Package 'dLagM'

October 22, 2018

Type Package

Title Time Series Regression Models with Distributed Lag Models

Version 1.0.10

Date 2018-10-22

Author Haydar Demirhan [aut, cre, cph] (<https://orcid.org/0000-0002-8565-4710>)

Maintainer Haydar Demirhan <haydar.demirhan@rmit.edu.au>

Description Provides time series regression models with one predictor using finite distributed lag models, polynomial (Almon) distributed lag models, geometric distributed lag models with Koyck transformation, and autoregressive distributed lag models. It also consists of functions for computation of h-step ahead forecasts from these models. See Baltagi (2011) <doi:10.1007/978-3-642-20059-5> for more information.

Depends stats, dynlm, wavethresh, AER, formula.tools, plyr

License GPL-3

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-22 11:00:08 UTC

R topics documented:

dLagM-package 2
ardlDlm 3
dlm 5
finiteDLMauto 8
forecast 10
koyckDlm 12
MASE 13
polyDlm 15
sortScore 17
warming 18

Index **19**

dLagM-package *Implementation of Time Series Regression Models with Distributed Lag Models*

Description

Provides time series regression models with one predictor using finite distributed lag models, polynomial (Almon) distributed lag models, geometric distributed lag models with Koyck transformation, and autoregressive distributed lag models. It also consists of functions for computation of h-step ahead forecasts from these models. See Baltagi (2011) for more information.

Details

Package:	dLagM
Type:	Package
Version:	1.0.10
Date:	2018-10-22
License:	GPL-3

To implement time series regression with finite distributed lag models, use `dlm` function.

To implement time series regression with polynomial distributed lag models, use `polyDlm` function.

To implement time series regression with geometric distributed lag models with Koyck transformation, use `koyckDlm` function.

To implement time series regression with autoregressive distributed lag models, use `ardlDlm` function.

To produce forecasts for any of the models, use `forecast` function.

To summarise the results of a model fitting, use `summary` function.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

References

B.H. Baltagi. *Econometrics*, Fifth Ed. Springer, 2011.

R.C. Hill, W.E. Griffiths, G.G. Judge. *Undergraduate Econometrics*. Wiley, 2000.

See Also

`dlm`, `polyDlm`, `koyckDlm`, `ardlDlm`

Examples

```
# --- For examples, please refer to specific functions ---
```

`ardlDlm` *Implement finite autoregressive distributed lag model*

Description

A function that applies autoregressive distributed lag models of order (p , q) with one predictor.

Usage

```
ardlDlm(formula = NULL , data = NULL , x = NULL , y = NULL , p = 1 , q = 1 ,
         remove = NULL )
```

Arguments

`formula`	A `formula` object for the model to be fitted. In the case of multiple predictor series, the model should be entered via a formula object.
`data`	A `data.frame` including all dependent and independent series. In the case of multiple predictor series, the data should be entered via the `data` argument.
`x`	A vector including the observations of predictor time series. This is not restricted to `ts` objects.
`y`	A vector including the observations of dependent time series. This is not restricted to `ts` objects.
`p`	An integer representing finite lag length.
`q`	An integer representing the order of autoregressive process.
`remove`	A list object having two elements showing the lags of independent series with `p` and the autoregressive lags with `q` to be removed from the full model for each independent series. Please see the details for the construction of this argument.

Details

The autoregressive DLM is a flexible and parsimonious infinite distributed lag model. The model ARDL(p, q) is written as

$$Y_t = \mu + \beta_0 X_t + \beta_1 X_{t-1} + \cdots + \beta_p X_{t-p} + \gamma_1 Y_{t-1} + \cdots + \gamma_q Y_{t-q} + e_t.$$

When there is only one predictor series, both of `model` and `formula` objects can be used. But when they are supplied, both `x` and `y` arguments should be `NULL`.

The variable names in `formula` must match with the names of variables in `data` argument and it must be in the form of a generic formula for R functions.

The argument `data` contains dependent series and independent series.

