Package 'forecTheta'

June 16, 2026

Type Package

Title Forecasting Time Series by Theta Models

Version 3.0.3

Maintainer Jose Augusto Fiorucci <jafiorucci@gmail.com>

Description Routines for forecasting univariate time series using Theta Models.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 2.0), forecast, tseries, foreach

BugReports https://github.com/jafiorucci/forecTheta/issues

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Jose Augusto Fiorucci [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-1201-9089>),
Francisco Louzada [aut, cph] (ORCID:
<https://orcid.org/0000-0001-7815-9554>),
Igor De Oliveira Barros Faluhelyi [aut, ctb] (ORCID:
<https://orcid.org/0009-0008-1637-4476>)

Repository CRAN

Date/Publication 2026-06-16 05:10:30 UTC

Contents

Bagged Theta Models 2
Cross Validation 5
Error Metric 6
expSmoot 8
forecTheta-Package 9
otm.arxiv 11

PI_eval 13
Plot 15
seasonal_test 16
Theta Models 17

Index **21**

`Bagged Theta Models` *Bagged Theta Models*

Description

Bagged implementation (Bergmeir et al, 2016) of Dynamic Optimised Theta Model, Dynamic Standard Theta Model, Optimised Theta Model and Standard Theta Model (Fiorucci et al, 2016).

Usage

```
bagged_dotm(y, h=5, level=c(80,90,95), num_bootstrap = 100, bs_bootstrap = NULL,
  s_type="multiplicative", s_test="default", lambda=NULL,
  par_ini=c(y[1]/2, 0.5, 2),estimation=TRUE, lower=c(-1e+10, 0.1, 1.0),
  upper=c(1e+10, 0.99, 1e+10), opt.method="Nelder-Mead")

bagged_dstm(y, h=5, level=c(80,90,95), num_bootstrap = 100, bs_bootstrap = NULL,
  s_type="multiplicative", s_test="default", lambda=NULL, par_ini=c(y[1]/2, 0.5),
   estimation=TRUE, lower=c(-1e+10, 0.1), upper=c(1e+10, 0.99),
   opt.method="Nelder-Mead")

bagged_otm(y, h=5, level=c(80,90,95), num_bootstrap = 100, bs_bootstrap = NULL,
  s_type="multiplicative", s_test="default", lambda=NULL, par_ini=c(y[1]/2, 0.5, 2),
   estimation=TRUE, lower=c(-1e+10, 0.1, 1.0), upper=c(1e+10, 0.99, 1e+10),
   opt.method="Nelder-Mead")

bagged_stm(y, h=5, level=c(80,90,95), num_bootstrap = 100, bs_bootstrap = NULL,
  s_type="multiplicative", s_test="default", lambda=NULL, par_ini=c(y[1]/2, 0.5),
   estimation=TRUE, lower=c(-1e+10, 0.1), upper=c(1e+10, 0.99),
   opt.method="Nelder-Mead")
```

Arguments

`y`	Object of time series class.
`h`	Number of required forecasting periods.
`level`	Levels for prediction intervals.
`num_bootstrap`	Number of bootstrap samples to be considered in the forecast (`num` argument of the `forecast::bld.mbb.bootstrap` function).
`bs_bootstrap`	Block size for bootstrap samples to be considered in the forecast (`block_size` argument of the `forecast::bld.mbb.bootstrap` function). Used only if `num_bootstrap` >= 1.

`s_type`	Use `"multiplicative"` for the classic multiplicative seasonal decomposition, `"additive"` for the classic additive seasonal decomposition and `"stl"` for the STL decomposition (in this case, `forecast::mstl` is considered).
`s_test`	If `TRUE`, seasonal decomposition is used. If `FALSE`, seasonal decomposition is not used. If `default`, the time series is tested for statistically seasonal behaviour. If `uni_root`, then first a difference is taken if a unit root is detected.
`lambda`	Parameter for Box-Cox transformation (`forecast::BoxCox` is considered). If `lambda=NULL`, then it is ignored. If `lambda="auto"`, then this parameter is automatically selected (see `forecast::BoxCox.lambda()`). Furthermore, `lambda=0` corresponds to the logarithmic transformation, which can be used to get strictly positive forecasts.
`par_ini`	Vector of initialization for `(ell, alpha, theta)` parameters.
`estimation`	If `TRUE`, the `optim()` function is consider for compute the minimum square estimator of parameters. If `FALSE`, the models/methods are computed for `par_ini` values.
`lower`	The lower limit of parametric space.
`upper`	The upper limit of parametric space.
`opt.method`	The numeric optimisation method for `optim()` function. Choose one among `'Nelder-Mead', 'L-BFGS-B', 'SANN'`.

Details

This version allows bagging to be used in conjunction with the models DOTM, DSTM, OTM and STM (Fiorucci et al, 2016), which usually generates considerably more accurate results, whether point or interval forecasts. In this case, *Box-Cox and Loess-based decomposition bootstrap (BLD-MBB)* is used, for details see Bergmeir et al, 2016.

