Package 'forunco'

March 1, 2020

Title Intelligent Algorithms to be Integrated in Business Intelligence solutions.

Version 0.0.0.9

Description This package implements intelligent algorithms such as time series forecasting that can be easily integrated into BI solutions designed using the Microsoft Stack by calling the sp_execute_external_script stored procedure. This document describes all the functions implemented in the package, whereas a user documentation, which describes how these can be used in SQL Server, can be found <here>.

Depends R (>= 3.3.2)

License MIT

Encoding UTF-8

LazyData true

Imports forecast (>= 8.0),
smooth (>= 1.9.0),
forecTheta (>= 2.2),
tidyr (>= 0.6.1),
ggplot2 (>= 2.2.1),
tibble (>= 1.3.0),
dplyr (>= 0.5.0),
lubridate (>= 1.6.0),
doSNOW (>= 1.0.16)

RoxygenNote 7.0.2

R topics documented:

.rx_forecast . . . 3
accuracy . . . 4
accuracy_detail . . . 5
accuracy_summary . . . 6
ae . . . 7
ape . . . 7
ase . . . 8
auto_arima . . . 8
auto_ces . . . 9

auto_damped . . . 9
auto_dotm . . . 10
auto_ets . . . 10
auto_holt . . . 11
auto_naive . . . 11
auto_nnar . . . 12
auto_ses . . . 12
auto_shd . . . 13
auto_snaive . . . 13
auto_theta . . . 14
auto_thetaf . . . 15
boxcox . . . 16
combine_theta_lines . . . 16
date_increment . . . 17
decimal_to_date . . . 17
diff . . . 18
exp_inverse . . . 18
forecast_forunco . . . 19
forunco . . . 21
generic_combine . . . 23
generic_forecast . . . 23
gmae . . . 24
gmrae . . . 25
g_mean . . . 25
inverse . . . 26
inv_boxcox . . . 26
inv_diff . . . 27
inv_log . . . 27
inv_normalize . . . 28
inv_no_pp . . . 28
inv_scale . . . 29
inv_seasonal_adjustment . . . 29
is_seasonal . . . 30
log . . . 30
mae . . . 31
mape . . . 31
mase . . . 32
mdrae . . . 32
mrae . . . 33
mse . . . 33
msis . . . 34
normalize . . . 34
no_pp . . . 35
plot_forunco . . . 35
pool_mean . . . 36
pool_median . . . 36
preprocessor . . . 37
produce_forecasts . . . 38

rae . . . 39
replace_outliers . . . 39
rmse . . . 40
rx_sql_forunco . . . 40
sape . . . 42
sase . . . 43
scale . . . 43
se . . . 44
seasonal_adjustment . . . 44
shd . . . 45
simple_combination . . . 45
smape . . . 46
smase . . . 47
squared_inverse . . . 47
theta_aea . . . 48
theta_aem . . . 48
theta_ala . . . 49
theta_alm . . . 49
theta_fit . . . 50
theta_forecast . . . 50
theta_line . . . 51
theta_line_zero . . . 51
theta_mea . . . 52
theta_mem . . . 52
theta_mla . . . 53
theta_mlm . . . 53
tidier_ts . . . 54
time_sequence . . . 54
validation_split . . . 55
weighted_average . . . 56
weight_error . . . 57

Index **58**

`.rx_forecast` *Wrapper to forecast with RevoscaleR*

Description

Create a wrapper around forunco that takes data from sql server or spark and predicts the future timesteps with forunco logic.

Usage

```
.rx_forecast(
  keys,
  data,
  h,
  levels,
  methods,
  pp_methods,
  point_combination,
  pi_combination_upper,
  pi_combination_lower,
  pool_limit,
  error_fun,
  weight_fun,
  val_h,
  sov_only,
  max_years,
  val_min_years,
  cv_min_years,
  cv_max_samples,
  allow_negatives,
  ...
)
```

Arguments

keys	key of the time series, integer
data	data of the time series passed by rxExecBy
params	function parameters by rxExecBy

Value

forunco object

---

accuracy *Summary and detail of accuracy measures*

Description

Summary and detail of accuracy measures

Usage

```
accuracy(
  y_true,
  y_pred,
```

```
  y_bench,
  upper,
  lower,
  alpha,
  in_sample,
  frequency,
  error_detail_functions = c("ae", "se", "ape", "sape", "rae", "ase", "sase"),
 error_summary_functions = c("mae", "mse", "gmae", "mape", "smape", "mrae", "mdrae",
    "gmrae", "mase", "smase", "msis")
)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
in_sample	in sample (hystorical values) of time series
frequency	frequency of time series

Value

list with summary and details of accuracy measures

`accuracy_detail` *Details (horizon specific) accuracy measures*

Description

Details (horizon specific) accuracy measures

Usage

```
accuracy_detail(
  y_true,
  y_pred,
  y_bench,
  in_sample,
  frequency,
  error_detail_functions = c("ae", "se", "ape", "sape", "rae", "ase", "sase")
)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
in_sample	in sample (hystorical values) of time series
frequency	frequency of time series

