

# Package ‘TSPred’

January 21, 2021

**Type** Package

**Title** Functions for Benchmarking Time Series Prediction

**Version** 5.1

**Date** 2021-01-14

**Author** Rebecca Pontes Salles [aut, cre, cph] (CEFET/RJ),  
Eduardo Ogasawara [ths] (CEFET/RJ)

**Maintainer** Rebecca Pontes Salles <rebeccapsalles@acm.org>

**Description** Functions for defining and conducting a time series prediction process including pre(post)processing, decomposition, modelling, prediction and accuracy assessment. The generated models and its yielded prediction errors can be used for benchmarking other time series prediction methods and for creating a demand for the refinement of such methods. For this purpose, benchmark data from prediction competitions may be used.

**Depends** R (>= 3.5.0)

**Imports** forecast, KFAS, stats, MuMIn, EMD, wavelets, vars,  
ModelMetrics, RSNNS, Rlibeemd, e1071, elmNNRcpp, nnet,  
randomForest, magrittr, plyr, methods, dplyr, keras, tfdatasets

**License** GPL (>= 2)

**BugReports** <https://github.com/RebeccaSalles/TSPred/issues>

**URL** <https://github.com/RebeccaSalles/TSPred/wiki>

**RoxygenNote** 7.1.1

**LazyData** true

**Encoding** UTF-8

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-01-21 11:30:03 UTC

**R topics documented:**

TSPred-package . . . . .	3
an . . . . .	4
ARIMA . . . . .	5
arimainterp . . . . .	8
arimaparameters . . . . .	9
arimapred . . . . .	10
BCT . . . . .	12
benchmark . . . . .	14
CATS . . . . .	16
CATS.cont . . . . .	17
detrend . . . . .	18
Diff . . . . .	19
emd . . . . .	20
EUNITE.Loads . . . . .	22
EUNITE.Loads.cont . . . . .	24
EUNITE.Reg . . . . .	28
EUNITE.Reg.cont . . . . .	29
EUNITE.Temp . . . . .	30
EUNITE.Temp.cont . . . . .	31
evaluate . . . . .	32
evaluate.tspred . . . . .	33
evaluating . . . . .	35
fittestArima . . . . .	36
fittestArimaKF . . . . .	38
fittestEMD . . . . .	41
fittestLM . . . . .	43
fittestMAS . . . . .	46
fittestPolyR . . . . .	49
fittestPolyRKF . . . . .	52
fittestWavelet . . . . .	55
ipeadata_d . . . . .	59
ipeadata_m . . . . .	60
LogT . . . . .	62
LT . . . . .	63
MAPE . . . . .	66
marimapar . . . . .	67
marimapred . . . . .	68
mas . . . . .	70
MAXError . . . . .	71
minmax . . . . .	72
modeling . . . . .	73
MSE . . . . .	75
MSE_eval . . . . .	76
NMSE . . . . .	77
NN3.A . . . . .	78
NN3.A.cont . . . . .	82

NN5.A . . . . .	86
NN5.A.cont . . . . .	90
outliers_bp . . . . .	94
pct . . . . .	95
plotarimapred . . . . .	96
postprocess.tspred . . . . .	98
predict . . . . .	99
predict.tspred . . . . .	101
preprocess . . . . .	102
preprocess.tspred . . . . .	103
processing . . . . .	105
SantaFe.A . . . . .	106
SantaFe.A.cont . . . . .	107
SantaFe.D . . . . .	108
SantaFe.D.cont . . . . .	109
sMAPE . . . . .	110
subset . . . . .	111
sw . . . . .	112
train . . . . .	113
train.tspred . . . . .	114
train_test_subset . . . . .	116
tspred . . . . .	117
WaveletT . . . . .	118
workflow . . . . .	120
<b>Index</b>	<b>123</b>

**Description**

Functions for time series pre(post)processing, decomposition, modelling, prediction and accuracy assessment. The generated models and its yielded prediction errors can be used for benchmarking other time series prediction methods and for creating a demand for the refinement of such methods. For this purpose, benchmark data from prediction competitions may be used.

**Author(s)**

Rebecca Pontes Salles

Maintainer: rebeccapsalles@acm.org

---

an *Adaptive Normalization*

---

### Description

The `an()` function normalizes data of the provided time series to bring values into the range [0,1]. The function applies the method of Adaptive Normalization designed for non-stationary heteroscedastic (with non-uniform volatility) time series. `an.rev()` reverses the normalization.