The argument `remove = list(p = list() , q = c())` is used to specify which lags of each independent series and the autoregressive lags of dependent series will be removed from the full model. Each element of the list `p` shows particular lags that will be removed from each independent series. To remove the main series from the model or to fit a model ARDL(0,q), include `0` within the elements of `p`. The element `q` is just a vector showing the autoregressive lags of dependent series to be removed. See the examples below for implementation details.

The standard function `summary()` prints model summary for the model of interest.

Value

`model`	An object of class `lm`. See the details of `lm` function.
`order`	A vector composed of p and q orders.
`removed.p`	A list or vector showing the lags of independent series to be removed from the full model.
`removed.q`	A vector showing the autoregressive lags to be removed from the full model.
`formula`	Model formula of the fitted model. This is returned if multiple independent series are entered.
`data`	A `data.frame` including all dependent and independent series. This is returned if multiple independent series are entered.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

References

B.H. Baltagi. *Econometrics*, Fifth Ed. Springer, 2011.

R.C. Hill, W.E. Griffiths, G.G. Judge. *Undergraduate Econometrics*. Wiley, 2000.

Examples

```
# Only one independent series
data(warming)
model.ardl = ardlDlm(x = warming$NoMotorVehicles,
                     y = warming$Warming, p = 1 , q = 1 )
summary(model.ardl)

# Remove some lags
# Remove some lags
rem.p = c(0,1) # 0 removes the main effect of X.t
rem.q = c(1,3)
remove = list(p = rem.p , q = rem.q)
model.ardl = ardlDlm(x = warming$NoMotorVehicles,
                     y = warming$Warming, p = 2 , q = 3 , remove = remove)
summary(model.ardl)

# Multiple independent series
```

```
data(M1Germany)
data = M1Germany[1:144,]
model.ardlDlm  = ardlDlm(formula = logprice ~ interest + logm1,
                         data = data.frame(data) , p = 2 , q = 1 )
summary(model.ardlDlm)

rem.p = list(interest = c(0,2) , logm1 = c(0))
# Remove the main series of interest and logm1 and the secont lag of
# interest from the model
rem.q = c(1)
remove = list(p = rem.p , q = rem.q)
remove
model.ardlDlm  = ardlDlm(formula = logprice ~ interest + logm1,
                         data = data.frame(data) , p = 2 , q = 2 ,
                         remove = remove)
summary(model.ardlDlm)
```

`dlm` *Implement finite distributed lag model*

Description

A function that applies distributed lag models with one or multiple predictor(s).

Usage

```
dlm(formula , data , x , y , q , remove )
```

Arguments

`formula`	A `formula` object for the model to be fitted. In the case of multiple predictor series, the model should be entered via a formula object.
`data`	A `data.frame` including all dependent and independent series. In the case of multiple predictor series, the data should be entered via the `data` argument.
`x`	A vector including the observations of predictor time series. This is not restricted to `ts` objects. If the series are supplied by `data`
`y`	A vector including the observations of dependent time series. This is not restricted to `ts` objects.
`q`	An integer representing finite lag length.
`remove`	A list object showing the lags to be removed from the model for each independent series in its elements. Please see the details for the construction of this argument.

Details

When a decision made on a variable, some of the related variables would be effected through time. For example, when income tax rate is increased, this would reduce expenditures of consumers on goods and services, which reduces profits of suppliers, which reduces the demand for productive inputs, which reduces the profits of the input suppliers, and so on (Judge and Griffiths, 2000). These effects occur over the future time periods; hence, they are distributed across the time.

In a distributed-lag model, the effect of an independent variable X on a dependent variable Y occurs over the time. Therefore, DLMs are dynamic models. A linear finite DLM with one independent variable is written as follows:

$$Y_t = \alpha + \sum_{s=0}^{q} \beta_s X_{t-s} + \epsilon_t,$$

where ϵ_t is a stationary error term with $E(\epsilon_t) = 0, Var(\epsilon_t) = \sigma^2, Cov(\epsilon_t, \epsilon_s) = 0$.

When there is only one predictor series, both of `model` and `formula` objects can be used. But when they are supplied, both `x` and `y` arguments should be `NULL`.

