Package 'dLagM'

October 2, 2023

Type Package

Title Time Series Regression Models with Distributed Lag Models

Version 1.1.13

Date 2023-10-02

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 Demirhan (2020)(<doi:10.1371/journal.pone.0228812>) and Baltagi (2011)(<doi:10.1007/978-3-642-20059-5>) for more information.

Depends graphics, stats, nardl, dynlm, R (>= 3.6.0)

Imports AER, formula.tools, plyr , lmtest, strucchange, wavethresh, MASS, roll, sandwich

License GPL-3

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-10-02 06:30:02 UTC

R topics documented:

`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 Demirhan (2020)(<doi:10.1371/journal.pone.0228812>) and Baltagi (2011)(<doi:10.1007/978-3-642-20059-5>) for more information.

Details

Package:	dLagM
Type:	Package
Version:	1.1.13
Date:	2023-10-02
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 implement ARDL Bounds test, use `ardlBound` 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 (https://orcid.org/0000-0002-8565-4710)

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

Acknowledgements: The author acknowledges the testing effort of of Dr. Rogerio Porto (https://orcid.org/0000-0002-6663-9531) on forecasting function.

References

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

H. Demirhan. dLagM: An R package for distributed lag models and ARDL bounds testing. *PLoS ONE*, 15(2): e0228812, 2020. DOI: 10.1371/journal.pone.0228812.

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

J. Soren, A.Q. Philips. "pss: Perform bounds test for cointegration and perform dynamic simulations."

P.K. Narayan. The Saving and Investment Nexus for China: Evidence from cointegration tests. *Applied Economics* 37(17):1979-1990, 2005.

M.H. Pesaran, S. Yongcheol, R.J. Smith. Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics* 16(3):289-326, 2001.

See Also

dlm, polyDlm, koyckDlm, ardlDlm

Examples

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

`ardlBound` *Implement ARDL bounds test*

Description

Applies ARDL bounds test with the approach of Pesaran et al. (2001).

Usage

```
ardlBound(data = NULL, formula = NULL, case = 3, p = NULL,
          remove = NULL, autoOrder = FALSE, HAC = FALSE,
          ic = c("AIC", "BIC", "MASE", "GMRAE"), max.p = 15,
          max.q = 15, ECM = TRUE, stability = TRUE)
```

Arguments

data	A data.frame including all dependent and independent series. Column names this data.frame must match the variable names in formula.
formula	A formula object showing the dependent and independent series.
case	An integer up to 5 showing the case number. See details.
p	An integer representing the order of short-run response or a data.frame to specify a different order of short-run response for each variable.
remove	A list showing the elements to be removed from the model. It includes elements p as a list and q as a vector. See details.
autoOrder	If TRUE, the order of ARDL will be found by the ardlBounOrders function.
HAC	If TRUE, the Newey-West estimate of variance-covariance function is used in testing.
ic	Information criterion to be used in the search for optimal orders.
max.p	Maximum order for the short-run coefficients.
max.q	Maximum auto-regressive order.
ECM	If TRUE, the error correction model corresponding to the case is also fitted and included in the outputs.
stability	If both ECM and stability are TRUE, the CUSUM, CUSUM of squares, and MOSUM charts are generated over recursive residuals using the package strucchange.

Details

The argument case takes the values 1 for "no intercept, no trend", 2 for "restricted intercept, no trend", 3 for "unrestricted intercept, no trend", 4 for "unrestricted intercept, restricted trend", and 5 for "unrestricted intercept, unrestricted trend."

If the argument p is entered as an integer, the same value is used to specify the order of short-run response for all variables.

We follow the formulation of Pesaran et al. (2001). So, the upper limit of summations in the model formulation goes up to (p-1). User should consider this while setting the value(s) of p.

The names of the element p of remove must match with those in the model. For example, to remove lags 2 and 4 of the first differences of X1 and X2 and remove the lags 2 and 5 of the dependent series, remove should be defined as `remove = list( p = list(dX1 = c(2,4), dX2 = c(2,4)), q = c(2,5) )`.

