

# Package ‘fdMA’

September 30, 2018

**Type** Package

**Title** Dynamic Model Averaging and Dynamic Model Selection for Continuous Outcomes

**Version** 2.2.4

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

**Suggests** R.rsp

**VignetteBuilder** R.rsp

**LinkingTo** Rcpp, RcppArmadillo

**Date** 2018-09-29

**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>.

**License** GPL-3

**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** 2018-09-29 22:00:02 UTC

**R topics documented:**

|                               |    |
|-------------------------------|----|
| altf . . . . .                | 3  |
| altf2 . . . . .               | 5  |
| altf3 . . . . .               | 8  |
| altf4 . . . . .               | 10 |
| archtest . . . . .            | 12 |
| coef.dma . . . . .            | 13 |
| crudeoil . . . . .            | 14 |
| descstat . . . . .            | 15 |
| dmtest . . . . .              | 16 |
| fDMA . . . . .                | 17 |
| fitted.dma . . . . .          | 21 |
| gNormalize . . . . .          | 22 |
| grid.DMA . . . . .            | 23 |
| grid.roll.reg . . . . .       | 25 |
| grid.tvp . . . . .            | 26 |
| hit.ratio . . . . .           | 27 |
| hmdmtest . . . . .            | 28 |
| mdmtest . . . . .             | 29 |
| normalize . . . . .           | 30 |
| onevar . . . . .              | 31 |
| plot.altf . . . . .           | 32 |
| plot.altf2 . . . . .          | 33 |
| plot.altf3 . . . . .          | 34 |
| plot.altf4 . . . . .          | 36 |
| plot.dma . . . . .            | 37 |
| plot.grid.dma . . . . .       | 39 |
| plot.grid.roll.reg . . . . .  | 40 |
| plot.grid.tvp . . . . .       | 42 |
| plot.reg . . . . .            | 43 |
| plot.tvp . . . . .            | 44 |
| predict.dma . . . . .         | 46 |
| print.altf . . . . .          | 46 |
| print.altf2 . . . . .         | 47 |
| print.altf3 . . . . .         | 48 |
| print.altf4 . . . . .         | 49 |
| print.dma . . . . .           | 49 |
| print.grid.dma . . . . .      | 50 |
| print.grid.roll.reg . . . . . | 51 |
| print.grid.tvp . . . . .      | 52 |
| print.reg . . . . .           | 53 |
| print.tvp . . . . .           | 53 |
| rec.reg . . . . .             | 54 |
| reduce.size . . . . .         | 55 |
| residuals.dma . . . . .       | 56 |
| roll.reg . . . . .            | 57 |
| rvi . . . . .                 | 58 |

|                                 |    |
|---------------------------------|----|
| standardize . . . . .           | 59 |
| stest . . . . .                 | 59 |
| summary.altf . . . . .          | 60 |
| summary.altf2 . . . . .         | 61 |
| summary.altf3 . . . . .         | 61 |
| summary.altf4 . . . . .         | 62 |
| summary.dma . . . . .           | 63 |
| summary.grid.dma . . . . .      | 64 |
| summary.grid.roll.reg . . . . . | 65 |
| summary.grid.tvp . . . . .      | 65 |
| summary.reg . . . . .           | 66 |
| summary.tvp . . . . .           | 67 |
| trends . . . . .                | 68 |
| tvps . . . . .                  | 69 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>71</b> |
|--------------|-----------|

---

|      |  |
|------|--|
| altf | <i>Computes a Few Alternative Forecasts.</i> |
|------|--|

---

## 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 [tvps](#) with  $V=1$  and  $\lambda=0.99$ .

Auto ARIMA (auto ARIMA) is computed as [auto.arima](#).

Markov Switching Models (MS) are computed as [msmFit](#). The object is a simple linear regression model. 2 regimes are assumed. All coefficients have switching.

ME (Mean Error), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), MPE (Mean Percentage Error) 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  |
| 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), TVP-AR(2) and MS – should be computed, if not specified f=c(rep(TRUE, 11)), 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**

```
## Not run:
# models for untransformed data
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
a1 <- altf(y=wti, x=drivers)

# do not include first 12 observations for forecast quality measures,
```

```

# i.e., treat first 12 observations as a 'training set'
a2 <- altf(y=wti,x=drivers,initial.period=12)

# models for log-differenced data
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
a3 <- altf(y=ld.wti,x=ld.drivers,d=TRUE)

# models where constant term is not included in modelled equations (if applicable)
a4 <- altf(y=ld.wti,x=ld.drivers,d=TRUE,c=FALSE)

# compute just selected models
fcomp <- c(TRUE,TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE,FALSE)
a5 <- altf(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp)

m1 <- fdma(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
a6 <- altf(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,fmod=m1)

## End(Not run)

