Package 'gammadlm'

December 11, 2021

Type Package

Title The Gamma Distributed-Lag Model with Multiple Explanatory Variables

Version 0.1.1

Date 2021-12-11

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. A panel structure can be taken into account.

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. A panel structure can be taken into account.

Details

Package:	gammadlm
Type:	Package
Version:	0.1.1
Date:	2021-12-11
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 ($j = 1, \ldots, J$) observed at time t. Under the assumption that the time series of Y and of $X_1, \ldots, X_J$ are all weakly stationary (i.e., expected value and autocorrelation function independent of time), the Gamma distributed-lag model explaining Y from $X_1, \ldots, X_J$ 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 $\{w_{j,k} : k = 0, 1, \ldots, \infty\}$ includes the weights for X_j and is defined by the shape parameters $0 \leq \delta_j < 1$ and $0 \leq \lambda_j < 1$ and the offset η_j (typically set to 0). The set $\{\beta_{j,k} : k = 0, 1, \ldots, \infty\}$ 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 ($j = 1, \ldots, J$) 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:

- adfTest and tsDiff, to check weak stationarity and to apply differencing in order to achieve it;
- 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.

`adfTest` *Augmented Dickey-Fuller test*

Description

Augmented Dickey-Fuller test with automated selection of the lag length for a multivariate time series, eventually structured as a panel.

Usage

```
adfTest(var.names=NULL, 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. 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`.
`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 indicating the parameter of the Box-Cox transformation for variables in `x.names`. If `box.cox` is of length one and has no names, the same parameter is used for all variables in `x.names`. If a variable in `x.names` has no name in `box.cox`, value 1 is assumed, meaning no transformation. Value 0 of the parameter equates to the logarithmic transformation. Default value of `box.cox` is 1, meaning no transformation 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` is of length one and has no names, the same number of differences is used for all variables in `x.names`. If a variable in `x.names` has no name in `ndiff`, value 0 is assumed, meaning no differencing. Default value of `ndiff` is 0, meaning no differencing for all variables in `x.names`.
`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 the squared root of the length of the time series.

.

Details

The variable subjected to the test must be quantitative.

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

The lag length at which the test is performed is selected through backward elimination starting from the lag length specified in argument `max.lag`.

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

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

Value

One list for each variable in `var.names`, each 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, which is a single value if `unit` is `NULL` and a vector of values otherwise, one for each unit of observation plus another one indicating the combined p-value;
- `box.cox`: parameter of the Box-Cox transformation;
- `ndiff`: order of differencing.

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 the parameter of the Box-Cox transformation is set to 0 for a non-negative variable, the transformation will not be applied. i.e., the parameter is set to 1, 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.

D. A. Dickey, and 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"),]

# adf test for time series in level ('box.cox'=1 by default):
#   there are some unit roots
adfTest(var.names=c("BTC","DJA","IXIC","GSPC"), time="Date",
  data=mydata)

# adf test for time series in log return: ok
adfTest(var.names=c("BTC","DJA","IXIC","GSPC"), time="Date",
  data=mydata, box.cox=0, ndiff=1)
```

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. A panel structure can be taken into account.

Usage

```
gammadlm(y.name, x.names, z.names=NULL, unit=NULL, time=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.
`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`	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. If equal to 1, then all the shape parameters are initialized to value 0. 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

All S3 methods for class `lm` are also available for class `gammadlm`. Furthermore, method lagCoef can be used to see the estimated dynamic coefficients, and method `plot` to display the estimated lag distributions.

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`.
- `unit.id`: list including the row names of the observations for each unit.
- `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. However, it is sufficient that all time series do not contain unit roots to get valid results (Granger \& Newbold, 1974). Before calling the function gammadlm, the user is strongly recommended to check the absence of unit roots in each time series through the function adfTest, which performs the Augmented Dickey-Fuller test (Dickey \& Fuller, 1981), and to apply differencing through the function tsDiff to each time series for which the test is not significant.

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

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

C. W. J. Granger, and P. Newbold (1974). Spurious regressions in econometrics. *Journal of Econometrics*, 2(2): 111-120.

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.

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

adfTest; lagCoef; plot.gammadlm.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
mydataLR <- tsDiff(time="Date", data=mydata, box.cox=0, ndiff=1)

## 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 the 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, 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, 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.

See Also

gammadlm; plot.gammadlm.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
mydataLR <- tsDiff(time="Date", data=mydata, box.cox=0, ndiff=1)

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

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. 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.

See Also

gammadlm.

Examples

```
data(USstock)
mydata <- USstock[which(USstock$Date>="2020-04-01"),]
mydataLR <- tsDiff(time="Date", data=mydata, box.cox=0, ndiff=1)

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'
```

`tsDiff` *Time series differencing*

Description

Application of differencing to a multivariate time series, eventually structured as a panel.

Usage

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

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`.
`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 indicating the parameter of the Box-Cox transformation for variables in `x.names`. If `box.cox` is of length one and has no names, the same parameter is used for all variables in `x.names`. If a variable in `x.names` has no name in `box.cox`, value 1 is assumed, meaning no transformation. Value 0 of the parameter equates to the logarithmic transformation. Default value of `box.cox` is 1, meaning no transformation 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` is of length one and has no names, the same number of differences is used for all variables in `x.names`. If a variable in `x.names` has no name in `ndiff`, value 0 is assumed, meaning no differencing. Default value of `ndiff` is 0, meaning no differencing for all variables in `x.names`.

Value

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

- `inits`: if `unit=NULL`, named vector indicating the first observed value for each variable in `x.names`, otherwise a named matrix indicating the first observed value for each unit of observation (by row) and each variable in `x.names` (by column);

- `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 the parameter of the Box-Cox transformation is set to 0 for a non-negative variable, the transformation will not be applied (parameter equal to 1) and a warning is returned.

See Also

adfTest.

Examples

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

# first order differencing ('box.cox'=1 by default)
mydataD <- tsDiff(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 <- tsDiff(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="")

# same result by omitting 'var.names'
mydataLR2 <- tsDiff(time="Date", data=mydata, box.cox=0, ndiff=1)
summary(mydataLR2)
plot(ts(mydataLR2[,c("BTC","DJA","IXIC","GSPC")]), main="")
```

`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