Package ‘gammadlm’

June 16, 2022

Type Package

Title Maximum Likelihood Estimation of the Gamma Distributed-Lag Model with Multiple Explanatory Variables

Version 0.1.1

Date 2022-06-16

Author Alessandro Magrini

Maintainer Alessandro Magrini <alessandro.magrini@unifi.it>

Description Implementation of the hill climbing algorithm for maximum likelihood estimation of the Gamma distributed-lag model with multiple explanatory variables (Magrini, 2022 <doi:10.17713/ajs.v51i2.1244>). Data may have a panel structure, even unbalanced.

Depends R (>= 3.5.0)

Imports graphics, stats

License GPL-2

NeedsCompilation no

R topics documented:

`gammadlm-package` *Maximum Likelihood Estimation of the Gamma Distributed-Lag Model with Multiple Explanatory Variables*

Description

Implementation of the hill climbing algorithm for maximum likelihood estimation of the Gamma distributed-lag model with multiple explanatory variables (Magrini, 2022 <doi:10.17713/ajs.v51i2.1244>). Data may have a panel structure, even unbalanced.

Details

Package:	gammadlm
Type:	Package
Version:	0.1.1
Date:	2022-06-16
License:	GPL-2

Let Y be the response variable and $X_1,\dots,X_J$ be J explanatory variables. Also, let y_t and $x_{j,t}$ be, respectively, the value of Y and of X_j ($j=1,\dots,J$) observed at time t. Under the assumption that the time series of Y and of $X_1,\dots,X_J$ are all weakly stationary (i.e., expected value and autocorrelation function independent of time), the Gamma distributed-lag model explaining Y from $X_1,\dots,X_J$ is defined as:

$$y_t = \alpha + \sum_{j=1}^{J}\sum_{k=0}^{\infty} \beta_{j,k}(\theta_j,\delta_j,\lambda_j,\eta_j)\, x_{j,t-k} + \varepsilon_t$$

$$\beta_{j,k}(\theta_j,\delta_j,\lambda_j,\eta_j) = \theta_j\, w_{j,k}(\delta_j,\lambda_j,\eta_j)$$

$$w_{j,k}(\delta_j,\lambda_j,\eta_j) = \frac{(k+1-\eta_j)^{\frac{\delta_j}{1-\delta_j}}\,\lambda_j^{k-\eta_j}}{\sum_{l=0}^{\infty}(l+1-\eta_j)^{\frac{\delta_j}{1-\delta_j}}\,\lambda_j^{l-\eta_j}}$$

where:

- α is the intercept;
- $\beta_{j,k}$ is the dynamic coefficient for X_j at time lag k, equal to the scale parameter θ_j times the weight $w_{j,k}$. The set $\{w_{j,k} : k=0,1,\dots,\infty\}$ includes the weights for X_j and is defined by the shape parameters $0 \le \delta_j < 1$ and $0 \le \lambda_j < 1$ and the offset η_j (typically set to 0). The set $\{\beta_{j,k} : k=0,1,\dots,\infty\}$ is called *lag distribution* of X_j;
- ε_t is the random error at time t.

In case of a panel structure, one intercept is specified for each unit of observation, while the lag distributions are the same for all units:

$$y_{i,t} = \alpha_i + \sum_{j=1}^{J}\sum_{k=0}^{\infty} \beta_{j,k}(\theta_j,\delta_j,\lambda_j,\eta_j)\, x_{i,j,t-k} + \varepsilon_{i,t}$$

where:

- $y_{i,t}$ and $x_{i,j,t}$ are, respectively, the value of Y and of X_j ($j = 1, \dots, J$) observed on unit i at time t;
- α_i is the intercept for unit i;
- $\varepsilon_{i,t}$ is the random error for unit i at time t.

The main functions of the package are:

- unirootTest, to check stationarity;
- preProcess, to apply the Box-Cox transformation and differencing in order to achieve stationarity;
- gammadlm, to estimate the model through the hill climbing algorithm;
- lagCoef, to see the estimated dynamic coefficients;
- plot.gammadlm, to display the estimated lag distributions.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

References

A. Magrini (2022). A hill climbing algorithm for maximum likelihood estimation of the Gamma distributed-lag model with multiple explanatory variables. *Austrian Journal of Statistics*, 51(2): 40-46. DOI: 10.17713/ajs.v51i2.1244

`btc` *Bitcoin and US stock indices data*

Description

Daily close exchange rate of Bitcoin and of three composite indices of the US stock market from 17 September 2014 to 30 September 2020.

