Package ‘fDMA’

January 24, 2025

Type Package

Title Dynamic Model Averaging and Dynamic Model Selection for Continuous Outcomes

Version 2.2.8

Imports doParallel, forecast, foreach, gplots, graphics, grDevices, iterators, itertools, parallel, psych, png, Rcpp, stats, tseries, utils, xts, zoo

Suggests R.rsp

VignetteBuilder R.rsp

LinkingTo Rcpp, RcppArmadillo

Date 2025-01-23

Author Krzysztof Drachal [aut, cre] (Faculty of Economic Sciences, University of Warsaw, Poland)

Maintainer Krzysztof Drachal <kdrachal@wne.uw.edu.pl>

Description Allows to estimate dynamic model averaging, dynamic model selection and median probability model. The original methods are implemented, as well as, selected further modifications of these methods. In particular the user might choose between recursive moment estimation and exponentially moving average for variance updating. Inclusion probabilities might be modified in a way using 'Google Trends'. The code is written in a way which minimises the computational burden (which is quite an obstacle for dynamic model averaging if many variables are used). For example, this package allows for parallel computations and Occam's window approach. The package is designed in a way that is hoped to be especially useful in economics and finance. Main reference: Raftery, A.E., Karny, M., Ettler, P. (2010) <doi:10.1198/TECH.2009.08104>.

LazyData TRUE

URL https://CRAN.R-project.org/package=fDMA

Note Research funded by the Polish National Science Centre grant under the contract number DEC-2015/19/N/HS4/00205.

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-01-24 10:00:07 UTC

Contents

`altf` *Computes a Few Alternative Forecasts.*

Description

It is necessary to compare a given forecast method with some alternative ones. This function computes selected forecast quality measures for a few selected forecast methods (which might be treated as alternative ones to Dynamic Model Averaging, Dynamic Model Selection, etc.).

Naive forecast (naive) is computed in a way that all forecasts are set to be the value of the last observation.

For rolling OLS forecast (roll. OLS) for the first periods (until the size of a `window` is obtained) are estimated through recursive OLS (rec. OLS).

Autoregressive models (AR(1) and AR(2)) are computed by ordinary least squares method.

Time-varying parameters models (TVP, TVP-AR(1) and TVP-AR(2)) are computed as `tvp` with `V=1` and `lambda=0.99`.

Auto ARIMA (auto ARIMA) is computed as `auto.arima`.

ME (Mean Error), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), MPE (Mean Percentage Errror) and MAPE (Mean Absolute Percentage Error) are computed as `accuracy`. HR (Hit Ratio) is computed as `hit.ratio`.

Usage

```
altf(y,x,window=NULL,initial.period=NULL,d=NULL,f=NULL,fmod=NULL,c=NULL)
```

Arguments

- `y` — numeric or a column matrix of a dependent variable
- `x` — matrix of independent variables, different columns correspond to different independent variables
- `window` — optional, numeric, a size of a rolling regression window (a number of observations), if not specified 10% of all observations are taken
- `initial.period` — optional, numeric, a number of observation since which forecast quality measures are computed, if not specified the whole sample is used, i.e., `initial.period=1`, this argument also divides the sample into in-sample and out-of-sample for non-recursive methods (OLS, AR(1), AR(2), auto ARIMA)
- `d` — optional, logical, a parameter used for HR (Hit Ratio) calculation, should be `d=FALSE` for level time-series and `d=TRUE` if time-series represent changes, if not specified `d=FALSE`
- `f` — optional, logical vector, indicating which of alternative forecasts – naive, OLS, rec. OLS, roll. OLS, TVP, AR(1), AR(2), auto ARIMA, TVP-AR(1) and TVP-AR(2) – should be computed, if not specified `f=c(rep(TRUE,10))`, i.e., all alternative forecasts are computed
- `fmod` — optional, class `dma` object, a model to be compared with alternative forecast
- `c` — optional, logical, a parameter indicating whether constant is included in models, if not specified `c=TRUE` is used, i.e., constant is included

Value

class `altf` object, list of

- `$summary` — matrix of forecast quality measures ordered by columns, forecast methods are ordered by rows
- `$y.hat` — list of predicted values from all forecasting methods which were applied
- `$y` — `y`, forecasted time-series
- `$coeff.` — list of coefficients from all forecasting methods which were applied (for naive forecast they are not computed)
- `$p.val.` — list of p-values for t-test of statistical significance for coefficients from all forecasting methods which were applied (for naive and TVP models they are not computed, and for auto ARIMA z-test is used)

See Also

`plot.altf`, `print.altf`, `summary.altf`, `rec.reg`, `roll.reg`, `altf2`, `altf3`, `altf4`.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
```

```
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

a1 <- altf(y=ld.wti,x=ld.drivers,d=TRUE,initial.period=60)

# models where constant term is not included in modelled equations (if applicable)
a2 <- altf(y=ld.wti,x=ld.drivers,d=TRUE,c=FALSE,initial.period=60)

# compute just selected models
fcomp <- c(TRUE,TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE)
a3 <- altf(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,initial.period=60)

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
a4 <- altf(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,fmod=m1,initial.period=60)
```

---

altf2 *Computes a Few Alternative Forecasts Based on Model Averaging.*

Description

It is necessary to compare a given forecast method with some alternative ones. This function computes selected forecast quality measures for a few selected forecast methods (which might be treated as alternative ones to Dynamic Model Averaging, Dynamic Model Selection, etc.).

ME (Mean Error), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), MPE (Mean Percentage Errror) and MAPE (Mean Absolute Percentage Error) are computed as `accuracy`. HR (Hit Ratio) is computed as `hit.ratio`.

Usage

```
altf2(y,x,mods.incl=NULL,gprob=NULL,omega=NULL,av=NULL,window=NULL,
initial.period=NULL,d=NULL,f=NULL,fmod=NULL,parallel=NULL)
```

Arguments

`y`	`numeric` or a column `matrix` of a dependent variable
`x`	`matrix` of independent variables, different columns correspond to different independent variables
`mods.incl`	optional, `matrix` indicating which models will be used in averaging, if not specified all possible models will be used, see `fDMA`
`gprob`	optional, `matrix` of Google probabilities as in Koop and Onorante (2014), columns should correspond to columns of x, see `fDMA`
`omega`	optional, `numeric`, a parameter between 0 and 1 used in probabilities estimations, used if `gprob` is specified, see `fDMA`

av	optional, a method for model averaging, av="ord" corresponds to equal weights for each model, av="aic" corresponds to information theoretic model averaging based on Akaike Information Criterion, av="aicc" corresponds to information theoretic model averaging based on Akaike Information Criterion with a correction for finite sample sizes, av="bic" corresponds to information theoretic model averaging based on Bayesian Information Criterion, av="mse" corresponds to setting weights proportional to the inverse of the models Mean Squared Error, if not specified av="ord" is used
window	optional, numeric, a size of a rolling regression window (a number of observations), if not specified 10% of all observations are taken
initial.period	optional, numeric, a number of observation since which forecast quality measures are computed, if not specified the whole sample is used, i.e., initial.period=1, this argument also divides the sample into in-sample and out-of-sample for av. OLS method
d	optional, logical, a parameter used for HR (Hit Ratio) calculation, should be d=FALSE for level time-series and d=TRUE if time-series represent changes, if not specified d=FALSE
f	optional, logical vector, indicating which of alternative forecast – av. OLS, av. rec. OLS, av. roll. OLS and av. TVP – should be averaged, if not specified f=c(rep(TRUE,4), i.e., all alternative forecast are computed
fmod	optional, class dma object, a model to be compared with alternative forecast
parallel	optional, logical, indicate whether parallel computations should be used, by default parallel=FALSE

Details

For each av method, in the initial period equal weights for each model are taken, and then successively updated based on the chosen criterion. For OLS models weights are not updated. The same weight for each model (estimated from the in-sample period) is taken for each period.

If gprob is used, then for OLS mean values from the in-sample period are taken, for rec. OLS – mean values from periods up to the current one, for roll. OLS – mean values from the last window periods, and for TVP – values from the current period.

Value

class altf2 object, list of

$summary	matrix of forecast quality measures ordered by columns, forecast methods are ordered by rows
$y.hat	list of predicted values from all forecasting methods which were applied
$y	y, forecasted time-series
$coeff.	list of coefficients from all forecasting methods which were applied
$weights	list of weights of models used in averaging for all forecasting methods which were applied

`$p.val.`	`list` of p-values (averaged with respect to suitable weights) for t-test of statistical significance for coefficients from all forecasting methods which were applied (for TVP they are not computed)
`$rel.var.imp.`	`list` of relative variable importance from all forecasting methods which were applied
`$exp.var.`	`list` of expected number of variables (incl. constant) from all forecasting methods which were applied

References

Burnham, K. P., Anderson, D. R., 2004. Multimodel inference: Understanding AIC and BIC in model selection. *Sociological Methods & Research* **33**, 261–304.

Burnham, K. P., Anderson, D. R., 2002. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*, Springer.

Gelman, A., Hwang, J., Vehtari, A., 2014. Understanding predictive information criteria for Bayesian models. *Statistics and Computing* **24**, 997–1016.

Kapetanios, G., Labhard, V., Price, S., 2008. Forecasting using Bayesian and information-theoretic model averaging. *Journal of Business & Economic Statistics* **26**, 33–41.

Koop, G., Onorante, L., 2014. Macroeconomic nowcasting using Google probabilities. https://goo.gl/ATsBN9

Timmermann, A., 2006. Forecast combinations. In: Elliott, G., et al. (eds.), *Handbook of Economic Forecasting*, Elsevier.

See Also

`plot.altf2`, `print.altf2`, `summary.altf2`, `rec.reg`, `roll.reg`, `tvp`, `altf`, `altf3`, `altf4`.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

a1 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,initial.period=60)

# compute just selected models
fcomp <- c(TRUE,TRUE,TRUE,FALSE)
a2 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,initial.period=60)
a3 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,av="aic",initial.period=60)

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
a4 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,fmod=m1,initial.period=60)

# models just with one independent variable and a constant will be averaged
mds <- diag(1,ncol(ld.drivers),ncol(ld.drivers))
```

```
mds <- cbind(rep(1,ncol(ld.drivers)),mds)
a5 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,mods.incl=mds,initial.period=60)

# Google trends are available since 2004
gp <- trends/100
s1 <- ld.wti['2004-01-01/']
s2 <- ld.drivers['2004-01-01/']
a6 <- altf2(y=s1,x=s2,d=TRUE,gprob=gp,omega=0.5,initial.period=60)
```

---

altf3 *Computes a Rolling Regression Averaged over Different Window Sizes.*

Description

It is necessary to compare a given forecast method with some alternative ones. This function computes selected forecast quality measures for a rolling regression averaged over different window sizes (which might be treated as alternative forecasting method to Dynamic Model Averaging, Dynamic Model Selection, etc.).