```

---

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 Error) 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

|                        |  |
|------------------------|--|
| <code>y</code>         | <a href="#">numeric</a> or a column <a href="#">matrix</a> of a dependent variable   |
| <code>x</code>         | <a href="#">matrix</a> of independent variables, different columns correspond to different independent variables   |
| <code>mods.incl</code> | optional, <a href="#">matrix</a> indicating which models will be used in averaging, if not specified all possible models will be used, see <a href="#">fdma</a>            |
| <code>gprob</code>     | optional, <a href="#">matrix</a> of Google probabilities as in Koop and Onorante (2014), columns should correspond to columns of <code>x</code> , see <a href="#">fdma</a> |

|                |   |
|----------------|---|
| omega          | optional, <b>numeric</b> , a parameter between 0 and 1 used in probabilities estimations, used if <code>gprob</code> is specified, see <code>fDMA</code>  |
| av             | optional, a method for model averaging, <code>av="ord"</code> corresponds to equal weights for each model, <code>av="aic"</code> corresponds to information theoretic model averaging based on Akaike Information Criterion, <code>av="aicc"</code> corresponds to information theoretic model averaging based on Akaike Information Criterion with a correction for finite sample sizes, <code>av="bic"</code> corresponds to information theoretic model averaging based on Bayesian Information Criterion, <code>av="mse"</code> corresponds to setting weights proportional to the inverse of the models Mean Squared Error, if not specified <code>av="ord"</code> is used |
| window         | optional, <b>numeric</b> , a size of a rolling regression window (a number of observations), if not specified 10% of all observations are taken   |
| initial.period | optional, <b>numeric</b> , a number of observation since which forecast quality measures are computed, if not specified the whole sample is used, i.e., <code>initial.period=1</code>   |
| d              | optional, <b>logical</b> , a parameter used for HR (Hit Ratio) calculation, should be <code>d=FALSE</code> for level time-series and <code>d=TRUE</code> if time-series represent changes, if not specified <code>d=FALSE</code>  |
| f              | optional, <b>logical</b> vector, indicating which of alternative forecast – av. OLS, av. rec. OLS, av. roll. OLS and av. TVP – should be averaged, if not specified <code>f=c(rep(TRUE, 4))</code> , i.e., all alternative forecast are computed  |
| fmod           | optional, class <code>dma</code> object, a model to be compared with alternative forecast   |
| parallel       | optional, <b>logical</b> , indicate whether parallel computations should be used, by default <code>parallel=FALSE</code>  |

## 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 whole sample) is taken for each period.

If `gprob` is used, then for OLS mean values from all periods 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

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

|                |  |
|----------------|--|
| \$p.val.       | <a href="#">list</a> 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. | <a href="#">list</a> of relative variable importance from all forecasting methods which were applied   |
| \$exp.var.     | <a href="#">list</a> 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

```
## Not run:
# models for untransformed data
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a1 <- altf2(y=wti,x=drivers)

# do not include first 12 observations for forecast quality measures,
# i.e., treat first 12 observations as a 'training set'
a2 <- altf2(y=wti,x=drivers,initial.period=12)

# models for log-differenced data
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
a3 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE)

fcomp <- c(TRUE,TRUE,TRUE,FALSE)
a4 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp)
a5 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,av="aic")

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

```

a6 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,f=fcomp,fmod=m1)

# 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)
a7 <- altf2(y=ld.wti,x=ld.drivers,d=TRUE,mods.incl=mds)

# Google trends are available since 2004
gp <- trends/100
s1 <- ld.wti['2004-01-01/']
s2 <- ld.drivers['2004-01-01/']
a8 <- altf2(y=s1,x=s2,d=TRUE,gprob=gp,omega=0.5)

## End(Not run)

```

---

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 Error) 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 <a href="#">matrix</a> of a dependent variable   |
| x       | <a href="#">matrix</a> of independent variables, different columns correspond to different independent variables, if not specified only constant term will be included   |
| windows | <a href="#">numeric vector</a> , 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 <a href="#">numeric</a> then weights are computed proportional to the av-th power of window size, if not specified av="ord" is used |



|                             |   |
|-----------------------------|---|
| <code>initial.period</code> | optional, <a href="#">numeric</a> , a number of observation since which forecast quality measures are computed, if not specified the whole sample is used, i.e., <code>initial.period=1</code>  |
| <code>d</code>              | optional, <a href="#">logical</a> , a parameter used for HR (Hit Ratio) calculation, should be <code>d=FALSE</code> for level time-series and <code>d=TRUE</code> if time-series represent changes, if not specified <code>d=FALSE</code> |
| <code>fmod</code>           | optional, class <code>dma</code> object, a model to be compared with alternative forecast   |
| <code>parallel</code>       | optional, <a href="#">logical</a> , indicate whether parallel computations should be used, by default <code>parallel=FALSE</code>   |
| <code>c</code>              | optional, see <a href="#">roll.reg</a>  |

### 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

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

```
## Not run:
# models for untransformed data
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
a1 <- altf3(y=wti, x=drivers, windows=c(36, 100, 150))
```

```

# do not include first 12 observations for forecast quality measures,
# i.e., treat first 12 observations as a 'training set'
a2 <- altf3(y=wti,x=drivers,initial.period=12,windows=c(36,100,150))

# models for log-differenced data
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
a3 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,windows=c(36,100,150))

a4 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,av="aic",windows=c(36,100,150))

a5 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,av=-2,windows=c(36,100,150))