If `num_bootstrap > 1`, then the bagged series are extrapolated through simulation. In the end, point forecasts are calculated as the mean (`$mean`) or the median (`$median`) of these series, while interval forecasts are computed based on quantiles. To ensure a substantial number of simulations, each bagged series can be extrapolated multiple times through simulation, aiming for a total simulation volume of at least 10,000 series.

By default, the 90% significance seasonal Z-test, used by Assimakopoulos and Nikolopoulos (2000), is applied for quarterly and monthly time series. The possibility of first checking the unit root was included because it was pointed out that this test presents many flaws for time series with this characteristic (Fiorucci et al, 2016). In this case, the KPSS test is performed with a significance level of 5% and in the case of a unit root, then the series is differentiated before checking for seasonal behavior.

Value

An object of `thetaModel` class with one list containing the elements:

`$method`	The name of the model/method
`$y`	The original time series.
`$s`	A binary indication for seasonal decomposition (original time series).
`type`	Seasonal decomposition type (original time series).

`opt.method`	The optimisation method used in the `optim()` function.
`$par`	The estimated values of `(ell, alpha, theta)` parameters (original time series)
`$weights`	The estimated weights values (original time series).
`$fitted`	A time series element with the fitted points (original time series).
`$residuals`	A time series element with the residual points (original time series).
`$mean`	The forecasting values calculed as the mean of bootstrapping simulations.
`$median`	The forecasting values calculed as the median of bootstrapping simulations.
`$level`	The levels for prediction intervals.
`$lower`	Lower limits for prediction intervals.
`$upper`	Upper limits for prediction intervals.

Author(s)

Jose Augusto Fiorucci, Francisco Louzada, Igor De Oliveira Barros Faluhelyi.

References

Assimakopoulos, V. and Nikolopoulos k. (2000). *The theta model: a decomposition approach to forecasting*. International Journal of Forecasting 16 (4), 521–530, <doi:10.1016/S0169-2070(00)00066-2>.

Bergmeir, C., Hyndman, R.J. and Benítez, J. M. (2016). *Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation*. International journal of forecasting 32 (2), 303–312, <doi:10.1016/j.ijforecast.2015.07.002>.

Fiorucci J.A., Pellegrini T.R., Louzada F., Petropoulos F., Koehler, A. (2016). *Models for optimising the theta method and their relationship to state space models*, International Journal of Forecasting, 32 (4), 1151–1161, <doi:10.1016/j.ijforecast.2016.02.005>.

See Also

`forecTheta-package`, `dotm`, `dstm`, `otm`, `stm`

Examples

```
#### additive seasonal decomposition ###
x = sin(2*pi*seq(0,9,len=300)) + exp((1:300)/150) + rnorm(mean=100,sd=0.5,n=300)
y = ts(x, frequency=33)
out <- bagged_dotm(y, h=50, s_type='additive', num_bootstrap = 5)
summary(out)
plot(out)
```

`Cross Validation` *Generalised Rolling Origin Evaluation*

Description

This function implements the Generalised Rolling Origin Evaluation of Fioruci et al (2015). Its particular cases include the cross validation methods: Rolling Origin Evaluation and Fixed Origin Evaluation of Tashman(2000).

Usage

```
groe(y, forecFunction, g="sAPE", n1=length(y)-10, m=5,
        H=length(y)-n1, p=1+floor((length(y)-n1)/m), ...)

rolOrig(y, forecFunction, g="sAPE", n1=length(y)-10, ...)

fixOrig(y, forecFunction, g="sAPE", n1=length(y)-10, ...)
```

Arguments

`y`	Object of time series class or a vector
`forecFunction`	A forecasting method as one object of the `forecast` class of forecast package.
`g`	The prediction error type of `errorMetric` function. The possible values are `"sAPE"`, `"APE"`, `"AE"` and `"SE"`.
`n1`	The index of the first origin element.
`m`	The number of movements of the origin in each update.
`H`	The number of predictions forward of each origin.
`p`	The number of origin updates. Default is the maximum.
`...`	Additional arguments for `forecFunction`.

Details

If `m=1` is computed the Rolling Origin Evaluation. If `m>=length(y)-n1` is computed the Fixed Origin Evaluation.

Value

The sum of the prediction errors.

Note

The `otm.arxiv` function use this function for estimate the theta parameter when the `theta` argument is `NULL`. Your computer may go into an infinite looping if you use `forecFunction = otm.arxiv` without specific a numeric value for the `theta` argument.

Author(s)

Jose Augusto Fiorucci and Francisco Louzada

References

Fioruci J.A., Pellegrini T.R., Louzada F., Petropoulos F. (2015). *The Optimised Theta Method*. arXiv preprint, arXiv:1503.03529.

Tashman, L.J. (2000). *Out-of-sample tests of forecasting accuracy: an analysis and review*. International Journal of Forecasting 16 (4), 437–450.