Value

horizon specific accuracy measures

`accuracy_summary` *Summary of accuracy measures*

Description

Summary of accuracy measures

Usage

```
accuracy_summary(
  y_true,
  y_pred,
  y_bench,
  in_sample,
  upper,
  lower,
  alpha,
  frequency,
 error_summary_functions = c("mae", "mse", "gmae", "mape", "smape", "mrae", "mdrae",
    "gmrae", "mase", "smase", "msis"),
  ...
)
```

Arguments

`y_true`	actual value of time series
`y_pred`	predicted value of time series
`in_sample`	in sample (hystorical values) of time series
`frequency`	frequency of time series

Value

summary and details of measures

ae *Absolut error (AE)*

Description

Absolut error (AE)

Usage

```
ae(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns a vector of absolute differences

ape *Absolute percentage error (APE)*

Description

Absolute percentage error (APE)

Usage

```
ape(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns vector of absolute percentage errors

ase *Absolut scaled error (ASE)*

Description

Absolut scaled error (ASE)

Usage

```
ase(y_true, y_pred, in_sample, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
in_sample	in sample (hystorical values) of time series

Value

Returns the absolute differences scaled by the mean in-sample error of naive

auto_arima *Auto Arima*

Description

Auto Arima

Usage

```
auto_arima(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_ces *Auto Complex exponential Smoothing*

Description

Auto Complex exponential Smoothing

Usage

```
auto_ces(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_damped *Auto Damped Exponentional Smoothing*

Description

Auto Damped Exponentional Smoothing

Usage

```
auto_damped(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_dotm *Auto Theta dotm*

Description

Auto Theta dotm

Usage

```
auto_dotm(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_ets *Auto ETS*

Description

Auto ETS

Usage

```
auto_ets(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_holt *Auto Holt Exponential Smoothing*

Description

Auto Holt Exponential Smoothing

Usage

```
auto_holt(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_naive *Auto Naive*

Description

Auto Naive

Usage

```
auto_naive(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_nnar *Auto autoregressive neural networke*

Description

Experimental state.

Usage

```
auto_nnar(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_ses *Auto Simple Exponential Smoothing*

Description

Auto Simple Exponential Smoothing

Usage

```
auto_ses(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_shd *Auto SHD Combo*

Description

Auto SHD Combo

Usage

```
auto_shd(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_snaive *Auto seasonal naive*

Description

Auto seasonal naive

Usage

```
auto_snaive(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

auto_theta *Auto Tehta*

Description

Combination of univariate theta methods, using a combination operator of choice for the mean forecasts and prediction intervals.

Usage

```
auto_theta(
  y,
  h,
  level = 95,
  point_combination = "weighted_average",
  pi_combination_upper = "median",
  pi_combination_lower = "median",
  pool_limit = 3,
  error_fun = "rmse",
  weight_fun = "inverse",
  val_h = h,
  sov_only = F,
  max_years = 30,
  val_min_years = 4,
  cv_min_years = 5,
  cv_max_samples = 3,
  allow_negatives = F,
  ...
)
```

Arguments

y	Time series object with historical values.
h	Horizons to be predicted.
point_combination	Point forecast combination operator. Optional, default meidan.
pi_combination_upper	combination operator of the upper bound of the prediction intervals. Optional, default max.
pi_combination_lower	combination operator for the lower bounds of the prediction intervals. Optional, default min.
pool_limit	number of methods selected from the pool to reach the final forecasts. Optional, default length(methods).
error_fun	error function to determine the validation performance Optional, default rmse.

weight_fun	weight function for calculating the definitive weights for combination. Optional, default inverse
val_h	The horizon for the validation samples. Optional, default h.
sov_only	Flag indicating whether only single origin validation should be considered. Optional, default TRUE.
max_years	Maxmium of years to consider during model fitting. Optional, default 30.
val_min_years	Minimum years required to conduct single origin validation. Optional, default 4.
cv_min_years	Minimum years required to conduct cross-origin validation. Optional, default 5.
cv_max_samples	Maximum samples that should be considered during cross-validation, i.e. 3 indicates the algorithm validates from 3 origins. Optional, default 3.
allow_negatives	Flag indicating whether to allow negative values or not. Optional, default FALSE.
...	passed to the forecasting functions
levels	Prediction interval levels. Optional, default c(95).
methods	Methods to be combined. Optional, default auto_ets,auto_arima,auto_dotm.

Value

combined forecasts of time series y including confidence intervals

---

`auto_thetaf` *Auto thetaf*

Description

Auto thetaf

Usage

```
auto_thetaf(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

boxcox *BoxCox transformation*

Description

BoxCox transformation wrapper around the function `forecast::BoxCox()`.

Usage

```
boxcox(y, ...)
```

Arguments

`y`	time series vector
`...`	not used

Value

list with two members; res is the transformed time series and param the used parameters for the transformation.

combine_theta_lines *Theta forecasts*

Description

Theta forecasts

Usage

```
combine_theta_lines(y, h, combination, tl_zero, tl_two, theta)
```

Arguments

`y`	historical observation in form of a time series object
`h`	number of horizons to predict
`combination`	combination type
`tl_zero`	theta line zero
`tl_two`	theta line two
`theta`	theta parameter

Value

point forecasts including one prediction interval

date_increment *Increment for date sequence*

Description

To be improved or removed

Usage