# models without a constant term
a6 <- altf3(y=ld.wti,x=ld.drivers,d=TRUE,av=-2,windows=c(36,100,150),c=FALSE)

# models only with a constant term
a7 <- altf3(y=ld.wti,d=TRUE,av=-2,windows=c(36,100,150))

## End(Not run)

```

---

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 choosing 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 Error) 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        | <a href="#">numeric vector</a> , sizes of a rolling regression windows (numbers of observations)   |
| V              | optional, <a href="#">numeric</a> , 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 <a href="#">tvp</a> |
| lambda         | optional, <a href="#">numeric</a> , a forgetting factor between 0 and 1 used in variance approximations, if not specified lambda=0.99 is taken, see <a href="#">tvp</a>  |
| alpha          | optional, <a href="#">numeric</a> , a forgetting factor $\alpha$ between 0 and 1 used in probabilities estimations, if not specified alpha=0.99 is taken, see <a href="#">fdMA</a>                                       |
| initial.period | optional, <a href="#">numeric</a> , 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, <a href="#">logical</a> , 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, <a href="#">logical</a> , indicate whether parallel computations should be used, by default parallel=FALSE   |
| c              | optional, see <a href="#">tvp</a>  |
| small.c        | optional, see <a href="#">fdMA</a>   |

## Value

class altf4 object, [list](#) of

|            |  |
|------------|--|
| \$summary  | <a href="#">matrix</a> of forecast quality measures ordered by columns   |
| \$y.hat    | <a href="#">list</a> of predicted values from a time-varying parameters rolling regression averaged over selected window sizes |
| \$y        | y, forecasted time-series  |
| \$coeff.   | <a href="#">list</a> of coefficients from a time-varying parameters rolling regression averaged over selected window sizes     |
| \$weights  | <a href="#">list</a> of weights of models used in averaging  |
| \$exp.win. | <a href="#">list</a> 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

```
## Not run:
# models for untransformed data
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
a1 <- altf4(y=wti, x=drivers, windows=c(36, 100, 150))

# do not include first 12 observations for forecast quality measures,
# i.e., treat first 12 observations as a 'training set'
a2 <- altf4(y=wti, x=drivers, initial.period=12, windows=c(36, 100, 150))

# models for log-differenced data
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
a3 <- altf4(y=ld.wti, x=ld.drivers, d=TRUE, windows=c(36, 100, 150))

win <- c(36, 100, 150)
a4 <- altf4(y=ld.wti, x=ld.drivers, d=TRUE, windows=win, alpha=0.9, lambda=0.95)

# models without a constant term
a5 <- 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)
a6 <- altf4(y=ld.wti, x=empty, d=TRUE, windows=win, alpha=0.9, lambda=0.95)

## End(Not run)
```

---

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               |

**Examples**

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
m1 <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
c <- coef(object=m1)

## End(Not run)
```

---

`crudeoil`*Crude Oil Data.*

---

**Description**

Selected data from oil market.

**Usage**

```
data(crudeoil)
```

**Format**

`crudeoil` is `xts` object such that

- `crudeoil$WTI` – WTI spot price in USD per barrel
- `crudeoil$MSCI` – MSCI World Index
- `crudeoil$TB3MS` – U.S. 3-month treasury bill secondary market rate in %
- `crudeoil$CSP` – Crude steel production in thousand tonnes
- `crudeoil$TWEXM` – Trade weighted U.S. dollar index (Mar, 1973 = 100)
- `crudeoil$PROD` – U.S. product supplied for crude oil and petroleum products in thousands of barrels
- `crudeoil$CONS` – Total consumption of petroleum products in OECD in quad BTU
- `crudeoil$VXO` – Implied volatility of S&P 100

**Details**

The data are in monthly frequency. They cover the period between Jan, 1990 and Dec, 2016.

**Source**

The data are provided by CBOE, Federal Reserve Bank of St. Louis, MSCI, U.S. Energy Information Administration and World Steel Association.

<https://www.cboe.com/products/vix-index-volatility/vix-options-and-futures/vix-index/vix-historical-data>

<https://www.eia.gov>

<https://fred.stlouisfed.org>

<https://goo.gl/P1WZXb>

<https://www.msci.com/end-of-day-data-search>

## Examples

```
data(crudeoil)
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
m <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")

## End(Not run)
```

---

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 personal perspective on the use and abuse of Diebold-Mariano tests. *Journal of Business & Economic Statistics* **33**, <http://dx.doi.org/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

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
m <- fDMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
m <- m$y.hat
a <- altf2(y=ld.wti,x=ld.drivers,d=TRUE)
a <- a$y.hat
a <- matrix(unlist(a),nrow=length(a),byrow=TRUE)
fc <- rbind(m,a)
dm <- dmtest(y=as.vector(ld.wti),f=fc)