Breusch-Godfrey and Ljung-Box tests are applied to test against the existence of autocorrelations in residuals and Breusch-Pagan test is applied to detect heteroskedasticity in residuals as a part of the bounds testing procedure. Ramsey's RESET test is conducted for correctness of functional form of the model.

The recursive CUSUM chart is plotted by epf function from the strucchange package. The recursive CUSUM of squares plot is plotted by the ardlBound function using the recursive residuals generated by recresid function of strucchange package. The testing limits on the recursive CUSUM of squares plot are calculated as described by Brown et al. (1975) based in Table 1 of Durbin (1969). The recursive MOSUM chart is plotted when there is no NA MOSUM values are calculated.

Value

model	An object including the fitted model under the null and alternative hypotheses.
F.stat	The value of F-statistic coming out of the Wald test.
p	p-orders in the lag structure.
k	The number of independent series.
bg	Breusch-Godfrey test results. Returns NULL if skipped.
lb	Ljung-Box test results. Returns NULL if skipped.
bp	Breusch-Pagan test results. Returns NULL if skipped.
sp	Shapiro-Wilk test results. Returns NULL if skipped.
ECM	A list including the error correction series in the element EC.t, the fitted error correction model in the element EC.model, and the coefficient of error correction part in the element EC.beta.
ARDL.model	A model object including the fitted ARDL model. Use this model object to display the long-run coefficients. To see the significance test results for the logn-run coefficients, use summary() function.

Author(s)

Haydar Demirhan

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

References

R.L. Brown, J. Durbin, J. M. Evans. Techniques for testing the constancy of regression relationships over time. Journal of the Royal Statistical Society - Series B, 37, 2, 149-192.

C-S. J. Chu, K. Hornik, C-M. Kuan. MOSUM tests for parameter constancy. Biometrika, 82, 603-617, 1995.

J. Durbin. Tests for serial correlation in regression analysis based on the periodogram of Least-Squares residuals. Biometrika, 56, 1, 1-15.

P.K. Narayan. The Saving and Investment Nexus for China: Evidence from Cointegration Tests. *Applied Economics* 37(17):1979-1990, 2005.

M.H. Pesaran, S. Yongcheol, R.J. Smith. Bounds testing approaches to the analysis of level relationships. *Journal of Applied Econometrics* 16(3):289-326, 2001.

J.B. Ramsey. Tests for specification error in classical linear Least Squares regression analysis. Journal of the Royal Statistical Society - Series B, 31, 350-371, 1969.

J. Soren, A.Q. Philips. "pss: Perform bounds test for cointegration and perform dynamic simulations."

A. Zeileis, F. Leisch, K. Hornik, C. Kleiber. strucchange: An R package for testing for structural change in linear regression models. Journal of Statistical Software, 7, 1-38, 2002.

Examples

```
## Not run:
data(M1Germany)
data <- M1Germany[1:144,]
model <- ardlBound(data = data , formula = logprice ~ interest + logm1 , case = 2 , p = 2)

# Let ardlBoundOrders() function find the orders
model <- ardlBound(data = data , formula = logprice ~ interest + logm1 , case = 2 ,
                   max.p = 3, max.q = 3)

## End(Not run)
```

`ardlBoundOrders` *Find optimal orders (lag structure) for ARDL bounds test*

Description

Computes optimal orders (lag structure) for the short-run relationships and autoregressive part of the ARDL model prior to ARDL bounds test with the approach of Pesaran et al. (2001).

Usage

```
ardlBoundOrders(data = NULL , formula = NULL, ic = c("AIC", "BIC", "MASE",
               "GMRAE"), max.p = 15,  max.q = 15, FullSearch = FALSE  )
```

Arguments

`data`	A `data.frame` including all dependent and independent series. Column names this `data.frame` must match the variable names in `formula`.
`formula`	A `formula` object showing the dependent and independent series.
`ic`	Information criterion to be used in the serach for optimal orders.
`max.p`	Maximum order for the short-run coefficients.
`max.q`	Maximum auto-regressive order.
`FullSearch`	If `TURE`, a search over all possbilble models is implemented. See the details.