### Usage

```
an(data, max = NULL, min = NULL, byRow = TRUE, outlier.rm = TRUE, alpha = 1.5)
```

```
an.rev(data, max, min, an)
```

### Arguments

<code>data</code>	A numeric matrix with sliding windows of time series data as returned by <code>sw</code> .
<code>max</code>	A numeric vector indicating the maximal values of each row (sliding window) in data. If NULL it is automatically computed.
<code>min</code>	A numeric vector indicating the minimum values of each row (sliding window) in data. If NULL it is automatically computed.
<code>byRow</code>	If TRUE, the normalization is performed by rows (sliding windows), the default.
<code>outlier.rm</code>	If TRUE, outlier values are removed from the data during the normalization process, the default.
<code>alpha</code>	The multiplier for the interquartile range used as base for outlier removal. The default is set to 1.5. The value 3.0 is also commonly used to remove only the extreme outliers.
<code>an</code>	The mean of each data window computed by <code>an()</code> and returned as attribute.

### Value

data normalized between 0 and 1. `max` and `min` are returned as attributes, as well as the mean values of each row (sliding window) in data (`an`).

### Author(s)

Rebecca Pontes Salles

### References

E. Ogasawara, L. C. Martinez, D. De Oliveira, G. Zimbrao, G. L. Pappa, and M. Mattoso, 2010, Adaptive Normalization: A novel data normalization approach for non-stationary time series, Proceedings of the International Joint Conference on Neural Networks.

**See Also**

Other normalization methods: [minmax\(\)](#)

**Examples**

```
data(CATS)
swin <- sw(CATS[,1],5)
d <- an(swin, outlier.rm=FALSE)
x <- an.rev(d, max=attributes(d)$max, min=attributes(d)$min, an=attributes(d)$an)
all(round(x,4)==round(swin,4))
```

---

ARIMA

*Time series prediction models*

---

**Description**

Constructors for the modeling class representing a time series modeling and prediction method based on a particular model.

**Usage**

```
ARIMA(train_par = list(), pred_par = list(level = c(80, 95)))
```

```
ETS(train_par = list(), pred_par = list(level = c(80, 95)))
```

```
HW(train_par = list(), pred_par = list(level = c(80, 95)))
```

```
TF(train_par = list(), pred_par = list(level = c(80, 95)))
```

```
NNET(
  size = 5,
  train_par = NULL,
  pred_par = list(level = c(80, 95)),
  sw = SW(window_len = size + 1),
  proc = list(MM = MinMax())
)
```

```
RFrst(
  ntree = 500,
  train_par = NULL,
  pred_par = list(level = c(80, 95)),
  sw = SW(window_len = 6),
  proc = list(MM = MinMax())
)
```

```
RBF(  
  size = 5,  
  train_par = NULL,  
  pred_par = list(level = c(80, 95)),  
  sw = SW(window_len = size + 1),  
  proc = list(MM = MinMax())  
)  
  
SVM(  
  train_par = NULL,  
  pred_par = list(level = c(80, 95)),  
  sw = SW(window_len = 6),  
  proc = list(MM = MinMax())  
)  
  
MLP(  
  size = 5,  
  train_par = NULL,  
  pred_par = list(level = c(80, 95)),  
  sw = SW(window_len = size + 1),  
  proc = list(MM = MinMax())  
)  
  
ELM(  
  train_par = list(),  
  pred_par = list(),  
  sw = SW(window_len = 6),  
  proc = list(MM = MinMax())  
)  
  
Tensor_CNN(  
  train_par = NULL,  
  pred_par = list(level = c(80, 95)),  
  sw = SW(window_len = 6),  
  proc = list(MM = MinMax())  
)  
  
Tensor_LSTM(  
  train_par = NULL,  
  pred_par = list(batch_size = 1, level = c(80, 95)),  
  sw = SW(window_len = 6),  
  proc = list(MM = MinMax())  
)
```

### Arguments

<code>train_par</code>	List of named parameters required by <code>train_func</code> .
<code>pred_par</code>	List of named parameters required by <code>pred_func</code> .

size	See <a href="#">mlp</a>
sw	A <a href="#">SW</a> object regarding sliding windows processing.
proc	A list of <a href="#">processing</a> objects regarding any pre(post)processing needed during training or prediction.
ntree	See <a href="#">randomForest</a>

**Value**

An object of class `modeling`.

