Package ‘gammadlm’

November 23, 2020

Type Package

Title The Gamma Distributed-Lag Model with Multiple Explanatory Variables

Version 0.0

Date 2020-11-17

Author Alessandro Magrini

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

Description Maximum likelihood estimation and inference for the Gamma distributed-lag model with multiple explanatory variables.

Depends R (>= 3.5.0)

Imports graphics, stats

License GPL-2

NeedsCompilation no

R topics documented:

---

gammadlm-package *The Gamma Distributed-Lag Model with Multiple Explanatory Variables*

Description

Maximum likelihood estimation and inference for the Gamma distributed-lag model with multiple explanatory variables.

Details

Package:	gammadlm
Type:	Package
Version:	0.0
Date:	2020-11-17
License:	GPL-2

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

$$y_t = \alpha + \sum_{j=1}^{J} \theta_j \sum_{k=0}^{\infty} w_{j,k} x_{j,t-k} + \varepsilon_t$$

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

where:

- α is the intercept;
- θ_j is the scale parameter of the lag distribution of X_j;
- $0 \leq \delta_j < 1$ and $0 \leq \lambda_j < 1$ are the shape parameters of the lag distribution of X_j;
- η_j is the offset of the lag distribution of X_j, typically set to 0;
- ε_t is the random error at time t.

The main functions of the package are:

- gammadlm, to estimate the model through the hill climbing algorithm;
- lagCoef, to obtain the estimates of dynamic coefficients.

Also, method lagCoef can be used to obtain dynamic coefficients, method `plot` to obtain graphics for the estimated lag distributions, and method `residuals` to obtain graphical model diagnostics.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

References

A. Magrini. A hill climbing algorithm for maximum likelihood estimation of the Gamma distributed-lag model with multiple explanatory variables. *Under review*.

J. M. Alston, M. A. Andersen, J. S. James, P. G. Pardey (2011). The economic returns to U.S. public agricultural research. *American Journal of Agricultural Economics*, 93(5), 1257-1277.

`adfTest` *Augmented Dickey-Fuller test*

Description

Augmented Dickey-Fuller test with automated selection of the lag length.

Usage

```
adfTest(x, max.lag=NULL)
```

Arguments

`x`	Numerical vector containing temporally ordered data.
`max.lag`	Non-negative integer representing the maximum lag length at which the test should be performed. If `NULL` (the default), it is taken as `trunc((length(x)-1)^(1/3))`

Details

The variable subjected to the test must be quantitative.

The test is performed starting from a specified lag length, which is sequentially decreased until the test statistic is lower than 1.6 in absolute value or the lag length is 0.

The null hypothesis is the presence of unit root, thus p-values higher than 0.05 indicate non-stationarity and the need of differencing the time series.

Missing values, if present, are deleted out and a warning is returned.

Value

A list with three components:

- `statistic`: the test statistic;
- `lag.selected`: the lag length selected to perform the test;
- `p.value`: the p-value of the test.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

References

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

See Also

tsDiff.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
adfTest(mydata$BTC)  ## unit root
adfTest(diff(log(mydata$BTC)))  ## data in log return: ok
```

---

gammadlm *Estimation of a Gamma distributed-lag model*

Description

Maximum likelihood estimation of a Gamma distributed-lag model with multiple explanatory variables using hill climbing algorithm.

Usage

```
gammadlm(y.name, x.names, z.names=NULL, time.name=NULL, data,
  offset=rep(0,length(x.names)), control=list(nstart=50, grid.by=0.05,
  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.

`time.name` Character including 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` and `time.name` cannot contain missing values.

`offset` Numerical vector indicating, in order, the offset for each variable in `x.names`. Default is 0. If the name of the response variable is indicated in `x.names` and its offset is less than 1, then it will be set automatically to 1.

`control` A list including control options for the hill climbing algorithm.

- `nstart`: positive integer value indicating the number of restarts. Default is 50.
- `grid.by`: positive value no greater than 0.1, indicating the increment in grid search. Default is 0.05.

- `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 displayed during the execution. Default is `TRUE`.

Details

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

$$y_t = \alpha + \sum_{j=1}^{J} \theta_j \sum_{k=0}^{\infty} w_{j,k} x_{j,t-k} + \varepsilon_t$$

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

where:

- α is the intercept;
- θ_j is the scale parameter of the lag distribution of X_j;
- $0 \leq \delta_j < 1$ and $0 \leq \lambda_j < 1$ are the shape parameters of the lag distribution of X_j;
- η_j is the offset of the lag distribution of X_j, typically set to 0;
- ε_t is the random error at time t.

All S3 methods for class `lm` are also available for class `gammadlm`. Furthermore, method lagCoef can be used to obtain dynamic coefficients, method `plot` to obtain graphics for the estimated lag distributions, and method `residuals` to obtain graphical model diagnostics.

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;
- `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` and `z.names`.
- `data`: data.frame including the data used for parameter estimation.

Note

Stationarity of the time series is a basic assumption of the model. Function adfTest performs the Augmented Dickey-Fuller test and can be used to test the presence of a unit root in each time series. Stationarity is stated in the case of a significant result of the test, otherwise differencing is required to achieve stationarity. Function tsDiff can be used to apply differencing to non-stationary time series.

Function gammadlm checks stationarity of the residuals and returns a warning in case of non-stationarity. However, the user is strongly recommended to check stationarity of the time series autonomously before running function gammadlm.

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 Heteroscedasticity 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.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

References

A. Magrini. A hill climbing algorithm for maximum likelihood estimation of the Gamma distributed-lag model with multiple explanatory variables. *Under review*.