```
date_increment(y, direction = "future", ...)
```

Arguments

y	teime series
direction	direction of the increment
...	not used

Value

increment

decimal_to_date *Decimal to date*

Description

Decimal to date

Usage

```
decimal_to_date(dec_date, ...)
```

Arguments

dec_date	decimal date
...	not used

Value

date

diff *Differencing of a vector*

Description

Calculates the differences of time series over time.

Usage

```
diff(y, n.diffs = 1, auto = T, ...)
```

Arguments

y	vector which is correctly ordered with regards to time
n.diffs	integer (default: 1) of differences to be conducted
auto	boolean (default: T), whether number of differences should be defined automatically
...	

Value

list with two members (res and param); where res is the preporcessed vector and param the used parameters

exp_inverse *Calculates the exponential inverse*

Description

Calculates the exponential inverse

Usage

```
exp_inverse(x, ...)
```

Arguments

x	error metrics (higher is worse)
...	not used

Value

the weights

forecast_forunco *Forunco function for batch forecasting in R*

Description

Implements the forunco function for optimal parallelisation with machine learning services of sql server 2017

Usage

```
forecast_forunco(
  ts_col,
  h = 12,
  num_cores = NULL,
  num_cores_ignore = 1,
  prog_bar = T,
  levels = c(95),
  methods = c("auto_ets", "auto_arima", "auto_dotm"),
  point_combination = "median",
  pi_combination_upper = "median",
  pi_combination_lower = "median",
  pool_limit = length(methods),
  error_fun = "rmse",
  weight_fun = "inverse",
  val_h = h,
  sov_only = F,
  max_years = 30,
  val_min_years = 4,
  cv_min_years = 5,
  cv_max_samples = 3,
  allow_negatives = F,
  ...
)
```

Arguments

ts_col	list of time series objects
h	number of horizons to predict
num_cores	number of cores to be used; default: NULL (all cores)
num_cores_ignore	number of cores to be ignored for e.g. OS; default: 1
prog_bar	boolean, whether or not a progress bar should be displyed
levels	Prediction interval levels. Optional, default `c(95)`.
methods	Methods to be combined. Optional, default `auto_ets,auto_arima,auto_dotm`.

point_combination
: Point forecast combination operator. Optional, default meidan.

pi_combination_upper
: combination operator of the upper bound of the prediction intervals. Optional, default max.

pi_combination_lower
: combination operator for the lower bounds of the prediction intervals. Optional, default min.

pool_limit
: number of methods selected from the pool to reach the final forecasts. Optional, default length(methods).

error_fun
: error function to determine the validation performance Optional, default rmse.

weight_fun
: weight function for calculating the definitive weights for combination. Optional, default inverse

val_h
: The horizon for the validation samples. Optional, default h.

sov_only
: Flag indicating whether only single origin validation should be considered. Optional, default TRUE.

max_years
: Maxmium of years to consider during model fitting. Optional, default 30.

val_min_years
: Minimum years required to conduct single origin validation. Optional, default 4.

cv_min_years
: Minimum years required to conduct cross-origin validation. Optional, default 5.

cv_max_samples
: Maximum samples that should be considered during cross-validation, i.e. 3 indicates the algorithm validates from 3 origins. Optional, default 3.

allow_negatives
: Flag indicating whether to allow negative values or not. Optional, default FALSE.

...

Value

list with mean, pis and data_frame containing the predicted values

Examples

```
## Not run:
library(tritelligence)
library(Mcomp)
m3 <- M3[2000:2025]
#a time series colllection has n elements, each of which is a time series
# object.
ts_col <- lapply(m3, function(x) {x$x})

result <- forecast_forunco(ts_col)
preds[[1]]
$mean
[1] 5054.439 5048.170 5048.170 5048.170 5048.170 5048.170 5048.170 5048.170 5048.170 5048.170 5048.170 5048.170

$upper
[1] 6348.204 6883.349 7294.081 7640.378 7945.488 8221.338 8475.014 8711.134 8932.906 9142.665 9342.174 9532.805
```

```
$lower
[1] 3760.6740 3149.1485 2453.7666 2132.0499 2054.5814 1887.5394 1633.8633 1397.7433 1175.9717  894.6316  766.7032

$preds
# A tibble: 36 X 3
date  type    value
<date>     <chr>    <dbl>
1989-07-02 Point 5054.439
1989-08-02 Point 5048.170
1989-09-02 Point 5048.170
1989-10-02 Point 5048.170
1989-11-02 Point 5048.170
1989-12-02 Point 5048.170
1990-01-02 Point 5048.170
1990-02-02 Point 5048.170
1990-03-02 Point 5048.170
1990-04-02 Point 5048.170

## End(Not run)
```

---

forunco *Forunco combination approach*

Description

Combination of univariate time seris methods, using a combination operator of choice for the mean forecasts and prediction intervals.

Usage

```
forunco(
  y,
  h,
  levels = c(95),
  methods = c("auto_ets", "auto_arima", "auto_thetaf"),
  pp_methods = c("boxcox"),
  point_combination = "median",
  pi_combination_upper = "median",
  pi_combination_lower = "median",
  pool_limit = length(methods),
  error_fun = "rmse",
  weight_fun = "inverse",
  val_h = h,
  sov_only = F,
  max_years = 30,
  val_min_years = 4,
  cv_min_years = 5,
  cv_max_samples = 3,
```