Details

If `FullSearch = FALSE`, this function first assumes that all p-orders are equal for the short-run relationships and finds the optimal p-order and autoregressive orders. Then, it finds the best subset of p-orders allowing them to change for each series in the short-run relationship part of the ARDL model under alternative hypothesis of ARDL bounds test.

Value

p	An integer or data.frame object including p-orders for the short-run relationship part.
q	The autoregressive order.
Stat.table	The selected statistic of all the considered models where all short-run relationship orders (p-orders) are equal.
Stat.p	The selected statistic of all possible combinations of short-run relationship orders (p-orders). The reported lag structure is the one that gives the minimum value to the chosen statistic among these combinations.

Author(s)

Haydar Demirhan

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

---

ardlDlm *Implement finite autoregressive distributed lag model*

Description

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 , type = 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.
type	An integer taking 1 if only x and y vectors are entered, 2 if a formula and data matrix is entered. It can be left NULL since the correct value is checked and fixed by the code.

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.

To remove the intercept from the model, if a `formula` is entered, just include "-1" in the model formula. Otherwise, include "-1" in the element `q` of the list `remove`. 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(seaLevelTempSOI)
model.ardl = ardlDlm(x = seaLevelTempSOI$LandOcean,
                     y = seaLevelTempSOI$GMSL, p = 1 , q = 1 )
summary(model.ardl)
# residuals(model.ardl)
# coef(model.ardl)
# fitted(model.ardl)

# 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 = seaLevelTempSOI$LandOcean,
                     y = seaLevelTempSOI$GMSL, p = 2 , q = 3 , remove = remove)
summary(model.ardl)

# To remove intercept as well
rem.q = c(1,3,-1)
remove = list(p = rem.p , q = rem.q)
model.ardl = ardlDlm(x = seaLevelTempSOI$LandOcean,
                     y = seaLevelTempSOI$GMSL, 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)

# To remove intercept as well
model.ardlDlm  = ardlDlm(formula = logprice ~ -1 + 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

Applies distributed lag models with one or multiple predictor(s).

Usage

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

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.
type	An integer taking 1 if only x and y vectors are entered, 2 if a formula and data matrix is entered. It can be left NULL since the correct value is checked and fixed by the code.

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.

To remove the intercept from the model, if a formula is entered, just include "-1" in the model formula. Otherwise, include "-1" in the element remove of the list remove. 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.
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(seaLevelTempSOI)
model.dlm = dlm(x = seaLevelTempSOI$LandOcean,
                y = seaLevelTempSOI$GMSL , q = 5)
summary(model.dlm)
residuals(model.dlm)
coef(model.dlm)
fitted(model.dlm)
```

```
removed = list(x = c(1))
model.dlm = dlm(x = seaLevelTempSOI$LandOcean,
                y = seaLevelTempSOI$GMSL , q = 5,
                remove = removed)
summary(model.dlm)

# Remove intercept as well
removed = list(x = c(1,-1))
model.dlm = dlm(x = seaLevelTempSOI$LandOcean,
                y = seaLevelTempSOI$GMSL , q = 5,
                remove = removed)
summary(model.dlm)

removed = list(x = c(0,1))
model.dlm = dlm(x = seaLevelTempSOI$LandOcean,
                y = seaLevelTempSOI$GMSL , q = 5,
                remove = removed)
summary(model.dlm)

model.dlm = dlm(formula =  GMSL ~ LandOcean ,
                 data = seaLevelTempSOI , q = 5)
summary(model.dlm)

removed = list(x = c(0,1))
model.dlm = dlm(formula =  GMSL ~ LandOcean ,
               data = seaLevelTempSOI , q = 5,
                remove = removed)
summary(model.dlm)

# Remove intercept as well
removed = list(x = c(0,-1))
model.dlm = dlm(formula =  GMSL ~ LandOcean ,
                data = seaLevelTempSOI , q = 2,
                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

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","GMRAE", "MBRAE", "radj"),
              trace = FALSE , type)
```

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. The optimal lag length is determined according to desired goodness-of-fit measure.
`trace`	If `TRUE`, prints all of the goodness-of-fit measures for all fitted models.
`type`	An integer taking 1 if only x and y vectors are entered, 2 if a formula and data matrix is entered. It can be left `NULL` since the correct value is checked and fixed by the code.

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