See Also

forecTheta-package, dotm, otm.arxiv

Examples

```
y1 = 2+ 0.15*(1:20) + rnorm(20,2)
y2 = y1[20]+ 0.3*(1:30) + rnorm(30,2)
y =  as.ts(c(y1,y2))

## Rolling Origin Evaluation
rolOrig( y=y, forecFunction = dotm, n1=40)
rolOrig( y=y, forecFunction = expSmoot, n1=40)
rolOrig( y=y, forecFunction = stheta, n1=40)
rolOrig( y=y, forecFunction = otm.arxiv, n1=40, theta=3)

## Fixed Origin Evaluation
fixOrig( y=y, forecFunction = dotm, n1=40)
fixOrig( y=y, forecFunction = expSmoot, n1=40)
fixOrig( y=y, forecFunction = stheta, n1=40)
fixOrig( y=y, forecFunction = otm.arxiv, n1=40, theta=3)

## Generalised Rolling Origin Evaluation with two origin updates.
## Where the first is the 40th element and second is the 45th element
groe( y=y, forecFunction = dotm, m=5, n1=40)
groe( y=y, forecFunction = expSmoot, m=5, n1=40)
groe( y=y, forecFunction = stheta, m=5, n1=40)
groe( y=y, forecFunction = otm.arxiv, m=5, n1=40, theta=3)
```

Error Metric *Error Metric Function*

Description

This function implements some of the more used error metrics. These metrics are "sMAPE", "MAPE", "MAE", "MSE" and they respectively versions with median "sMdAPE", "MdAPE", "MdAE", "MdSE".

Usage

```
errorMetric(obs, forec, type="sAPE", statistic="M")
```

Arguments

obs	A vector or a matrix with the real values.
forec	A vector or a matrix with the estimated values.
type	The error type of "sAPE", "APE", "AE" and "SE".
statistic	The statistic to be returned. Use "M" or "Md" for return the mean or median of the errors. If "N" so a vector with all errors will be returned.

Details

The metric sMAPE is obtained using `type = "sAPE"` and `statistic = "M"`

The metric sMdAPE is obtained using `type = "sAPE"` and `statistic = "Md"`

The metric MAPE is obtained using `type = "APE"` and `statistic = "M"`

The metric MdAPE is obtained using `type = "APE"` and `statistic = "Md"`

The metric MAE is obtained using `type = "AE"` and `statistic = "M"`

The metric MdAE is obtained using `type = "AE"` and `statistic = "Md"`

The metric MSE is obtained using `type = "SE"` and `statistic = "M"`

The metric MdSE is obtained using `type = "SE"` and `statistic = "Md"`

Value

If `statistic="M"` or `statistic="Md"` it is returned the respectively error metric result. If `statistic="N"` so is returned a vector with all errors points according to the chosen error type.

Author(s)

Jose Augusto Fiorucci and Francisco Louzada

See Also

forecTheta-package, PIEval, groe

Examples

```
############################################################

y1 = 2+ 0.15*(1:20) + rnorm(20,2)
y2 = y1[20]+ 0.3*(1:30) + rnorm(30,2)
y =  as.ts(c(y1,y2))

out <- dotm(y=as.ts(y[1:40]), h=10)

### sMAPE metric
errorMetric(obs=as.ts(y[41:50]), forec=out$mean)
```

```
### sMdAPE metric
errorMetric(obs=as.ts(y[41:50]), forec=out$mean, statistic = "Md")

### MASE metric
meanDiff1 = mean(abs(diff(as.ts(y[1:40]), lag = 1)))
errorMetric(obs=as.ts(y[41:50]), forec=out$mean, type = "AE", statistic = "M") / meanDiff1
```

`expSmoot` *Simple Exponential Smoothing Method*

Description

Estimation of Simple Exponential Smoothing Method

Usage

```
expSmoot(y, h=5, ell0=NULL, alpha=NULL, lower = c(-1e+10, 0.1),
upper = c(1e+10, 0.99))
```

Arguments

`y`	Object of time series class.
`h`	Number of required forecasting periods.
`ell0`	The value of `ell0^*` parameter.
`alpha`	The value of `alpha` parameter.
`lower`	The lower limit of parametric space.
`upper`	The upper limit of parametric space.

Value

A list containing the elements:

`$y`	The original time series.
`$par`	The estimated values for (`ell^*`, `alpha`) parameters
`$mean`	The forecasting values
`$fitted`	A time series element with the fitted points.
`$residuals`	A time series element with the residual points.

Author(s)

Jose Augusto Fiorucci, Francisco Louzada and Bao Yiqi

See Also

`forecTheta-package`, `stheta`, `dotm`

Examples

```
y1 = 2+ 0.15*(1:20) + rnorm(20,2)
y2 = y1[20]+ 0.3*(1:30) + rnorm(30,2)
y =  as.ts(c(y1,y2))

expSmoot(y, h=10)
```

---

forecTheta-Package *Forecasting Time Series by Theta Models*

Description

This package provides functions for forecasting univariate time series using several Theta models, originally proposed by Assimakopoulos and Nikolopoulos (2000) and later extended by Fiorucci et al. (2016). This version also includes implementations of bagging methods, based on the work of Bergmeir et al. (2016), applied to the DOTM, DSTM, OTM, and STM models.