```
  allow_negatives = F,
  remove_outliers = F,
  ...
)
```

Arguments

y	Time series object with historical values.
h	Horizons to be predicted.
levels	Prediction interval levels. Optional, default `c(95)`.
methods	Methods to be combined. Optional, default `auto_ets,auto_arima,auto_thetaf`.
point_combination	Point forecast combination operator. Optional, default `meidan`.
pi_combination_upper	combination operator of the upper bound of the prediction intervals. Optional, default `max`.
pi_combination_lower	combination operator for the lower bounds of the prediction intervals. Optional, default `min`.
pool_limit	number of methods selected from the pool to reach the final forecasts. Optional, default `length(methods)`.
error_fun	error function to determine the validation performance Optional, default `rmse`.
weight_fun	weight function for calculating the definitive weights for combination. Optional, default `inverse`
val_h	The horizon for the validation samples. Optional, default `h`.
sov_only	Flag indicating whether only single origin validation should be considered. Optional, default `TRUE`.
max_years	Maxmium of years to consider during model fitting. Optional, default `30`.
val_min_years	Minimum years required to conduct single origin validation. Optional, default `4`.
cv_min_years	Minimum years required to conduct cross-origin validation. Optional, default `5`.
cv_max_samples	Maximum samples that should be considered during cross-validation, i.e. 3 indicates the algorithm validates from 3 origins. Optional, default `3`.
allow_negatives	Flag indicating whether to allow negative values or not. Optional, default `FALSE`.
remove_outliers	Flag indicating whether to automatically remove outliers form the time series. default `FALSE`
...	passed to the forecasting functions

Value

combined forecasts of time series y including confidence intervals

Examples

```
# not run
library(Mcomp)
ts <- M3[[2104]]$x
fcs <- forunco(y = ts, h = 18)
# end not run
```

`generic_combine` *Generic combination function*

Description

This function is used for simpler combination methods supported out of the box from the stats package in R; such as e.g. `mean` or `median`

Usage

```
generic_combine(y_hat, c_fun, ...)
```

Arguments

`y_hat`	predicted values for test set
`...`	
`c.fun`	combination funciton

Value

combined forecasts

`generic_forecast` *Wrapper to preprecess, predict and postprocess forecasts*

Description

Wrapper to preprecess, predict and postprocess forecasts

Usage

```
generic_forecast(
  fcs.fun,
  y,
  h,
  level,
  pp.operations = c("seasonal_adjustment"),
  ...
)
```

Arguments

fcs.fun	forecasting function
y	historical values
h	number of horizons to predict
level	prediction interval levels
pp.operations	preprocessing operations to be conducted
...	not used

Value

forecast object

---

gmae *Geomatric mean absolute error (GMAE)*

Description

Geomatric mean absolute error (GMAE)

Usage

```
gmae(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns geomatric mean of squared errors

gmrae *Geomatric mean relative absolut error (GMRAE)*

Description

Geomatric mean relative absolut error (GMRAE)

Usage

```
gmrae(y_true, y_pred, y_bench, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
y_bench	forecasts of benchmark method

Value

Returns geometric mean of relative absolute differences

g_mean *Geometric Mean*

Description

Geometric Mean

Usage

```
g_mean(x, na.rm = TRUE)
```

Arguments

x	Vector to take geometric mean from
na.rm	Boolean (Optional, default: True) Wheater to remove NA values.

Value

geometric mean of vector

inverse *Calculates the inverse*

Description

Calculates the inverse

Usage

```
inverse(x, ...)
```

Arguments

x	error metrics (higher is worse)
...	not used

Value

the weights

inv_boxcox *Inverse Boxcox transformation*

Description

Inverses the BoxCox transformation conducted in function `boxcox`. The funciton is a wrapper around the function `forecast::InvBoxCox()`.

Usage

```
inv_boxcox(y, param, ...)
```

Arguments

y	time series
param	parameter list
...	not used

Value

backtransformed time series vector

inv_diff *Inverses diff transformations*

Description

Backtransformes diff transformations performed in `diff()`. Support for multiple transformations.

Usage

```
inv_diff(y, param, is_fcs = T, ...)
```

Arguments

`y`	time series vector
`param`	parameter of the transformation function
`is_fcs`	boolean, default (T). whether it is a forecast or in-sample vector.
`...`	not used

Value

backtransformed time series

inv_log *Inverse of log transformation*

Description

Used to automatically perfrom multiply transformations using the preprocessor environment; basicall calculates `exp` of input `y`.

Usage

```
inv_log(y, ...)
```

Arguments

`y`	time series vector.
`...`	not used

Value

backtransformed time series

inv_normalize *Inverses normalization*

Description

Reverses the normalization conducted in function `normalize()`.

Usage

```
inv_normalize(y, param = NA, ...)
```

Arguments

`y`	vector to be rescaled
`param`	parameters used for the original scaling
`...`	

Value

renormalized vector

inv_no_pp *Dummy inv PP*

Description

Dummy inv PP

Usage