Usage

```
data(btc)
```

Format

A data.frame with a total of 1521 observations on the following 5 variables:

`Date` Day of observation.

`DJA` Exchange rate of Dow Jones Average index.

`IXIC` Exchange rate of Nasdaq Composite index.

`GSPC` Exchange rate of Standard\&Poor 500 index.

`BTC` Exchange rate of Bitcoin.

`fadn` *Farm performance and subsidies in EU countries*

Description

Aggregated data on farm performance and public subsidies in EU countries in the period 2004-2019. Source: Farm Accountancy Data Network (FADN, European Commission, 2020).

Usage

```
data(fadn)
```

Format

A data.frame with a total of 433 observations on the following 15 variables:

`Country` Country name.

`Country_code` Country code.

`Year` Time of measurement (year).

`Labour` Total labour input (annual work unit), item SE010.

`Land` Total utilised agricultural area (ha), SE025.

`Capital` Total intermediate consumption (euro), item SE275.

`Int_cons` Depreciation of fixed capital, excluding land (euro), SE360.

`Inputs_total` Total inputs (euro), item SE270.

`Output_total` Total output (euro), item SE131.

`TFP` Total factor productivity, computed as the ratio of total output to total input, item SE132.

`Net_income` Farm net income (euro), item SE420.

`Subs_prod` Subsidies on production (euro), item SE605.

`Subs_inv` Subsidies on investments (euro), item SE406.

`Subs_rur` Total support for rural development (euro), item SE624.

`Subs_dec` Decoupled payments (euro), item SE630.

References

European Commission (2020). Farm Accountancy Data Network (FADN).

https://agridata.ec.europa.eu/extensions/FADNPublicDatabase/FADNPublicDatabase.html

gammadlm *Estimation of a Gamma distributed-lag model*

Description

Maximum likelihood estimation of a Gamma distributed-lag model with multiple explanatory variables using the hill climbing algorithm. Data may have a panel structure, even unbalanced.

Usage

```
gammadlm(y.name, x.names, z.names=NULL, unit=NULL, time=NULL, data,
  offset=rep(0,length(x.names)), add.intercept=TRUE, control=list(nstart=NULL,
  delta.lim=NULL, lambda.lim=NULL, peak.lim=NULL, length.lim=NULL), quiet=FALSE)
```

Arguments

`y.name`	Character including the name of the response variable, that must be a quantitative variable. If a vector with length greater than 1 is provided, only the first element is considered.
`x.names`	Character vector of length 1 or greater including the names of the explanatory variables with lags, that must be quantitative variables. If the name of the response variable is indicated in `x.names`, then the autoregressive lag distribution will be estimated.
`z.names`	Character vector including the names of the explanatory variables without lags (optional). They may be either quantitative or qualitative variables. If `NULL` (the default), no explanatory variable without lags is included in the model.
`unit`	Character containing the name of the units of observation in case of a panel structure. If a vector with length greater than 1 is provided, only the first element is considered. If `NULL` (the default), a single unit of observation is assumed.
`time`	Character containing the name of the time variable, which must be in numeric or date format. If a vector with length greater than 1 is provided, only the first element is considered. If `NULL` (the default), data are assumed to be temporally ordered.
`data`	Object of class `data.frame` containing the variables in the model. Variables in `y.name`, `x.names`, `z.names`, `unit` and `time` cannot contain missing values.
`offset`	Named vector with non-negative real values indicating the offset for variables in `x.names`. If `offset` is of length one and has no names, the same offset is used for all variables in `x.names`. If a variable in `x.names` has no name in `offset` or a negative value is provided, value 0 is assumed. If the name of the response variable is indicated in `x.names` and a value less than 1 is provided, value 1 is assumed. Default value of `offset` is 0.
`add.intercept`	Logical value indicating whether the intercept should be included in the model. Default is `TRUE`. See 'Details'.
`control`	A list including control options for the hill climbing algorithm. • `nstart`: positive integer value indicating the number of restarts. If equal to 1 or `NULL` (the default), shape parameters are initialized for one explanatory variable at a time based on OLS-grid search.