ME (Mean Error), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), MPE (Mean Percentage Errror) and MAPE (Mean Absolute Percentage Error) are computed as `accuracy`. HR (Hit Ratio) is computed as `hit.ratio`.

Usage

```
altf3(y,x=NULL,windows,av=NULL,initial.period=NULL,d=NULL,fmod=NULL,parallel=NULL,c=NULL)
```

Arguments

`y`	`numeric` or a column `matrix` of a dependent variable
`x`	`matrix` of independent variables, different columns correspond to different independent variables, if not specified only constant term will be included
`windows`	`numeric vector`, sizes of a rolling regression windows (numbers of observations)
`av`	optional, a method for model averaging, `av="ord"` corresponds to equal weights for each model, `av="aic"` corresponds to information theoretic model averaging based on Akaike Information Criterion, `av="aicc"` corresponds to information theoretic model averaging based on Akaike Information Criterion with a correction for finite sample sizes, `av="bic"` corresponds to information theoretic model averaging based on Bayesian Information Criterion, `av="mse"` corresponds to setting weights proportional to the inverse of the models Mean Squared Error, if `av` is `numeric` then weights are computed proportional to the `av`-th power of window size, if not specified `av="ord"` is used
`initial.period`	optional, `numeric`, a number of observation since which forecast quality measures are computed, if not specified the whole sample is used, i.e., `initial.period=1`

d	optional, logical, a parameter used for HR (Hit Ratio) calculation, should be d=FALSE for level time-series and d=TRUE if time-series represent changes, if not specified d=FALSE
fmod	optional, class dma object, a model to be compared with alternative forecast
parallel	optional, logical, indicate whether parallel computations should be used, by default parallel=FALSE
c	optional, see roll.reg

Details

For each av method, in the initial period equal weights for each model are taken, and then successively updated based on the chosen criterion.

Value

class altf3 object, list of

$summary	matrix of forecast quality measures ordered by columns
$y.hat	list of predicted values from a rolling regression averaged over selected window sizes
$y	y, forecasted time-series
$coeff.	list of coefficients from a rolling regression averaged over selected window sizes
$weights	list of weights of models used in averaging
$p.val.	list of p-values (averaged over selected window sizes) for t-test of statistical significance for coefficients from a rolling regression
$exp.win.	list of expected window size

References

Pesaran, M. H., Pick, A., 2011. Forecast combination across estimation windows. *Journal of Business & Economic Statistics* **29**, 307–318.

See Also

plot.altf3, print.altf3, summary.altf3, roll.reg, altf, altf2, altf4.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

a1 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,windows=c(36,100,150))
```

```
a2 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,av="aic",windows=c(36,100,150))

a3 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,av=-2,windows=c(36,100,150))

# models without a constant term
a4 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,av=-2,windows=c(36,100,150),c=FALSE)

# models only with a constant term
a5 <- altf3(y=ld.wti,d=TRUE,av=-2,windows=c(36,100,150))
```

---

altf4 *Computes a Time-Varying Parameters Rolling Regression Averaged over Different Window Sizes.*

Description

It is necessary to compare a given forecast method with some alternative ones. This function computes selected forecast quality measures for a time-varying parameters rolling regression averaged over different window sizes (which might be treated as alternative forecasting method to Dynamic Model Averaging, Dynamic Model Selection, etc.). The averaging is performed as in Raftery et al. (2010). The only difference is that the state space of the models are constructed not by chosing different combinations of independent variables, but for a fixed set of independent variables various rolling windows sizes are chosen and models constructed in such a way constitute the state space.

ME (Mean Error), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), MPE (Mean Percentage Errror) and MAPE (Mean Absolute Percentage Error) are computed as `accuracy`. HR (Hit Ratio) is computed as `hit.ratio`.

Usage

```
altf4(y,x,windows,V=NULL,alpha=NULL,lambda=NULL,initial.period=NULL,
d=NULL,fmod=NULL,parallel=NULL,c=NULL,small.c=NULL)
```

Arguments

y	`numeric` or a column `matrix` of a dependent variable
x	`matrix` of independent variables, different columns correspond to different independent variables
windows	`numeric vector`, sizes of a rolling regression windows (numbers of observations)
V	optional, `numeric`, initial variance in the state space equation for the recursive moment estimator updating method, as in Raftery et al. (2010), if not specified V=1 is taken, see `tvp`
lambda	optional, `numeric`, a forgetting factor between 0 and 1 used in variance approximations, if not specified `lambda=0.99` is taken, see `tvp`

alpha	optional, numeric, a forgetting factor α between 0 and 1 used in probabilities estimations, if not specified alpha=0.99 is taken, see fDMA
initial.period	optional, numeric, a number of observation since which forecast quality measures are computed, if not specified the whole sample is used, i.e., initial.period=1
d	optional, logical, a parameter used for HR (Hit Ratio) calculation, should be d=FALSE for level time-series and d=TRUE if time-series represent changes, if not specified d=FALSE
fmod	optional, class dma object, a model to be compared with alternative forecast
parallel	optional, logical, indicate whether parallel computations should be used, by default parallel=FALSE
c	optional, see tvp
small.c	optional, see fDMA

Value

class altf4 object, list of

$summary	matrix of forecast quality measures ordered by columns
$y.hat	list of predicted values from a time-varying parameters rolling regression averaged over selected window sizes
$y	y, forecasted time-series
$coeff.	list of coefficients from a time-varying parameters rolling regression averaged over selected window sizes
$weights	list of weights of models used in averaging
$exp.win.	list of expected window size

References

Pesaran, M. H., Pick, A., 2011. Forecast combination across estimation windows. *Journal of Business & Economic Statistics* **29**, 307–318.

Raftery, A. E., Gneiting, T., Balabdaoui, F., Polakowski, M., 2005. Using Bayesian Model Averaging to calibrate forecast ensembles. *Monthly Weather Review* **133**, 1155–1174.

Raftery, A. E., Karny, M., Ettler, P., 2010. Online prediction under model uncertainty via Dynamic Model Averaging: Application to a cold rolling mill. *Technometrics* **52**, 52–66.

See Also

plot.altf4, print.altf4, summary.altf4, roll.reg, tvp, altf, altf2, altf3.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
```

```
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

a1 <- altf4(y=ld.wti,x=ld.drivers,d=TRUE,windows=c(36,100,150))

win <- c(36,100,150)
a2 <- altf4(y=ld.wti,x=ld.drivers,d=TRUE,windows=win,alpha=0.9,lambda=0.95)

# models without a constant term
a3 <- altf4(y=ld.wti,x=ld.drivers,d=TRUE,windows=win,alpha=0.9,lambda=0.95,c=FALSE)

# models only with a constant term
empty <- matrix(,nrow=nrow(ld.drivers),ncol=0)
a4 <- altf4(y=ld.wti,x=empty,d=TRUE,windows=win,alpha=0.9,lambda=0.95)
```

`archtest` *Computes Engle's ARCH Test.*

Description

This function computes Engle's ARCH test. The null hypothesis of this Lagrange Multiplier test is that a series of residuals exhibits no ARCH effects. The alternative hypothesis is that ARCH(lag) effects are present. The `lag` is specified by the User.

Usage

```
archtest(ts,lag=NULL)
```

Arguments

`ts`	`vector`, the tested time-series
`lag`	`numeric`, suspected order of ARCH process, if not specified `lag=1` is taken

Value

class `htest` object, `list` of

`statistic`	test statistic
`parameter`	`lag` used in the test
`alternative`	alternative hypothesis of the test
`p.value`	p-value
`method`	name of the test
`data.name`	name of the tested time-series

References

Engle, R. F., 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* **50**, 987–1007.

Examples

```
wti <- crudeoil[-1,1]
ld.wti <- (diff(log(wti)))[-1,]
arch <- archtest(ts=as.vector(ld.wti),lag=10)
```

---

coef.dma — *Extracts Averaged Coefficients from* dma *Model.*

Description

The function extracts the expected values of regression coefficients from the fDMA model.