Details

Package:	forecTheta
Type:	Package
Version:	3.0.3
Date:	2026-06-15
License:	GPL (>=2.0)

```
dotm(y, h)

bagged_dotm(y, h)

stheta(y, h)

errorMetric(obs, forec, type = "sAPE", statistic = "M")

PI_eval(obs, forec, lower_bounds, upper_bounds, name = c("MSIS", "ACD"), alpha = 0.05)

groe(y, forecFunction = ses, g = "sAPE", n1 = length(y)-10)
```

Author(s)

Jose Augusto Fiorucci, Francisco Louzada, Igor De Oliveira Barros Faluhelyi.

Maintainer: Jose Augusto Fiorucci <jafiorucci@gmail.com>

References

Assimakopoulos, V. and Nikolopoulos k. (2000). *The theta model: a decomposition approach to forecasting*. International Journal of Forecasting 16, 4, 521–530, <doi:10.1016/S0169-2070(00)00066-2>.

Bergmeir, C., Hyndman, R.J. and Benítez, J. M. (2016). *Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation*. International journal of forecasting 32 (2), 303–312, <doi:10.1016/j.ijforecast.2015.07.002>.

Fiorucci J.A., Pellegrini T.R., Louzada F., Petropoulos F., Koehler, A. (2016). *Models for optimising the theta method and their relationship to state space models*, International Journal of Forecasting, 32 (4), 1151–1161, <doi:10.1016/j.ijforecast.2016.02.005>.

Fioruci J.A., Pellegrini T.R., Louzada F., Petropoulos F. (2015). *The Optimised Theta Method*. arXiv preprint, arXiv:1503.03529.

Gneiting, T. and Raftery, A.E. (2007). Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association 102 (477), 359–378, doi:10.1198/016214506000001437.

Tashman, L.J. (2000). *Out-of-sample tests of forecasting accuracy: an analysis and review*. International Journal of Forecasting, 16 (4), 437–450, <doi:10.1016/S0169-2070(00)00065-0>.

See Also

dotm, bagged_dotm, stheta, otm.arxiv, groe, rolOrig, fixOrig, errorMetric

Examples

```
############################################################

y1 = 2+ 0.15*(1:20) + rnorm(20)
y2 = y1[20]+ 0.3*(1:30) + rnorm(30)
y =  as.ts(c(y1,y2))
out <- dotm(y, h=10)
summary(out)
plot(out)

out <- dotm(y=as.ts(y[1:40]), h=10)
summary(out)
plot(out)

out2 <- bagged_dotm(y=as.ts(y[1:40]), h=10)
summary(out2)
plot(out2)

out3 <- stheta(y=as.ts(y[1:40]), h=10)
summary(out3)
plot(out3)

### sMAPE metric
errorMetric(obs=as.ts(y[41:50]), forec=out$mean, type = "sAPE", statistic = "M")
errorMetric(obs=as.ts(y[41:50]), forec=out2$mean, type = "sAPE", statistic = "M")
errorMetric(obs=as.ts(y[41:50]), forec=out3$mean, type = "sAPE", statistic = "M")
```

```
### sMdAPE metric
errorMetric(obs=as.ts(y[41:50]), forec=out$mean, type = "sAPE", statistic = "Md")
errorMetric(obs=as.ts(y[41:50]), forec=out2$mean, type = "sAPE", statistic = "Md")
errorMetric(obs=as.ts(y[41:50]), forec=out3$mean, type = "sAPE", statistic = "Md")

### MASE metric
meanDiff1 = mean(abs(diff(as.ts(y[1:40]), lag = 1)))
errorMetric(obs=as.ts(y[41:50]), forec=out$mean, type = "AE", statistic = "M") / meanDiff1
errorMetric(obs=as.ts(y[41:50]), forec=out2$mean, type = "AE", statistic = "M") / meanDiff1
errorMetric(obs=as.ts(y[41:50]), forec=out3$mean, type = "AE", statistic = "M") / meanDiff1
```

`otm.arxiv` *Optimised Theta Method*

Description

Functions for forecast univariate time series using the Optimised Theta Method presented in the arxiv paper (Fioruci et al, 2015). If the theta parameter is not specified so the Generalised Rolling Origin Evaluation is used for select the theta value over the thetaList argument.

