

Package ‘dynlm’

September 8, 2009

Version 0.2-3

Date 2009-09-08

Title Dynamic Linear Regression

Author Achim Zeileis

Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>

Description Dynamic linear models and time series regression.

Depends R (>= 2.0.0), stats, zoo, lmtest, car

Suggests sandwich, strucchange

Imports stats, strucchange

LazyLoad yes

LazyData yes

License GPL-2

Repository CRAN

Date/Publication 2009-09-08 17:35:57

R topics documented:

dynlm	2
M1Germany	4
Index	6

Description

Interface to `lm.wfit` for fitting dynamic linear models and time-series regression relationships.

Usage

```
dynlm(formula, data, subset, weights, na.action, method = "qr",
      model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
      contrasts = NULL, offset, start = NULL, end = NULL, ...)
```

Arguments

<code>formula</code>	a "formula" describing the linear model to be fit. For details see below and lm .
<code>data</code>	an optional "data.frame" or time-series object (e.g., "ts" or "zoo"), containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>lm</code> is called.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>weights</code>	an optional vector of weights to be used in the fitting process. If specified, weighted least squares is used with weights <code>weights</code> (that is, minimizing $\sum(w \cdot e^2)$); otherwise ordinary least squares is used.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of options , and is <code>na.fail</code> if that is unset. The "factory-fresh" default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Note, that for time-series regression special methods like na.contiguous , na.locf and na.approx are available.
<code>method</code>	the method to be used; for fitting, currently only <code>method = "qr"</code> is supported; <code>method = "model.frame"</code> returns the model frame (the same as with <code>model = TRUE</code> , see below).
<code>model, x, y, qr</code>	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
<code>singular.ok</code>	logical. If <code>FALSE</code> (the default in S but not in R) a singular fit is an error.
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>offset</code>	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. An <code>offset</code> term can be included in the formula instead or as well, and if both are specified their sum is used.
<code>start</code>	start of the time period which should be used for fitting the model.
<code>end</code>	end of the time period which should be used for fitting the model.
<code>...</code>	additional arguments to be passed to the low level regression fitting functions.

Details

The interface and internals of `dynlm` are very similar to `lm`, but currently `dynlm` offers three advantages over the direct use of `lm`: 1. extended formula processing, 2. preservation of time-series attributes, 3. instrumental variables regression (via two-stage least squares).

For specifying the formula of the model to be fitted, there are additional functions available which facilitate the specification of dynamic models. An example would be $d(y) \sim L(y, 2)$, where $d(x, k)$ is `diff(x, lag = k)` and $L(x, k)$ is `lag(x, lag = -k)`, note the difference in sign. The default for k is in both cases 1. For $L()$, it can also be vector-valued, e.g., $y \sim L(y, 1:4)$. Furthermore, there is an additional function `season(x, ref = NULL)` which creates a factor with levels for each cycle of the season. Using the `ref` argument, the reference level can be changed from the default first level to any other. See below for examples and [M1Germany](#) for a more elaborate application.

The specification of dynamic relationships only makes sense if there is an underlying ordering of the observations. Currently, `lm` offers only limited support for such data, hence a major aim of `dynlm` is to preserve time-series properties of the data. Explicit support is currently available for "ts" and "zoo" series. Internally, the data is kept as a "zoo" series and coerced back to "ts" if the original dependent variable was of that class (and no internal NAs were created by the `na.action`).

To specify a set of instruments, formulas of type $y \sim x_1 + x_2 + x_3 | z_1 + z_2$ can be used where z_1 and z_2 represent the instruments. Again, the extended formula processing described above can be employed for all variables in the model.