Usage

```
## S3 method for class 'dma'
coef(object, ...)
```

Arguments

object	an object of dma class
...	not used

Value

matrix of expected values of regression coefficients

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
c <- coef(object=m1)
```

`crudeoil` *Crude Oil Data.*

Description

Selected data from oil market.

Usage

```
data(crudeoil)
```

Format

`crudeoil` is `xts` object such that

- `crudeoil$p_oil` – average spot price of crude oil (Brent, Dubai and WTI) in USD per barrel
- `crudeoil$prod` – U.S. field production of crude oil in thousand barrels
- `crudeoil$cons` – U.S. product supplied of crude oil and petroleum products in thousand barrels
- `crudeoil$econ_act` – Index of Global Real Economic Activity
- `crudeoil$r` – U.S. 3-month treasury bill secondary market rate in %
- `crudeoil$stocks` – U.S. share prices index, 2015=100
- `crudeoil$risk` – Geopolitical risk (GPR) index
- `crudeoil$ex_rate` – U.S. real effective exchange rate index (broad basket), 2020=100

Details

The data are in monthly frequency. They cover the period between Jan, 1998 and Oct, 2024.

Source

The data are provided by Bank for International Settlements, Board of Governors of the Federal Reserve System, Caldara and Iacoviello (2022), Federal Reserve Bank of Dallas, OECD, U.S. Energy Information Administration and World Bank.

https://www.bis.org

https://www.dallasfed.org

https://www.eia.gov

https://www.federalreserve.gov

https://www.matteoiacoviello.com/gpr.htm

https://www.oecd.org/en.html

https://www.worldbank.org/ext/en/home

References

Bank for International Settlements, 2025. Effective exchange rates, BIS WS_EER 1.0 (data set). https://data.bis.org/topics/EER/BIS%2CWS_EER%2C1.0/M.R.B.US

Board of Governors of the Federal Reserve System, 2025. Selected interest rates. https://www.federalreserve.gov/releases/h15/

Caldara, D., Iacoviello, M., 2022. Measuring geopolitical risk. *American Economic Review* **112**, 1194–1225.

Federal Reserve Bank of Dallas, 2025. Index of global real economic activity. https://www.dallasfed.org/research/igrea

Kilian, L., 2009. Not all oil price shocks are alike: Disentangling demand and supply shocks in the crude oil market. *American Economic Review* **99**, 1053–1069.

OECD, 2025. Share prices. https://www.oecd.org/en/data/indicators/share-prices.html

U.S. Energy Information Administration, 2025. Petroleum /& other liquids. https://www.eia.gov/petroleum/data.php

World Bank, 2025. Commodity markets. https://www.worldbank.org/en/research/commodity-markets

Examples

```
data(crudeoil)
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
```

---

descstat *Computes Basic Descriptive Statistics.*

Description

This function computes descriptive statistics which are most useful for Dynamic Model Averaging.

It is a wrapper of `describe`.

If the argument is not a `matrix`, the function tries to convert the object into a `matrix`. For example, it works smoothly for `xts` objects.

Usage

```
descstat(data)
```

Arguments

`data` matrix, observations are put in rows, and variables are grouped by columns

Details

See describe.

Value

matrix

Examples

```
descstat(crudeoil)
```

---

dmtest — *Computes Diebold-Mariano Test.*

Description

This is a wrapper for dm.test from forecast package. This function computes the original Diebold-Mariano test.

Usage

```
dmtest(y,f)
```

Arguments

`y` vector of the forecasted time-series

`f` matrix of the predicted values from various methods, forecasts are ordered in rows, the first row should correspond to the method that is compared with alternative ones (corresponding to subsequent rows)

Details

The null hypothesis is that the two methods have the same forecast accuracy. This function assumes that one-step ahead forecasts are compared and the second power is used in the loss function (see dm.test). "The Diebold-Mariano (DM) test was intended for comparing forecasts; it has been, and remains, useful in that regard. The DM test was not intended for comparing models." (Diebold, 2015)

Value

matrix, first column contains tests statistics, next p-values are given for the alternative hypothesis that alternative forecasts have different accuracy than the compared forecast, alternative forecasts are less accurate and alternative forecasts have greater accuracy, tests outcomes for different forecasts are ordered by rows

References

Diebold, F. X., 2015. Comparing predictive accuracy, Twenty years later: A peersonal perspective on the use and abuse of Diebold-Mariano tests. *Journal of Business & Economic Statistics* **33**, doi:10.1080/07350015.2014.983236.

Diebold, F. X., Mariano, R. S., 1995. Comparing predictive accuracy. *Journal of Business & Economic Statistics* **13**, 253–263.

See Also

hmdmtest, mdmtest.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
m <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,initial.period=241)
m <- m$y.hat
a <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,initial.period=241)
a <- a$y.hat
a <- matrix(unlist(a),nrow=length(a),byrow=TRUE)
fc <- rbind(m,a)
dm <- dmtest(y=as.vector(ld.wti)[-(1:240)],f=fc[,-(1:240)])
```

---

fDMA *Computes Dynamic Model Averaging.*

Description

The function estimates Dynamic Model Averaging (and some of its variations). The method is described in Raftery et al. (2010).

Usage

```
fDMA(y,x,alpha,lambda,initvar,W=NULL,initial.period=NULL,V.meth=NULL,kappa=NULL,
gprob=NULL,omega=NULL,model=NULL,parallel=NULL,m.prior=NULL,mods.incl=NULL,
DOW=NULL,DOW.nmods=NULL,DOW.type=NULL,DOW.limit.nmods=NULL,progress.info=NULL,
forced.models=NULL,forbidden.models=NULL,forced.variables=NULL,bm=NULL,
small.c=NULL,fcores=NULL,mods.check=NULL,red.size=NULL,av=NULL)
```

Arguments

- `y` — numeric or a column matrix of a dependent variable, if `y` is xts object, then plots will have time index on the x axis
- `x` — matrix of independent variables, different columns correspond to different variables
- `alpha` — numeric, a forgetting factor α between 0 and 1 used in probabilities estimations
- `lambda` — numeric, a forgetting factor λ between 0 and 1 used in variance approximations
- `initvar` — numeric, initial variance in the state space equation, i.e., the number by which the unit matrix is multiplied
- `W` — optional, a method for setting the initial values of variance for the models equations, `W="reg"` corresponds to the method based on the linear regression as in the paper by Raftery et al. (2010), alternatively an arbitrary positive number (numeric) can be specified, by default the method of Raftery et al. (2010) is used
- `initial.period` — optional, numeric, a number of observation since which MSE (Mean Squared Error) and MAE (Mean Absolute Error) are computed, by default the whole sample is used, i.e., `initial.period=1`
- `V.meth` — optional, a method for the state space equation variance updating, `V.meth="rec"` corresponds to the recursive moment estimator, as in the paper by Raftery et al. (2010), `V.meth = "ewma"` corresponds to the exponentially weighted moving average as in, for example, Koop and Korobilis (2012), by default `V.meth = "rec"` is used
- `kappa` — optional, numeric, a parameter in the exponentially weighted moving average, between 0 and 1, used if `V.meth = "ewma"`
- `gprob` — optional, matrix of Google probabilities as in Koop and Onorante (2014), columns should correspond to columns of x
- `omega` — optional, numeric, a parameter between 0 and 1 used in probabilities estimations, used if `gprob` is specified
- `model` — optional, `model="dma"` for Dynamic Model Averaging, `model="dms"` for Dynamic Model Selection, or `model="med"` for Median Probability Model as in Barbieri and Berger (2004), by default `model="dma"` is used
- `parallel` — optional, logical, indicate whether parallel computations should be used, by default `parallel=FALSE`
- `m.prior` — optional, numeric, a parameter for general model prior (Mitchell and Beauchamp, 1988), by default `m.prior=0.5`, which corresponds to the uniform distribution, i.e., non-informative priors, see also Eicher et al. (2011)
- `mods.incl` — optional, matrix indicating which models should be used for estimation, the first column indicates inclusion of a constant, by default all possible models with a constant are used, inclusion of a variable is indicated by 1, omitting by 0
- `DOW` — optional, numeric, a threshold for Dynamic Occam's Window (Onorante and Raftery, 2016), should be a number between 0 and 1, if `DOW=0`, then no Dynamic Occam's Window is applied, by default `DOW=0`, Dynamic Occam's Window can be applied only to Dynamic Model Averaging, i.e., when `model="dma"`

DOW.nmods
: optional, numeric, initial number of models for Dynamic Occam's Window, should be less than the number of all possible models and larger than or equal to 2, they are randomly chosen, if DOW.nmods=0, then initially models with exactly one variable are taken, by default DOW.nmods=0

DOW.type
: optional, DOW.type="r" corresponds to DMA-R from Onorante and Raftery (2016), DOW.type="e" to DMA-E, by default DOW.type="r"

DOW.limit.nmods
: optional, numeric, maximum number of models selected by Dynamic Occam's Window, an additional limitation to the threshold given by DOW, by default no limit is set

progress.info
: optional, logical, applicable only if Dynamic Occam's Window is used, otherwise ignored, if progress.info=TRUE number of the current recursive DMA computation round and number of models selected for this round are printed, by default progress.info=FALSE

forced.models
: optional, matrix, applicable only if Dynamic Occam's Window is used, otherwise ignored, indicates models that have to be always included in the set of expanded models, similar as mods.incl, by default forced.models=NULL

forbidden.models
: optional, matrix, applicable only if Dynamic Occam's Window is used, otherwise ignored, indicates models that cannot be used in the set of expanded models, similar as mods.incl, by default forbidden.models=NULL

forced.variables
: optional, vector, applicable only if Dynamic Occam's Window is used, otherwise ignored, indicates variables that have to be always included in models constituting the set of expanded models, similar as mods.incl, first slot indicates inclusion of constant, by default forced.variables=NULL

bm
: optional, logical, indicate whether benchmark forecast should be computed, these benchmarks are naive forecast (all forecasts are set to be the value of the last observation) and Auto Arima auto.arima, by default bm=FALSE

small.c
: optional, numeric, small constant added to posterior model proabilities as in Raftery et al. (2010) to prevent potential reduction them to 0 due to the computational issues, if not specified the value computed as in Raftery et al. (2010) is taken

fcores
: optional, numeric, used only if parallel=TRUE, otherwise ignored, indicates the number of cores that should not be used, by default fcores=1

mods.check
: optional, logical, indicates if mods.incl should be checked for duplicated entries, etc., by default mods.check=FALSE

red.size
: optional, logical, indicates if outcomes should be reduced to save memory, by default red.size=FALSE

av
: optional, av="dma" corresponds to the original DMA averaging scheme, av="mse" corresponds to averaging based on Mean Squared Error, av="hr1" corresponds to averaging based on Hit Ratio, assuming time-series are in levels, av="hr2" corresponds to averaging based on Hit Ratio, assuming time-series represent changes, by default av="dma"

Details

It is possible to use numeric vector for lambda. Its values are automatically ordered in descending order and if numbers are not unique they are reduced to become unique. If more than one value is given for lambda, then model state space, i.e., mods.incl, is expanded by considering all these models with given values of lambda. The outcomes are then ordered by columns in a way that first outcomes from models with first value of lambda are presented, then from models with second value of lambda, etc. (Raftery et al., 2010).