## End(Not run)
```

---

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 $\alpha$ between 0 and 1 used in probabilities estimations  |
| lambda  | numeric, a forgetting factor $\lambda$ 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 |

|                              |   |
|------------------------------|---|
| <code>initial.period</code>  | optional, <b>numeric</b> , 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., <code>initial.period=1</code>  |
| <code>V.meth</code>          | optional, a method for the state space equation variance updating, <code>V.meth="rec"</code> corresponds to the recursive moment estimator, as in the paper by Raftery et al. (2010), <code>V.meth = "ewma"</code> corresponds to the exponentially weighted moving average as in, for example, Koop and Korobilis (2012), by default <code>V.meth = "rec"</code> is used |
| <code>kappa</code>           | optional, <b>numeric</b> , a parameter in the exponentially weighted moving average, between 0 and 1, used if <code>V.meth = "ewma"</code>  |
| <code>gprob</code>           | optional, <b>matrix</b> of Google probabilities as in Koop and Onorante (2014), columns should correspond to columns of <code>x</code>  |
| <code>omega</code>           | optional, <b>numeric</b> , a parameter between 0 and 1 used in probabilities estimations, used if <code>gprob</code> is specified   |
| <code>model</code>           | optional, <code>model="dma"</code> for Dynamic Model Averaging, <code>model="dms"</code> for Dynamic Model Selection, or <code>model="med"</code> for Median Probability Model as in Barbieri and Berger (2004), by default <code>model="dma"</code> is used  |
| <code>parallel</code>        | optional, <b>logical</b> , indicate whether parallel computations should be used, by default <code>parallel=FALSE</code>  |
| <code>m.prior</code>         | optional, <b>numeric</b> , a parameter for general model prior (Mitchell and Beauchamp, 1988), by default <code>m.prior=0.5</code> , which corresponds to the uniform distribution, i.e., non-informative priors, see also Eicher et al. (2011)   |
| <code>mods.incl</code>       | optional, <b>matrix</b> 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  |
| <code>DOW</code>             | optional, <b>numeric</b> , a threshold for Dynamic Occam's Window (Onorante and Raftery, 2016), should be a number between 0 and 1, if <code>DOW=0</code> , then no Dynamic Occam's Window is applied, by default <code>DOW=0</code> , Dynamic Occam's Window can be applied only to Dynamic Model Averaging, i.e., when <code>model="dma"</code>                         |
| <code>DOW.nmods</code>       | optional, <b>numeric</b> , 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 <code>DOW.nmods=0</code> , then initially models with exactly one variable are taken, by default <code>DOW.nmods=0</code>   |
| <code>DOW.type</code>        | optional, <code>DOW.type="r"</code> corresponds to DMA-R from Onorante and Raftery (2016), <code>DOW.type="e"</code> to DMA-E, by default <code>DOW.type="r"</code>   |
| <code>DOW.limit.nmods</code> | optional, <b>numeric</b> , maximum number of models selected by Dynamic Occam's Window, an additional limitation to the threshold given by <code>DOW</code> , by default no limit is set  |
| <code>progress.info</code>   | optional, <b>logical</b> , applicable only if Dynamic Occam's Window is used, otherwise ignored, if <code>progress.info=TRUE</code> number of the current recursive DMA computation round and number of models selected for this round are printed, by default <code>progress.info=FALSE</code>   |
| <code>forced.models</code>   | optional, <b>matrix</b> , 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 <code>mods.incl</code> , by default <code>forced.models=NULL</code>  |

|                               |  |
|-------------------------------|--|
| <code>forbidden.models</code> | optional, <b>matrix</b> , 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 <code>mods.incl</code> , by default <code>forbidden.models=NULL</code>  |
| <code>forced.variables</code> | optional, <b>vector</b> , 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 <code>mods.incl</code> , first slot indicates inclusion of constant, by default <code>forced.variables=NULL</code>   |
| <code>bm</code>               | optional, <b>logical</b> , 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 <code>auto.arima</code> , by default <code>bm=FALSE</code>   |
| <code>small.c</code>          | optional, <b>numeric</b> , small constant added to posterior model probabilities 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  |
| <code>fcores</code>           | optional, <b>numeric</b> , used only if <code>parallel=TRUE</code> , otherwise ignored, indicates the number of cores that should not be used, by default <code>fcores=1</code>  |
| <code>mods.check</code>       | optional, <b>logical</b> , indicates if <code>mods.incl</code> should be checked for duplicated entries, etc., by default <code>mods.check=FALSE</code>  |
| <code>red.size</code>         | optional, <b>logical</b> , indicates if outcomes should be reduced to save memory, by default <code>red.size=FALSE</code>  |
| <code>av</code>               | optional, <code>av="dma"</code> corresponds to the original DMA averaging scheme, <code>av="mse"</code> corresponds to averaging based on Mean Squared Error, <code>av="hr1"</code> corresponds to averaging based on Hit Ratio, assuming time-series are in levels, <code>av="hr2"</code> corresponds to averaging based on Hit Ratio, assuming time-series represent changes, by default <code>av="dma"</code> |

## 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

|                          |   |
|--------------------------|---|
| <code>\$y.hat</code>     | forecasted values   |
| <code>\$post.incl</code> | posterior inclusion probabilities for independent variables |
| <code>\$MSE</code>       | 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.       | predictive 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. <https://doi.org/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 <- (diff(log(drivers)))[-1,]
## Not run:
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)

## End(Not run)

# Google trends are available since 2004
## Not run:
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)

## End(Not run)

# 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)
## Not run:
m6 <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,mods.incl=mds)

## End(Not run)

# 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)
## Not run:
m7 <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10,mods.incl=mds.nc)

## End(Not run)

# model with multiple lambda
## Not run:
m8 <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=c(0.99,0.95,0.90),initvar=10)

## End(Not run)
```

**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               |

**Examples**

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
m1 <- fdma(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
f <- fitted(object=m1)

## End(Not run)
```

---

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 | <a href="#">matrix</a> , 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 | <i>Computes <a href="#">fdMA</a> Function for Multiple Values of alpha and lambda.</i> |
|----------|--|

---

## 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 <a href="#">fdMA</a>   |
| x              | see <a href="#">fdMA</a>   |
| grid.alpha     | a <a href="#">numeric vector</a> of different values of alpha  |
| grid.lambda    | a <a href="#">numeric vector</a> of different values of lambda or a <a href="#">list</a> of numeric vectors for multiple lambda in one model (see <a href="#">fdMA</a> ) |
| initvar        | see <a href="#">fdMA</a>   |
| W              | see <a href="#">fdMA</a>   |
| initial.period | see <a href="#">fdMA</a>   |
| V.meth         | see <a href="#">fdMA</a>   |
| kappa          | see <a href="#">fdMA</a>   |
| gprob          | see <a href="#">fdMA</a>   |
| omega          | see <a href="#">fdMA</a>   |
| model          | see <a href="#">fdMA</a>   |

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 <- (diff(log(drivers)))[-1,]
## Not run:
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)

## End(Not run)

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

# models with various multiple lambdas
```



```
## Not run:
gra <- c(0.99,0.98,0.97)
gr1 <- 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=gr1,initvar=10)

## End(Not run)
```

