Package 'ecm'

October 13, 2022

Type Package

Title Build Error Correction Models

Imports stats, utils, car, sandwich, lmtest, urca, earth

Version 6.3.0

Author Gaurav Bansal

Maintainer Gaurav Bansal <gaurbans@gmail.com>

Description Functions for easy building of error correction models (ECM) for time series regression.

URL https://github.com/gaurbans/ecm

License GPL (>= 2)

LazyData TRUE

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-28 15:20:02 UTC

R topics documented:

`cumcount` *Get cumulative count*

Description

Get the cumulative count of a variable of interest

Usage

```
cumcount(x)
```

Arguments

`x`	A vector for which to get cumulative count

Value

The cumulative count of all items in x

`durbinH` *Calculate Durbin's h-statistic*

Description

Calculates Durbin's h-statistic for autoregressive models.

Usage

```
durbinH(model, ylag1var)
```

Arguments

`model`	The model being assessed
`ylag1var`	The variable in the model that represents the lag of the y-term

Details

Using the Durbin-Watson (DW) test for autoregressive models is inappropriate because the DW test itself tests for first order autocorrelation. This doesn't apply to an ECM model, for which the DW test is still valid, but the durbinH function in included here in case an autoregressive model has been built. If Durbin's h-statistic is greater than 1.96, it is likely that autocorrelation exists.

Value

Numeric Durbin's h statistic

See Also

`lm`

Examples

```
##Not run

#Build a simple AR1 model to predict performance of the Wilshire 5000 Index
data(Wilshire)
Wilshire$Wilshire5000Lag1 <- c(NA, Wilshire$Wilshire5000[1:(nrow(Wilshire)-1)])
Wilshire <- Wilshire[complete.cases(Wilshire),]
AR1model <- lm(Wilshire5000 ~ Wilshire5000Lag1, data=Wilshire)

#Check Durbin's h-statistic on AR1model
durbinH(AR1model, "Wilshire5000Lag1")
#The h-statistic is 4.23, which means there is likely autocorrelation in the data.
```

`ecm` *Build an error correction model*

Description

Builds an lm object that represents an error correction model (ECM) by automatically differencing and lagging predictor variables according to ECM methodology.

Usage

```
ecm(
  y,
  xeq,
  xtr,
  lags = 1,
  includeIntercept = TRUE,
  weights = NULL,
  linearFitter = "lm",
  ...
)
```

Arguments

`y`	The target variable
`xeq`	The variables to be used in the equilibrium term of the error correction model
`xtr`	The variables to be used in the transient term of the error correction model
`lags`	The number of lags to use

`includeIntercept`	Boolean whether the y-intercept should be included (should be set to TRUE if using 'earth' as linearFitter)
`weights`	Optional vector of weights to be passed to the fitting process
`linearFitter`	Whether to use 'lm' or 'earth' to fit the model
`...`	Additional arguments to be passed to the 'lm' or 'earth' function (careful that some arguments may not be appropriate for ecm!)

Details

The general format of an ECM is

$$\Delta y_t = \beta_0 + \beta_1 \Delta x_{1,t} + ... + \beta_i \Delta x_{i,t} + \gamma(y_{t-1} - (\alpha_1 x_{1,t-1} + ... + \alpha_i x_{i,t-1})).$$

The ecm function here modifies the equation to the following:

$$\Delta y = \beta_0 + \beta_1 \Delta x_{1,t} + ... + \beta_i \Delta x_{i,t} + \gamma y_{t-1} + \gamma_1 x_{1,t-1} + ... + \gamma_i x_{i,t-1},$$

$$where \gamma_i = -\gamma \alpha_i,$$

so it can be modeled as a simpler ordinary least squares (OLS) function using R's lm function.

Ordinarily, the ECM uses lag=1 when differencing the transient term and lagging the equilibrium term, as specified in the equation above. However, the ecm function here gives the user the ability to specify a lag greater than 1.

Notice that an ECM models the change in the target variable (y). This means that the predictors will be lagged and differenced, and the model will be built on one observation less than what the user inputs for y, xeq, and xtr. If these arguments contain vectors with too few observations (eg. one single observation), the function will not work. Additionally, for the same reason, if using weights in the ecm function, the length of weights should be one less than the number of rows in xeq or xtr.

When inputting a single variable for xeq or xtr in base R, it is important to input it in the format "xeq=df['col1']" so they inherit the class 'data.frame'. Inputting such as "xeq=df[,'col1']" or "xeq=df$col1" will result in errors in the ecm function. You can load data via other R packages that store data in other formats, as long as those formats also inherit the 'data.frame' class.