If nrow(gprob)<length(y), then the method by Koop and Onorante (2014) is used for the last nrow(gprob) observations. For the preceding ones the original method by Raftery et al. (2010) is used. In such case a warning is generated.

Value

class dma object, list of

$y.hat	forecasted values
$post.incl	posterior inclusion probabilities for independent variables
$MSE	Mean Squared Error of forecast
$MAE	Mean Absolute Error of forecast
$models	models included in estimations, or models used in the last step of Dynamic Occam's Window method (if this method has been selected)
$post.mod	posterior probabilities of all used models, or NA if Dynamic Occam's Window method has been selected
$exp.var	expected number of variables (incl. constant)
$exp.coef.	expected values of regression coefficients
$parameters	parameters of the estimated model
$yhat.all.mods	predictions from every sub-model used in estimations
$y	y, dependent variable
$benchmarks	Root Mean Squared Error and Mean Absolute Error of naive and auto ARIMA forecast
$DOW.init.mods	models initially selected to Dynamic Occam's Window, if this method has been selected
$DOW.n.mods.t	number of models used in Dynamic Model Averaging at time t, if Dynamic Occam's Window method has been selected
$p.dens.	predicitive densities from the last period of all sub-models used in estimations
$exp.lambda	expected values of lambda parameter

Source

Raftery, A. E., Karny, M., Ettler, P., 2010. Online prediction under model uncertainty via Dynamic Model Averaging: Application to a cold rolling mill. *Technometrics* **52**, 52–66.

References

Barbieri, M. M., Berger, J. O., 2004. Optimal predictive model selection. *The Annals of Statistics* **32**, 870–897.

Eicher, T. S., Papageorgiou, C., Raftery, A. E., 2011. Default priors and predictive performance in Bayesian Model Averaging, with application to growth determinants. *Journal of Applied Econometrics* **26**, 30–55.

Koop, G., Korobilis, D., 2012. Forecasting inflation using Dynamic Model Averaging. *International Economic Review* **53**, 867–886.

Koop, G., Korobilis, D., 2018. Variational Bayes inference in high-dimensional time-varying parameter models. https://arxiv.org/pdf/1809.03031

Koop, G., Onorante, L., 2014. Macroeconomic nowcasting using Google probabilities. https://goo.gl/ATsBN9

Mitchell, T. J., Beauchamp, J. J., 1988. Bayesian variable selection in linear regression (with discussion). *Journal of the American Statistical Association* **83**, 1023–1036.

Onorante, L., Raftery, A. E., 2016. Dynamic model averaging in large model spaces using dynamic Occam's window. *European Economic Review* **81**, 2–14.

Yin, X., Peng, J., Tang, T., 2018. Improving the forecasting accuracy of crude oil prices. *Sustainability* **10**, 454. doi:10.3390/su10020454

See Also

grid.DMA, print.dma, summary.dma, plot.dma, hit.ratio.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
m2 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,model="dms")
m3 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,V.meth="ewma",kappa=0.9)
m4 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,DOW=0.7)


# Google trends are available since 2004

gp <- trends/100
s <- ld.drivers['2004-01-01/']
m5 <- fDMA(y=ld.wti['2004-01-01/'],x=s,alpha=0.99,lambda=0.90,initvar=10,gprob=gp,omega=0.5)


# models just with one independent variable and a constant will be averaged
mds <- diag(1,ncol(ld.drivers),ncol(ld.drivers))
```

```
mds <- cbind(rep(1,ncol(ld.drivers)),mds)

m6 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,mods.incl=mds)


# models just with one independent variable (without a constant) will be averaged
mds.nc <- diag(1,ncol(ld.drivers),ncol(ld.drivers))
mds.nc <- cbind(rep(0,ncol(ld.drivers)),mds.nc)

m7 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,mods.incl=mds.nc)

# model with multiple lambda

m8 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=c(0.99,0.95,0.90),initvar=10)
```

---

fitted.dma *Extracts Fitted Values from* dma *Model.*

Description

The function extracts predictions made by the fDMA model.

Usage

```
## S3 method for class 'dma'
fitted(object, ...)
```

Arguments

object	an object of dma class
...	not used

Value

vector of forecasted values

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
f <- fitted(object=m1)
```

gNormalize — *Normalizes a Numeric Matrix by Rows.*

Description

For example, Google Trends data are given as numbers between 0 and 100. If the Users divide them by 100, they can be interpreted in a certain sense as probabilities.

However, if there are such probabilities for several variables, sometimes it might be desirable to have the sum of these probabilities for all variables to sum up to 1. This function does not divide the values of an argument by 100, but rescales every row to sum up to 1. In other words, values in each row of an argument are divided by the sum of all values in this row.

Usage

```
gNormalize(data)
```

Arguments

`data`	`matrix`, observations are put in rows, and variables are grouped by columns

Value

`matrix`

References

Koop, G., Onorante, L., 2014. Macroeconomic nowcasting using Google probabilities. https://goo.gl/ATsBN9

Examples

```
gt <- gNormalize(trends)

gNormalize(rbind(c(0,1,2),c(1,2,3)))
```

grid.DMA — *Computes `fDMA` Function for Multiple Values of `alpha` and `lambda`.*

Description

Sometimes it is necessary to consider various values of parameters `alpha` and `lambda` in Dynamic Model Averaging (or Dynamic Model Selection, etc.). This function computes `fDMA` function for all combinations of `alpha` and `lambda` for given grids.

This function is a wrapper of `fDMA`.

Usage

```
grid.DMA(y,x,grid.alpha,grid.lambda,initvar,W=NULL,initial.period=NULL,V.meth=NULL,
kappa=NULL,gprob=NULL,omega=NULL,model=NULL,parallel.grid=NULL,m.prior=NULL,
mods.incl=NULL,DOW=NULL,DOW.nmods=NULL,DOW.type=NULL,DOW.limit.nmods=NULL,
forced.models=NULL,forbidden.models=NULL,forced.variables=NULL,bm=NULL,
small.c=NULL,av=NULL)
```

Arguments

y	see fDMA
x	see fDMA
grid.alpha	a numeric vector of different values of alpha
grid.lambda	a numeric vector of different values of lambda or a list of numeric vectors for multiple lambda in one model (see fDMA)
initvar	see fDMA
W	see fDMA
initial.period	see fDMA
V.meth	see fDMA
kappa	see fDMA
gprob	see fDMA
omega	see fDMA
model	see fDMA
parallel.grid	optional, logical, indicate whether parallel computations should be used, by default parallel.grid=FALSE
m.prior	see fDMA
mods.incl	see fDMA
DOW	see fDMA
DOW.nmods	see fDMA
DOW.type	see fDMA
DOW.limit.nmods	see fDMA
forced.models	see fDMA
forbidden.models	see fDMA
forced.variables	see fDMA
bm	see fDMA
small.c	see fDMA
av	see fDMA

Value

an object of class `grid.dma`, `list` of

`$models`	`list` of `list` of models
`$RMSE`	`matrix` with Root Mean Squared Error (RMSE) for all estimated models
`$MAE`	`matrix` with Mean Absolute Error (MAE) for all estimated models

See Also

`print.grid.dma`, `summary.grid.dma`, `plot.grid.dma`.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

gra <- c(0.99,0.98,0.97)
grl <- c(0.99,0.95)
g1 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=10)


# extract model with alpha=0.97 and lambda=0.95
model <- g$models[[3]][[2]]

# models with various multiple lambdas

gra <- c(0.99,0.98,0.97)
grl <- list(c(0.99,0.95,0.90),c(0.99,0.98,0.97,0.96,0.95))
g2 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=10)
```

`grid.roll.reg` *Computes* `roll.reg` *Function for Multiple Values of* `window`.

Description

Sometimes it is necessary to consider various values of parameter `window` in Rolling Regression. This function computes `roll.reg` function for all values of `window` for a given grid.

This function is a wrapper of `roll.reg`.

Usage

```
grid.roll.reg(y,x=NULL,grid.window,parallel.grid=NULL,c=NULL)
```

Arguments

y	see roll.reg
x	see roll.reg
grid.window	a numeric vector of different values of window, see roll.reg
parallel.grid	optional, logical, indicate whether parallel computations should be used, by default parallel=FALSE
c	optional, see roll.reg

Value

an object of class grid.roll.reg, list of

$models	list of reg objects
$fq	matrix with Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for all estimated models

See Also

print.grid.roll.reg, summary.grid.roll.reg, plot.grid.roll.reg.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grw <- c(50,100,150)
g <- grid.roll.reg(y=ld.wti,x=ld.drivers,grid.window=grw)

# extract model with window=100
model <- g$models[[2]]
```

---

grid.tvp — *Computes* tvp *Function for Multiple Values of* lambda.

Description

Sometimes it is necessary to consider various values of parameter lambda in Time-Varying Parameters Regression. This function computes tvp function for all values of lambda for a given grid.

This function is a wrapper of tvp.