Usage

```
otm.arxiv( y, h=5, s=NULL, theta=NULL, tLineExtrap=expSmoot, g="sAPE",
approach="c", n1=NULL, m=NULL, H=NULL, p=NULL,
thetaList=seq(from=1,to=5,by=0.5), mc.cores=1, ...)
```

Arguments

`y`	Object of time series class
`h`	Number of required forecasting periods
`s`	If `TRUE`, the multiplicative seasonal decomposition is used. If `NULL`, quarterly and monthly time series are tested for statistically seasonal behaviour, with 95% of significance. Default is NULL.
`theta`	The value of theta parameter. If `theta = NULL` the theta parameter is estimated using the Generalised Rolling Origin Evaluation.
`tLineExtrap`	A forecasting function for extrapolation the second theta-line. Default is `expSmoot`.
`g`	The error type that will be used by `groe` function for select the `theta` value in the estimation process. The possibility values for g is `"sAPE"`, `"APE"`, `"AE"` and `"SE"`. If `theta` is not `NULL` the g argument is not used. Default is "sAPE".
`approach`	The approach set-up for `groe` parameters (`n1`, `m`, `H`, `p`). One letter between `'a'` to `'h'` according to Fioruci et al (2015).
`n1`	The first origin for Generalised Rolling Origin Evaluation. This argument is not used if `theta!=NULL` or `approach!=NULL`.
`m`	The number of movements of the origin in each step. This argument is not used if `theta!=NULL` or `approach!=NULL`.

H	The number of predictions in each step. This argument is not used if theta!=NULL or approach!=NULL.
p	The number of origin updates. This argument is not used if theta!=NULL or approach!=NULL.
thetaList	A vector with the possible values for theta. This argument is not used if theta argument is not NULL.
mc.cores	Number of cores that will be used for estimate the theta parameter. It is not accepted mc.cores>1 on Windows SO.
...	Additional arguments for tLineExtrap.

Details

These functions are fully automatic, you just need to pass your time series. Particular cases are obtained by: If theta = 1 the tLineExtrapModel method is computed; If theta = 2 so the Standard Theta Method of Assimakopoulos and Nikolopoulos (2000) is computed.

By default (s=NULL), the 90% significance seasonal Z-test, used by Assimakopoulos and Nikolopoulos (2000), is applied for quarterly and monthly time series.

Value

An list containing the elements:

$y	The original time series.
$mean	A time series element with the forecasting points.
$fitted	A time series element with the fitted points.
$residuals	A time series element with the residual points.
$theta	The estimated theta value.
$tLineExtrap_par	The estimated parameters of tLineExtrap method.
$weights	The estimated weights values.

Note

The thetaM function is just a particular case of otm with theta=2.

Author(s)

Jose Augusto Fiorucci, Francisco Louzada

References

Fioruci J.A., Pellegrini T.R., Louzada F., Petropoulos F. (2015). *The Optimised Theta Method*. arXiv preprint, arXiv:1503.03529.

Assimakopoulos, V. and Nikolopoulos k. (2000). *The theta model: a decomposition approach to forecasting*. International Journal of Forecasting 16, 4, 521-530.

See Also

forecTheta-package, dotm, groe

Examples

```
y1 = 2+ 0.15*(1:20) + rnorm(20,2)
y2 = y1[20]+ 0.3*(1:30) + rnorm(30,2)
y =  as.ts(c(y1,y2))

otm.arxiv(y, h=10)

### running the M3-competition data base by OTM approach (a) ###
#require(Mcomp)
#data(M3)
#
#forec = matrix(NA, nrow=3003, ncol=18)
#obs = matrix(NA, nrow=3003, ncol=18) #matrix of the out-sample values
#
#for(i in 1:3003){
# if(i %% 100 == 0){print(i)}
# x=M3[[i]]$x
# h=M3[[i]]$h
# out = otm.arxiv(x,h,approach='a',tLineExtrap=ses)
# forec[i,1:h] = out$mean
# obs[i,1:h] = M3[[i]]$xx
#}
#
#sAPE = errorMetric(obs, forec, type="sAPE", statistic="N") ## sAPE matrix
#
##### sMAPE results ##
### Yearly
#mean( sAPE[1:645, 1:6] )
### QUARTERLY
#mean( sAPE[646:1401, 1:8] )
### MONTHLY
#mean( sAPE[1402:2829, 1:18] )
### Other
#mean( sAPE[2830:3003, 1:8] )
### ALL
#mean( sAPE, na.rm=TRUE )
```

PI_eval *Prediction Interval (PI) evaluation*

Description

Implementation of Mean Scaled Interval Score (MSIS) and Absolute Coverage Difference (ACD).

Usage

```
PI_eval(obs, forec, lower_bounds, upper_bounds, name = c("MSIS", "ACD"), alpha = 0.05)
```

Arguments

obs	Observed time series values.
forec	Forecasted values corresponding to the observed data.
lower_bounds	Lower bounds of the prediction interval.
upper_bounds	Upper bounds of the prediction interval.
name	Evaluation metric to be used. Options are "MSIS" or "ACD".
alpha	Significance level for the prediction interval. Default is 0.05.

Details

The MSIS is computed by combining the width of the prediction interval with a penalty for observations falling outside the interval. This score is averaged over time and scaled by the mean absolute seasonal difference, following the same approach used for the MASE in the M4 competition, to ensure scale invariance. As a complementary metric, the ACD measures the absolute difference between the empirical and nominal coverage (e.g., 0.95), helping to assess interval reliability.

Value

Returns a numeric value corresponding to the selected evaluation metric (MSIS or ACD).