By default, base R's 'lm' is used to fit the model. However, users can opt to use 'earth', which uses Jerome Friedman's Multivariate Adaptive Regression Splines (MARS) to build a regression model, which transforms each continuous variable into piece-wise linear hinge functions. This allows for non-linear features in both the transient and equilibrium terms.

ECM models are used for time series data. This means the user may need to consider stationarity and/or cointegration before using the model.

Value

an lm object representing an error correction model

See Also

`lm`, `earth`

Examples

```
##Not run

#Use ecm to predict Wilshire 5000 index based on corporate profits,
#Federal Reserve funds rate, and unemployment rate.
data(Wilshire)

#Use 2015-12-01 and earlier data to build models
trn <- Wilshire[Wilshire$date<='2015-12-01',]

#Assume all predictors are needed in the equilibrium and transient terms of ecm.
xeq <- xtr <- trn[c('CorpProfits', 'FedFundsRate', 'UnempRate')]
model1 <- ecm(trn$Wilshire5000, xeq, xtr, includeIntercept=TRUE)

#Assume CorpProfits and FedFundsRate are in the equilibrium term,
#UnempRate has only transient impacts.
xeq <- trn[c('CorpProfits', 'FedFundsRate')]
xtr <- trn['UnempRate']
model2 <- ecm(trn$Wilshire5000, xeq, xtr, includeIntercept=TRUE)

#From a strictly statistical standpoint, Wilshire data may not be stationary
#and hence model1 and model2 may have heteroskedasticity in the residuals.
#Let's check for that.
lmtest::bptest(model1)
lmtest::bptest(model2)
#The Breush-Pagan tests suggest we should reject homoskedasticity in the residuals for both models.

lmtest::bgtest(model1)
lmtest::bgtest(model2)
#The Breusch-Godfrey tests suggest we should reject that there is no serial correlation
#in the residuals.

#Given the above issues, see adjusted std. errors and p-values for our models.
lmtest::coeftest(model1, vcov=sandwich::NeweyWest)
lmtest::coeftest(model2, vcov=sandwich::NeweyWest)
```

---

ecmave *Build an averaged error correction model*

Description

Builds multiple ECM models on subsets of the data and averages them. See the lmave function for more details on the methodology and use cases for this approach.

Usage

```
ecmave(
  y,
  xeq,
  xtr,
  lags = 1,
  includeIntercept = TRUE,
  k,
  method = "boot",
  seed = 5,
  weights = NULL,
  ...
)
```

Arguments

`y`	The target variable
`xeq`	The variables to be used in the equilibrium term of the error correction model
`xtr`	The variables to be used in the transient term of the error correction model
`lags`	The number of lags to use
`includeIntercept`	Boolean whether the y-intercept should be included
`k`	The number of models or data partitions desired
`method`	Whether to split data by folds ("fold"), nested folds ("nestedfold"), or bootstrapping ("boot")
`seed`	Seed for reproducibility (only needed if method is "boot")
`weights`	Optional vector of weights to be passed to the fitting process
`...`	Additional arguments to be passed to the 'lm' function (careful in that these may need to be modified for ecm or may not be appropriate!)

Details

In some cases, instead of building an ECM on the entire dataset, it may be preferable to build k ECM models on k subsets of the data, each subset containing (k-1)/k*nrow(data) observations of the full dataset, and then average their coefficients. Reasons to do this include controlling for overfitting or extending the training sample. For example, in many time series modeling exercises, the holdout test sample is often the latest few months or years worth of data. Ideally, it's desirable to include these data since they likely have more future predictive power than older observations. However, including the entire dataset in the training sample could result in overfitting, or using a different time period as the test sample may be even less representative of future performance. One potential solution is to build multiple ECM models using the entire dataset, each with a different holdout test sample, and then average them to get a final ECM. This approach is somewhat similar to the idea of random forest regression, in which multiple regression trees are built on subsets of the data and then averaged.

This function only works with the 'lm' linear fitter.

Value

an lm object representing an error correction model

See Also

`lm`

Examples

```
##Not run

#Use ecm to predict Wilshire 5000 index based on corporate profits,
#Federal Reserve funds rate, and unemployment rate
data(Wilshire)

#Use 2015-12-01 and earlier data to build models
trn <- Wilshire[Wilshire$date<='2015-12-01',]

#Build five ECM models and average them to get one model
xeq <- xtr <- trn[c('CorpProfits', 'FedFundsRate', 'UnempRate')]
model1 <- ecmave(trn$Wilshire5000, xeq, xtr, includeIntercept=TRUE, k=5)
```

---

ecmaveback *Backwards selection using an averaged error correction model*

Description

Much like the ecmback function, ecmaveback uses backwards selection to build an error correction model. However, it uses the averaging method of ecmave to build models and then choose variables based on lowest AIC or BIC, or highest adjusted R-squared.