J. M. Alston, M. A. Andersen, J. S. James, P. G. Pardey (2011). The economic returns to U.S. public agricultural research. *American Journal of Agricultural Economics*, 93(5), 1257-1277.

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

lagCoef; plot.gammadlm; residuals.gammadlm.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
mydataLR <- tsDiff(time.name="Date", data=mydata, ndiff=1, log=TRUE)

## estimation with fixed values of delta and lambda parameters: 1-step OLS
dval <- list(DJA=0.85,IXIC=0.75,GSPC=0.55)
lval <- list(DJA=0.5, IXIC=0.35,GSPC=0.45)
mod <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(delta.lim=dval, lambda.lim=lval))
summary(mod)  ## summary of estimation

## estimation through hill climbing algorithm: NOT RUN
##
## * no constraints with 50 random restarts (by default)
#set.seed(100)
#m1 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR)
#summary(m1)
##
## * no contraints with 100 random restarts
```

```
#set.seed(100)
#m1a <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
#  control=list(nstart=100))
#summary(m1a)
##
## * constraints: peak>=1 and 3<=length<=10
#pklim <- list(DJA=c(1,Inf),IXIC=c(1,Inf),GSPC=c(1,Inf))
#lenlim <- list(DJA=c(3,10),IXIC=c(3,10),GSPC=c(3,10))
#set.seed(100)
#m2 <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
#  control=list(peak.lim=pklim, length.lim=lenlim, nstart=100))
#summary(m2)
```

---

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, 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.
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))
```

`lagCoef` *Estimated dynamic coefficients*

Description

Estimated dynamic coefficients for each explanatory variable.

Usage

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

Arguments

`x`	An object of class `gammadlm`.
`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` with lags as observations and two columns containing estimation and asymptotic standard error.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

See Also

gammadlm; plot.gammadlm.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
mydataLR <- tsDiff(time.name="Date", data=mydata, ndiff=1, log=TRUE)

dval <- list(DJA=0.85,IXIC=0.75,GSPC=0.55)
lval <- list(DJA=0.5, IXIC=0.35,GSPC=0.45)
mod <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(delta.lim=dval, lambda.lim=lval))

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

---

plot.gammadlm *Graphics for the estimated lag distributions*

Description

Obtain the graphic for 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. If `NULL` (the default), the lag distribution of all the variables with lags will be displayed.
`conf`	Numerical value indicating the confidence level. Default is 0.95.
`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.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

See Also

gammadlm.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
mydataLR <- tsDiff(time.name="Date", data=mydata, ndiff=1, log=TRUE)

dval <- list(DJA=0.85,IXIC=0.75,GSPC=0.55)
lval <- list(DJA=0.5, IXIC=0.35,GSPC=0.45)
mod <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(delta.lim=dval, lambda.lim=lval))

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

---

residuals.gammadlm *Residuals and graphical model diagnostics*

Description

Obtain residuals and graphical diagnostics for an estimated Gamma distributed-lag model.

Usage

```
## S3 method for class 'gammadlm'
residuals(object, plot=FALSE, cex.lab=1, cex.axis=1, ...)
```

Arguments

object	Object of class gammadlm.
plot	Logical value indicating whether the graphical diagnostics should be produced.
cex.lab	Size of the axis text labels.
cex.axis	Size of the axis tick labels.
...	Further graphical parameters.

Value

Residuals extracted from `object`, or `NULL` if argument `plot` is set to `TRUE`.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

See Also

gammadlm.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
mydataLR <- tsDiff(time.name="Date", data=mydata, ndiff=1, log=TRUE)

dval <- list(DJA=0.85,IXIC=0.75,GSPC=0.55)
lval <- list(DJA=0.5, IXIC=0.35,GSPC=0.45)
mod <- gammadlm(y.name="BTC", x.names=c("DJA","IXIC","GSPC"), data=mydataLR,
  control=list(delta.lim=dval, lambda.lim=lval))

residuals(mod)  ## returns the residuals
residuals(mod, plot=TRUE)  ## displays graphical model diagnostics
```

---

tsDiff — *Time series differencing*

Description

Application of differencing to several time series.

Usage

```
tsDiff(var.names=NULL, time.name=NULL, data, ndiff=0, log=F)
```

Arguments

`var.names`	Character vector including the name of the variables to be differenced. If `NULL` (the default), all the quantitative variables in the dataset provided to argument `data` will be differenced, with the exception of the variable indicated in `time.name`.
`time.name`	Character including 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` and `time.name`.
`ndiff`	Non-negative integer value indicating the order of differencing for all variables in `var.names`, or a vector of non-negative integer values indicating the order of differencing for each variable in `var.names`. In the second case, the length of `ndiff` must be the same of `var.names`. Default is 0, meaning no differencing.

`log` Logical value indicating whether the logarithmic transformation should be applied to all variables in `var.names`, or a vector of logical values indicating whether the logarithmic transformation should be applied to each variable in `var.names`. In the second case, the length of `log` must be the same of `var.names`. Default is `FALSE`, meaning no logarithmic transformation.

Value

The object provided to argument `data` where variables in `var.names` have been differenced and/or log transformed.

Author(s)

Alessandro Magrini <alessandro.magrini@unifi.it>

See Also

adfTest.

Examples

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

# setting ndiff=0 and log=T produces the log return
mydataLR <- tsDiff(var.names=c("BTC","DJA","IXIC","GSPC"), time.name="Date",
  data=mydata, ndiff=1, log=TRUE)
summary(mydataLR)

# same result by omitting 'var.names'
mydataLR2 <- tsDiff(time.name="Date", data=mydata, ndiff=1, log=TRUE)
summary(mydataLR2)
```

USstock *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(USstock)
```

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.

Index