Author(s)

Jose Augusto Fiorucci, Francisco Louzada, Igor De Oliveira Barros Faluhelyi.

References

Gneiting, T. and Raftery, A.E. (2007). Strictly proper scoring rules, prediction, and estimation. Journal of the American Statistical Association 102 (477), 359–378, doi:10.1198/016214506000001437.

Makridakis, S., Spiliotis, E. and Assimakopoulos, V. (2018). The M4 Competition: 100,000 time series and 61 forecasting methods. International Journal of Forecasting 36 (1), 54–74, doi:10.1016/j.ijforecast.2019.04.014.

See Also

forecTheta-package, errorMetricFunctions, dotm, dstm, otm, stm

Examples

```
y1 <- 2 + 0.15 * (1:20) + rnorm(20)
y2 <- y1[20] + 0.3 * (1:30) + rnorm(30)
y <- as.ts(c(y1, y2))
out <- dotm(y, h = 10)

# MSIS metric
```

```
PI_eval(
  obs = y, forec = out$mean,
  lower_bounds = out$lower[, 3], upper_bounds = out$upper[, 3],
  name = "MSIS", alpha = 0.05
)

# ACD metric
PI_eval(
  obs = y, forec = out$mean,
  lower_bounds = out$lower[, 3], upper_bounds = out$upper[, 3],
  name = "ACD", alpha = 0.05
)
```

`Plot` *Plot forecasts points and prediction intervals for* `thetaModel` *objects*

Description

Produces a figure of the time series and the forecasts points from Optimised Theta Method.

Usage

```
## S3 method for class 'thetaModel'
plot(x, ylim=NULL, xlim=NULL, ylab=NULL, xlab=NULL, main=NULL, ...)
```

Arguments

`x`	Object of class "thetaModel".
`ylim`	the y limits of the plot.
`xlim`	the x limits of the plot.
`ylab`	a label for the y axis.
`xlab`	a label for the x axis.
`main`	a main title for the plot.
`...`	Other plotting parameters passed to `par`.

Value

None. Function produces a plot

Author(s)

Jose A Fiorucci

See Also

`dotm`, `forecTheta-package`

Examples

```
y1 = 2+ 0.15*(1:20) + rnorm(20,2)
y2 = y1[20]+ 0.3*(1:30) + rnorm(30,2)
y =  as.ts(c(y1,y2))
out <- dotm(y, h=10)
plot(out)
```

`seasonal_test` *Statistical test for seasonal behavior*

Description

Statistical test for seasonal behavior

Usage

```
seasonal_test(y, s_test = c("default", "unit_root"))
```

Arguments

y	Object of time series class
s_test	If "default" time series is tested for statistically seasonal behaviour. If "unit_root", then First a difference is taken if a unit root is detected.

Details

By default, the 90 The possibility of first checking the unit root was included because it was pointed out that this test presents many flaws for time series with this characteristic (Fiorucci et al, 2016)). In this case, the KPSS test is performed with a significance level of 5

Value

A logical result indicating whether or not seasonal behavior was detected.

References

Fiorucci J.A., Pellegrini T.R., Louzada F., Petropoulos F., Koehler, A. (2016). Models for optimising the theta method and their relationship to state space models, International Journal of Forecasting, 32 (4), 1151–1161, <doi:10.1016/j.ijforecast.2016.02.005>. Assimakopoulos, V. and Nikolopoulos k. (2000). The theta model: a decomposition approach to forecasting. International Journal of Forecasting 16, 4, 521–530, <doi:10.1016/S0169-2070(00)00066-2>.

Examples

```
seasonal_test(AirPassengers)
seasonal_test(AirPassengers, "unit")
```

Theta Models *Theta Models*

Description

Functions for forecast univariate time series using the Dynamic Optimised Theta Model, Dynamic Standard Theta Model, Optimised Theta Model and Standard Theta Model (Fiorucci et al, 2016). We also provide an implementation for the Standard Theta Method (STheta) of Assimakopoulos and Nikolopoulos (2000).

Usage

```
dotm(y, h=5, level=c(80,90,95),s_type="multiplicative", s_test="default",
   lambda=NULL, par_ini=c(y[1]/2, 0.5, 2),estimation=TRUE,
   lower=c(-1e+10, 0.1, 1.0), upper=c(1e+10, 0.99, 1e+10),
   opt.method="Nelder-Mead", xreg=NULL, s=NULL)

dstm(y, h=5, level=c(80,90,95), s_type="multiplicative", s_test="default",
   lambda=NULL, par_ini=c(y[1]/2, 0.5), estimation=TRUE,
   lower=c(-1e+10, 0.1), upper=c(1e+10, 0.99),
   opt.method="Nelder-Mead", xreg=NULL, s=NULL)

otm(y, h=5, level=c(80,90,95), s_type="multiplicative", s_test="default",
   lambda=NULL, par_ini=c(y[1]/2, 0.5, 2), estimation=TRUE,
   lower=c(-1e+10, 0.1, 1.0), upper=c(1e+10, 0.99, 1e+10),
   opt.method="Nelder-Mead", xreg=NULL, s=NULL)

stm(y, h=5, level=c(80,90,95), s_type="multiplicative", s_test="default",
   lambda=NULL, par_ini=c(y[1]/2, 0.5), estimation=TRUE,
   lower=c(-1e+10, 0.1), upper=c(1e+10, 0.99),
   opt.method="Nelder-Mead", xreg=NULL, s=NULL)

stheta(y, h=5, s_type="multiplicative", s_test="default", s=NULL)
```