Usage

```
ecmaveback(
  y,
  xeq,
  xtr,
  lags = 1,
  includeIntercept = T,
  criterion = "AIC",
  k,
  method = "boot",
  seed = 5,
  weights = NULL,
  keep = NULL,
  ...
)
```

Arguments

`y`	The target variable
`xeq`	The variables to be used in the equilibrium term of the error correction model
`xtr`	The variables to be used in the transient term of the error correction model
`lags`	The number of lags to use
`includeIntercept`	Boolean whether the y-intercept should be included
`criterion`	Whether AIC (default), BIC, or adjustedR2 should be used to select variables
`k`	The number of models or data partitions desired
`method`	Whether to split data by folds ("fold"), nested folds ("nestedfold"), or bootstrapping ("boot")
`seed`	Seed for reproducibility (only needed if method is "boot")
`weights`	Optional vector of weights to be passed to the fitting process
`keep`	Optional character vector of variables to forcibly retain
`...`	Additional arguments to be passed to the 'lm' function (careful in that these may need to be modified for ecm or may not be appropriate!)

Details

When inputting a single variable for xeq or xtr, it is important to input it in the format "xeq=df['col1']" in order to retain the data frame class. Inputting such as "xeq=df[,'col1']" or "xeq=df$col1" will result in errors in the ecm function.

If using weights, the length of weights should be one less than the number of rows in xeq or xtr.

This function only works with the 'lm' linear fitter.

Value

an lm object representing an error correction model using backwards selection

See Also

`lm`

Examples

```
##Not run

#Use ecm to predict Wilshire 5000 index based on corporate profits,
#Federal Reserve funds rate, and unemployment rate
data(Wilshire)

#Use 2015-12-01 and earlier data to build models
trn <- Wilshire[Wilshire$date<='2015-12-01',]

#Use backwards selection to choose which predictors are needed
xeq <- xtr <- trn[c('CorpProfits', 'FedFundsRate', 'UnempRate')]
```

```
modelaveback <- ecmaveback(trn$Wilshire5000, xeq, xtr, k = 5)
print(modelaveback)
#Backwards selection chose CorpProfits and FedFundsRate in the equilibrium term,
#CorpProfits and UnempRate in the transient term.

modelavebackFFR <- ecmaveback(trn$Wilshire5000, xeq, xtr, k = 5, keep = 'UnempRate')
print(modelavebackFFR)
#Backwards selection was forced to retain UnempRate in both terms.
```

ecmback — *Backwards selection to build an error correction model*

Description

Much like the ecm function, this builds an error correction model. However, it uses backwards selection to select the optimal predictors based on lowest AIC or BIC, or highest adjusted R-squared, rather than using all predictors.

Usage

```
ecmback(
  y,
  xeq,
  xtr,
  lags = 1,
  includeIntercept = T,
  criterion = "AIC",
  weights = NULL,
  keep = NULL,
  ...
)
```

Arguments

`y`	The target variable
`xeq`	The variables to be used in the equilibrium term of the error correction model
`xtr`	The variables to be used in the transient term of the error correction model
`lags`	The number of lags to use
`includeIntercept`	Boolean whether the y-intercept should be included
`criterion`	Whether AIC (default), BIC, or adjustedR2 should be used to select variables
`weights`	Optional vector of weights to be passed to the fitting process
`keep`	Optional character vector of variables to forcibly retain
`...`	Additional arguments to be passed to the 'lm' function (careful in that these may need to be modified for ecm or may not be appropriate!)

Details

When inputting a single variable for xeq or xtr, it is important to input it in the format "xeq=df['col1']" in order to retain the data frame class. Inputting such as "xeq=df[,'col1']" or "xeq=df$col1" will result in errors in the ecm function.

If using weights, the length of weights should be one less than the number of rows in xeq or xtr.

This function only works with the 'lm' linear fitter. The 'earth' linear fitter already does some variable selection, so one can use that via that 'ecm' function.