```
inv_no_pp(y, ...)
```

Arguments

`y`	time series
`...`	

Value

same time series

inv_scale *Inverses scaling*

Description

Reverses the scaling conducted in function `scale()`.

Usage

```
inv_scale(y, param = NA, ...)
```

Arguments

`y`	vector to be rescaled
`param`	parameters used for the original scaling
`...`	

Value

rescaled vector

inv_seasonal_adjustment *Inversed seasonal adjustment*

Description

Backtransforms the seasonal adjustment conducted in function `seasonal_adjustment`.

Usage

```
inv_seasonal_adjustment(y, param, ...)
```

Arguments

`y`	time series
`param`	parameter list
`...`	not used

Value

Re-seasonalized time series

References

Makridakis SG, Wheelwright SC, Hyndman RJ (1998). Forecasting: Methods and applications (Third Edition). New York: Wiley.

is_seasonal *Seasonality test*

Description

Seasonality test

Usage

```
is_seasonal(y, frequency = frequency(y), ...)
```

Arguments

y	time series
frequency	frequency of time series vector
...	not used

Value

boolean (true if seasonal, false otherwise)

References

Makridakis SG, Wheelwright SC, Hyndman RJ (1998). Forecasting: Methods and applications (Third Edition). New York: Wiley.

log *Log tranformation*

Description

Log tranformation

Usage

```
log(y, ...)
```

Arguments

y	time series vecotr
...	not used

Value

transformed time series

mae *Mean absolute error (MAE)*

Description

Mean absolute error (MAE)

Usage

```
mae(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns ean of absolute differences

mape *Mean absolute percentage error (MAPE)*

Description

Mean absolute percentage error (MAPE)

Usage

```
mape(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns mean of absolute percentage differences

mase *Mean absolut scaled error (MASE)*

Description

Mean absolut scaled error (MASE)

Usage

```
mase(y_true, y_pred, in_sample, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
in_sample	in sample (hystorical values) of time series

Value

Returns mean of the absolute differences scaled by the mean in-sample error of naive

mdrae *Median relative absolut error (MRAE)*

Description

Median relative absolut error (MRAE)

Usage

```
mdrae(y_true, y_pred, y_bench, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
y_bench	forecasts of benchmark method

Value

Returns median of relative absolute differences

mrae *Mean relative absolut error (MRAE)*

Description

Mean relative absolut error (MRAE)

Usage

```
mrae(y_true, y_pred, y_bench, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
y_bench	forecasts of benchmark method

Value

Returns mean of relative absolute differences

mse *Mean square error (MSE)*

Description

Mean square error (MSE)

Usage

```
mse(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns mean of squared errors

msis *Mean Scaled interval score (MSIS)*

Description

Mean Scaled interval score (MSIS)

Usage

```
msis(y_true, upper, lower, alpha, in_sample, frequency, ...)
```

Arguments

y_true	true values of forecasts
upper	upper bound
lower	lower bound
alpha	significance level
in_sample	in_sample data
frequency	frequency of data
...	additional parameters, not used

Value

Average MSIS

normalize *Normalize vector*

Description

Normalize vector

Usage

```
normalize(y, ...)
```

Arguments

y	vector of occurences
...	

Value

list with two members (res and param); where res is the preporcessed vector and param the used parameters

no_pp *Dummy PP*

Description

Dummy PP

Usage

```
no_pp(y, ...)
```

Arguments

y	time series
...	

Value

same time series

plot_forunco *Plot of forunco object*

Description

Plot of forunco object

Usage

```
plot_forunco(forunco, interactive = F, future_vals = NA, ...)
```

Arguments

forunco	forunco object
interactive	whether or not to return an interactive plot or a static one
...	addtional argument, currently not used
add_data	additional data

Value

ggplot/plotly object

pool_mean *Calculates the mean of the pool*

Description

Calculates the mean of the pool

Usage

```
pool_mean(x, ...)
```

Arguments

x	error metrics (higher is worse)
...	not used

Value

the weights

pool_median *Calculates the mean of the pool*

Description

Calculates the mean of the pool

Usage

```
pool_median(fcs, ...)
```

Arguments

fcs	pooled forecasts
...	not used

Value

the weights

preprocessor *Preprocessor environment*

Description

Can be used to flexibly transforming and backtransforming data. Generally all functions - including custom ones in the current environment are supported. The function assumes that each transformation funcotion (e.g. `normalize <-function (x,...) {}`) has a backtransformation function with the same name but a `inv_` prefix (e.g. `inv_normalize <-function(x,param,...){}`). All parameter used for transforming the variables can be stored in the param variable; which will be passed automatically back to the backtransformation function.

Usage

```
preprocessor()
```

Details

The preprocessor has to "instantiated" and can then be called using the following parameters:

- "y"time series or forecast to be transformed/backtransformed
- "action"action to be taked, either 'transform' or 'backtransform'
- "operations"transformatoin operations to be takes (functions to be called)
- "n.diffs"integer default (NA). Sets the number of differencing to be taken (used only when diff is part of operator)
- "auto"boolean, default (T), whether the number of differencing is to be determined automatically
- "is_fcs"boolean ,default (T), wheter it is forecast or in-sample data to be transformed
- "frequency"inteher, default (NA), the frequency of the time series (used for seasonal adjustment)
- "h"horizon, default (1)

Value

returns either transformed or backtransformed time series vector

Examples