---

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 <a href="#">roll.reg</a>   |
| x             | see <a href="#">roll.reg</a>   |
| grid.window   | a numeric <a href="#">vector</a> of different values of window, see <a href="#">roll.reg</a>                         |
| parallel.grid | optional, <a href="#">logical</a> , indicate whether parallel computations should be used, by default parallel=FALSE |
| c             | optional, see <a href="#">roll.reg</a>   |

### Value

an object of class `grid.roll.reg`, [list](#) of

|          |   |
|----------|---|
| \$models | <a href="#">list</a> of reg objects   |
| \$fq     | <a href="#">matrix</a> 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 <- (diff(log(drivers)))[-1,]
## Not run:
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]]

## End(Not run)

```

---

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 <a href="#">tvp</a>  |
| x             | see <a href="#">tvp</a>  |
| V             | see <a href="#">tvp</a>  |
| grid.lambda   | a numeric <a href="#">vector</a> of different values of lambda, see <a href="#">tvp</a>                              |
| W             | optional, see <a href="#">tvp</a>  |
| kappa         | optional, see <a href="#">tvp</a>  |
| parallel.grid | optional, <a href="#">logical</a> , indicate whether parallel computations should be used, by default parallel=FALSE |
| c             | optional, see <a href="#">tvp</a>  |

**Value**

an object of class `grid.tvp`, [list](#) of

|          |   |
|----------|---|
| \$models | <a href="#">list</a> of <a href="#">tvp</a> objects   |
| \$fq     | <a href="#">matrix</a> 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 <- (diff(log(drivers)))[-1,]
## Not run:
gr1 <- c(0.99,0.98,0.97,0.96,0.95)
g <- grid.tvp(y=ld.wti,x=ld.drivers,V=1,grid.lambda=gr1)

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

## End(Not run)
```

---

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 <- (diff(log(drivers)))[-1,]
## Not run:
m1 <- fdMA(y=wti, x=drivers, 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)

## End(Not run)
```

---

|          |  |
|----------|--|
| hmdmtest | <i>Computes Diebold-Mariano Test when Presence of ARCH Effects is Suspected.</i> |
|----------|--|

---

## 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

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
m <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
m <- m$y.hat
a <- altf2(y=ld.wti,x=ld.drivers,d=TRUE)
a <- a$y.hat
fc <- matrix(unlist(a),nrow=length(a),byrow=TRUE)
hmdm <- hmdmtest(y=as.vector(ld.wti),f=fc)

## End(Not run)
```

---

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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
m <- fdMA(y=ld.wti, x=ld.drivers, alpha=0.99, lambda=0.90, initvar=10)
m <- m$y.hat
a <- altf2(y=ld.wti, x=ld.drivers, d=TRUE)
a <- a$y.hat
a <- matrix(unlist(a), nrow=length(a), byrow=TRUE)
fc <- rbind(m, a)
mdm <- mdmtest(y=as.vector(ld.wti), f=fc)

## End(Not run)
```

---

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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]

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))

## End(Not run)
```

---

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, <b>logical</b> , 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working directory).

Chosing 0 exits the plot command.

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

### 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 executing 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

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf(y=wti,x=drivers)

plot(a)

## End(Not run)
```

---

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, <b>logical</b> , 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working directory),

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

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

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

Choosing 0 exits the plot command.

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

## Note

It is suggested to execute `graphics.off` before executing 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

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf2(y=wti,x=drivers,av="aic")

plot(a)

## End(Not run)
```

---

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, <b>logical</b> , 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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.

**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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf3(y=wti,x=drivers,windows=c(36,100,150))

plot(a)

## End(Not run)
```

---

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, <a href="#">logical</a> , 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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.

### Note

It is suggested to execute [graphics.off](#) before executing 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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
a <- altf4(y=wti, x=drivers, windows=c(36, 100, 150))

plot(a)

## End(Not run)
```

---

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, <a href="#">logical</a> , 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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.

Choosing 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working directory),

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

Choosing 0 exits the plot command.

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

## Note

It is suggested to execute `graphics.off` before executing 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

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]

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)
# graphics.off()
```

```
plot(m2)
## End(Not run)
```

---

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

|                              |   |
|------------------------------|---|
| <code>x</code>               | an object of <code>grid.dma</code> class  |
| <code>non.interactive</code> | optional, <b>logical</b> , indicate whether plots should be made in non-interactive mode, by default <code>non.interactive=FALSE</code> , i.e., the user specifies in the interactive menu which plots will be made |
| <code>...</code>             | 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 current working directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the current working directory),
  - 4 - for plotting expected coefficients (including constant) for all estimated models, the outcomes are saved in separate png files in the current working directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the current working 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 current working directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the current working directory).

Chosing 0 exits the plot command.

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

### Note

It is suggested to execute `graphics.off` before executing 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

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
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)
# graphics.off()
plot(g2)

## End(Not run)
```

---

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, <b>logical</b> , 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 current working directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the current working 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 current working directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the current working directory),

Choosing 0 exits the plot command.

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

**Note**

It is suggested to execute **graphics.off** before executing 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 <- (diff(log(drivers)))[-1,]
## Not run:
grw <- c(50,100,150)
g <- grid.roll.reg(y=ld.wti,x=ld.drivers,grid.window=grw)

plot(g)
```

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

---

```
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 <code>grid.tvp</code> class   |
| non.interactive | optional, <a href="#">logical</a> , indicate whether plots should be made in non-interactive mode, by default <code>non.interactive=FALSE</code> , 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 current working directory, and additionally, plots for different variables are collected into one big plot (also saved as a png file in the current working directory).