The variable names in `formula` must match with the names of variables in `data` argument and it must be in the form of a generic formula for R functions.

The argument `data` contains dependent series and independent series. Required lags of dependent series are generated by the `dlm` function automatically.

The argument `remove = list()` is used to specify which lags will be removed from the full model. Each element of the list `remove` shows particular lags that will be removed from each independent series. Notice that it is possible to fit a model with different lag lengths for each independent series by removing the appropriate lags of independent series with `remove` argument. To remove the main series from the model include `0` within the elements of `remove`.

The standard function `summary()` prints model summary for the model of interest.

Value

`model`	An object of class `lm`.
`designMatrix`	The design matrix composed of transformed z-variables.
`k`	The number of independent series. This is returned if multiple independent series are entered.
`q`	The lag length.
`removed`	A list or vector showing the removed lags from the model for independent series. Returns `NULL` if the fitted model is full.
`formula`	Model formula of the fitted model. This is returned if multiple independent series are entered.
`data`	A `data.frame` including all dependent and independent series. This is returned if multiple independent series are entered.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

References

B.H. Baltagi. *Econometrics*, Fifth Ed. Springer, 2011.

R.C. Hill, W.E. Griffiths, G.G. Judge. *Undergraduate Econometrics*. Wiley, 2000.

Examples

```
# Only one independent series
data(warming)
model.dlm = dlm(x = warming$NoMotorVehicles ,
                y = warming$Warming , q = 2)
summary(model.dlm)

removed = list(x = c(1,3))
model.dlm = dlm(x = warming$NoMotorVehicles ,
                y = warming$Warming , q = 5,
                remove = removed)
summary(model.dlm)

removed = list(x = c(0,1,3))
model.dlm = dlm(x = warming$NoMotorVehicles ,
                y = warming$Warming , q = 5,
                remove = removed)
summary(model.dlm)

model.dlm = dlm(formula = Warming ~ NoMotorVehicles ,
                data = warming , q = 2)
summary(model.dlm)

removed = list(x = c(0,3))
model.dlm = dlm(formula = Warming ~ NoMotorVehicles ,
                data = warming , q = 4,
                remove = removed)
summary(model.dlm)

# Multiple independent series
data(M1Germany)
data = M1Germany[1:144,]
model.dlm  = dlm(formula = logprice ~ interest + logm1,
                 data = data.frame(data) , q = 4)
summary(model.dlm)

removed = list(interest = c(1,3), logm1 = c(2))
removed
model.dlm  = dlm(formula = logprice ~ interest + logm1,
                 data = data.frame(data) , q = 4 , remove = removed)
summary(model.dlm)

removed = list(interest = c(0,1,3), logm1 = c(0,2))
removed
model.dlm  = dlm(formula = logprice ~ interest + logm1,
                 data = data.frame(data) , q = 4 , remove = removed)
summary(model.dlm)
```

```
removed = list( logm1 = c(1,2))
removed
model.dlm  = dlm(formula = logprice ~ interest + logm1,
                 data = data.frame(data) , q = 4 , remove = removed)
summary(model.dlm)
```

`finiteDLMauto` *Find the optimal lag length for finite DLMs*

Description

A function that fits finite DLMs for a range of lag lengths and orders the fitted models according to a desired measure.

Usage

```
finiteDLMauto(formula , data, x, y, q.min = 1, q.max = 10, k.order = NULL,
model.type = c("dlm","poly"), error.type = c("MASE","AIC","BIC","radj"),
trace = FALSE)
```