```
## Not run:
library(dLagM)
# Multiple independent series
data(M1Germany)
data = M1Germany[1:44,]
# Run the search over finite DLMs according to AIC values
finiteDLMauto(formula = logprice ~ interest + logm1,
              data = data.frame(data), q.min = 2, q.max = 5,
              model.type = "dlm", error.type = "AIC", trace = FALSE)

## End(Not run)
```

---

forecast *Compute forecasts for distributed lag models*

Description

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	A fitted model by one of dlm(), koyckDlm(), ploydlm() or ardlDlm functions.
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.

When x and y are used to fit the model and some elements of the model are removed, you must use model and formula instead of x and y to fit the model and send the new data as a matrix into the forecast() function.

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

```
## Not run:
# Only one independent series
data(seaLevelTempSOI)
#--- ARDL dlm ---
model.ardl = ardlDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL, p = 1 , q = 1 )
forecast(model = model.ardl , x = c(0.15, 0.45) ,
```

```
                h = 2 , interval = FALSE)
forecast(model = model.ardl , x = c(0.15, 0.45, 0.20) ,
                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, 0.08, 9.2), ncol = 3,
                nrow = 2)
forecast(model = model.ardlDlm1 , x = x.new , h = 3 ,
                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 = seaLevelTempSOI$LandOcean,  y = seaLevelTempSOI$GMSL,
                q = 2)
forecast(model = model.dlm , x = c(0.15, 0.45, 0.20) , 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 -1,
              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 = seaLevelTempSOI$LandOcean,  y = seaLevelTempSOI$GMSL)
forecast(model = model.koyck  , x = c(0.15, 0.45, 0.20), h = 3 ,
                interval = FALSE)

#--- Polynomial dlm ---
model.poly = polyDlm(x = seaLevelTempSOI$LandOcean,  y = seaLevelTempSOI$GMSL,
                q = 2 , k = 2 , show.beta = TRUE)
forecast(model = model.poly , x = c(0.15, 0.45) , h = 1 ,
                interval = FALSE)


## End(Not run)
```

GoF *Compute goodness-of-fit measures for DLMs*

Description

Computes mean absolute error (MAE), mean squared error (MSE), mean percentage error (MPE), symmetric mean absolute percentage error (sMAPE), mean absolute percentage error (MAPE), mean absolute scaled error (MASE), mean relative absolute error (MRAE), geometric mean relative absolute error (GMRAE), mean bounded relative absolute error (MBRAE), unscaled MBRAE (UMBRAE) for distributed lag models.

Usage

```
GoF(model, ...)
MASE(model, ...)
sMAPE(model, ...)
MAPE(model, ...)
MRAE(model, ...)
GMRAE(model, ...)
MBRAE(model, ...)
```

Arguments

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

Details

See Chen et al. (2017) for the definitions of MSE, MAE, MAPE, sMAPE, MRAE, GMRAE, MBRAE, UMBMRAE.

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.

Author(s)

Haydar Demirhan

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

References

Chen, C., Twycross, J., Garibaldi, J.M. (2017). A new accuracy measure based on bounded relative error for time series forecasting. *PLoS ONE*, 12(3), e0174202.

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

Examples

```
## Not run:
data(seaLevelTempSOI)
# Fit a bunch of polynomial DLMs
model.poly1 = polyDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL,
                      q = 2 , k = 2 , show.beta = TRUE)
model.poly2 = polyDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL,
                       q = 3 , k = 2 , show.beta = TRUE)
model.poly3 = polyDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL,
                       q = 4 , k = 2 , show.beta = TRUE)
MASE(model.poly1, model.poly2, model.poly3)

# Fit a bunch of finite DLMs
model.dlm1 = dlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL, q =2)
model.dlm2 = dlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL, q =3)
model.dlm3 = dlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL, q =4)
MASE(model.dlm1, model.dlm2, model.dlm3)
MBRAE(model.dlm1, model.dlm2, model.dlm3)
GoF(model.dlm1, model.dlm2, model.dlm3)