Choosing 0 exits the plot command.

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

## Note

It is suggested to execute [graphics.off](#) before executing 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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]

gr1 <- c(0.99, 0.98, 0.97, 0.96, 0.95)
g <- grid.tvp(y=ld.wti, x=ld.drivers, V=1, grid.lambda=gr1)

plot(g)

## End(Not run)
```

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, <b>logical</b> , 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working directory).

Chosing 0 exits the plot command.

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

### 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

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]

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

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

# graphics.off()
plot(roll)

# graphics.off()
plot(rec)

## End(Not run)
```

---

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, <a href="#">logical</a> , 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 current working directory, and moreover, to paste them into one big plot (also saved as a png file in the current working directory).

Choosing 0 exits the plot command.

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

**Note**

It is suggested to execute [graphics.off](#) before executing 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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]

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

# graphics.off()
plot(tvp)

## End(Not run)
```

---

|             |   |
|-------------|---|
| predict.dma | <i>Computes Predictions from dma Model.</i> |
|-------------|---|

---

### 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 <a href="#">fdma</a>  |
| 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  |

### Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
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")

## End(Not run)
```

---

|            |                            |
|------------|----------------------------|
| print.altf | <i>Prints altf Object.</i> |
|------------|----------------------------|

---

### 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](#).

**Examples**

```
## Not run:  
wti <- crudeoil[-1,1]  
drivers <- (lag(crudeoil[, -1], k=1))[-1,]  
a <- altf(y=wti, x=drivers)  
  
print(a)  
  
## End(Not run)
```

---

|             |                             |
|-------------|-----------------------------|
| print.altf2 | <i>Prints altf2 Object.</i> |
|-------------|-----------------------------|

---

**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](#).

### Examples

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
a <- altf2(y=wti, x=drivers)

print(a)

## End(Not run)
```

---

|             |                             |
|-------------|-----------------------------|
| print.altf3 | <i>Prints altf3 Object.</i> |
|-------------|-----------------------------|

---

### 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](#).

### Examples

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
a <- altf3(y=wti, x=drivers, windows=c(36, 100, 150))

print(a)

## End(Not run)
```



---

|             |                             |
|-------------|-----------------------------|
| print.altf4 | <i>Prints altf4 Object.</i> |
|-------------|-----------------------------|

---

**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](#).

**Examples**

```
## Not run:  
wti <- crudeoil[-1,1]  
drivers <- (lag(crudeoil[,-1],k=1))[-1,]  
a <- altf4(y=wti,x=drivers,windows=c(36,100,150))  
  
print(a)  
  
## End(Not run)
```

---

|           |                           |
|-----------|---------------------------|
| print.dma | <i>Prints dma Object.</i> |
|-----------|---------------------------|

---

**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](#).

**Examples**

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
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")

## End(Not run)

print(m1)
print(m2)
```

---

|                |                                |
|----------------|--------------------------------|
| print.grid.dma | <i>Prints grid.dma Object.</i> |
|----------------|--------------------------------|

---

**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.

### Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
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")

## End(Not run)

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.

### Examples

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
grw <- c(50,100,150)
g <- grid.roll.reg(y=ld.wti,x=ld.drivers,grid.window=grw)

print(g)
```

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

---

```
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.

### Examples

```
wti <- crudeoil[-1,1]  
drivers <- (lag(crudeoil[,-1],k=1))[-1,]  
ld.wti <- (diff(log(wti)))[-1,]  
ld.drivers <- (diff(log(drivers)))[-1,]  
## Not run:  
gr1 <- c(0.99,0.98,0.97,0.96,0.95)  
g <- grid.tvp(y=ld.wti,x=ld.drivers,V=1,grid.lambda=gr1)  
  
print(g)  
  
## End(Not run)
```

---

|           |                           |
|-----------|---------------------------|
| print.reg | <i>Prints reg Object.</i> |
|-----------|---------------------------|

---

### 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.

### Examples

```
wti <- crudeoil[-1,1]  
drivers <- (lag(crudeoil[, -1], k=1))[-1,]  
ld.wti <- (diff(log(wti)))[-1,]  
ld.drivers <- (diff(log(drivers)))[-1,]  
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 | <i>Prints tvp Object.</i> |
|-----------|---------------------------|

---

### 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.

**Examples**

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

|                       |   |
|-----------------------|---|
| <code>\$y.hat</code>  | fitted (forecasted) values  |
| <code>\$AIC</code>    | Akaike Information Criterion (from the current set of observations)   |
| <code>\$AICc</code>   | Akaike Information Criterion with a correction for finite sample sizes (from the current set of observations) |
| <code>\$BIC</code>    | Bayesian Information Criterion (from the current set of observations)   |
| <code>\$MSE</code>    | Mean Squared Error (from the current set of observations)   |
| <code>\$coeff.</code> | regression coefficients   |
| <code>\$p.val</code>  | p-values for t-test for statistical significance of regression coefficients                                   |
| <code>\$y</code>      | 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 <- (diff(log(drivers)))[-1,]
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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
m1 <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.90,initvar=10)
m2 <- reduce.size(m1)
## End(Not run)
```

---

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               |

**Examples**

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
m1 <- fdMA(y=ld.wti,x=ld.drivers,alpha=0.99,lambda=0.99,initvar=1,model="dma")
r <- residuals(object=m1)

## End(Not run)
```



---

|          |                                     |
|----------|-------------------------------------|
| roll.reg | <i>Computes Rolling Regression.</i> |
|----------|-------------------------------------|