**Linear models**

ARIMA: ARIMA model. `train_func` set as [auto.arima](#) and `pred_func` set as [forecast](#).

ETS: Exponential Smoothing State Space model. `train_func` set as [ets](#) and `pred_func` set as [forecast](#).

HW: Holt-Winter's Exponential Smoothing model. `train_func` set as [hw](#) and `pred_func` set as [forecast](#).

TF: Theta Forecasting model. `train_func` set as [thetaf](#) and `pred_func` set as [forecast](#).

**Machine learning models**

NNET: Artificial Neural Network model. `train_func` set as [nnet](#) and `pred_func` set as [predict](#).

RFrst: Random Forest model. `train_func` set as [randomForest](#) and `pred_func` set as [predict](#).

RBF: Radial Basis Function (RBF) Network model. `train_func` set as [rbf](#) and `pred_func` set as [predict](#).

SVM: Support Vector Machine model. `train_func` set as [tune.svm](#) and `pred_func` set as [predict](#).

MLP: Multi-Layer Perceptron (MLP) Network model. `train_func` set as [mlp](#) and `pred_func` set as [predict](#).

ELM: Extreme Learning Machine (ELM) model. `train_func` set as [elm\\_train](#) and `pred_func` set as [elm\\_predict](#).

Tensor\_CNN: Convolutional Neural Network - TensorFlow. `train_func` based on functions from `tensorflow` and `keras`, and `pred_func` set as [predict](#).

Tensor\_LSTM: Long Short Term Memory Neural Networks - TensorFlow. `train_func` based on functions from `tensorflow` and `keras`, and `pred_func` set as [predict](#).

**Author(s)**

Rebecca Pontes Salles

**See Also**

Other constructors: [LT\(\)](#), [MSE\\_eval\(\)](#), [evaluating\(\)](#), [modeling\(\)](#), [processing\(\)](#), [tspred\(\)](#)

---

arimainterp	<i>Interpolation of unknown values using automatic ARIMA fitting and prediction</i>
-------------	---

---

### Description

The function predicts nonconsecutive blocks of N unknown values of a single time series using the [arimapred](#) function and an interpolation approach.

### Usage

```
arimainterp(
  TimeSeries,
  n.ahead,
  extrap = TRUE,
  xreg = NULL,
  newxreg = NULL,
  se.fit = FALSE
)
```

### Arguments

TimeSeries	A matrix, or data frame which contains a set of time series used for fitting ARIMA models. Each column corresponds to one time series. Each time series in TimeSeries is assumed to be a sequence of known values of the single time series that intercalates blocks of unknown values. The time series values in column 1 are lagged values of the ones in column 2, and the values in these two columns are assumed to be intercalated by the first block of N unknown values to be predicted. This is also valid for columns 2 and 3, and so forth.
n.ahead	A numeric value (N) with the number of consecutive unknown values of each block which is to be predicted of TimeSeries, that is, the length of the blocks of N unknown values.
extrap	A Boolean parameter which defines whether one of the blocks of N unknown values to be predicted follows the last sequence of known values in TimeSeries. If extrap is TRUE, the last block of N unknown values will be extrapolated from the last time series in TimeSeries.
xreg	A list of vectors, matrices, data frames or times series of external regressors used for fitting the ARIMA models. The first component of the list contains external regressors for the first time series in TimeSeries and therefore must have the same number of rows as this respective time series. This is also valid for the second component, and so forth. Ignored if NULL.
newxreg	A list of vectors, matrices, data frames or times series with further values of xreg to be used for prediction of the blocks of N unknown values. Each component of the list must have at least n.ahead rows. Ignored if NULL.
se.fit	If se.fit is TRUE, the standard errors of the predictions are returned.



## Details

In order to avoid error accumulation, when possible, the function provides the separate prediction of each half of the blocks of unknown values using their past and future known values, respectively. If `extrap` is `TRUE`, this strategy is not possible for the last of the blocks of unknown values, for whose prediction the function uses only its past values. By default the function omits any missing values found in `TimeSeries`.

## Value

A vector of time series of predictions, or if `se.fit` is `TRUE`, a vector of lists, each one with the components `pred`, the predictions, and `se`, the estimated standard errors. Both components are time series. See the [predict.Arima](#) function in the `stats` package and the function [arimapred](#).