```
# not run
# load demo data
library(Mcomp)
# get demo time series
demo_ts <- M3[[1560]]$x
# show demo ts
demo_ts
# get preprocessor "object" (environment)
```

```
pp <- preprocessor()
# preprocess time series
y_pp <- pp(y = demo_ts,
           action = 'transform',
           operations = c('seasonal_adjustment', 'scale'),
           frequency = frequency(demo_ts))
# show preprocessed time series
y_pp
# ok, now go back to the roots
y_back <- pp(y = y_pp, action = 'backtransform', is_fcs = F)
y_back - demo_ts
# end not run
```

`produce_forecasts` *Produces forecasts*

Description

Using the predefined methods, this function produces point forecasts as well as prediction intervals of any given level.

Usage

```
produce_forecasts(
  y,
  h,
  levels = c(80, 95),
  methods = c("auto_ets", "auto_arima", "auto_thetaf"),
  ...
)
```

Arguments

`y`	Time series object.
`h`	Horizons to be predicted.
`levels`	Prediction interval levels. Optional, default `c(95)`.
`methods`	Methods to be combined. Optional, default `auto_ets,auto_arima,auto_thetaf`.
`...`	passed to the forecasting functions
`pp_obj`	preprocessing parameters

Value

combined forecasts of time series y including confidence intervals

rae *Relative absolut error (RAE)*

Description

Relative absolut error (RAE)

Usage

```
rae(y_true, y_pred, y_bench, epsilon = 0.001, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
y_bench	forecasts of benchmark method

Value

Returns a vector of relative absolute differences

replace_outliers *Replacement of outliers*

Description

This function is a wrapper around the funciton `forecast::tsoutliers()`.

Usage

```
replace_outliers(y, frequency, start = 1, ...)
```

Arguments

y	time series vector
frequency	frequency of time series vector
start	start date, default (1)
...	not used

Value

timeseries without outliers

rmse *Root mean square error (MSE)*

Description

Root mean square error (MSE)

Usage

```
rmse(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns root mean of squared errors

rx_sql_forunco *forunco function for SQL Server compute context*

Description

Implements the forunco function for optimal parallelisation with machine learning services of sql server 2017

Usage

```
rx_sql_forunco(
  connection_string,
  table,
  num_cores = 8,
  h = 2,
  levels = c(95),
  methods = c("auto_ets", "auto_arima", "auto_thetaf"),
  pp_methods = c("boxcox"),
  point_combination = "median",
  pi_combination_upper = "median",
  pi_combination_lower = "median",
  pool_limit = length(methods),
  error_fun = "rmse",
  weight_fun = "inverse",
  val_h = h,
```

```
  sov_only = F,
  max_years = 30,
  val_min_years = 4,
  cv_min_years = 5,
  cv_max_samples = 3,
  allow_negatives = F,
  ...
)
```

Arguments

connection_string	mandatory: Connectionstring to the database
table	mandatory: Table name that is the source for the forecasting objects
num_cores	number of cores to be used, default 8.
h	number of horizons to predict.
levels	Prediction interval levels. Optional, default c(95).
methods	Methods to be combined. Optional, default auto_ets,auto_arima,auto_dotm.
point_combination	Point forecast combination operator. Optional, default meidan.
pi_combination_upper	combination operator of the upper bound of the prediction intervals. Optional, default max.
pi_combination_lower	combination operator for the lower bounds of the prediction intervals. Optional, default min.
pool_limit	number of methods selected from the pool to reach the final forecasts. Optional, default length(methods).
error_fun	error function to determine the validation performance Optional, default rmse.
weight_fun	weight function for calculating the definitive weights for combination. Optional, default inverse
val_h	The horizon for the validation samples. Optional, default h.
sov_only	Flag indicating whether only single origin validation should be considered. Optional, default TRUE.
max_years	Maxmium of years to consider during model fitting. Optional, default 30.
val_min_years	Minimum years required to conduct single origin validation. Optional, default 4.
cv_min_years	Minimum years required to conduct cross-origin validation. Optional, default 5.
cv_max_samples	Maximum samples that should be considered during cross-validation, i.e. 3 indicates the algorithm validates from 3 origins. Optional, default 3.
allow_negatives	Flag indicating whether to allow negative values or not. Optional, default FALSE.
...	

Value

forecest data.frame

Examples

```
## Not run:
connection_string <- "Server=LTYMA01\\QWERTZ;Database=FORECASTING_DATA;UID=ruser;PWD=ruser;"
table <- paste0("m3.vw_Micro")
result <- rx_sql_forunco(connection_string=connection_string, table=table)
#    .id  TSN       date  type       value
#     P1 1403 1994-03-02 Point  1468.48451
#     P1 1403 1994-03-30 Point  1468.48451
#     P1 1403 1994-04-30 Point  1468.48451
#     P1 1403 1994-05-30 Point  1468.48451
#     P1 1403 1994-06-30 Point  1468.48451