- `delta.lim`: a named list with one component for each variable in `x.names`, that must be either a numerical vector of length 2 indicating the minimum and the maximum value of δ, or a numerical value indicating the exact value of δ. If there is no component in `delta.lim` for a certain variable in `x.names`, then the theoretical range [0,1) is assumed for δ.
- `lambda.lim`: the same as `delta.lim`, but it is about λ parameters.
- `peak.lim`: the same as `delta.lim`, but it is about the peak of the lag distributions.
- `length.lim`: the same as `delta.lim`, but it is about the 99.9th percentile of the lag distributions.

`quiet`	Logical value indicating whether prompt messages should be suppressed. Default is `FALSE`.

Details

If the response variable is not differenced, the intercept captures the effect of time-invariant factors. Otherwise, the intercept represents the drift. To avoid the drift, argument `add.intercept` should be set to `FALSE` when differencing is applied to the response variable.

All S3 methods for class `lm` are also available for class `gammadlm`. Notably:

- `plot` displays the estimated lag distributions;

- `summary` provides a summary of parameter estimation;

- `residuals` returns the residuals;

- `fitted.values` returns the fitted values;

- `predict` performs h step ahead forecast.

Furthermore, method lagCoef can be used to see the estimated dynamic coefficients.

Value

An object of class `lm` and `gammadlm`, including all the components of an object of class `lm` plus the following components:

- `offset`: vector including the offset of the lag distributions;
- `add.intercept`: logical value indicating whether the model includes the intercept;
- `par`: matrix including the shape parameters for each variable in `x.names` (by column);
- `variables`: list including the names of the variables provided to arguments `y.name`, `x.names`, `z.names`, `unit` and `time`.
- `control`: list including control options used in parameter estimation;
- `unit.id`: list including the row names of the observations for each unit. `NULL` if `unit` is `NULL`.
- `data`: data.frame including the data used for parameter estimation.
- `local.max`: list including all the models fitted at each restart.

Note

Weak stationarity of all time series (expected value and autocorrelation function independent of time) is a basic assumption of the model, that is guaranteed if no time series contains unit roots.

Before calling the function gammadlm, the user is strongly recommended to check the absence of unit roots in each time series through the function unirootTest, and to apply differencing through the function preProcess to the ones containing unit roots.

Function gammadlm automatically checks the absence of unit roots in the residuals and returns a warning in case of failure.

When the `summary` method is called on an object of class `gammadlm`, the order of auto-correlation of the residuals is estimated based on the Bayesian Information Criterion. If it is greater than 0, then the Heteroskedasticity and Autocorrelation Consistent (HAC, Newey & West, 1987) estimator of the covariance matrix of least squares estimates is applied to get robust standard errors. The same holds for the `confint` method.

References

A. Magrini (2022). A hill climbing algorithm for maximum likelihood estimation of the Gamma distributed-lag model with multiple explanatory variables. *Austrian Journal of Statistics*, 51(2): 40-46. DOI: 10.17713/ajs.v51i2.1244

W. K. Newey, K. D. West (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3): 703-708

See Also

unirootTest; lagCoef; plot.gammadlm.