Usage

```
grid.tvp(y,x,V,grid.lambda,W=NULL,kappa=NULL,parallel.grid=NULL,c=NULL)
```

Arguments

y	see tvp
x	see tvp
V	see tvp
grid.lambda	a numeric vector of different values of lambda, see tvp
W	optional, see tvp
kappa	optional, see tvp
parallel.grid	optional, logical, indicate whether parallel computations should be used, by default parallel=FALSE
c	optional, see tvp

Value

an object of class grid.tvp, list of

$models	list of tvp objects
$fq	matrix with Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for all estimated models

See Also

print.grid.tvp, summary.grid.tvp, plot.grid.tvp.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grl <- c(0.99,0.98,0.97,0.96,0.95)
g <- grid.tvp(y=ld.wti,x=ld.drivers,V=1,grid.lambda=grl)

# extract model with lambda=0.95
model <- g$models[[5]]
```

`hit.ratio` *Computes Hit Ratio (HR) for Forecast.*

Description

Sometimes it is interesting to analyze just whether the forecast can predict the direction of a change in a modelled time-series. This function computes the proportion of correctly predicted signs (i.e., in which cases the direction of a change given by forecast agrees with the change in real data).

Usage

```
hit.ratio(y,y.hat,d=NULL)
```

Arguments

y	`numeric`, `vector`, or one row or one column `matrix` or `xts` object, representing a forecasted time-series
y.hat	`numeric`, `vector`, or one row or one column `matrix` or `xts` object, representing forecast predictions
d	optional, `logical`, `d=FALSE` for level time-series, `d=TRUE` if time-series already represent changes, by default `d=FALSE`

Value

`numeric`

References

Baur, D. G., Beckmann, J., Czudaj, R., 2016. A melting pot – Gold price forecasts under model and parameter uncertainty. *International Review of Financial Analysis* **48**, 282–291.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=wti,x=drivers[,5:7],alpha=0.99,lambda=0.99,initvar=10)
hit.ratio(y=wti,y.hat=m1$y.hat)

m2 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=10)
hit.ratio(y=ld.wti,y.hat=m2$y.hat,d=TRUE)
```

hmdmtest — *Computes Diebold-Mariano Test when Presence of ARCH Effects is Suspected.*

Description

This is a wrapper for `dm.test` from `forecast` package. This function computes the modified Diebold-Mariano test. The modification is useful if the presence of ARCH effects is suspected in forecast errors. It is also useful for small samples. This is a modification of `mdmtest` for the presence of ARCH effects in forecast errors.

Usage

```
hmdmtest(y,f)
```

Arguments

y	`vector` of the forecasted time-series
f	`matrix` of the predicted values from various methods, forecasts are ordered in rows, the first row should correspond to the method that is compared with alternative ones (corresponding to subsequent rows)

Details

The null hypothesis is that the two methods have the same forecast accuracy. This function assumes that one-step ahead forecasts are compared and the second power is used in the loss function (see `dm.test`).

Value

`matrix`, first column contains tests statistics, next p-values are given for the alternative hypothesis that alternative forecasts have different accuracy than the compared forecast, alternative forecasts are less accurate and alternative forecasts have greater accuracy, tests outcomes for different forecasts are ordered by rows

References

Newbold, P., Harvey, D. J., 2002. Forecast combinations. In: Clements, M. P., Hendry, D. F. (eds.), *A Companion to Economic Forecasting*, Blackwell Publishing Ltd.

See Also

`archtest`, `dmtest`, `mdmtest`.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
m <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,initial.period=241)
m <- m$y.hat
a <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,initial.period=241)
a <- a$y.hat
a <- matrix(unlist(a),nrow=length(a),byrow=TRUE)
fc <- rbind(m,a)
hmdm <- hmdmtest(y=as.vector(ld.wti)[-(1:240)],f=fc[,-(1:240)])
```

---

mdmtest *Computes Harvey-Leybourne-Newbold Test.*

Description

This is a wrapper for `dm.test` from `forecast` package. This function computes the modified Diebold-Mariano test. The modification is useful for small samples.

Usage

```
mdmtest(y,f)
```

Arguments

y	`vector` of the forecasted time-series
f	`matrix` of the predicted values from various methods, forecasts are ordered in rows, the first row should correspond to the method that is compared with alternative ones (corresponding to subsequent rows)

Details

The null hypothesis is that the two methods have the same forecast accuracy. This function assumes that one-step ahead forecasts are compared and the second power is used in the loss function (see `dm.test`).

Value

`matrix`, first column contains tests statistics, next p-values are given for the alternative hypothesis that alternative forecasts have different accuracy than the compared forecast, alternative forecasts are less accurate and alternative forecasts have greater accuracy, tests outcomes for different forecasts are ordered by rows

References

Harvey, D., Leybourne, S., Newbold, P., 1997. Testing the equality of prediction mean squared errors. *International Journal of Forecasting* **13**, 281–291.

See Also

dmtest, hmdmtest.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
m <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,initial.period=241)
m <- m$y.hat
a <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,initial.period=241)
a <- a$y.hat
a <- matrix(unlist(a),nrow=length(a),byrow=TRUE)
fc <- rbind(m,a)
mdm <- mdmtest(y=as.vector(ld.wti)[-(1:240)],f=fc[,-(1:240)])
```

---

normalize — *Normalizes a Numeric Matrix by Columns.*

Description

For a variable considered to be used in Dynamic Model Averaging (or Dynamic Model Selection, etc.), sometimes it is desirable to have all its values between 0 and 1. This function rescales the values to fit between 0 and 1.

If the argument is not a matrix, the function tries to convert the object into a matrix. For example, it works smoothly for xts objects.

Usage

```
normalize(data)
```

Arguments

data	matrix, observations are put in rows, and variables are grouped by columns

Value

matrix

See Also

standardize

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]

nwti <- normalize(wti)

nd <- normalize(drivers)

normalize(cbind(c(0,1,2),c(1,2,3),c(0,1,3)))
```

---

onevar *Creates a matrix of one-variable models.*

Description

This function simplifies working with one-variable models in, for example, fDMA. It produces a matrix corresponding to the set of models consisting of models with a constant and just one extra variable, and a model with a constant only.

Usage

```
onevar(x)
```

Arguments

x	matrix of independent variables, see mods.incl in fDMA

Value

matrix, inclusion of a variable is indicated by 1, omitting by 0

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

mds <- diag(1,ncol(ld.drivers),ncol(ld.drivers))
mds <- cbind(rep(1,ncol(ld.drivers)),mds)
mds <- rbind(rep(0,ncol(mds)),mds)
mds[1,1] <- 1
```

```
m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,mods.incl=mds)

# Equivalently:

m2 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,mods.incl=onevar(ld.drivers))
```

---

plot.altf *Plots Selected Outcomes from* altf *Object.*

Description

The function plots selected outcomes from altf object.

Usage

```
## S3 method for class 'altf'
plot(x,non.interactive=NULL, ...)
```

Arguments

x an object of altf class

non.interactive
optional, logical, indicate whether plots should be made in non-interactive mode, by default non.interactive=FALSE, i.e., the user specifies in the interactive menu which plots will be made

... not used

Details

After executing the command, the User is asked to choose

1 - for plotting regression coefficients in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

2 - for plotting p-values for t-test of statistical significance for regression coefficients from applied models, in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory).

Chosing 0 exits the plot command.

If non.interactive=TRUE all the above plots are made.

Value

Called for making a plot.

Note

Coefficients are plotted only for rec. OLS, roll. OLS, TVP, TVP-AR(1) and TVP-AR(2) models. P-values – for rec. OLS and roll. OLS.

It is suggested to execute `graphics.off` before exectuing `plot` command for `altf` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `altf` object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf(y=wti,x=drivers)

plot(a,non.interactive=TRUE)
```

`plot.altf2` *Plots Selected Outcomes from* `altf2` *Object.*

Description

The function plots selected outcomes from `altf2` object.

Usage

```
## S3 method for class 'altf2'
plot(x,non.interactive=NULL, ...)
```

Arguments

`x`	an object of `altf2` class
`non.interactive`	optional, `logical`, indicate whether plots should be made in non-interactive mode, by default `non.interactive=FALSE`, i.e., the user specifies in the interactive menu which plots will be made
`...`	not used

Details

After executing the command, the User is asked to choose

1 - for plotting expected coefficients in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

2 - for plotting p-values (averaged over selected models) for t-test of statistical significance for regression coefficients from applied models, in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

3 - for plotting weights of all models used in averaging,

4 - for plotting relative variable importance in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

5 - for plotting expected number of variables (incl. constant) from all models used in averaging.

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute `graphics.off` before exectuing `plot` command for `altf2` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `altf2` object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf2(y=wti,x=drivers[,5:7],av="aic")

plot(a,non.interactive=TRUE)
```

`plot.altf3` *Plots Selected Outcomes from* `altf3` *Object.*

Description

The function plots selected outcomes from `altf3` object.

Usage

```
## S3 method for class 'altf3'
plot(x,non.interactive=NULL, ...)
```

Arguments

`x`	an object of `altf3` class
`non.interactive`	optional, `logical`, indicate whether plots should be made in non-interactive mode, by default `non.interactive=FALSE`, i.e., the user specifies in the interactive menu which plots will be made
`...`	not used

Details

After executing the command, the User is asked to choose

1 - for plotting expected coefficients in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

2 - for plotting p-values (averaged over selected window sizes) for t-test of statistical significance for coefficients from a rolling regression, in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

3 - for plotting weights of all models used in averaging,

4 - for plotting expected window size.

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute `graphics.off` before exectuing `plot` command for `altf3` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `altf3` object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf3(y=wti,x=drivers,windows=c(36,100,150))

plot(a,non.interactive=TRUE)
```

`plot.altf4` *Plots Selected Outcomes from* `altf4` *Object.*

Description

The function plots selected outcomes from `altf4` object.

Usage

```
## S3 method for class 'altf4'
plot(x,non.interactive=NULL, ...)
```

Arguments

`x` an object of `altf4` class

`non.interactive`
optional, `logical`, indicate whether plots should be made in non-interactive mode, by default `non.interactive=FALSE`, i.e., the user specifies in the interactive menu which plots will be made

`...` not used

Details

After executing the command, the User is asked to choose

1 - for plotting expected coefficients in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

2 - for plotting weights of all models used in averaging,

3 - for plotting expected window size.

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute `graphics.off` before exectuing `plot` command for `altf4` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `altf4` object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf4(y=wti,x=drivers,windows=c(36,100,150))

plot(a,non.interactive=TRUE)
```

---

plot.dma *Plots Selected Outcomes from fDMA Function.*

Description

The function plots selected outcomes from fDMA.