## End(Not run)
```

---

sape *Symmetric absolute percentage error (sAPE)*

Description

Symmetric absolute percentage error (sAPE)

Usage

```
sape(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns vector of symmetric absolute percentage differences

sase *Seasonal absolut scaled error (ASE)*

Description

Seasonal absolut scaled error (ASE)

Usage

```
sase(y_true, y_pred, in_sample, frequency, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
in_sample	in sample (hystorical values) of time series
frequency	frequency of time series

Value

Returns the absolute differences scaled by the mean in-sample error of seasonal naive

scale *Scale vector*

Description

Scale vector

Usage

```
scale(y, ...)
```

Arguments

y	vector of occurences
...	

Value

list with two members (res and param); where res is the preporcessed vector and param the used parameters

se *Squared error (SE)*

Description

Squared error (SE)

Usage

```
se(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns a vector of saured differences

seasonal_adjustment *Conductes a seasonal adjustment*

Description

Seasonally adjusts a time series vector using the multiplicative decomposition approach.

Usage

```
seasonal_adjustment(y, frequency = frequency, h = h, type = "M", ...)
```

Arguments

y	time series vector
frequency	frequency of time series
h	horizon of time series
...	not used

Value

list with two members; res is the seasonally adjusted time series; param the seasonal components to reconstruct the time series or forecasts in `inv_seasonal_adjustment`.

shd *Conventional SHD Combination*

Description

Conventional SHD Combination

Usage

```
shd(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

simple_combination *Simple combination of univariate time series forecasting methods*

Description

Combines muultiple forecasts using definable combination operators.

Usage

```
simple_combination(
  y,
  h,
  fcs_mats = NA,
  levels,
  methods,
  point_combination = "median",
  pi_combination_upper = "median",
  pi_combination_lower = "median",
  allow_negatives = F,
  ...
)
```

Arguments

y	Time series object.
h	Horizons to be predicted.
levels	Prediction interval levels. Optional, default c(95).
methods	Methods to be combined. Optional, default auto_ets,auto_arima,auto_thetaf.
point_combination	Point forecast combination operator. Optional, default meidan.
pi_combination_upper	combination operator of the upper bound of the prediction intervals. Optional, default max.
pi_combination_lower	combination operator for the lower bounds of the prediction intervals. Optional, default min.
allow_negatives	Flag indicating whether to allow negative values or not. Optional, default FALSE.
...	passed to the forecasting functions

Value

combined forecasts of time series y including confidence intervals

---

smape *Symmetric mean absolute percentage error (sMAPE)*

Description

Symmetric mean absolute percentage error (sMAPE)

Usage

```
smape(y_true, y_pred, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series

Value

Returns mean of symmetric absolute percentage differences

smase *Seasonal Mean absolut scaled error (sMASE)*

Description

Seasonal Mean absolut scaled error (sMASE)

Usage

```
smase(y_true, y_pred, in_sample, frequency, ...)
```

Arguments

y_true	actual value of time series
y_pred	predicted value of time series
in_sample	in sample (hystorical values) of time series
frequency	frequency of time series

Value

Returns the mean of the absolute differences scaled by the mean in-sample error of seasonal naive

squared_inverse *Calculates the squared inverse*

Description

Calculates the squared inverse

Usage

```
squared_inverse(x, ...)
```

Arguments

x	error metrics (higher is worse)
...	not used

Value

the weights

theta_aea *AEA theta model*

Description

AEA theta model

Usage

```
theta_aea(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

theta_aem *AEM theta model*

Description

AEM theta model

Usage

```
theta_aem(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

theta_ala *ALM theta model*

Description

ALM theta model

Usage

```
theta_ala(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

theta_alm *ALM theta model*

Description

ALM theta model

Usage

```
theta_alm(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

theta_fit *Theta fit*

Description

This method is inspired by the implementation of E. Spiliotis and V. Assimakopoulos (2017) / Forecasting & Strategy Unit - NTUA.

Usage

```
theta_fit(y, h, level = 95, theta, curve, combination, seasonality)
```

Arguments

y	historical observation in form of a time series object
h	number of horizons to predict
level	prediction interval level, only one allowed.
theta	theta parameter.
curve	curve type.
seasonality	seasonality type.
model	model type.

Value

point forecasts including one prediction interval level

theta_forecast *theta forecast*

Description

theta forecast

Usage

```
theta_forecast(y, h, level = 95, model = "MEM")
```

Arguments

y	historical observations
h	horizons to be predicted
level	prediction interval level
model	model to be predicted

Value

forecast object

`theta_line` *Theta line*

Description

Theta line

Usage

```
theta_line(y, tl_zero, theta, combination)
```

Arguments

`y`	historical observarions
`tl_zero`	theta line zero
`theta`	theta parameter
`combination`	combination type

Value

theta line to be fitted by ses

`theta_line_zero` *Theta line zero*

Description

Theta line zero

Usage

```
theta_line_zero(y, h, curve, level, ...)
```

Arguments

`y`	historical observations
`h`	horizons to predict
`curve`	curve type
`level`	prediction interval level
`...`	

Value

thea line zero

theta_mea *MEA theta model*

Description

MEA theta model

Usage

```
theta_mea(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

theta_mem *MEM theta model*

Description

MEM theta model

Usage

