Package 'dyn'

July 22, 2025

Version 0.2-9.6

Date 2018-03-18

Title Time Series Regression

Author G. Grothendieck

Maintainer M. Leeds <markleeds2@gmail.com>

Description Time series regression. The dyn class interfaces ts, irts(), zoo() and zooreg() time series classes to lm(), glm(), loess(), quantreg::rq(), MASS::rlm(), MCMCpack::MCMCregress(), quantreg::rq(), randomForest::randomForest() and other regression functions allowing those functions to be used with time series including specifications that may contain lags, diffs and missing values.

Depends R (>= 2.6.0), zoo (>= 1.0-0)

Suggests lattice, MASS, MCMCpack, quantreg (>= 3.82), randomForest, sandwich, tseries

License GPL

Repository CRAN

Date/Publication 2018-03-19 06:14:06 UTC

NeedsCompilation no

Contents

dyn-package *Time Series Regression*

Description

Time series regression. The dyn class interfaces ts, irts, its, zoo and zooreg time series classes to lm, glm, loess, quantreg::rq, MASS::rlm, quantreg::rq, randomForest::randomForest and other regression functions allowing those functions to be used with time series including specifications that may contain lags, diffs and missing values.

Details

`"dyn"` allows one to use time series with regression functions that were not originally written to support time series by simply prefacing the call to the regression function with `"dyn$"`. The following are sources of information on `"dyn"`:

Overview	file.show(system.file("README", package = "dyn"))
News	RShowDoc("NEWS", package = "dyn")
Acknowledgements	RShowDoc("THANKS", package = "dyn")
Wish List	RShowDoc("WISHLIST", package = "dyn")
License	RShowDoc("COPYING", package = "dyn")
Citation	citation(package = "dyn")
List of all demo files	demo(package = "dyn"))
Invoking a demo file	demo("dyn-rq")
Source of demo file	file.show(system.file("dyn-rq.R", package = "dyn"))
This File	package?dyn
Help file	?dyn

Examples

```
x <- ts(seq(10)^2)
dyn$lm(x ~ lag(x,-1))
dyn$glm(x ~ lag(x,-1))
dyn$loess(x ~ lag(x,-1))
```

baltimore *Baltimore energy data*

Description

Heating costs for a home in Baltimore.

Usage

```
data(baltimore)
```

Format

The data set is a zoo series with a "Date" class time index and 6 numeric numeric columns.

start	date of start of billing period
therms	integer number of therms used in this billing period
gas	Total cost of gas (including delivery and commodity charges) for natural gas
KWHs	integer number of KWH used in this billing period
elect	Total cost of electricity (including delivery and commodity charges)
temp	average daily outdoor temperature in degrees Fahrenheit, as printed on the bill
days	number of days in billing period.

Details

Heating system is a 10-15 year old natural gas steam boiler supplying iron radiators. Hot water heater, clothes dryer and stove and oven are also natural gas. Air conditioning is by various numbers of window units. If surface area of house is desired, I can add this at a later time.

Some interesting points in time:

22-Apr-04	Date when house was upgraded 2 failed, older storm windows to more modern ones.
1-Sep-04	Date when house was upgraded 4 failed, older storm windows to more modern ones. Interesting ques
last week of July 1999	Spouse moved in; both adults absent during the work day, setback thermostat used. Interesting questi
18-Dec-2005	Brought home son; spouse and son home during the day, setback thermostat no longer used. Interesti

Examples

```
library(lattice)
data(baltimore)
xyplot(baltimore)
cor(baltimore)
xyplot(elect + gas ~ temp,
data = as.data.frame(baltimore), pch = 20, auto.key = TRUE)
```

---

dyn *dynamic regression class*

Description

dyn is used to construct objects of class "dyn"

Usage

```
dyn(x)
```

Arguments

x	an object, typically a "formula" object or an object produced by "lm", "glm" or other regression function.

Details

"dyn" enables regression functions that were not written to handle time series to handle them. Both the dependent and independent variables may be time series and they may have different time indexes (in which case they are automatically aligned). The time series may also have missing values including internal missing values.

"dyn" currently works with any regression function that makes use of "model.frame" and is written in the style of "lm". This includes "lm", "glm", "loess", "rlm" (from **"MASS"**), "lqs" (from **"MASS"**), "MCMCregress" (from **"MCMCpack"**), "randomForest" (from **"randomForest"**), "rq" (from **"quantreg"**) and others. The time series objects can be one of the following classes: "ts", "irts", "its", "zoo" or "zooreg".