Usage

```
## S3 method for class 'dma'
plot(x,non.interactive=NULL, ...)
```

Arguments

x	an object of dma class
non.interactive	optional, logical, indicate whether plots should be made in non-interactive mode, by default non.interactive=FALSE, i.e., the user specifies in the interactive menu which plots will be made
...	not used

Details

If x comes from estimation of Dynamic Model Averaging (DMA), after executing the command, the User is asked to choose

1 - for plotting actual and predicted values,

2 - for plotting residuals,

3 - for plotting the expected number of variables (including constant),

4 - for plotting posterior inclusion probabilities (including constant) on one plot,

5 - for plotting posterior inclusion probabilities (including constant) in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

6 - for plotting expected coefficients (including constant) on one plot,

7 - for plotting expected coefficients (including constant) in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

8 - for plotting the expected value of `lambda`,

9 - for plotting posterior model probabilities, if Dynamic Occam's Window method has not been selected, or plotting the number of models used in Dynamic Model Averaging, if Dynamic Occam's Window method has been selected.

Chosing 0 exits the `plot` command.

If `x` comes from estimation of Dynamic Model Selection (DMS) or Median Probability Model (MED), after executing `plot` the User is asked to choose

1 - for plotting actual and predicted values,

2 - for plotting residuals,

3 - for plotting the expected number of variables (including constant),

4 - for producing a plot showing which variables (including constant) are included in the DMS or MED model in each time,

5 - for plotting expected coefficients (including constant) on one plot,

6 - for plotting expected coefficients (including constant) in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

7 - for plotting the expected value of `lambda` (only for DMS).

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute `graphics.off` before exectuing `plot` command for `dma` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `dma` object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
m2 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dms")

# graphics.off()
```

```
plot(m1,non.interactive=TRUE)
# graphics.off()
plot(m2,non.interactive=TRUE)
```

---

plot.grid.dma *Plots Selected Outcomes from* grid.DMA *Function.*

Description

The function plots selected outcomes from grid.DMA.

Usage

```
## S3 method for class 'grid.dma'
plot(x,non.interactive=NULL, ...)
```

Arguments

x	an object of grid.dma class
non.interactive	optional, logical, indicate whether plots should be made in non-interactive mode, by default non.interactive=FALSE, i.e., the user specifies in the interactive menu which plots will be made
...	not used

Details

If x comes from estimation of Dynamic Model Averaging (DMA), after executing the command, the User is asked to choose

1 - for plotting Root Mean Squared Error (RMSE) for all estimated models,

2 - for plotting Mean Absolute Error (MAE) for all estimated models,

3 - for plotting posterior inclusion probabilities (including constant) for all estimated models, the outcomes are saved in separate png files in the temporary directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the temporary directory),

4 - for plotting expected coefficients (including constant) for all estimated models, the outcomes are saved in separate png files in the temporary directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the temporary directory).

Chosing 0 exits the plot command.

If x comes from estimation of Dynamic Model Selection (DMS) or Median Probability Model (MED), after executing the command, the User is asked to choose

1 - for plotting Root Mean Squared Error (RMSE) for all estimated models,

2 - for plotting Mean Absolute Error (MAE) for all estimated models,

3 - for plotting expected coefficients (including constant) for all estimated models, the outcomes are saved in separate png files in the temporary directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the temporary directory).

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute `graphics.off` before exectuing `plot` command for `grid.dma` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `grid.dma` object, sometimes a legend might cover the important parts of the plot.

If any of the models comes from using multiple lambda (see `fDMA`), then RMSE and MAE are not plotted.

Also, if `length(grid.alpha)` or `length(grid.lambda)` is less than 2, then RMSE and MAE are not plotted.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
gra <- c(0.99,0.98,0.97)
grl <- c(0.99,0.95)

g1 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=1)
g2 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=1,model="dms")

# graphics.off()
plot(g1,non.interactive=TRUE)
# graphics.off()
plot(g2,non.interactive=TRUE)
```

plot.grid.roll.reg *Plots Selected Outcomes from grid.roll.reg Function.*

Description

The function plots selected outcomes from grid.roll.reg.

Usage

```
## S3 method for class 'grid.roll.reg'
plot(x,non.interactive=NULL, ...)
```

Arguments

x	an object of grid.roll.reg class
non.interactive	optional, logical, indicate whether plots should be made in non-interactive mode, by default non.interactive=FALSE, i.e., the user specifies in the interactive menu which plots will be made
...	not used

Details

After executing the command, the User is asked to choose

1 - for plotting Root Mean Squared Error (RMSE) for all estimated models,

2 - for plotting Mean Absolute Error (MAE) for all estimated models,

3 - for plotting coefficients (including constant) for all estimated models, the outcomes are saved in separate png files in the temporary directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the temporary directory),

4 - for plotting p-values for t-test of statistical significance for regression coefficients for all estimated models, the outcomes are saved in separate png files in the temporary directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the temporary directory),

Chosing 0 exits the plot command.

If non.interactive=TRUE all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute graphics.off before exectuing plot command for grid.roll.reg object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If graphics.off is not executed before plotting grid.roll.reg object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grw <- c(50,100,150)
g <- grid.roll.reg(y=ld.wti,x=ld.drivers,grid.window=grw)

plot(g,non.interactive=TRUE)
```

---

plot.grid.tvp *Plots Selected Outcomes from grid.tvp Function.*

Description

The function plots selected outcomes from grid.tvp.

Usage

```
## S3 method for class 'grid.tvp'
plot(x,non.interactive=NULL, ...)
```

Arguments

x	an object of grid.tvp class
non.interactive	optional, logical, indicate whether plots should be made in non-interactive mode, by default non.interactive=FALSE, i.e., the user specifies in the interactive menu which plots will be made
...	not used

Details

After executing the command, the User is asked to choose

1 - for plotting Root Mean Squared Error (RMSE) for all estimated models,

2 - for plotting Mean Absolute Error (MAE) for all estimated models,

3 - for plotting coefficients (including constant) for all estimated models, the outcomes are saved in separate png files in the temporary directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the temporary directory).

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute `graphics.off` before exectuing `plot` command for `grid.tvp` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `grid.tvp` object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grl <- c(0.99,0.98,0.97,0.96,0.95)
g <- grid.tvp(y=ld.wti,x=ld.drivers,V=1,grid.lambda=grl)

plot(g,non.interactive=TRUE)
```

`plot.reg` *Plots Selected Outcomes from* `reg` *Object.*

Description

The function plots selected outcomes from `reg` object.

Usage

```
## S3 method for class 'reg'
plot(x,non.interactive=NULL, ...)
```

Arguments

`x` an object of `reg` class

`non.interactive` optional, `logical`, indicate whether plots should be made in non-interactive mode, by default `non.interactive=FALSE`, i.e., the user specifies in the interactive menu which plots will be made

`...` not used

Details

After executing the command, the User is asked to choose

1 - for plotting actual and predicted values,

2 - for plotting residuals,

3 - for plotting regression coefficients on one plot,

4 - for plotting regression coefficients in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory),

5 - for plotting p-values for t-test of statistical significance for regression coefficients on one plot,

6 - for plotting p-values for t-test of statistical significance for regression coefficients in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory).

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute `graphics.off` before exectuing `plot` command for `reg` object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If `graphics.off` is not executed before plotting `reg` object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
```

```
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

roll <- roll.reg(y=ld.wti,x=ld.drivers,window=100)

rec <- rec.reg(y=ld.wti,x=ld.drivers)

# graphics.off()
plot(roll,non.interactive=TRUE)

# graphics.off()
plot(rec,non.interactive=TRUE)
```

`plot.tvp` *Plots Selected Outcomes from* `tvp` *Object.*

Description

The function plots selected outcomes from `tvp` object.

Usage

```
## S3 method for class 'tvp'
plot(x,non.interactive=NULL, ...)
```

Arguments

`x`	an object of `tvp` class
`non.interactive`	optional, `logical`, indicate whether plots should be made in non-interactive mode, by default `non.interactive=FALSE`, i.e., the user specifies in the interactive menu which plots will be made
`...`	not used

Details

After executing the command, the User is asked to choose

1 - for plotting actual and predicted values,

2 - for plotting residuals,

3 - for plotting regression coefficients on one plot,

4 - for plotting regression coefficients in separate png files, saved in the temporary directory, and moreover, to paste them into one big plot (also saved as a png file in the temporary directory).

Chosing 0 exits the `plot` command.

If `non.interactive=TRUE` all the above plots are made.

Value

Called for making a plot.

Note

It is suggested to execute graphics.off before exectuing plot command for tvp object. However, the User should take care to save all other plots before executing this command, as they can be lost.

If graphics.off is not executed before plotting tvp object, sometimes a legend might cover the important parts of the plot.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

tvp <- tvp(y=ld.wti,x=ld.drivers,V=1,lambda=0.99)

# graphics.off()
plot(tvp,non.interactive=TRUE)
```

---

predict.dma *Computes Predictions from* dma *Model.*

Description

The function computes predictions based on the model obtained from fDMA.

Usage

```
## S3 method for class 'dma'
predict(object, newdata, type, ...)
```

Arguments

object	an object of dma class
newdata	a matrix as x object in fDMA
type	type="backward" computes predictions of y with the already estimated coefficients, but with x given by newdata, type="forward" computes predictions of y with the coefficients estimated in the last period, for various combinations of x given in rows of newdata
...	not used

Value

vector of forecasted values

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
p1 <- predict(object=m1,newdata=ld.drivers,type="backward")
p2 <- predict(object=m1,newdata=ld.drivers[1,],type="forward")
p3 <- predict(object=m1,newdata=ld.drivers[1:3,],type="forward")
```

---

print.altf *Prints* altf *Object.*

Description

The function prints selected outcomes obtained from altf.

Usage

```
## S3 method for class 'altf'
print(x, ...)
```

Arguments

x	an object of altf class
...	not used

Details

The function prints forecast quality measures from x. For details see accuracy.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf(y=wti,x=drivers)

print(a)
```

---

print.altf2 *Prints* altf2 *Object.*

Description

The function prints selected outcomes obtained from altf2.