```
theta_mem(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

theta_mla *MLA theta model*

Description

MLA theta model

Usage

```
theta_mla(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

theta_mlm *MLM theta model*

Description

MLM theta model

Usage

```
theta_mlm(y, h, level, ...)
```

Arguments

y	historical values
h	number of horizons to predict
level	prediction interval levels
...	not used

Value

forecast object

tidier_ts *Tidy time series in data frame*

Description

Tidy time series in data frame

Usage

```
tidier_ts(y, type = "actual", ...)
```

Arguments

y	observations
type	type of observations
...	not used

Value

tidy ts

time_sequence *Time sequence*

Description

To be improved or removed in the future.

Usage

```
time_sequence(y, start_pos = "max", out_length)
```

Arguments

y	observations
start_pos	where to start
out_length	how many to generate

Value

time sequence

validation_split *Validation Split*

Description

This function splits a time series into validation sets with training and test data; the output can be used to backtest models and/or select/combine models based on how they perfrom on this sets.

Usage

```
validation_split(
  y,
  val_h,
  sov_only = F,
  val_min_years = 4,
  cv_min_years = 5,
  cv_max_samples = 3,
  ...
)
```

Arguments

y	Time series onject.
val_h	Horizon to be used for validation.
sov_only	Flag, whether only single-origin validation should be considered. Optional, default TRUE.
val_min_years	Minimum years required to conduct single origin validation. Optional, default 4.
cv_min_years	Minimum years required to conduct cross-origin validation. Optional, default 5.
cv_max_samples	Maximum samples that should be considered during cross-validation, i.e. 3 indicates the algorithm validates from 3 origins. Optional, default 3.
...	not used

Value

list of validation sets

`weighted_average` *Averaging forecasts using weights*

Description

Combines forecasting methods based on the validation error. See Nowotarski, J., Raviv, E., Tr\"uck, S., and Weron, R. (2014) for more examples.

Usage

```
weighted_average(
  val_hat,
  val_true,
  y_hat,
  error_fun = "rmse",
  weight_fun = "inverse",
  pool_limit = 3,
  ...
)
```

Arguments

`val_hat`	forcasts on the validation set
`val_true`	true values of the validation set
`y_hat`	forecasts for the test set
`weight_fun`	the function to determine the weights
`pool_limit`	how many methods should be considered for combination
`...`	not used
`error_measure`	error measure to be used for error calculation; error measures that calculate errors per horizons will lead to a horizon specific combination, error measures over all horizons on the other hand, will lead to a simple weighted combination.

Value

combined forecasts using imrse for weighting

References

Nowotarski, J., Raviv, E., Tr\"uck, S., and Weron, R. (2014). An Empirical Comparison of Alternative Schemes for Combining Electricity Spot Price Forecasts. *Energy Economics*, **46**, 395–412.

`weight_error` *Weighting and pooling methods on the basis of their error*

Description

Weighting and pooling methods on the basis of their error

Usage

```
weight_error(weight_fun, error, y_hat, pool_limit, ...)
```

Arguments

`weight_fun`	weight function
`error`	the error vector which should be weightened
`pool_limit`	the pool size to be considered (e.g. 3 would indicate that only the best three methods would be selected)
`...`	passed to weight_fun

Value

the weights

Index

.rx_forecast, 3

accuracy, 4
accuracy_detail, 5
accuracy_summary, 6
ae, 7
ape, 7
ase, 8
auto_arima, 8
auto_ces, 9
auto_damped, 9
auto_dotm, 10
auto_ets, 10
auto_holt, 11
auto_naive, 11
auto_nnar, 12
auto_ses, 12
auto_shd, 13
auto_snaive, 13
auto_theta, 14
auto_thetaf, 15

boxcox, 16

combine_theta_lines, 16

date_increment, 17
decimal_to_date, 17
diff, 18

exp_inverse, 18

forecast_forunco, 19
forunco, 21

g_mean, 25
generic_combine, 23
generic_forecast, 23
gmae, 24
gmrae, 25

inv_boxcox, 26
inv_diff, 27
inv_log, 27
inv_no_pp, 28
inv_normalize, 28
inv_scale, 29
inv_seasonal_adjustment, 29
inverse, 26
is_seasonal, 30

log, 30

mae, 31
mape, 31
mase, 32
mdrae, 32
mrae, 33
mse, 33
msis, 34

no_pp, 35
normalize, 34

plot_forunco, 35
pool_mean, 36
pool_median, 36
preprocessor, 37
produce_forecasts, 38

rae, 39
replace_outliers, 39
rmse, 40
rx_sql_forunco, 40

sape, 42
sase, 43
scale, 43
se, 44
seasonal_adjustment, 44
shd, 45
simple_combination, 45

smape, 46
smase, 47
squared_inverse, 47

theta_aea, 48
theta_aem, 48
theta_ala, 49
theta_alm, 49
theta_fit, 50
theta_forecast, 50
theta_line, 51
theta_line_zero, 51
theta_mea, 52
theta_mem, 52
theta_mla, 53
theta_mlm, 53
tidier_ts, 54
time_sequence, 54

validation_split, 55

weight_error, 57
weighted_average, 56