---

### 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

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

|                       |   |
|-----------------------|---|
| <code>\$y.hat</code>  | fitted (forecasted) values  |
| <code>\$AIC</code>    | Akaike Information Criterion (from the current window size)   |
| <code>\$AICc</code>   | Akaike Information Criterion with a correction for finite sample sizes (from the current window size) |
| <code>\$BIC</code>    | Bayesian Information Criterion (from the current window size)   |
| <code>\$MSE</code>    | Mean Squared Error (from the current window size)   |
| <code>\$coeff.</code> | regression coefficients   |
| <code>\$p.val</code>  | p-values for t-test for statistical significance of regression coefficients                           |
| <code>\$window</code> | window size   |
| <code>\$y</code>      | 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 <- (diff(log(drivers)))[-1,]
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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
m1 <- fdMA(y=ld.wti, x=ld.drivers, alpha=0.99, lambda=0.90, initvar=10)
r <- rvi(m1)
## End(Not run)
```

---

|             |  |
|-------------|--|
| standardize | <i>Standardizes a Numeric Matrix by Columns.</i> |
|-------------|--|

---

**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 | <i>Computes a Few Stationarity Tests.</i> |
|-------|---|

---

**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**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
x <- cbind(ld.wti,ld.drivers)
stest(x)

## End(Not run)
```

---

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**

|                     |                                      |
|---------------------|--------------------------------------|
| <code>object</code> | an object of <code>altf</code> class |
| <code>...</code>    | 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.

**Examples**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf(y=wti,x=drivers)

summary(a)

## End(Not run)
```

---

|               |   |
|---------------|---|
| summary.altf2 | <i>Summarizes Outcomes from altf2 Object.</i> |
|---------------|---|

---

### 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.

### Examples

```
## Not run:  
wti <- crudeoil[-1,1]  
drivers <- (lag(crudeoil[, -1], k=1))[-1,]  
a <- altf2(y=wti, x=drivers)  
  
summary(a)  
  
## End(Not run)
```

---

|               |   |
|---------------|---|
| summary.altf3 | <i>Summarizes Outcomes from altf3 Object.</i> |
|---------------|---|

---

### 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.

**Examples**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
a <- altf3(y=wti,x=drivers,windows=c(36,100,150))

summary(a)

## End(Not run)
```

---

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.

**Examples**

```
## Not run:
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1],k=1))[-1,]
a <- altf4(y=wti,x=drivers,windows=c(36,100,150))

summary(a)

## End(Not run)
```

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 comparison 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 comparison to the whole analyzed period) a given variable is present in the selected model.

**Examples**

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
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")

## End(Not run)

summary(m1)
summary(m2)

```

---

|                  |   |
|------------------|---|
| summary.grid.dma | <i>Summarizes Outcomes from grid.dma Objects.</i> |
|------------------|---|

---

## 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).

## Examples

```

wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[,-1],k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
gra <- c(0.99,0.98,0.97)
gr1 <- c(0.99,0.95)
g1 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=gr1,initvar=1)
g2 <- grid.DMA(y=ld.wti,x=ld.drivers,grid.alpha=gra,grid.lambda=gr1,initvar=1,model="dms")

## End(Not run)

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).

### Examples

```
wti <- crudeoil[-1,1]  
drivers <- (lag(crudeoil[,-1],k=1))[-1,]  
ld.wti <- (diff(log(wti)))[-1,]  
ld.drivers <- (diff(log(drivers)))[-1,]  
## Not run:  
grw <- c(50,100,150)  
g <- grid.roll.reg(y=ld.wti,x=ld.drivers,grid.window=grw)  
  
summary(g)  
  
## End(Not run)
```

---

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).

**Examples**

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
## Not run:
gr1 <- c(0.99, 0.98, 0.97, 0.96, 0.95)
g <- grid.tvp(y=ld.wti, x=ld.drivers, V=1, grid.lambda=gr1)

summary(g)

## End(Not run)
```

---

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.

**Examples**

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

**Examples**

```
wti <- crudeoil[-1,1]
drivers <- (lag(crudeoil[, -1], k=1))[-1,]
ld.wti <- (diff(log(wti)))[-1,]
ld.drivers <- (diff(log(drivers)))[-1,]
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\$stock\_markets – Google Trends for "stock markets"
- trends\$interest\_rate – Google Trends for "interest rate"
- trends\$economic\_activity – Google Trends for "economic activity"
- trends\$exchange\_rate – Google Trends for "exchange rate"
- trends\$oil\_production – Google Trends for "oil production"
- trends\$oil\_consumption – Google Trends for "oil consumption"
- trends\$market\_stress – Google Trends for "market stress"

### Details

The data are in monthly frequency. They cover the period between Jan, 2004 and Dec, 2016.

### 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 <- (diff(log(drivers)))[-1,]
ld.wti <- ld.wti['2004-01-01-/' ]
ld.drivers <- ld.drivers['2004-01-01-/' ]
## Not run:
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)