Arguments

`formula`	A `formula` object for the model to be fitted. In the case of multiple predictor series, the model should be entered via a formula object.
`data`	A `data.frame` including all dependent and independent series. In the case of multiple predictor series, the data should be entered via the `data` argument.
`x`	A vector including the observations of predictor time series. This is not restricted to `ts` objects.
`y`	A vector including the observations of dependent time series. This is not restricted to `ts` objects.
`q.min`	An integer representing the lower limit of the range of lag lengths to be considered. If missing, it will be set to 1.
`q.max`	An integer representing the upper limit of the range of lag lengths to be considered. If missing, it will be set to 10.
`k.order`	An integer representing order of polynomial distributed lags.
`model.type`	The type of model to be fitted. If set to `dlm`, finite distributed lag models are fitted. If set to `poly`, polynomial distributed lag models are fitted.
`error.type`	The type of goodness-of-fit measure to be used for the selection of optimal lag length. If set to `MASE`, the optimal lag length is determined according to MASE. If set to `AIC`, the optimal lag length is determined according to AIC. If set to `BIC`, the optimal lag length is determined according to BIC. If set to `radj`, the optimal lag length is determined according to Adjusted R-square.
`trace`	If `TRUE`, prints all of the goodness-of-fit measures for all fitted models.

Details

When there is only one predictor series, both of `model` and `formula` objects can be used. But when they are supplied, both `x` and `y` arguments should be `NULL`.

The variable names in `formula` must match with the names of variables in `data` argument and it must be in the form of a generic formula for R functions.

The argument `data` contains dependent series and independent series. Required lags of dependent series are generated by the `dlm` function automatically.

If `q.max` is entered greater than the length of the series, its value will be adjusted to have the length of the series for fitting the regression model.

Value

Returns a `data.frame` including the values of goodness-of-fit measures and corresponding lag lengths.

Author(s)

Agung Andiojaya <agung.andiojaya@gmail.com>, Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

Examples

```
library(dLagM)
# Only one independent series
data(warming)
# Run the search over polynomial DLMs according to MASE values
finiteDLMauto(x = warming$NoMotorVehicles , y = warming$Warming ,
              q.max = 11, k.order = 3, model.type = "poly",
              error.type = "MASE", trace = TRUE)

# Run the search over finite DLMs according to AIC values
finiteDLMauto(x = warming$NoMotorVehicles , y = warming$Warming ,
              q.min = 2, q.max = 8, model.type = "dlm", error.type = "AIC",
              trace = TRUE)

# Multiple independent series
data(M1Germany)
data = M1Germany[1:144,]
# Run the search over finite DLMs according to AIC values
finiteDLMauto(formula = logprice ~ interest + logm1,
              data = data.frame(data), q.min = 2, q.max = 14,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

`forecast` *Compute forecasts for distributed lag models*

Description

A function that computes forecasts for the finite distributed lag models, autoregressive distributed lag models, Koyck transformation of distributed lag models, and polynomial distributed lag models.

Usage

```
forecast(model , x , h = 1 , interval = FALSE, level = 0.95 , nSim = 500)
```

Arguments

`model`	An object of class `lm` including the fitted model with `ardl.dlm()` function.
`x`	A vector or matrix including the new observations of independent time series. This is not restricted to `ts` objects. Please see the details for construction of this argument.
`h`	The number of ahead forecasts.
`interval`	If `TRUE`, $(1-\alpha)\%$ prediction intervals for forecasts are displayed along with forecasts.
`level`	Confidence level of prediction interval.
`nSim`	An integer showing the number of Monte Carlo simulations used to compute prediction intervals for forecasts.

Details

This function directly uses the model formula and estimates of model coefficients to find forecast one-by-one starting from the one-step ahead forecast.

Prediction intervals are found by the Monte Carlo approach using a Gaussian error distribution with zero mean and empirical variance of the dependent series.

When the `model` argument includes multiple independent series, `x` must be entered as a matrix including the new observations of each independent series in its rows. The number of columns of `x` must be equal to the forecast horizon `h` and the rows of `x` must match with the independent series in the order they appear in the `data`.

This function can still be used when some of the lags of independent series are removed from the model.

Value

`forecasts`	A vector including forecasts.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

Examples

```
# Only one independent series
data(warming)
#--- ARDL dlm ---
model.ardl = ardlDlm(x = warming$NoMotorVehicles,
             y = warming$Warming, p = 1 , q = 1 )
forecast(model = model.ardl , x = c(95, 98) ,
                h = 2 , interval = FALSE)
forecast(model = model.ardl , x = c(95, 98, 87) ,
                h = 3 , interval = FALSE)