## Author(s)

Rebecca Pontes Salles

## References

H. Cheng, P.-N. Tan, J. Gao, and J. Scripps, 2006, "Multistep-Ahead Time Series Prediction", In: W.-K. Ng, M. Kitsuregawa, J. Li, and K. Chang, eds., *Advances in Knowledge Discovery and Data Mining*, Springer Berlin Heidelberg, p. 765-774.

## See Also

[arimapred](#), [marimapred](#)

## Examples

```
data(CATS)
arimainterp(CATS[,c(2:3)],n.ahead=20,extrap=TRUE)
```

---

arimaparameters	<i>Get ARIMA model parameters</i>
-----------------	-----------------------------------

---

## Description

The function returns the parameters of a fitted ARIMA model, including non-seasonal and seasonal orders and drift.

## Usage

```
arimaparameters(fit)
```

**Arguments**

`fit` An object of class "Arima" containing a fitted ARIMA model.

**Details**

The `fit` object could possibly be the result of `auto.arima` or `Arima` of the `forecast` package, or `arima` of the `stats` package.

**Value**

A list giving the number of AR, MA, seasonal AR and seasonal MA coefficients, plus the period and the number of non-seasonal and seasonal differences of the provided ARIMA model. The value of the fitted drift constant is also presented.

**Author(s)**

Rebecca Pontes Salles

**References**

R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.

R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

`fittestArima`, `arimapred`

**Examples**

```
data(SantaFe.A)
arimaparameters(forecast::auto.arima(SantaFe.A[,1]))
```

---

arimapred

*Automatic ARIMA fitting and prediction*

---

**Description**

The function predicts and returns the next `n` consecutive values of a time series using an automatically fitted ARIMA model. It may also plot the predicted values against the actual ones using the function `plotarimapred`.

**Usage**

```

arimapred(
  timeseries,
  timeseries.cont = NULL,
  n.ahead = NULL,
  na.action = stats::na.omit,
  xreg = NULL,
  newxreg = NULL,
  se.fit = FALSE,
  plot = FALSE,
  range.p = 0.2,
  ylab = NULL,
  xlab = NULL,
  main = NULL
)

```

**Arguments**

<code>timeseries</code>	A vector or univariate time series which contains the values used for fitting an ARIMA model.
<code>timeseries.cont</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. Ignored if NULL.
<code>n.ahead</code>	Number of consecutive values of the time series, which are to be predicted. If <code>n.ahead</code> is NULL, the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.cont</code> . Required when <code>timeseries.cont</code> is NULL.
<code>na.action</code>	A function for treating missing values in <code>timeseries</code> and <code>timeseries.cont</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.cont</code> .
<code>xreg</code>	A vector, matrix, data frame or times series of external regressors used for fitting the ARIMA model. It must have the same number of rows as <code>timeseries</code> . Ignored if NULL.
<code>newxreg</code>	A vector, matrix, data frame or times series with new values of <code>xreg</code> to be used for prediction. Must have at least <code>n.ahead</code> rows or the number of rows in <code>timeseries.cont</code> . Ignored if NULL.
<code>se.fit</code>	If <code>se.fit</code> is TRUE, the standard errors of the predictions are returned.
<code>plot</code>	If <code>plot</code> is TRUE, the function will generate a graphic of the predicted values against the actual ones in <code>timeseries.cont</code> .
<code>range.p</code>	A percentage which defines how much the range of the graphic's y-axis will be increased from the minimum limits imposed by data.
<code>ylab</code>	A title for the graphic's y-axis. Ignored if NULL.
<code>xlab</code>	A title for the graphic's x-axis. Ignored if NULL.
<code>main</code>	An overall title for the graphic. Ignored if NULL.

### Details

The ARIMA model used for time series prediction is automatically fitted by the `auto.arima` function in the `forecast` package. In order to avoid drift errors, the function introduces an auxiliary regressor whose values are a sequence of consecutive integer numbers starting from 1. The fitted ARIMA model is used for prediction by the `predict.Arima` function in the `stats` package. For more details, see the `auto.arima` function in the `forecast` package and the `predict.Arima` function in the `stats` package.