Examples

```
# load data on Bitcoin and US stock market price
data(btc)
mydata <- btc[which(btc$Date>="2020-04-01"),]

# compute logarithmic differences (yearly relative changes)
#   to achieve weak stationarity
mydataLR <- preProcess(c("BTC","DJA","IXIC","GSPC"),
  time="Date", data=mydata, box.cox=0, ndiff=1)

# estimation with pre-specified shape parameters (simple OLS)
dval <- list(DJA=0.85,IXIC=0.75,GSPC=0.55)
lval <- list(DJA=0.05, IXIC=0.35,GSPC=0.45)
m0 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(delta.lim=dval, lambda.lim=lval))
summary(m0)  ## summary of estimation

# hill climbing algorithm without random restarts
#   (initialization based on independent OLS-grid searches)
m1 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR)
summary(m1)

# add constraints: peak>=1 and 3<=length<=15
pklim <- list(DJA=c(1,Inf),IXIC=c(1,Inf),GSPC=c(1,Inf))
lenlim <- list(DJA=c(3,15),IXIC=c(3,15),GSPC=c(3,15))
m2 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(peak.lim=pklim, length.lim=lenlim))
summary(m2)

## 50 random restarts without constraints: NOT RUN
#set.seed(100)
#m3 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
#  control=list(nstart=50))
#summary(m3)  ## better fit than m1
```

```
##
## 50 random restarts with constraints: NOT RUN
#set.seed(100)
#m4 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
#  control=list(peak.lim=pklim, length.lim=lenlim, nstart=50))
#summary(m4)  ## better fit than m2


###  Example with panel data  ###

# load data on farm performance and subsidies in EU countries, 2004-2019
data(fadn)

# selected variables:
#  Y: productivity
#  X: subsidies
#  Z: utilized agricultural area (dimensional class),
#     total output (economic class)
y_name <- "TFP"
x_name <- c("Subs_prod","Subs_inv","Subs_rur","Subs_dec")
z_name <- c("Land","Output_total")

# compute logarithmic differences (yearly relative changes)
#   to achieve weak stationarity
fadnLR <- preProcess(c(y_name,x_name,z_name),
  unit="Country",time="Year", data=fadn, box.cox=0, ndiff=1)

# model
#  (since data are differenced, intercepts represent the
#   coefficients of country-specific linear trends)
m_fadn <- gammadlm(y.name=y_name, x.names=x_name, z.names=z_name,
  unit="Country", time="Year", data=fadnLR, add.intercept=FALSE,
  control=list(peak.lim=c(1,Inf), length.lim=c(3,10)))
summary(m_fadn)
```

`gammaWeights` *Functionalities for the Gamma lag distribution*

Description

Obtain weights, quantiles and kernel projection for the desired Gamma lag distribution.

Usage

```
gammaWeights(k, par, offset=0, normalize=TRUE)
gammaQuantile(prob, par, offset=0)
gammaKernel(x, par, unit=NULL, offset=0, normalize=TRUE)
```

Arguments

k	Numerical vector indicating the lags for which the weights should be computed.
prob	Numerical vector indicating the order of the quantiles to be computed.
x	Numerical vector representing temporally ordered data for which the kernel projection should be returned.

`par`	Numerical vector of length 2 representing the shape parameters of the Gamma lag distribution.
`unit`	Character containing the name of the units of observation in case of a panel structure. If a vector with length greater than 1 is provided, only the first element is considered. If `NULL` (the default), a single unit of observation is assumed.
`offset`	Numerical value representing the offset of the Gamma lag distribution. Default is 0.
`normalize`	Logical value indicating whether the weights should be normalized to have sum 1. Default is `TRUE`.

Details

Function `gammaWeights` provides the weights, function `gammaQuantile` computes the quantiles, and function `gammaKernel` returns the kernel projection.

Examples

```
## examples for a Gamma lag distribution with delta=0.6 and lambda=0.3

# weights
gammaWeights(0:12, par=c(0.6,0.3))  ## at lags from 0 to 12
gammaWeights(10, par=c(0.6,0.3))    ## at lag 10

# quantiles
gammaQuantile(0.5, par=c(0.6,0.3))   ## median
gammaQuantile(0.95, par=c(0.6,0.3))  ## 95th percentile
gammaQuantile(0.99, par=c(0.6,0.3))  ## 99th percentile

# kernel projection
set.seed(100); xval <- rnorm(10)
gammaKernel(xval, par=c(0.6,0.3))

# kernel projection under a panel structure
set.seed(100); xval <- rnorm(20)
gr <- c(rep(0,10),rep(1,10))
gammaKernel(xval, par=c(0.6,0.3), unit=gr)
```

`lagCoef` *Estimated dynamic coefficients*

Description

See the estimated dynamic coefficients for each explanatory variable with lags.

Usage

```
lagCoef(x, conf=0.95, cumulative=FALSE, max.lag=NULL, max.quantile=0.999)
```

Arguments

`x`	An object of class `gammadlm`.
`conf`	Numerical value indicating the level of confidence intervals. If `NULL`, confidence intervals are not provided. Default is 0.95.
`cumulative`	Logical value indicating whether cumulative coefficients should be returned. Default is `FALSE`.
`max.lag`	Non-negative integer value indicating the lag up to which coefficients should be returned. If `NULL` (the default), it is set accordingly to argument `max.quantile`.
`max.quantile`	Numerical value indicating the order of the quantile lag up to which coefficients should be returned. Default is 0.999 (99.9th percentile).