Value

an lm object representing an error correction model using backwards selection

See Also

`lm`

Examples

```
##Not run

#Use ecm to predict Wilshire 5000 index based on corporate profits,
#Federal Reserve funds rate, and unemployment rate
data(Wilshire)

#Use 2015-12-01 and earlier data to build models
trn <- Wilshire[Wilshire$date<='2015-12-01',]

#Use backwards selection to choose which predictors are needed
xeq <- xtr <- trn[c('CorpProfits', 'FedFundsRate', 'UnempRate')]
modelback <- ecmback(trn$Wilshire5000, xeq, xtr)
print(modelback)
#Backwards selection chose CorpProfits and FedFundsRate in the equilibrium term,
#CorpProfits and UnempRate in the transient term.

modelbackFFR <- ecmback(trn$Wilshire5000, xeq, xtr, keep = 'UnempRate')
print(modelbackFFR)
#Backwards selection was forced to retain UnempRate in both terms.
```

`ecmpredict` *Predict using an ecm object*

Description

Takes an ecm object and uses it to predict based on new data. This prediction does the undifferencing required to transform the change in y back to y itself.

Usage

```
ecmpredict(model, newdata, init)
```

Arguments

`model`	ecm object used to make predictions
`newdata`	Data frame to on which to predict
`init`	Initial value for prediction

Details

Since error correction models only model the change in the target variable, an initial value must be specified. Additionally, the 'newdata' parameter should have at least 3 rows of data.

Value

Numeric predictions on new data based ecm object

Examples

```
##Not run

data(Wilshire)

#Rebuilding model1 from ecm example
trn <- Wilshire[Wilshire$date<='2015-12-01',]
xeq <- xtr <- trn[c('CorpProfits', 'FedFundsRate', 'UnempRate')]
model1 <- ecm(trn$Wilshire5000, xeq, xtr)
model2 <- ecm(trn$Wilshire5000, xeq, xtr, linearFitter='earth')

#Use 2015-12-01 and onwards data as test data to predict
tst <- Wilshire[Wilshire$date>='2015-12-01',]

#predict on tst using model1 and initial FedFundsRate
tst$model1Pred <- ecmpredict(model1, tst, tst$Wilshire5000[1])
tst$model2Pred <- ecmpredict(model2, tst, tst$Wilshire5000[1])
```

`lagpad` *Lag a vector*

Description

Create a vector of the lag of a variable and fill missing values with NA's.

Usage

```
lagpad(x, k = 1)
```

Arguments

`x`	A vector to be lagged
`k`	The number of lags to output

Value

The lagged vector with NA's in missing values

`lmave` *Build multiple lm models and average them*

Description

Builds k lm models on k partitions of the data and averages their coefficients to get create one model. Each partition excludes k/nrow(data) observations. See links in the References section for further details on this methodology.

Usage

```
lmave(formula, data, k, method = "boot", seed = 5, weights = NULL, ...)
```

Arguments

`formula`	The formula to be passed to lm
`data`	The data to be used
`k`	The number of models or data partitions desired
`method`	Whether to split data by folds ("fold"), nested folds ("nestedfold"), or bootstrapping ("boot")
`seed`	Seed for reproducibility (only needed if method is "boot")
`weights`	Optional vector of weights to be passed to the fitting process
`...`	Additional arguments to be passed to the 'lm' function

Details

In some cases–especially in some time series modeling (see ecmave function)–rather than building one model on the entire dataset, it may be preferable to build multiple models on subsets of the data and average them. The lmave function splits the data into k partitions of size (k-1)/k*nrow(data), builds k models, and then averages the coefficients of these models to get a final model. This is similar to averaging multiple tree regression models in algorithms like random forest.

Unlike the 'ecm' functin, this function only works with the 'lm' linear fitter.

Value

an lm object

References

Jung, Y. & Hu, J. (2016). "A K-fold Averaging Cross-validation Procedure". https://www.ncbi.nlm.nih.gov/pmc/articles/PMC

Cochrane, C. (2018). "Time Series Nested Cross-Validation". https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9

See Also

`lm`

Examples

```
##Not run

#Build linear models to predict Wilshire 5000 index based on corporate profits,
#Federal Reserve funds rate, and unemployment rate
data(Wilshire)

#Build one model on the entire dataset
modelall <- lm(Wilshire5000 ~ ., data = Wilshire[-1])

#Build a five fold averaged linear model on the entire dataset
modelave <- lmave('Wilshire5000 ~ .', data = Wilshire[-1], k = 5)
```

`Wilshire` *FRED data on the Wilshire 5000 index and other economic factors*

Description

A dataset containing quarterly performance of the Wilshire 5000 index, corporate profits, Federal Reserve funds rate, and the unemployment rate.

Usage

`data(Wilshire)`

Format

A data frame with 188 rows and 5 variables:

date monthly date

Wilshire5000 quarterly Wilshire 5000 index, in value

CorpProfits quarterly corporate profits, in value

FedFundsRate quarterly federal funds rate, in percent

UnempRate quarterly unemployment rate, in percent

Source

https://fred.stlouisfed.org/

Index