### Value

A time series of predictions, or if `se.fit` is TRUE, a list with the components `pred`, the predictions, and `se`, the estimated standard errors. Both components are time series. See the `predict.Arima` function in the `stats` package.

### Author(s)

Rebecca Pontes Salles

### References

- R.J. Hyndman and G. Athanasopoulos, 2013, *Forecasting: principles and practice*. OTexts.
- R.H. Shumway and D.S. Stoffer, 2010, *Time Series Analysis and Its Applications: With R Examples*. 3rd ed. 2011 edition ed. New York, Springer.

### See Also

[auto.arima](#), [predict.Arima](#), [plotarimapred](#), [marimapred](#)

### Examples

```
data(SantaFe.A, SantaFe.A.cont)
arimapred(SantaFe.A[,1], SantaFe.A.cont[,1])
arimapred(SantaFe.A[,1], n.ahead=100)
```

---

BCT

*Box Cox Transformation*

---

### Description

The `BCT()` function returns a transformation of the provided time series using a Box-Cox transformation. `BCT.rev()` reverses the transformation. Wrapper functions for `BoxCox` and `InvBoxCox` of the `forecast` package, respectively.

**Usage**

```
BCT(x, lambda = NULL, ...)
```

```
BCT.rev(x, lambda, ...)
```

**Arguments**

x	A numeric vector or univariate time series of class <code>ts</code> .
lambda	Box-Cox transformation parameter. If <code>NULL</code> , lambda is selected using <code>BoxCox.lambda</code> of the <code>forecast</code> package.
...	Additional arguments passed to the <code>BoxCox.lambda</code> function for <code>BCT()</code> , and to the <code>InvBoxCox</code> function for <code>BCT.rev()</code> .

**Details**

If lambda is not 0, the Box-Cox transformation is given by

$$f_{\lambda}(x) = \frac{x^{\lambda} - 1}{\lambda}$$

If  $\lambda = 0$ , the Box-Cox transformation is given by

$$f_0(x) = \log(x)$$

**Value**

A vector of the same length as x containing the transformed values.

**Author(s)**

Rebecca Pontes Salles

**References**

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *JRSS B* **26** 211–246.

**See Also**

[DIF](#), [detrend](#), [MAS](#), [LT](#), [PCT](#)

**Examples**

```
data(CATS)
BCT(CATS[,1])
```

---

 benchmark

*Benchmarking a time series prediction process*


---

### Description

benchmark is a generic function for benchmarking results based on particular metrics. The function invokes particular *methods* which depend on the class of the first argument.

### Usage

```
benchmark(obj, ...)
```

```
## S3 method for class 'tspred'
benchmark(obj, bmrk_objs, rank.by = c("MSE"), ...)
```

### Arguments

obj	An object of class <code>tspred</code> defining a particular time series prediction process.
...	Ignored
bmrk_objs	A list of objects of class <code>tspred</code> to be compared against obj.
rank.by	A vector of the given names of the metrics that should base the ranking.

### Details

The function `benchmark.tspred` benchmarks a time series prediction process defined by a `tspred` object based on a particular metric. The metrics resulting from its execution are compared against the ones produced by other time series prediction processes (defined in a list of `tspred` objects).

### Value

A list containing:

rank	A data.frame with the ranking of metrics computed for the benchmarked <code>tspred</code> objects.
ranked_tspred_objs	A list of the benchmarked <code>tspred</code> objects ordered according to the produced rank.

### Author(s)

Rebecca Pontes Salles

### See Also

[`tspred()`] for defining a particular time series prediction process.

**Examples**

```

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class
mod1 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()
eval2 <- MAPE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod1,
                  evaluating=list(MSE=eval1,
                                MAPE=eval2)
                  )
summary(tspred_1)

#Obtaining objects of the processing class
proc4 <- SW(window_len = 6)
proc5 <- MinMax()

#Obtaining objects of the modeling class
mod12 <- NNET(size=5,sw=proc4,proc=list(MM=proc5))

#Defining a time series prediction process
tspred_2 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod12,
                  evaluating=list(MSE=eval1,
                                MAPE=eval2)
                  )
summary(tspred_2)

data("CATS")
data <- CATS[3]

tspred_1_run <- workflow(tspred_1,data=data,prep_test=TRUE,onestep=TRUE)
tspred_2_run <- workflow(tspred_2,data=data,prep_test=TRUE,onestep=TRUE)

b <- benchmark(tspred_1_run,list(tspred_2_run),rank.by=c("MSE"))

```

---

CATS

*Time series of the CATS Competition*

---

### Description

A univariate artificial time series presenting 5 non-consecutive blocks of 20 unknown points.