Value

A list with one component for each explanatory variable with lags. Each component is an object of class `data.frame` containing estimation, asymptotic standard error and confidence interval for each lag up to `max.lag`.

See Also

gammadlm; plot.gammadlm.

Examples

```
data(btc)
mydata <- btc[which(btc$Date>="2020-04-01"),]
mydataLR <- preProcess(c("BTC","DJA","IXIC","GSPC"),
  time="Date", data=mydata, box.cox=0, ndiff=1)

dval <- list(DJA=0.85,IXIC=0.75,GSPC=0.55)
lval <- list(DJA=0.05, IXIC=0.35,GSPC=0.45)
m0 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(delta.lim=dval, lambda.lim=lval))

lagCoef(m0)  ## coefficients shown up to the 99.9th percentile lag
lagCoef(m0, max.lag=11)  ## coefficients shown up to lag 11
lagCoef(m0, cumulative=TRUE)  ## cumulative coefficients
```

`plot.gammadlm` *Graphics for the estimated lag distributions*

Description

Display the estimated lag distribution of each explanatory variable with lags.

Usage

```
## S3 method for class 'gammadlm'
plot(x, x.names=NULL, conf=0.95, max.lag=NULL, max.quantile=0.999,
  xlim=NULL, ylim=NULL, add.legend=TRUE, cex.legend=1, digits=4, grid.length=100,
  main=NULL, ylab=NULL, xlab=NULL, ...)
```

Arguments

`x`	Object of class `gammadlm`.
`x.names`	Character vector including the name of the variables for which the lag distribution should be displayed. Unknown variables will be ignored. If `NULL` (the default), the lag distribution of all the variables with lags will be displayed.
`conf`	Numerical value indicating the level of confidence bands. Default is 0.95. If `NULL`, confidence bands will not be displayed.
`max.lag`	Non-negative integer value indicating the lag up to which each lag distribution should be displayed. If `NULL` (the default), it is set accordingly to argument `max.quantile`.
`max.quantile`	Numerical value indicating the order of the quantile up to which each lag distribution should be displayed. Default is 0.999 (99.9th percentile).
`xlim`	Numerical vector of length 2 indicating the range of the x-axis, which is applied to all graphics (optional).
`ylim`	Numerical vector of length 2 indicating the range of the y-axis, which is applied to all graphics (optional).
`add.legend`	Logical value indicating whether a legend with numerical information should be added to the graphics. Default is `TRUE`.
`cex.legend`	Size of the legend. Default is 1.
`digits`	Integer non-negative value indicating the number of decimal places to be used in the legend. Default is 4. Ignored if `add.legend=FALSE`.
`grid.length`	Numerical value no less than 100 indicating the resolution of the interpolation. Default is 100.
`main`	Vector of characters including the title for each graphic. If `NULL` (the default), the name of the explanatory variables is used.
`ylab`	Text for y-axis, which is applied to all graphics (optional).
`xlab`	Text for x-axis, which is applied to all graphics (optional).
`...`	Further graphical parameters.

See Also

gammadlm.

Examples

```
data(btc)
mydata <- btc[which(btc$Date>="2020-04-01"),]
mydataLR <- preProcess(c("BTC","DJA","IXIC","GSPC"),
  time="Date", data=mydata, box.cox=0, ndiff=1)

dval <- list(DJA=0.85,IXIC=0.75,GSPC=0.55)
lval <- list(DJA=0.05, IXIC=0.35,GSPC=0.45)
m0 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(delta.lim=dval, lambda.lim=lval))

plot(m0)  ## all the lag distributions
plot(m0, x.names=c("DJA","IXIC"))  ## just the ones of 'DJA' and 'IXIC'
```

`preProcess` *Preprocessing of time series data*

Description

Application of the Box-Cox transformation and/or differencing to a multivariate time series, eventually structured as a balanced or unbalanced panel.