# Fit a linear model
```

```
model.lm = lm(GMSL ~ LandOcean , data = seaLevelTempSOI)
MASE(model.lm)
GoF(model.lm)

# Fit a Koyck model
model.koyck = koyckDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL)
MASE(model.koyck)
GoF(model.koyck)

# Fit a bunch of ARDLs
model.ardl1 = ardlDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL, p=1, q=2)
model.ardl2 = ardlDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL, p=2, q=2)
model.ardl3 = ardlDlm(x = seaLevelTempSOI$LandOcean, y = seaLevelTempSOI$GMSL, p=3, q=2)
MASE(model.ardl1 , model.ardl2 , model.ardl3)
GoF(model.ardl1 , model.ardl2 , model.ardl3)

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

## End(Not run)
```

`grainProduction` *World oats, corn, rice, wheat production, CO2 emissions, temperature anomalies, cropland data*

Description

This data set is composed of annual world total CO2 emissions, oats, corn, rice, and wheat production, cropland area, and annual average temperature anomalies series between 1961 and 2018.

Usage

```
data(grainProduction)
```

Format

Multiple time series

`CO2(Tons)` column shows the global mean annual carbon dioxide (CO2) emissions measured in tons.

`Land-OceanTempIndex(C)` column shows the land-ocean temperature index without smoothing.

`Cropland(1Mha)` column shows the annual cropland area in the world scale in million hectares.

`Oats(1Mt)` column shows the annual world oats production in million tons.

`Corn(1Mt)` column shows the annual world corn production in million tons.

`Rice(1Mt)` column shows the annual world rice production in million tons.

`Wheat(1Mt)` column shows the annual world wheat production in million tons.

Source

CO2 emissions data: Hannah Ritchie and Max Roser (2020) - "CO2 and Greenhouse Gas Emissions". Published online at OurWorldInData.org.

Temperature anomalies data: NASA

Grain production data: United States Department of Agriculture, Foreign Agricultural Service

Cropland data: Food and Agriculture Organization of United Nations

References

CO2 emissions data: https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions

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

Grain production data: psd_grains_pulses.csv from https://apps.fas.usda.gov/psdonline/app/index.html#/app/downloads

Cropland data: https://www.fao.org/faostat/en/#data/RL/visualize

Examples

```
data(grainProduction)
oatsProduction.ts = ts(grainProduction[,5], start = 1961)
plot(oatsProduction.ts, main="Time series plot
of world oats production series.")
```

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

Description

Applies distributed lag models with Koyck transformation with one predictor.

Usage

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

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.
`intercept`	Set to `TRUE` to inclue intercept in the model.

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.

AIC/BIC of a fitted KOyck model is displayed by setting the `class` attribute of model to `lm`. See the example.

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(seaLevelTempSOI)
model.koyck = koyckDlm(x = seaLevelTempSOI$LandOcean,
                       y = seaLevelTempSOI$GMSL)
summary(model.koyck, diagnostics = TRUE)
residuals(model.koyck)
coef(model.koyck)
fitted(model.koyck)
AIC(model.koyck)
BIC(model.koyck)
```

polyDlm *Implement finite polynomial distributed lag model*

Description

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(seaLevelTempSOI)
model.poly = polyDlm(x = seaLevelTempSOI$LandOcean,  y = seaLevelTempSOI$GMSL ,
                     q = 4 , k = 2 , show.beta = TRUE)
summary(model.poly)
residuals(model.poly)
coef(model.poly)
fitted(model.poly)
```

---

rolCorPlot *PLot the rolling correlations*

Description

Plots the rolling correlations along with other required statistics to visualise the approach of Gershunov et al. (2001) to test the significance of signal from rolling correlation analysis.

Usage