# Multiple independent series
data(M1Germany)
data = M1Germany[1:144,]
model.ardlDlm1  = ardlDlm(formula = logprice ~ interest + logm1,
       data = data.frame(data) , p = 2 , q = 1 )
x.new =  matrix(c(0.07 , 9.06 , 0.071 , 9.09), ncol = 2,
                nrow = 2)
forecast(model = model.ardlDlm1 , x = x.new , h = 2 ,
                interval = TRUE, nSim = 100)

rem.p = list(interest = c(1,2))
rem.q = c(1)
remove = list(p = rem.p , q = rem.q)
model.ardlDlm2  = ardlDlm(formula = logprice ~ interest + logm1,
       data = data.frame(data) , p = 2 , q = 2 ,
       remove = remove)

forecast(model = model.ardlDlm2 , x = x.new , h = 2 ,
                interval = FALSE)

#--- Finite dlm ---
model.dlm = dlm(x = warming$NoMotorVehicles ,
                y = warming$Warming , q = 2)
forecast(model = model.dlm , x = c(95 , 98, 101) , h = 3)


# Multiple independent series
model.dlm  = dlm(formula = logprice ~ interest + logm1,
          data = data.frame(data) , q = 4)

x.new =  matrix(c(0.07 , 9.06 , 0.071 , 9.09),
                ncol = 2, nrow = 2)
forecast(model = model.dlm , x = x.new , h = 2 ,
            interval = FALSE)

# Some lags are removed:
# Remove lags 0 and 2 from "interest" and
# lags 1 and 3 from "logm1"
removed = list(interest = c(0,2), logm1 = c(1,3))
removed
model.dlm  = dlm(formula = logprice ~ interest + logm1 ,
```

```
              data = data.frame(data) , q = 4 , remove = removed)

x.new =  matrix(c(0.07 , 9.06 , 0.071 , 9.09 , 0.079 , 9.19 ,
                  0.069 , 9.21) , ncol = 4, nrow = 2)
forecast(model = model.dlm , x = x.new , h = 4 ,
         interval = FALSE)
forecast(model = model.dlm , x = x.new , h = 4 ,
            interval = FALSE)

x.new =  matrix(c(0.07 , 9.06 , 0.071 , 9.09, 0.08 , 9.12), ncol = 3,
                nrow = 2)
forecast(model = model.dlm , x = x.new , h = 3,  interval = FALSE)

#--- Koyck dlm ---
model.koyck = koyckDlm(x = warming$NoMotorVehicles ,
                       y = warming$Warming)
forecast(model = model.koyck  , x = c(95, 98, 101), h = 3 ,
                interval = FALSE)

#--- Polynomial dlm ---
model.poly = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
                q = 2 , k = 2 , show.beta = TRUE)
forecast(model = model.poly , x = c(95, 98) , h = 1 ,
                interval = FALSE)
```

`koyckDlm` *Implement distributed lag models with Koyck transformation*

Description

A function that applies distributed lag models with Koyck transformation with one predictor.

Usage

```
koyckDlm(x , y)
```

Arguments

`x`	A vector including the observations of predictor time series. This is not restricted to `ts` objects.
`y`	A vector including the observations of dependent time series. This is not restricted to `ts` objects.

Details

To deal with infinite DLMs, we can use the Koyck transformation. When we apply Koyck transformation, we get the following:

$$Y_t - \phi Y_{t-1} = \alpha(1-\phi) + \beta X_t + (\epsilon_t - \phi\epsilon_{t-1}).$$

When we solve this equation for Y_t, we obtain Koyck DLM as follows:

$$Y_t = \delta_1 + \delta_2 Y_{t-1} + \delta_3 X_t + \nu_t,$$

where $\delta_1 = \alpha(1-\phi), \delta_2 = \phi, \delta_3 = \beta$ and the random error after the transformation is $\nu_t = (\epsilon_t - \phi\epsilon_{t-1})$ (Judge and Griffiths, 2000).

Then, instrumental variables estimation is employed to fit the model.

The standard function `summary()` prints model summary for the model of interest.

Value

`model` An object of class `ivreg`. See the details of `ivreg` function.

`geometric.coefficients`
A vector composed of corresponding geometric distributed lag model coefficients.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

References

B.H. Baltagi. *Econometrics*, Fifth Ed. Springer, 2011.

R.C. Hill, W.E. Griffiths, G.G. Judge. *Undergraduate Econometrics*. Wiley, 2000.

Examples