Usage

```
preProcess(var.names, unit=NULL, time=NULL, data, box.cox=1, ndiff=0,
  imputation=TRUE, em.control=list(nlags=NULL, tol=1e-4, maxit=1000, quiet=FALSE))
```

Arguments

`var.names`	Character vector including the name of the variables to be differenced.
`unit`	Character containing the name of the units of observation in case of a panel structure. If a vector with length greater than 1 is provided, only the first element is considered. If `NULL` (the default), a single unit of observation is assumed.
`time`	Character containing the name of the time variable, which must be in numeric or date format. If a vector with length greater than 1 is provided, only the first element is considered. If `NULL` (the default), data are assumed to be temporally ordered.
`data`	Object of class `data.frame` containing the variables in `var.names`, `unit` and `time`.
`box.cox`	Named vector with non-negative real values indicating the parameters of the Box-Cox transformation (Box & Cox, 1964) for variables in `x.names`. If `box.cox` has no names and length greater than one, the same ordering as in `x.names` is assumed. If `box.cox` has no names and length equal to one, the same parameter is used for all variables in `x.names`. Value 0 of the parameter equates to the logarithmic transformation, while value 1 means no transformation. Default is 1 for all variables in `x.names`.
`ndiff`	Named vector with non-negative integer values indicating the number of differences for variables in `x.names`. If `ndiff` has no names and length greater than one, the same ordering as in `x.names` is assumed. If `ndiff` has no names and length equal to one, the same number of differences is used for all variables in `x.names`. Value 0 means no differencing. Default is 0 for all variables in `x.names`.
`imputation`	Logical value indicating whether imputation of missing values should be performed (see 'Details'). Default is `TRUE`.
`em.control`	List including control options for the EM algorithm (see 'Details') with the following components: • `nlags`: Non-negative integer value indicating the number of lags to consider. If `NULL` (the default), it is taken as `trunc((n-1)^(1/3))`, where `n` is the sample size, otherwise it is right-truncated at `trunc(n*2/3)`. • `tol`: Positive number indicating the tolerance. Default is 0.001. • `maxit`: Positive integer indicating the number of iterations. Default is 1000. • `quiet`: Logical value indicating whether prompt messages should be suppressed. Default is `FALSE`. Ignored if `imputation` is `FALSE`.

Details

Imputation of missing values is performed after the Box-Cox transformation (Box & Cox, 1964) and differencing. A vector autoregressive model is assumed where the expectation of missing values is computed through the Expectation-Maximization (EM, Dempster et al., 1977) algorithm.

Value

The object provided to argument `data` where variables in `var.names` have been transformed and/or differenced, and the following attributes are added:

- `box.cox`: named vector indicating the parameter of the Box-Cox transformation for each variable in `x.names`;

- `ndiff`: named vector indicating the number of differences for each variable in `x.names`.

Note

The first order difference of logarithmic values (`box.cox`=0 and `ndiff`=1) provides the log returns, which approximate the proportional changes with respect to the previous time point.

If a variable contains negative values, the Box-Cox transformation will be not applied and a warning is returned.

If the number of differencing exceeds `n-5`, where `n` is the sample size, differencing will be not applied and a warning is returned.

References

G. E. P. Box, and D. R. Cox (1964). An analysis of transformations. *Journal of the Royal Statistical Society*, Series B (Methodological), 26(2): 211-252. DOI: 10.1111/j.2517-6161.1964.tb00553.x

A. P. Dempster, N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*n Series B (Methodological), 39(1): 1-38. DOI: 10.1111/j.2517-6161.1977.tb01600.x

See Also

unirootTest.

Examples

```
data(btc)
mydata <- btc[which(btc$Date>="2020-04-01"),]

# first order differencing ('box.cox'=1 by default)
mydataD <- preProcess(var.names=c("BTC","DJA","IXIC","GSPC"), time="Date",
  data=mydata, ndiff=1)
summary(mydataD)
plot(ts(mydataD[,c("BTC","DJA","IXIC","GSPC")]), main="")

# setting box.cox=0 and ndiff=1 produces the log returns
mydataLR <- preProcess(var.names=c("BTC","DJA","IXIC","GSPC"), time="Date",
  data=mydata, box.cox=0, ndiff=1)
summary(mydataLR)
plot(ts(mydataLR[,c("BTC","DJA","IXIC","GSPC")]), main="")
```

`unirootTest` *Unit root tests*

Description

Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests for a multivariate time series, eventually structured as a balanced or unbalanced panel.