```
rolCorPlot(x , y , width, level = 0.95, main = NULL,
           SDtest = TRUE, N = 500)
```

Arguments

x	A ts object.
y	A ts object.
width	A numeric vector of window lengths of the rolling correlation analysis.
level	Confidence level for intervals.
main	The main title of the plot.
SDtest	Set to TRUE to run test the significance of signal from rolling correlation analysis along with plotting.
N	An integer showing the number of series to be generated in Monte Carlo simulation.

Value

rolCor	A matrix showing rolling correlations for each width on its columns.
rolcCor.avr.filtered	A vector showing average rolling correlations filtered by running median nonlinear filter against outliers.
rolcCor.avr.raw	A vector showing unfiltered average rolling correlations.
rolCor.sd	A vector showing standard deviations of rolling correlations for each width.
rawCor	Pearson correlation between two series.
sdPercentiles	Percentiles of MC distribution of standard deviations of rolling correlations as the test limits.
test	A data frame showing the standard deviations of rolling correlations for each width along with level and (1-level) limits.

Author(s)

Haydar Demirhan

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

References

Gershunov, A., Scheider, N., Barnett, T. (2001). Low-Frequency Modulation of the ENSO-Indian Monsoon Rainfall Relationship: Signal or Noise? *Journal of Climate*, 14, 2486 - 2492.

Examples

```
## Not run:
data(wheat)
prod.ts <-ts(wheat[,5], start = 1960)
CO2.ts <- ts(wheat[,2], start = 1960)
rolCorPlot(x = prod.ts, y = CO2.ts , width = c(7, 11, 15), level = 0.95, N = 50)

## End(Not run)
```

sdPercentiles *Test the significance of signal from rolling correlation analysis*

Description

Implements the approach of Gershunov et al. (2001) to test the significance of signal from rolling correlation analysis.

Usage

```
sdPercentiles(n = 150, cor = 0.5, width = 5, N = 500,
              percentiles = c(.05, .95))
```

Arguments

n	The length of the series in the rolling correlation analysis.
cor	The magnitude of raw correaltion betweeen two time series in the rolling correlation analysis.
width	Window length of the rolling correlation analysis.
N	Number of Monte Carlo replications for simulations.
percentiles	Percentiles to be reported for the Monte Carlo distribution of standard deviations of rolling correlations for the given window width.

Details

N samples of correlated white noise series are generated with a magnitude of cor; rolling correlations analysis is applied with the window length of width; Monte Carlo distribution of standard deviations of rolling correlations are generated; and desired percentiles of the MC distribution of standard deviations are reported (Gershunov et al. 2001).

Value

rollCorSd.limits

Percentiles of MC distribution of standard deviations of rolling correlations as the test limits.

Author(s)

Haydar Demirhan

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

References

Gershunov, A., Scheider, N., Barnett, T. (2001). Low-Frequency Modulation of the ENSO-Indian Monsoon Rainfall Relationship: Signal or Noise? *Journal of Climate*, 14, 2486 - 2492.

Examples

```
# sdPercentiles(n = 50, cor = 0.5, width = 5, N = 50,
#              percentiles = c(.025, .975))
```

`seaLevelTempSOI` *Global mean sea level (GMSL), mean land and ocean temperature anomalies, and Southern Oscillation Index (SOI) data*

Description

This data set is composed of monthly global mean sea level (compared to 1993-2008 average) series by CSIRO, land ocean temperature anomalies (1951-1980 as a baseline period) by GISS, NASA, and monthly Southern Oscillation Index (SOI) by Australian Government Bureau of Meteorology (BOM) between July 1885 and June 2013. GMSL and temperature anomalies series are smoothed and seasonally adjusted.

Usage

```
data(seaLevelTempSOI)
```

Format

Multiple time series

Source

Goddard Institute for Space Studies, NASA, US. Marine and Atmospheric Research, CSIRO, Australia. Australian Government Bureau of Meteorology (BOM).

References

Church, J. A. and N.J. White (2011), Sea-level rise from the late 19th to the early 21st Century. Surveys in Geophysics, doi:10.1007/s10712-011-9119-1.

https://www.cmar.csiro.au/sealevel/sl_data_cmar.html

https://data.giss.nasa.gov/gistemp/graphs_v4/

https://www.bom.gov.au/climate/current/soihtm1.shtml

Examples