See Also

[zoo](#), [merge.zoo](#)

Examples

```
## multiplicative SARIMA(1,0,0)(1,0,0)_12 model fitted
## to UK seatbelt data
uk <- log10(UKDriverDeaths)
dfm <- dynlm(uk ~ L(uk, 1) + L(uk, 12))
dfm
## explicitly set start and end
dfm <- dynlm(uk ~ L(uk, 1) + L(uk, 12), start = c(1975, 1), end = c(1982, 12))
dfm

## remove lag 12
dfm0 <- update(dfm, . ~ . - L(uk, 12))
anova(dfm0, dfm)

## add season term
dfm1 <- dynlm(uk ~ 1, start = c(1975, 1), end = c(1982, 12))
dfm2 <- dynlm(uk ~ season(uk), start = c(1975, 1), end = c(1982, 12))
anova(dfm1, dfm2)

plot(uk)
lines(fitted(dfm0), col = 2)
lines(fitted(dfm2), col = 4)
```

```

## regression on multiple lags in a single L() call
dfm3 <- dynlm(uk ~ L(uk, c(1, 11, 12)), start = c(1975, 1), end = c(1982, 12))
anova(dfm, dfm3)

## Examples 7.11/7.12 from Greene (1993)
if(require("lmtest")) {
  data("USDistLag")
  dfm1 <- dynlm(consumption ~ gnp + L(consumption), data = USDistLag)
  dfm2 <- dynlm(consumption ~ gnp + L(gnp), data = USDistLag)
  plot(USDistLag[, "consumption"])
  lines(fitted(dfm1), col = 2)
  lines(fitted(dfm2), col = 4)
  encomptest(dfm1, dfm2)
}

```

M1Germany

German M1 Money Demand

Description

German M1 money demand.

Usage

```
data(M1Germany)
```

Format

M1Germany is a "zoo" series containing 4 quarterly time series from 1960(1) to 1996(3).

logm1 logarithm of real M1 per capita,

logprice logarithm of a price index,

loggnp logarithm of real per capita gross national product,

interest long-run interest rate,

Details

This is essentially the same data set as [GermanM1](#), the important difference is that it is stored as a `zoo` series and not as a data frame. It does not contain differenced and lagged versions of the variables (as `GermanM1`) does, because these do not have to be computed explicitly before applying `dynlm`.

The (short) story behind the data is the following (for more detailed information see [GermanM1](#)): Lütkepohl et al. (1999) investigate the linearity and stability of German M1 money demand: they find a stable regression relation for the time before the monetary union on 1990-06-01 but a clear structural instability afterwards. Zeileis et al. (2005) re-analyze this data set in a monitoring situation.

Source

The data is provided by the German central bank and is available online in the data archive of the Journal of Applied Econometrics <http://qed.econ.queensu.ca/jae/1999-v14.5/lutkepohl-terasvirta-wolters/>.

References

Lütkepohl H., Teräsvirta T., Wolters J. (1999), Investigating Stability and Linearity of a German M1 Money Demand Function, *Journal of Applied Econometrics*, **14**, 511–525.

Zeileis A., Leisch F., Kleiber C., Hornik K. (2005), Monitoring Structural Change in Dynamic Econometric Models, *Journal of Applied Econometrics*, **20**, 99–121.

See Also

[GermanM1](#)

Examples

```
data("M1Germany")
## fit the model of Lütkepohl et al. (1999) on the history period
## before the monetary unification
histfm <- dynlm(d(logm1) ~ d(L(loggnp, 2)) + d(interest) + d(L(interest)) + d(logprice) +
               L(logm1) + L(loggnp) + L(interest) +
               season(logm1, ref = 4),
               data = M1Germany, start = c(1961, 1), end = c(1990, 2))

## fit on extended sample period
fm <- update(histfm, end = c(1995, 4))

if(require("strucchange")) {
  scus <- gefp(fm, fit = NULL)
  plot(scus, functional = supLM(0.1))
}
```

Index

*Topic **datasets**

M1Germany, 4

*Topic **regression**

dynlm, 2

dynlm, 2

end.dynlm(*dynlm*), 2

GermanM1, 4, 5

index.dynlm(*dynlm*), 2

lm, 2, 3

lm.wfit, 2

M1Germany, 3, 4

merge.zoo, 3

na.approx, 2

na.contiguous, 2

na.fail, 2

na.locf, 2

na.omit, 2

offset, 2

options, 2

print.dynlm(*dynlm*), 2

print.summary.dynlm(*dynlm*), 2

start.dynlm(*dynlm*), 2

summary.dynlm(*dynlm*), 2

time.dynlm(*dynlm*), 2

zoo, 3, 4