default) but `b0` is not `NULL`, it takes `h0=b0`. If both `b0` and `h0` are `NULL`, a quarter of the range of t_i is considered for both `b0` and `h0`.

`lag.max` if `Tau.eps=NULL`, `lag.max` contains the maximum delay used to construct the default for `Tau.eps`. The default is 50.

`p.max` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=`p.max` and 0<=q<=`q.max`. The default is 3.

`q.max` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, the ARMA model is selected between the models ARMA(p,q) with 0<=p<=`p.max` and 0<=q<=`q.max`. The default is 3.

`ic` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `ic` contains the information criterion used to suggest the ARMA model. It allows us to choose between: "AIC", "AICC" or "BIC" (the default).

`num.lb` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, it checks the suitability of the selected ARMA model according to the Ljung-Box test and the t-test. It uses up to `num.lb` delays in the Ljung-Box test. The default is 10.

`alpha` if `Var.Cov.eps=NULL` and/or `Tau.eps=NULL`, `alpha` contains the significance level which the ARMA model is checked. The default is 0.05.

Details

A weight function (specifically, the indicator function $\mathbf{1}_{[w[1],w[2]]}$) is introduced in the test statistic for testing $H0 : m = m_0$ to allow elimination (or at least significant reduction) of boundary effects from the estimate of $m(t_i)$.

If `Var.Cov.eps=NULL` and the routine is not able to suggest an approximation for `Var.Cov.eps`, it warns the user with a message saying that the model could be not appropriate and then it shows

the results. In order to construct Var.Cov.eps, the procedure suggested in Aneiros-Perez and Vieu (2013) can be followed.

If Tau.eps=NULL and the routine is not able to suggest an approximation for Tau.eps, it warns the user with a message saying that the model could be not appropriate and then it shows the results. In order to construct Tau.eps, the procedures suggested in Aneiros-Perez (2008) can be followed.

The implemented procedures generalize those ones in expressions (9) and (10) in Gonzalez-Manteiga and Aneiros-Perez (2003) by allowing some dependence condition in $(X_{i1}, ..., X_{ip}) : i = 1, ..., n$ and including a weight function (see above), respectively.

Value

A list with two dataframes:

parametric.test
: a dataframe containing the bandwidths, the statistics and the p-values when one tests $H0 : \beta = \beta_0$

nonparametric.test
: a dataframe containing the bandwidths b and h, the statistics, the normalised statistics and the p-values when one tests $H0 : m = m_0$

Moreover, if data is a time series and Tau.eps or Var.Cov.eps are not especified:

pv.Box.test	p-values of the Ljung-Box test for the model fitted to the residuals.
pv.t.test	p-values of the t.test for the model fitted to the residuals.
ar.ma	ARMA orders for the model fitted to the residuals.

Author(s)

German Aneiros Perez <ganeiros@udc.es>

Ana Lopez Cheda <ana.lopez.cheda@udc.es>

References

Aneiros-Perez, G. (2008) Semi-parametric analysis of covariance under dependence conditions within each group. *Aust. N. Z. J. Stat.* **50**, 97-123.

Aneiros-Perez, G., Gonzalez-Manteiga, W. and Vieu, P. (2004) Estimation and testing in a partial linear regression under long-memory dependence. *Bernoulli* **10**, 49-78.

Aneiros-Perez, G. and Vieu, P. (2013) Testing linearity in semi-parametric functional data analysis. *Comput. Stat.* **28**, 413-434.

Gao, J. (1997) Adaptive parametric test in a semiparametric regression model. *Comm. Statist. Theory Methods* **26**, 787-800.

Gonzalez-Manteiga, W. and Aneiros-Perez, G. (2003) Testing in partial linear regression models with dependent errors. *J. Nonparametr. Statist.* **15**, 93-111.

See Also

Other related functions are plrm.est, par.gof and np.gof.

Examples

```
# EXAMPLE 1: REAL DATA
data(barnacles1)
data <- as.matrix(barnacles1)
data <- diff(data, 12)
data <- cbind(data,1:nrow(data))

plrm.gof(data)
plrm.gof(data, beta0=c(-0.1, 0.35))



# EXAMPLE 2: SIMULATED DATA
## Example 2a: dependent data

set.seed(1234)
# We generate the data
n <- 100
t <- ((1:n)-0.5)/n
beta <- c(0.05, 0.01)
m <- function(t) {0.25*t*(1-t)}
f <- m(t)
f.function <- function(u) {0.25*u*(1-u)}

x <- matrix(rnorm(200,0,1), nrow=n)
sum <- x%*%beta
epsilon <- arima.sim(list(order = c(1,0,0), ar=0.7), sd = 0.01, n = n)
y <-  sum + f + epsilon
data <- cbind(y,x,t)

## Example 2a.1: true null hypotheses
plrm.gof(data, beta0=c(0.05, 0.01), m0=f.function, time.series=TRUE)

## Example 2a.2: false null hypotheses
plrm.gof(data, time.series=TRUE)
```

Index