```
data(seaLevelTempSOI)
level.ts <- ts(seaLevelTempSOI[,1], start = c(1880,7), freq = 12)
temp.ts <- ts(seaLevelTempSOI[,2], start = c(1880,7), freq = 12)
plot(level.ts, main="Time series plot of GMSL series.")
```

sortScore *Sort AIC, BIC, MASE, MAPE, sMAPE, MRAE, GMRAE, or MBRAE scores*

Description

Displays sorted AIC, BIC, and MASE scores.

Usage

```
sortScore(x, score = c("bic", "aic", "mase", "smape", "mape", "mrae", "gmrae", "mbrae"))
```

Arguments

x	A vector of AIC, BIC, MASE, MAPE, sMAPE, MRAE, GMRAE, or MBRAE values.
score	The type of scores to be sorted.

Details

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

Author(s)

Cameron Doyle

Maintainer: Cameron Doyle <cdoyle305@gmail.com>

Examples

```
## Not run:
data(seaLevelTempSOI)
model.poly1 = polyDlm(x = seaLevelTempSOI$LandOcean,  y = seaLevelTempSOI$GMSL ,
                      q = 2 , k = 2 , show.beta = TRUE)
model.poly2 = polyDlm(x = seaLevelTempSOI$LandOcean,  y = seaLevelTempSOI$GMSL ,
                       q = 3 , k = 2 , show.beta = TRUE)
model.poly3 = polyDlm(x = seaLevelTempSOI$LandOcean,  y = seaLevelTempSOI$GMSL ,
                       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$model)
mbrae = MBRAE(model.poly1$model, model.poly2$model, model.poly3$model)

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

```
## End(Not run)
```

---

sunspotTemp *Sunspot numbers and mean temperature anomalies data*

Description

This data set is composed of monthly mean global surface temperature series by GISS NASA and sunspot numbers recorded by SWPC Space Weather Operations (SWO) between January 1991 and November 2019.

Usage

```
data(sunspotTemp)
```

Format

Multiple time series

Source

Goddard Institute for Space Studies, NASA, US. Space Weather Prediction Center National Oceanic and Atmospheric Administration, US.

References

https://data.giss.nasa.gov/gistemp/graphs_v4/

ftp://ftp.swpc.noaa.gov/pub/weekly/RecentIndices.txt

Examples

```
data(sunspotTemp)
sunspots.ts <- ts(sunspotTemp[,3], start = c(1991,1), freq = 12)
temp.ts <- ts(sunspotTemp[,4], start = c(1991,1), freq = 12)
plot(sunspots.ts, main="Time series plots
of sunspot numbers series.")
```

warming *Global warming and vehicle production 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 (in millions) 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/

https://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.")
```

wheat *World wheat production, CO2 emissions, and temperature anomalies data*

Description

This data set is composed of annual world total CO2 emissions, wheat production, harvested area, wheat production per hectare, and annual average temperature anomalies series between 1960 and 2017.

Usage

```
data(wheat)
```

Format

Multiple time series

`CO2..ppm.` column shows the global mean annual concentration of carbon dioxide (CO2) measured in parts per million (ppm).

`CO2..tons` column shows the global mean annual carbon dioxide (CO2) emissions measured in tons.

`HarvestedArea..million.ha.` column shows the annual harvested area in the world scale in million hectare.

`Production..Mt.` column shows the annual world wheat production in million tons.

`ProducationPerArea` column shows the annual wheat production per hectare in tons.

`TempAnomaly..C.degrees.` column shows the land-ocean temperature index without smoothing.

Source

Australian Government Department of Agriculture and Water Services, ABARES CO2 and other Greenhouse Gas Emissions by Hannah Ritchie and Max Roser

References

https://www.agriculture.gov.au/abares/research-topics/agricultural-commodities/agricultural-commodities-trade-data#australian-crop-report-data

https://www.agriculture.gov.au/abares/research-topics/agricultural-outlook/data

https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions#annual-co2-emissions

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

Examples

```
data(wheat)
wheatProduction.ts = ts(wheat[,4], start = 1960)
plot(wheatProduction.ts, main="Time series plot
of world wheat production series.")
```

Index