```
data(warming)
model.koyck = koyckDlm(x = warming$NoMotorVehicles ,
                       y = warming$Warming)
summary(model.koyck)
```

MASE *Compute mean absolute scaled error (MASE) for distributed lag models*

Description

A function that computes mean absolute scaled error for fitted DLMs.

Usage

```
MASE(model, ...)
```

Arguments

model	Model object fitted for time series data.
...	Optionally, more fitted models.

Details

Let $e_t = Y_t - \hat{Y}_t$ be the one-step-ahead forecast error. Then, a scaled error is defined as

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|},$$

which is independent of the scale of the data. Mean absolute scaled error is defined as

$$MASE = mean(|q_t|)$$

(Hyndman and Koehler, 2006).

Fitted models would be finite, polynomial, Koyck, ARDL DLMs, or linear model fitted with `lm()` function. This function also computes MASE values of multiple models when fed at the same time.

Value

MASE	Mean absolute scaled error (MASE) for the observed and fitted series sent into the function.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

References

Hyndman, R.J. and Koehler, A.B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22, 679-688.

Examples

```
data(warming)
# Fit a bunch of polynomial DLMs
model.poly1 = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
                      q = 2 , k = 2 , show.beta = TRUE)
model.poly2 = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
                       q = 3 , k = 2 , show.beta = TRUE)
model.poly3 = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
                       q = 4 , k = 2 , show.beta = TRUE)
MASE(model.poly1, model.poly2, model.poly3)

# Fit a bunch of finite DLMs
model.dlm1 = dlm(x = warming$NoMotorVehicles , y = warming$Warming, q =2)
model.dlm2 = dlm(x = warming$NoMotorVehicles , y = warming$Warming, q =3)
model.dlm3 = dlm(x = warming$NoMotorVehicles , y = warming$Warming, q =4)
```

```
MASE(model.dlm1, model.dlm2, model.dlm3)

# Fit a linear model
model.lm = lm(Warming ~ NoMotorVehicles , data = warming)
MASE(model.lm)

# Fit a Koyck model
model.koyck = koyckDlm(x = warming$NoMotorVehicles , y = warming$Warming )
MASE(model.koyck)

# Fit a bunch of ARDLs
model.ardl1 = ardlDlm(x = warming$NoMotorVehicles , y = warming$Warming, p=1, q=2)
model.ardl2 = ardlDlm(x = warming$NoMotorVehicles , y = warming$Warming, p=2, q=2)
model.ardl3 = ardlDlm(x = warming$NoMotorVehicles , y = warming$Warming, p=3, q=2)
MASE(model.ardl1 , model.ardl2 , model.ardl3)

# Find MASEs of different model objects
MASE(model.ardl1 , model.dlm1 , model.poly1, model.lm)
```

---

polyDlm *Implement finite polynomial distributed lag model*

Description

A function that applies polynomial distributed lag models with one predictor.

Usage