Usage

```
unirootTest(var.names, unit=NULL, time=NULL, data, box.cox=1, ndiff=0, max.lag=NULL)
```

Arguments

`var.names`	Character vector including the name of the variables to be differenced.
`unit`	Character containing the name of the units of observation in case of a panel structure. If a vector with length greater than 1 is provided, only the first element is considered. If `NULL` (the default), a single unit of observation is assumed.
`time`	Character containing the name of the time variable, which must be in numeric or date format. If a vector with length greater than 1 is provided, only the first element is considered. If `NULL` (the default), data are assumed to be temporally ordered.
`data`	Object of class `data.frame` containing the variables in `var.names`, `unit` and `time`.
`box.cox`	Named vector with non-negative real values indicating the parameters of the Box-Cox transformation (Box & Cox, 1964) for variables in `x.names`. If `box.cox` has no names and length greater than one, the same ordering as in `x.names` is assumed. If `box.cox` has no names and length equal to one, the same parameter is used for all variables in `x.names`. Value 0 of the parameter equates to the logarithmic transformation, while value 1 means no transformation. Default is 1 for all variables in `x.names`.
`ndiff`	Named vector with non-negative integer values indicating the number of differences for variables in `x.names`. If `ndiff` has no names and length greater than one, the same ordering as in `x.names` is assumed. If `ndiff` has no names and length equal to one, the same number of differences is used for all variables in `x.names`. Value 0 means no differencing. Default is 0 for all variables in `x.names`.
`max.lag`	Non-negative integer value representing the maximum lag length at which to perform the tests (see 'Details'). If `NULL` (the default), it is taken as the squared root of the length of the time series. .

Details

The variables subjected to the tests must be quantitative. Missing values internal to the time series are imputed through linear interpolation, otherwise they are deleted out.

The lag length at which to perform the tests is selected through AIC-based backward elimination starting from the lag length specified in argument `max.lag`.

The null hypothesis of the ADF test is the presence of unit roots, while the null hypothesis of the KPSS test is the absence of unit roots. Therefore, p-value higher than 0.05 for the ADF test or p-value lower than 0.05 for the KPSS test suggest the presence of unit roots and the need of further differencing.

In case of a panel structure, p-values are combined according to the method by Demetrescu *et al.* (2006).

Value

One list for each variable in `var.names`, each with three components:

- `statistic`: test statistic for each test;
- `lag.selected`: lag length selected for each test;
- `p.value`: p-value of each test, which is a single value if `unit` is `NULL`, otherwise one value for each unit of observation plus another one indicating the combined p-value;
- `box.cox`: parameter of the Box-Cox transformation for each variable subjected to the tests;
- `ndiff`: order of differencing for each variable subjected to the tests.

Note

The first order difference of logarithmic values (`box.cox`=0 and `ndiff`=1) provides the log returns, which approximate the proportional changes with respect to the previous time point.

If a variable contains negative values, the Box-Cox transformation will be not applied and a warning is returned.

If the number of differencing exceeds `n-5`, where `n` is the sample size, differencing will be not applied and a warning is returned.

References

M. Demetrescu, U. Hassler, and A. Tarcolea (2006). Combining significance of correlated statistics with application to panel data. *Oxford Bulletin of Economics and Statistics*, 68(5), 647-663. DOI: 10.1111/j.1468-0084.2006.00181.x

D. A. Dickey, and W. A. Fuller (1981). Likelihood ratio statistics for autoregressive time series with a unit root. *Econometrica*, 49(4): 1057-1072. DOI: 10.2307/1912517

D. Kwiatkowski, P. C. B. Phillips, P. Schmidt, and Y. Shin (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54(1-3): 159-178. DOI: 10.1016/0304-4076(92)90104-Y

See Also

preProcess.

Examples

```
data(btc)
mydata <- btc[which(btc$Date>="2020-04-01"),]

# tests on time series in level ('box.cox'=1 by default):
#   there are some unit roots
unirootTest(c("BTC","DJA","IXIC","GSPC"), time="Date",
  data=mydata)
```

```
# tests on time series in log return: ok
unirootTest(c("BTC","DJA","IXIC","GSPC"), time="Date",
  data=mydata, box.cox=0, ndiff=1)
```

Index