Arguments

y	Object of time series class.
h	Number of required forecasting periods.
level	Levels for prediction intervals.
s_type	Use "multiplicative" for the classic multiplicative seasonal decomposition, "additive" for the classic additive seasonal decomposition and "stl" for the STL decomposition (in this case, forecast::mstl is considered).
s_test	If TRUE, seasonal decomposition is used. If FALSE, seasonal decomposition is not used. If default, the time series is tested for statistically seasonal behaviour. If uni_root, then first a difference is taken if a unit root is detected.

lambda	Parameter for Box-Cox transformation (forecast::BoxCox is considered). If lambda=NULL, then it is ignored. If lambda="auto", then this parameter is automatically selected (see forecast::BoxCox.lambda()). Furthermore, lambda=0 corresponds to the logarithmic transformation, which can be used to get strictly positive forecasts.
par_ini	Vector of initialization for (ell, alpha, theta) parameters.
estimation	If TRUE, the optim() function is consider for compute the minimum square estimator of parameters. If FALSE, the models/methods are computed for par_ini values.
lower	The lower limit of parametric space.
upper	The upper limit of parametric space.
opt.method	The numeric optimisation method for optim() function. Choose one among 'Nelder-Mead', 'L-BFGS-B', 'SANN'.
xreg	A matrix with the regressor variables including the out-of-sample data.
s	The argument s is deprecated and has been replaced by s_type and s_test for improved functionality and clarity. While s is still supported in this version for backward compatibility, it may be removed in future releases. Users are strongly encouraged to update their code accordingly.

Details

By default, the 90% significance seasonal Z-test, used by Assimakopoulos and Nikolopoulos (2000), is applied for quarterly and monthly time series. The possibility of first checking the unit root was included because it was pointed out that this test presents many flaws for time series with this characteristic (Fiorucci et al, 2016). In this case, the KPSS test is performed with a significance level of 5% and in the case of a unit root, then the series is differentiated before checking for seasonal behavior.

For details of each model see Fiorucci et al, 2016. If you are looking for the methods presented in the arXiv paper (Fiorucci et al, 2015), see otm.arxiv function.

This version allows bagging to be used in conjunction with these models, see bagged_dotm, bagged_dstm, bagged_otm and bagged_stm functions.

Value

An object of thetaModel class with one list containing the elements:

$method	The name of the model/method
$y	The original time series.
$s	A binary indication for seasonal decomposition.
type	Classical seasonal decomposition type.
opt.method	The optimisation method used in the optim() function.
$par	The estimated values for (ell, alpha, theta) parameters
$weights	The estimated weights values.
$fitted	A time series element with the fitted points.

`$residuals`	A time series element with the residual points.
`$mean`	The forecasting values.
`$level`	The levels for prediction intervals.
`$lower`	Lower limits for prediction intervals.
`$upper`	Upper limits for prediction intervals.
`$tests`	The p.value of Teraesvirta Neural Network test applied on unseasoned time series and the p.value of Shapiro-Wilk test applied on unseasoned residuals.

Author(s)

Jose Augusto Fiorucci, Francisco Louzada

References

Fiorucci J.A., Pellegrini T.R., Louzada F., Petropoulos F., Koehler, A. (2016). *Models for optimising the theta method and their relationship to state space models*, International Journal of Forecasting, 32 (4), 1151–1161, <doi:10.1016/j.ijforecast.2016.02.005>.

Assimakopoulos, V. and Nikolopoulos k. (2000). *The theta model: a decomposition approach to forecasting*. International Journal of Forecasting 16 (4), 521–530, <doi:10.1016/S0169-2070(00)00066-2>.