```
polyDlm(x , y , q , k , show.beta = TRUE)
```

Arguments

x	A vector including the observations of predictor time series. This is not restricted to ts objects.
y	A vector including the observations of dependent time series. This is not restricted to ts objects.
q	An integer representing finite lag length.
k	An integer representing order of polynomial distributed lags.
show.beta	If TRUE, generates original beta parameters and associated t-tests and prints the results.

Details

Finite distributed lag models, in general, suffer from the multicollinearity due to inclusion of the lags of the same variable in the model. To reduce the impact of this multicollinearity, a polynomial shape is imposed on the lag distribution (Judge and Griffiths, 2000). The resulting model is called Polynomial Distributed Lag model or Almond Distributed Lag Model.

Imposing a polynomial pattern on the lag distribution is equivalent to representing β parameters with another \$k\$th order polynomial model of time. So, the effect of change in X_{t-s} on the expected value of Y_t is represented as follows:

$$\frac{\partial E(Y_t)}{\partial X_{t-s}} = \beta_s = \gamma_0 + \gamma_1 s + \gamma_2 s^2 + \cdots + \gamma_k s^k$$

where $s = 0, \ldots, q$ (Judge and Griffiths, 2000). Then the model becomes:

$$Y_t = \alpha + \gamma_0 Z_{t0} + \gamma_1 Z_{t1} + \gamma_2 Z_{t2} + \cdots + \gamma_k Z_{tk} + \epsilon_t.$$

The standard function `summary()` prints model summary for the model of interest.

Value

`model` An object of class `lm`.

`designMatrix` The design matrix composed of transformed z-variables.

`designMatrix.x` The design matrix composed of original x-variables.

`beta.coefficients`
Estimates and t-tests of original beta coefficients. This will be generated if `show.beta` is set to `TRUE`.

Author(s)

Haydar Demirhan

Maintainer: Haydar Demirhan <haydar.demirhan@rmit.edu.au>

References

B.H. Baltagi. *Econometrics*, Fifth Ed. Springer, 2011.

R.C. Hill, W.E. Griffiths, G.G. Judge. *Undergraduate Econometrics*. Wiley, 2000.

Examples

```
data(warming)
model.poly = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
q = 2 , k = 2 , show.beta = TRUE)
summary(model.poly)
```

sortScore *Sort ACI, BIC and MASE scores*

Description

A function that displays sorted AIC, BIC, and MASE scores.

Usage

```
sortScore(x, score = c("bic", "aic", "mase"))
```

Arguments

x	A vector of AIC, BIC, or MASE values.
score	The type of scores to be sorted.

Details

This function sorts the AIC, BIC, or MASE scores to display the smallest one at the top of a bunch of AIC, BIC, or MASE scores.

Author(s)

Cameron Doyle

Maintainer: Cameron Doyle <cdoyle305@gmail.com>

Examples

```
data(warming)
model.poly1 = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
                      q = 2 , k = 2 , show.beta = TRUE)
model.poly2 = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
                       q = 3 , k = 2 , show.beta = TRUE)
model.poly3 = polyDlm(x = warming$NoMotorVehicles , y = warming$Warming ,
                       q = 4 , k = 2 , show.beta = TRUE)

aic = AIC(model.poly1$model, model.poly2$model, model.poly3$model)
bic = BIC(model.poly1$model, model.poly2$model, model.poly3$model)
mase = MASE(model.poly1$model, model.poly2$model, model.poly3$model)

sortScore(aic , score = "aic")
sortScore(bic , score = "bic")
sortScore(mase , score = "mase")
```

warming *Global warming and vehicle prediction data*

Description

This data set is composed of annual mean global warming series between 1997 and 2016 showing the change in global surface temperature relative to 1951-1980 average temperatures and the number of vehicles produced within the same time span over the globe.

Usage

```
data(warming)
```

Format

Multiple time series

Source

Global Climate Center, NASA Organisation Internationale des Constructeurs d'Automobiles (OICA)

References

https://climate.nasa.gov/vital-signs/global-temperature/

http://www.oica.net/category/production-statistics/

Examples

```
data(warming)
vehicleWarming.ts = ts(warming[,2:3], start = 1997)
plot(vehicleWarming.ts, main="Time series plots
of global warming and the nuber of produced motor
vehciles series.")
```

Index

∗Topic **datasets**
warming, 18
∗Topic **distributed lag model, time series, regression model, polynomial lag, predictor**
dLagM-package, 2

ardlDlm, *2*, 3

dLagM-package, 2
dlm, *2*, 5

finiteDLMauto, 8
forecast, 10

koyckDlm, *2*, 12

MASE, 13

polyDlm, *2*, 15

sortScore, 17

warming, 18