Usage

```
## S3 method for class 'altf2'
print(x, ...)
```

Arguments

x	an object of altf2 class
...	not used

Details

The function prints forecast quality measures from x. For details see accuracy.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf2(y=wti,x=drivers[,5:7])

print(a)
```

print.altf3 — *Prints* `altf3` *Object.*

Description

The function prints selected outcomes obtained from `altf3`.

Usage

```
## S3 method for class 'altf3'
print(x, ...)
```

Arguments

`x`	an object of `altf3` class
`...`	not used

Details

The function prints forecast quality measures from `x`. For details see `accuracy`.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf3(y=wti,x=drivers,windows=c(36,100,150))

print(a)
```

print.altf4 — *Prints* `altf4` *Object.*

Description

The function prints selected outcomes obtained from `altf4`.

Usage

```
## S3 method for class 'altf4'
print(x, ...)
```

Arguments

x	an object of `altf4` class
...	not used

Details

The function prints forecast quality measures from `x`. For details see `accuracy`.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf4(y=wti,x=drivers,windows=c(36,100,150))

print(a)
```

---

print.dma *Prints* dma *Object.*

Description

The function prints selected outcomes obtained from `fDMA`.

Usage

```
## S3 method for class 'dma'
print(x, ...)
```

Arguments

x	an object of dma class
...	not used

Details

The function prints parameters of an argument `x`, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) from the estimated model. It also shows the number of observations, the number of models in averaging (selecting) procedure and the number of variables (including constant) used in the model. The number of models does not include the increase, if multiple `lambda` is used. The function also shows forecast quality measures for alternative forecasting methods, i.e., naive forecast (see also `altf`) and, if computed, for Auto ARIMA `auto.arima`.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
m2 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dms")


print(m1)
print(m2)
```

---

print.grid.dma *Prints* grid.dma *Object.*

Description

The function prints selected outcomes obtained from grid.DMA.

Usage

```
## S3 method for class 'grid.dma'
print(x, ...)
```

Arguments

x	an object of grid.dma class
...	not used

Details

The function prints Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for all estimated models.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

gra <- c(0.99,0.98,0.97)
grl <- c(0.99,0.95)
g1 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=1)
g2 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=1,model="dms")


print(g1)
print(g2)
```

`print.grid.roll.reg` *Prints* `grid.roll.reg` *Object.*

Description

The function prints selected outcomes obtained from `grid.roll.reg`.

Usage

```
## S3 method for class 'grid.roll.reg'
print(x, ...)
```

Arguments

`x`	an object of `grid.roll.reg` class
`...`	not used

Details

The function prints Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for all estimated models.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grw <- c(50,100,150)
g <- grid.roll.reg(y=ld.wti,x=ld.drivers,grid.window=grw)

print(g)
```

---

print.grid.tvp   *Prints* `grid.tvp` *Object.*

Description

The function prints selected outcomes obtained from `grid.tvp`.

Usage

```
## S3 method for class 'grid.tvp'
print(x, ...)
```

Arguments

`x`	an object of `grid.tvp` class
`...`	not used

Details

The function prints Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for all estimated models.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
```

```
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grl <- c(0.99,0.98,0.97,0.96,0.95)
g <- grid.tvp(y=ld.wti,x=ld.drivers,V=1,grid.lambda=grl)

print(g)
```

---

print.reg *Prints* reg *Object.*

Description

The function prints selected outcomes obtained from roll.reg and rec.reg.

Usage

```
## S3 method for class 'reg'
print(x, ...)
```

Arguments

x	an object of reg class
...	not used

Details

The function prints mean regression coefficients from the analyzed period, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) from the estimated model. For roll.reg it also shows the size of a rolling window.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
roll <- roll.reg(y=ld.wti,x=ld.drivers,window=100)
rec <-  rec.reg(y=ld.wti,x=ld.drivers)
print(roll)
print(rec)
```

print.tvp *Prints* tvp *Object.*

Description

The function prints selected outcomes obtained from tvp.

Usage

```
## S3 method for class 'tvp'
print(x, ...)
```

Arguments

x	an object of tvp class
...	not used

Details

The function prints mean regression coefficients from the analyzed period, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) from the estimated model.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
tvp <- tvp(y=ld.wti,x=ld.drivers,V=1,lambda=0.99)
print(tvp)
```

`rec.reg` *Computes Recursive Regression.*

Description

This function computes Recursive Regression.

Usage

```
rec.reg(y,x=NULL,c=NULL)
```

Arguments

y	`numeric` or a column `matrix` of a dependent variable
x	`matrix` of independent variables, different columns should correspond to different variables, if not specified only a constant will be used
c	optional, `logical`, a parameter indicating whether constant is included, if not specified `c=TRUE` is used, i.e., constant is included

Details

It might happen during computations that `lm` (which is used inside `rec.reg`) will produce `NA` or `NaN`. In such a case regression coefficients for a given period are taken as 0 and p-values for t-test for statistical significance of regression coefficients are taken as 1.

It is not possible to set `c=FALSE` if `x=NULL`. In such a case the function will automatically reset `c=TRUE` inside the code.

Value

class `reg` object, `list` of

`$y.hat`	fitted (forecasted) values
`$AIC`	Akaike Information Criterion (from the current set of observations)
`$AICc`	Akaike Information Criterion with a correction for finite sample sizes (from the current set of observations)
`$BIC`	Bayesian Information Criterion (from the current set of observations)
`$MSE`	Mean Squared Error (from the current set of observations)
`$coeff.`	regression coefficients
`$p.val`	p-values for t-test for statistical significance of regression coefficients
`$y`	y, forecasted time-series

See Also

`print.reg`, `summary.reg`, `plot.reg`.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
rec1 <- rec.reg(y=ld.wti,x=ld.drivers)
rec2 <- rec.reg(y=ld.wti)
```

---

reduce.size *Reduces the Size of* `fDMA` *or* `grid.DMA` *Outcomes.*

Description

This functions reduces the size of `dma` or `grid.dma` object.

Usage

```
reduce.size(dma.object)
```

Arguments

`dma.object`	`dma` or `grid.dma` object

Details

The information corresponding to each sub-model is erased. In particular, for the object produced by `fDMA` `$models` is reduced to one-row matrix to keep only `colnames`, and `$postmod`, `$yhat.all.mods` and `$p.dens.` are replaced by `NA`. It can be useful if large number of models is considered.

Value

`dma` or `grid.dma` object, with the information corresponding to each sub-model erased

See Also

`fDMA`, `grid.DMA`.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
```

```
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
m2 <- reduce.size(m1)
```

residuals.dma *Extracts Residuals from* dma *Model.*

Description

The function extracts residuals from the fDMA model.

Usage

```
## S3 method for class 'dma'
residuals(object, ...)
```

Arguments

object	an object of dma class
...	not used

Value

vector of residuals

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
r <- residuals(object=m1)
```

`roll.reg` *Computes Rolling Regression.*

Description

This function computes Rolling Regression. For the first `window-1` observations Recursive Regression is computed. Since `window`-th observation the rolling is performed.

Usage

```
roll.reg(y,x=NULL,window,c=NULL)
```

Arguments

`y`	`numeric` or a column `matrix` of a dependent variable
`x`	`matrix` of independent variables, different columns should correspond to different variables, if not specified only a constant will be used
`window`	`numeric`, a size of a window for rolling
`c`	optional, `logical`, a parameter indicating whether constant is included, if not specified `c=TRUE` is used, i.e., constant is included

Details

It might happen during computations that `lm` (which is used inside `roll.reg`) will produce `NA` or `NaN`. In such a case regression coefficients for a given period are taken as 0 and p-values for t-test for statistical significance of regression coefficients are taken as 1.

It is not possible to set `c=FALSE` if `x=NULL`. In such a case the function will automatically reset `c=TRUE` inside the code.

Value

class `reg` object, `list` of

`$y.hat`	fitted (forecasted) values
`$AIC`	Akaike Information Criterion (from the current window size)
`$AICc`	Akaike Information Criterion with a correction for finite sample sizes (from the current window size)
`$BIC`	Bayesian Information Criterion (from the current window size)
`$MSE`	Mean Squared Error (from the current window size)
`$coeff.`	regression coefficients
`$p.val`	p-values for t-test for statistical significance of regression coefficients
`$window`	window size
`$y`	`y`, forecasted time-series

See Also

grid.roll.reg, print.reg, summary.reg, plot.reg.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
roll1 <- roll.reg(y=ld.wti,x=ld.drivers,window=100)
roll2 <- roll.reg(y=ld.wti,window=100)
```

---

rvi — *Extracts Relative Variable Importances from fDMA Model.*

Description

This functions extracts posterior inclusion probabilities for independent variables from dma object.

Usage

```
rvi(dma.object)
```

Arguments

dma.object	dma object

Value

matrix of posterior inclusion probabilities for independent variables

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
r <- rvi(m1)
```

`standardize` *Standardizes a Numeric Matrix by Columns.*

Description

Sometimes it is desirable to have all variables to have mean 0 and standard deviation 1. This function rescales the values in such a way.

If the argument is not a `matrix`, the function tries to convert the object into a `matrix`. For example, it works smoothly for `xts` objects.

Usage

```
standardize(data)
```

Arguments

`data`	`matrix`, observations are put in rows, and variables are grouped by columns

Value

`matrix`

See Also

`normalize`

Examples

```
standardize(crudeoil)
```

`stest` *Computes a Few Stationarity Tests.*

Description

This is a wrapper for three functions from `tseries` package. Augmented Dickey-Fuller (ADF, `adf.test`), Phillips-Perron (PP, `pp.test`) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS, `kpss.test`) tests for stationarity are performed.