See Also

forecTheta-package, bagged_dotm, bagged_dstm, bagged_otm, bagged_stm, otm.arxiv

Examples

```
y1 = 2+ 0.15*(1:20) + rnorm(20)
y2 = y1[20]+ 0.3*(1:30) + rnorm(30)
y =  as.ts(c(y1,y2))
out <- dotm(y, h=10)
summary(out)
plot(out)

#### additive seasonal decomposition ###
x = sin(2*pi*seq(0,9,len=300)) + exp((1:300)/150) + rnorm(mean=0,sd=0.5,n=300)
y = ts(x, frequency=33)
out <- dotm(y, h=50, s_type='additive')
summary(out)
plot(out)

# ##########################################################
# ######### Reproducing the M3 results by DOTM ############
# ##########################################################
#
# library(Mcomp)
# data(M3)
#
# forec = matrix(NA, nrow=3003, ncol=18)
```

```
# obs = matrix(NA, nrow=3003, ncol=18) #matrix of the out-sample values
# meanDiff <- rep(1, 3003)
#
# for(i in 1:3003){
#   x=M3[[i]]$x
#   h=M3[[i]]$h
#   out = dotm(x,h,level=NULL)
#   forec[i,1:h] = out$mean
#   obs[i,1:h] = M3[[i]]$xx
#   meanDiff[i] = mean(abs(diff(x, lag = frequency(x))))
# }
#
# ############## sMAPE ###################
# sAPE_matrix = errorMetric(obs=obs, forec=forec, type="sAPE", statistic="N")
# #### Yearly ###
# mean( sAPE_matrix[1:645, 1:6] )
# #### QUARTERLY ###
# mean( sAPE_matrix[646:1401, 1:8] )
# #### MONTHLY ###
# mean( sAPE_matrix[1402:2829, 1:18] )
# #### Other ###
# mean( sAPE_matrix[2830:3003, 1:8] )
# #### ALL ###
# mean( sAPE_matrix, na.rm=TRUE )
# #
# ############# MASE ######################
# AE_matrix = errorMetric(obs=obs, forec=forec, type="AE", statistic="N")
# ASE_matrix=AE_matrix/meanDiff
# #### Yearly ###
# mean( ASE_matrix[1:645, 1:6] )
# #### QUARTERLY ###
# mean( ASE_matrix[646:1401, 1:8] )
# #### MONTHLY ###
# mean( ASE_matrix[1402:2829, 1:18] )
# #### Other ###
# mean( ASE_matrix[2830:3003, 1:8] )
# #### ALL ###
# mean( ASE_matrix, na.rm=TRUE )
# ##########################################################
```

Index

∗ **ACD**
PI_eval, 13
∗ **Cross Validation**
Cross Validation, 5
∗ **DOTM**
Bagged Theta Models, 2
Theta Models, 17
∗ **DSTM**
Bagged Theta Models, 2
Theta Models, 17
∗ **Dynamic Optimised Theta Model**
forecTheta-Package, 9
∗ **Fixed Origin Evaluation**
Cross Validation, 5
∗ **Generalised Rolling Origin Evaluation**
Cross Validation, 5
∗ **MAE**
Error Metric, 6
∗ **MSE**
Error Metric, 6
∗ **MSIS**
PI_eval, 13
∗ **MdAE**
Error Metric, 6
∗ **MdSE**
Error Metric, 6
∗ **OTM**
Bagged Theta Models, 2
Theta Models, 17
∗ **Rolling Origin Evaluation**
Cross Validation, 5
∗ **STM**
Bagged Theta Models, 2
Theta Models, 17
∗ **STheta**
Theta Models, 17
∗ **Theta Method**
forecTheta-Package, 9
∗ **error metric**
Error Metric, 6
∗ **otm**
otm.arxiv, 11
∗ **plot**
Plot, 15
∗ **sMAPE**
Error Metric, 6
∗ **sMdAPE**
Error Metric, 6
∗ **simple exponential smoothing**
expSmoot, 8
∗ **theta-method**
otm.arxiv, 11
∗ **thetaM**
otm.arxiv, 11
∗ **time series forecasting**
Bagged Theta Models, 2
forecTheta-Package, 9
otm.arxiv, 11
PI_eval, 13
Theta Models, 17

Bagged Theta Models, 2
bagged_dotm, *10*, *18, 19*
bagged_dotm *(*Bagged Theta Models*)*, 2
bagged_dstm, *18, 19*
bagged_dstm *(*Bagged Theta Models*)*, 2
bagged_otm, *18, 19*
bagged_otm *(*Bagged Theta Models*)*, 2
bagged_stm, *18, 19*
bagged_stm *(*Bagged Theta Models*)*, 2

Cross Validation, 5

dotm, *4*, *6*, *8*, *10*, *13–15*
dotm *(*Theta Models*)*, 17
dstm, *4*, *14*
dstm *(*Theta Models*)*, 17

Error Metric, 6

errorMetric, *10*
errorMetric *(*Error Metric*)*, 6
errorMetricFunctions, *14*
expSmoot, 8

fixOrig, *10*
fixOrig *(*Cross Validation*)*, 5
forecTheta *(*forecTheta-Package*)*, 9
forecTheta-Package, 9
forecTheta-package
*(*forecTheta-Package*)*, 9

groe, *7*, *10*, *13*
groe *(*Cross Validation*)*, 5

otm, *4*, *14*
otm *(*Theta Models*)*, 17
otm.arxiv, *6*, *10*, 11, *18, 19*

par, *15*
PI_eval, 13
PIEval, *7*
Plot, 15
plot.thetaModel *(*Plot*)*, 15

rolOrig, *10*
rolOrig *(*Cross Validation*)*, 5

seasonal_test, 16
stheta, *8*, *10*
stheta *(*Theta Models*)*, 17
stm, *4*, *14*
stm *(*Theta Models*)*, 17

Theta Models, 17