Typically "dyn" is used like this "dyn$lm(y ~ lag(y, -1))". That is, one prepends the usual "lm" or other regression function with "dyn$" and then uses time series including "lag" and "diff" operators in the formula. The returned object has a class vector beginning with "dyn" and includes all classes that it would have had without "dyn". "dyn" methods include "model.frame", "fitted", "residuals", "predict", "update", "anova" and "$" methods. These methods preprocess their arguments, call the real method which does the actual work and then post process the returned object. In the case of "fitted", "residuals" and "predict" they ensure that the result is a time series. In the case of anova the times of the objects are intersected so that they all have the same time indexes to ensure that a comparable input is provided to "anova".

Value

"dyn" returns its argument with the class name "dyn" prepended to its class vector. The "fitted", "residuals" and "predict" "dyn" methods return time series of the appropriate class. "model.frame" creates a model frame with an attribute of "series" that contains a data frame of the time series and factor variables as columns. "model.matrix" returns a model matrix of class "matrix".

Note

"dyn" relies on the underlying time series classes and regression routines for all substantive functionality. In particular note these limitations: "irts" has no "lag" or "diff" methods. The lag function of "its" is called "lagIts". "ts" and "zooreg" series can be lagged outside of the data range (both forward and backward) but other time series classes cannot represent such data and therefore will drop them. If the regression function in question does not have an associated "fitted", "residuals", etc. method then such method will not be available with "dyn" either.

Internally the system uses "zoo". Additional time series classes not already defined to work with "dyn" can be added by simply defining "as" methods between the new class and "zoo" and then creating new methods (for "model.frame", "predict", "fitted", etc.) In most cases these method names can be set equal to the corresponding "zoo" method name (e.g. "model.frame.newclass <- model.frame.zoo" so that no new function bodies need be written).

The main requirements for new regression routines to work with "dyn" are that they use "model.frame", that their "fitted", "residuals" and "predict" methods return named vectors whose names are the corresponding indexes in the original data and that they follow the same style of processing as "lm". There is no "dyn" code specific to any particular regression routine.

"dyn$lm(formula, ...)" is equivalent to "dyn(lm(dyn(formula), ...))" (where "formula" is assumed to be the first argument) but is easier to write. When "dyn" is used with an argument, as

just shown, then its effect is simply to return its argument with the "dyn" class prepended to the class vector so that further processing of the result is intercepted by other "dyn" methods.

See Also

See Also model.frame, predict, fitted, residuals, anova, update, lm, glm, loess

Examples

```
y <- ts(1:12, start = c(2000,2), freq = 4)^3
x <- ts(1:9, start = c(2000,3), freq = 4)^2

# can be used with numerous different regression functions
y.lm <- dyn$lm( window(y, start = c(2000,4)) ~ diff(x) )
y.lm <- dyn$lm( y ~ diff(x) )
y.glm <- dyn$glm( y ~ diff(x) )
y.loess <- dyn$loess( y ~ diff(x) )

y.lm <- dyn(lm(dyn(y ~ diff(x))))  # same
y.lm
summary(y.lm)
residuals(y.lm)
fitted(y.lm)
y2.lm <- update(y.lm, . ~ . + lag(x,-1))
y2.lm
anova(y.lm, y2.lm)

# examples of using data
dyn$lm(y ~ diff(x), list(y = y, x = x))
dyn$lm(y ~ diffx, list(y = y, diffx = diff(x)))

# invoke model.frame on formula as a dyn object
dyn$model.frame( y ~ diff(x) )

# superimpose a loess fit on Nile time series data
plot(Nile)
lines(fitted(dyn$loess(Nile ~ time(Nile))), col = "red")

# lag.zoo can take vector lags
set.seed(1)
yz <- zoo(rnorm(100)); xz <- zoo(rnorm(100))
yz.lm <- dyn$lm(yz ~ lag(xz, 0:-3))


###
# simulate series and then NA out 7:10 and predict them
###

library(dyn)
set.seed(123)
tz <- zoo(cbind(Y = 0, x = rnorm(10), z = rnorm(10)))

# simulate values
```

```
for(i in 2:10) {
 tz$Y[i] <- with(as.data.frame(tz),
2*Y[i-1] + 3*z[i] +4* x[i] + 5*x[i-1] + rnorm(1))
}

# keep copy of tz to compare later to simulated Y's
tz.orig <- tz

# NA out Y's that are to be predicted
tz[7:10, "Y"] <- NA

L <- function(x, k = 1) lag(x, -k)

# predict 1 ahead each iteration
for(i in 7:10) {
   # fit based on first i-1 values
   fit <- dyn$lm(Y ~ L(Y) + z + L(x, 0:1), tz, subset = seq_len(i-1))
   # get prediction for ith value
   tz[i, "Y"] <- tail(predict(fit, tz[1:i,]), 1)
}
cbind(pred = tz[7:10, "Y"], act = tz.orig[7:10, "Y"])
```

Index