Usage

```
stest(data)
```

Arguments

`data`	`matrix` of variables, different columns correspond to different variables

Value

matrix, tests statistics and p-values are given by columns, tests outcomes for different variables are ordered by rows

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
x <- cbind(ld.wti,ld.drivers)
stest(x)
```

---

summary.altf *Summarizes Outcomes from* altf *Object.*

Description

The function summarizes selected outcomes obtained from altf.

Usage

```
## S3 method for class 'altf'
summary(object, ...)
```

Arguments

object	an object of altf class
...	not used

Details

The function produces the outcomes as print.altf.

Additionally, it provides mean values of coefficients and how often p-values for t-test of statistical significance for each independent variable in the model are below 1%, 5% and 10%, respectively.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf(y=wti,x=drivers)

summary(a)
```

---

summary.altf2 *Summarizes Outcomes from* altf2 *Object.*

Description

The function summarizes selected outcomes obtained from altf2.

Usage

```
## S3 method for class 'altf2'
summary(object, ...)
```

Arguments

object	an object of altf2 class
...	not used

Details

The function produces the outcomes as print.altf2.

Additionally, it provides mean values of coefficients, min, max and mean relative variable importance for each independent variable in the model, frequency when relative variable importance is over 0.5 for each independent variable in the model, and how often p-values (averaged over selected models) for t-test of statistical significance for each independent variable in the model are below 1%, 5% and 10%, respectively.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf2(y=wti,x=drivers[,5:7],initial.period=60)

summary(a)
```

summary.altf3 *Summarizes Outcomes from* altf3 *Object.*

Description

The function summarizes selected outcomes obtained from altf3.

Usage

```
## S3 method for class 'altf3'
summary(object, ...)
```

Arguments

object	an object of altf3 class
...	not used

Details

The function produces the outcomes as print.altf3.

Additionally, it provides mean values of coefficients and how often p-values (averaged over selected window sizes) for t-test of statistical significance for each independent variable in the model are below 1%, 5% and 10%, respectively.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf3(y=wti,x=drivers,windows=c(36,100,150))

summary(a)
```

summary.altf4 *Summarizes Outcomes from* `altf4` *Object.*

Description

The function summarizes selected outcomes obtained from `altf4`.

Usage

```
## S3 method for class 'altf4'
summary(object, ...)
```

Arguments

`object`	an object of `altf4` class
`...`	not used

Details

The function produces the outcomes as `print.altf4`.

Additionally, it provides mean values of coefficients.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf4(y=wti,x=drivers,windows=c(36,100,150))

summary(a)
```

summary.dma *Summarizes Outcomes from* `dma` *Object.*

Description

The function summarizes outcomes obtained from `fDMA`.

Usage

```
## S3 method for class 'dma'
summary(object, ...)
```

Arguments

`object`	an object of `dma` class
`...`	not used

Details

The function produces the outcomes as `print.dma`.

Additionally:

If `object` comes from Dynamic Model Averaging (DMA), it shows how often (in comparision to the whole analyzed period) a posterior inclusion probability for a given variable exceeds 1/2. It also shows minimum, maximum and mean posterior inclusion probability for every variable throughout the analyzed period.

If `object` comes from Dynamic Model Selection (DMS) or Median Probability Model (MED), it shows how often (in comparision to the whole analyzed period) a given variable is present in the selected model.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

m1 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
m2 <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dms")


summary(m1)
summary(m2)
```

`summary.grid.dma` *Summarizes Outcomes from* `grid.dma` *Objects.*

Description

The function summarizes outcomes obtained from `grid.DMA`.

Usage

```
## S3 method for class 'grid.dma'
summary(object, ...)
```

Arguments

`object`	an object of `grid.dma` class
`...`	not used

Details

The function produces the outcomes as `print.grid.dma`.

Additionally, it finds the indices for a model minimizing Root Mean Squared Error (RMSE) and for a model minimizing Mean Absolute Error (MAE).

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

gra <- c(0.99,0.98,0.97)
grl <- c(0.99,0.95)
g1 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=1)
g2 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=grl,initvar=1,model="dms")


summary(g1)
summary(g2)
```

`summary.grid.roll.reg` *Summarizes Outcomes from* `grid.roll.reg` *Objects.*

Description

The function summarizes outcomes obtained from `grid.roll.reg`.

Usage

```
## S3 method for class 'grid.roll.reg'
summary(object, ...)
```

Arguments

`object`	an object of `grid.roll.reg` class
`...`	not used

Details

The function produces the outcomes as `print.grid.roll.reg`.

Additionally, it finds the model minimizing Root Mean Squared Error (RMSE) and minimizing Mean Absolute Error (MAE).

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grw <- c(50,100,150)
g <- grid.roll.reg(y=ld.wti,x=ld.drivers,grid.window=grw)

summary(g)
```

`summary.grid.tvp` *Summarizes Outcomes from* `grid.tvp` *Objects.*

Description

The function summarizes outcomes obtained from `grid.tvp`.

Usage

```
## S3 method for class 'grid.tvp'
summary(object, ...)
```

Arguments

`object`	an object of `grid.tvp` class
`...`	not used

Details

The function produces the outcomes as `print.grid.tvp`.

Additionally, it finds the model minimizing Root Mean Squared Error (RMSE) and minimizing Mean Absolute Error (MAE).

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

grl <- c(0.99,0.98,0.97,0.96,0.95)
g <- grid.tvp(y=ld.wti,x=ld.drivers,V=1,grid.lambda=grl)

summary(g)
```

---

summary.reg *Summarizes Outcomes from* reg *Object.*

Description

The function summarizes selected outcomes obtained from roll.reg and rec.reg.

Usage

```
## S3 method for class 'reg'
summary(object, ...)
```

Arguments

object	an object of reg class
...	not used

Details

The function produces the outcomes as print.reg.

Additionally, it provides how often p-values for t-test of statistical significance for each independent variable in the model is below 1%, 5% and 10%, respectively.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
roll <- roll.reg(y=ld.wti,x=ld.drivers,window=100)
rec <- rec.reg(y=ld.wti,x=ld.drivers)
summary(roll)
summary(rec)
```

---

summary.tvp — *Summarizes Outcomes from* `tvp` *Object.*

Description

The function summarizes selected outcomes obtained from `tvp`.

Usage

```
## S3 method for class 'tvp'
summary(object, ...)
```

Arguments

`object`	an object of `tvp` class
`...`	not used

Details

The function produces the outcomes as `tvp`.

Value

Called for printing.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
tvp <- tvp(y=ld.wti,x=ld.drivers,V=1,lambda=0.99)
summary(tvp)
```

trends — *Google Trends for Crude Oil Data.*

Description

Google Trends for Crude Oil Data.

Usage

```
data(trends)
```

Format

`trends` is `xts` object such that

- `trends$prod` – Google Trends for "oil production"
- `trends$cons` – Google Trends for "oil consumption"
- `trends$econ_act` – Google Trends for "economic activity"
- `trends$r` – Google Trends for "interest rate"
- `trends$stocks` – Google Trends for "stock markets"
- `trends$risk` – Google Trends for "market stress"
- `trends$ex_rate` – Google Trends for "exchange rate"

Details

The data are in monthly frequency. They cover the period between Jan, 2004 and Oct, 2024.

Source

The data are provided by Google.

https://trends.google.com/trends

Examples

```
data(trends)
gtrends <- trends/100
data(crudeoil)
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100
ld.wti <- ld.wti['2004-01-01-/']
ld.drivers <- ld.drivers['2004-01-01-/']
```

```
xx <- ld.drivers
m <- fDMA(y=ld.wti,x=xx,alpha=0.99,lambda=0.99,initvar=1,model="dma",gprob=gtrends,omega=0.5)
```

---

tvp *Computes Time-Varying Parameters Regression.*

Description

This function computes Time-Varying Parameters Regression (TVP) with the updating procedure as in Raftery et. al (2010).

Usage

```
tvp(y,x,V,lambda,W=NULL,kappa=NULL,c=NULL)
```

Arguments

y	numeric or a column matrix of a dependent variable
x	matrix of independent variables, different columns should correspond to different variables
V	numeric, initial variance in the state space equation for the recursive moment estimator updating method, as in Raftery et al. (2010)
lambda	numeric, a forgetting factor between 0 and 1 used in variance approximations
W	optional, numeric, initial value of variance for the model equations, if not specified the method based on the linear regression, as in Raftery et al. (2010) is used
kappa	optional, numeric, a parameter in the exponentially weighted moving average in variance updating (see also fDMA), between 0 and 1, if not specified the method as in Raftery et al. (2010) is used
c	optional, logical, a parameter indicating whether constant is included, if not specified c=TRUE is used, i.e., constant is included

Details

It is not possible to set c=FALSE if ncol(x)=0. In such a case the function will automatically reset c=TRUE inside the code.

Value

class tvp object, list of

$y.hat	fitted (forecasted) values
$thetas	estimated regression coefficients
$pred.dens.	predicitive densities from each period
$y	y, forecasted time-series

References

Raftery, A. E., Karny, M., Ettler, P., 2010. Online prediction under model uncertainty via Dynamic Model Averaging: Application to a cold rolling mill. *Technometrics* **52**, 52–66.

Sanderson, C., Curtin, R., 2016. Armadillo: A template-based C++ library for linear algebra. *Journal of Open Source Software* **1**, https://arma.sourceforge.net/armadillo_joss_2016.pdf.

See Also

grid.tvp, print.tvp, summary.tvp, plot.tvp.

Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- drivers[-1,]
ld.drivers[,c(4,6)] <- (diff(drivers[,c(4,6)]))[-1,]
ld.drivers[,c(1:2,5,7)] <- (diff(log(drivers[,c(1:2,5,7)])))[-1,]
ld.drivers[,c(3,6)] <- ld.drivers[,c(3,6)]/100

t1 <- tvp(y=ld.wti,x=ld.drivers,V=1,lambda=0.99)



t2 <- tvp(y=ld.wti,x=ld.drivers,V=1,lambda=0.99,W=1)



t3 <- tvp(y=ld.wti,x=ld.drivers,V=1,lambda=0.99,W=1,kappa=0.75)


# Model with constant only
empty <- matrix(,nrow=nrow(ld.drivers),ncol=0)
t4 <- tvp(y=ld.wti,x=empty,lambda=0.99,V=1)
```

Index