## End(Not run)
```

---

|     |   |
|-----|---|
| tvp | <i>Computes Time-Varying Parameters Regression.</i> |
|-----|---|

---

### 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 <code>fdma</code> ), 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 <code>c=TRUE</code> 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. | predictive 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**, [http://arma.sourceforge.net/armadillo\\_joss\\_2016.pdf](http://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 <- (diff(log(drivers)))[-1,]
## Not run:
t1 <- tvp(y=ld.wti, x=ld.drivers, V=1, lambda=0.99)

## End(Not run)

## Not run:
t2 <- tvp(y=ld.wti, x=ld.drivers, V=1, lambda=0.99, W=1)

## End(Not run)

## Not run:
t3 <- tvp(y=ld.wti, x=ld.drivers, V=1, lambda=0.99, W=1, kappa=0.75)

## End(Not run)

# 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

accuracy, [3](#), [5](#), [8](#), [10](#), [47–49](#)  
adf.test, [59](#)  
altf, [3](#), [7](#), [9](#), [11](#), [46](#), [50](#), [60](#)  
altf2, [4](#), [5](#), [9](#), [11](#), [47](#), [61](#)  
altf3, [4](#), [7](#), [8](#), [11](#), [48](#), [61](#)  
altf4, [4](#), [7](#), [9](#), [10](#), [49](#), [62](#)  
archtest, [12](#), [29](#)  
auto.arima, [3](#), [19](#), [50](#)

coef (coef.dma), [13](#)  
coef.dma, [13](#)  
colnames, [55](#)  
crudeoil, [14](#)

describe, [15](#)  
descstat, [15](#)  
dm.test, [16](#), [28](#), [29](#)  
dmtest, [16](#), [29](#), [30](#)

fDMA, [5](#), [6](#), [11](#), [13](#), [17](#), [22–24](#), [31](#), [37](#), [40](#), [46](#),  
[49](#), [55](#), [56](#), [58](#), [63](#), [69](#)  
fitted (fitted.dma), [21](#)  
fitted.dma, [21](#)

gNormalize, [22](#)  
graphics.off, [33–36](#), [38](#), [40–42](#), [44](#), [45](#)  
grid.DMA, [21](#), [23](#), [39](#), [50](#), [55](#), [56](#), [64](#)  
grid.roll.reg, [25](#), [40](#), [51](#), [58](#), [65](#)  
grid.tvp, [26](#), [42](#), [52](#), [65](#), [70](#)

hit.ratio, [3](#), [5](#), [8](#), [10](#), [21](#), [27](#)  
hmdmtest, [16](#), [28](#), [30](#)

kpss.test, [59](#)

length, [40](#)  
list, [4](#), [6](#), [7](#), [9](#), [11](#), [13](#), [19](#), [23–26](#), [55](#), [57](#), [69](#)  
lm, [54](#), [57](#)  
logical, [4](#), [6](#), [9](#), [11](#), [18](#), [19](#), [24–27](#), [32](#), [33](#),  
[35–37](#), [39](#), [41–43](#), [45](#), [54](#), [57](#), [69](#)

matrix, [4–6](#), [8–11](#), [15–19](#), [22](#), [24–31](#), [54](#), [57](#),  
[59](#), [60](#), [69](#)  
mdmtest, [16](#), [28](#), [29](#), [29](#)  
msmFit, [3](#)

NA, [20](#), [54](#), [55](#), [57](#)  
NaN, [54](#), [57](#)  
normalize, [30](#), [59](#)  
numeric, [4–6](#), [8–12](#), [17–19](#), [23](#), [27](#), [54](#), [57](#), [69](#)

onevar, [31](#)

plot (plot.dma), [37](#)  
plot.altf, [4](#), [32](#)  
plot.altf2, [7](#), [33](#)  
plot.altf3, [9](#), [34](#)  
plot.altf4, [11](#), [36](#)  
plot.dma, [21](#), [37](#)  
plot.grid.dma, [24](#), [39](#)  
plot.grid.roll.reg, [25](#), [40](#)  
plot.grid.tvp, [27](#), [42](#)  
plot.reg, [43](#), [55](#), [58](#)  
plot.tvp, [44](#), [70](#)  
pp.test, [59](#)  
predict (predict.dma), [46](#)  
predict.dma, [46](#)  
print (print.dma), [49](#)  
print.altf, [4](#), [46](#), [60](#)  
print.altf2, [7](#), [47](#), [61](#)  
print.altf3, [9](#), [48](#), [62](#)  
print.altf4, [11](#), [49](#), [62](#)  
print.dma, [21](#), [49](#), [63](#)  
print.grid.dma, [24](#), [50](#), [64](#)  
print.grid.roll.reg, [25](#), [51](#), [65](#)  
print.grid.tvp, [27](#), [52](#), [66](#)  
print.reg, [53](#), [55](#), [58](#), [67](#)  
print.tvp, [53](#), [70](#)

rec.reg, [4](#), [7](#), [53](#), [54](#), [66](#)  
reduce.size, [55](#)

rep, [4](#), [6](#)  
residuals (residuals.dma), [56](#)  
residuals.dma, [56](#)  
roll.reg, [4](#), [7](#), [9](#), [11](#), [25](#), [53](#), [57](#), [66](#)  
rvi, [58](#)

standardize, [31](#), [59](#)  
stest, [59](#)  
summary (summary.dma), [63](#)  
summary.altf, [4](#), [60](#)  
summary.altf2, [7](#), [61](#)  
summary.altf3, [9](#), [61](#)  
summary.altf4, [11](#), [62](#)  
summary.dma, [21](#), [63](#)  
summary.grid.dma, [24](#), [64](#)  
summary.grid.roll.reg, [25](#), [65](#)  
summary.grid.tvp, [27](#), [65](#)  
summary.reg, [55](#), [58](#), [66](#)  
summary.tvp, [67](#), [70](#)

trends, [68](#)  
tvp, [3](#), [7](#), [11](#), [26](#), [53](#), [67](#), [69](#)

vector, [8](#), [11](#), [12](#), [16](#), [19](#), [23](#), [25–29](#)

xts, [14](#), [15](#), [17](#), [27](#), [30](#), [59](#), [68](#)