### Usage

CATS

### Format

A data frame with 980 observations on the following 5 variables.

**V1** a numeric vector containing the known points 1-980 of the CATS time series.

**V2** a numeric vector containing the known points 1001-1980 of the CATS time series.

**V3** a numeric vector containing the known points 2001-2980 of the CATS time series.

**V4** a numeric vector containing the known points 3001-3980 of the CATS time series.

**V5** a numeric vector containing the known points 4001-4980 of the CATS time series.

### Details

The CATS Competition presented an artificial time series with 5,000 points, among which 100 are unknown. The competition proposed that the competitors predicted the 100 unknown values from the given time series, which are grouped into five non-consecutive blocks of 20 successive values ([CATS.cont](#)). The unknown points of the series are the 981-1000, 1981-2000, 2981-3000, 3981-4000 and 4981-5000. The performance evaluation done by the CATS Competition was based on the MSEs computed on the 100 unknown values (E1) and on the 80 first unknown values (E2). The E2 error was considered relevant because some of the proposed methods used interpolation techniques, which cannot be applied in the case of the fifth set of unknown points.

### References

A. Lendasse, E. Oja, O. Simula, M. Verleysen, and others, 2004, Time Series Prediction Competition: The CATS Benchmark, In: IJCNN'2004-International Joint Conference on Neural Networks

A. Lendasse, E. Oja, O. Simula, and M. Verleysen, 2007, Time series prediction competition: The CATS benchmark, *Neurocomputing*, v. 70, n. 13-15 (Aug.), p. 2325-2329.

### See Also

[CATS.cont](#)



## Examples

```
data(CATS)
str(CATS)
plot(ts(CATS["V5"]))
```

---

CATS.cont

*Continuation dataset of the time series of the CATS Competition*

---

## Description

A dataset of providing the 5 blocks of 20 unknown points of the univariate time series in [CATS](#)

## Usage

```
CATS.cont
```

## Format

A data frame with 20 observations on the following 5 variables.

- V1** a numeric vector containing the unknown points 981-1000 of the CATS time series in [CATS](#)
- V2** a numeric vector containing the unknown points 1981-2000 of the CATS time series in [CATS](#)
- V3** a numeric vector containing the unknown points 2981-3000 of the CATS time series in [CATS](#)
- V4** a numeric vector containing the unknown points 3981-4000 of the CATS time series in [CATS](#)
- V5** a numeric vector containing the unknown points 4981-5000 of the CATS time series in [CATS](#)

## Details

Contains the 100 unknown observations which were to be predicted of the CATS time series in ([CATS](#)) as demanded by the CATS Competition.

## Source

A. Lendasse, E. Oja, O. Simula, M. Verleysen, and others, 2004, Time Series Prediction Competition: The CATS Benchmark, In: IJCNN'2004-International Joint Conference on Neural Networks

## References

A. Lendasse, E. Oja, O. Simula, and M. Verleysen, 2007, Time series prediction competition: The CATS benchmark, *Neurocomputing*, v. 70, n. 13-15 (Aug.), p. 2325-2329.

## See Also

[CATS](#)

## Examples

```
data(CATS.cont)
str(CATS.cont)
plot(ts(CATS.cont["V5"]))
```

---

detrend	<i>Detrending Transformation</i>
---------	----------------------------------

---

## Description

The `detrend()` function performs a detrending transformation and removes a trend from the provided time series. `detrend.rev()` reverses the transformation.

## Usage

```
detrend(x, trend)
```

## Arguments

<code>x</code>	A numeric vector or univariate time series of class <code>ts</code> .
<code>trend</code>	A numeric vector or univariate time series containing the trend to be removed. Generally, the fitted values of a model object.

## Value

A vector of the same length as `x` containing the residuals of `x` after trend removal.

## Author(s)

Rebecca Pontes Salles

## References

R. H. Shumway, D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples*, Springer, New York, NY, 4 edition, 2017.

## See Also

[DIF](#), [BCT](#), [MAS](#), [LT](#), [PCT](#)

**Examples**

```

data(CATS,CATS.cont)
fpoly <- fittestPolyR(CATS[,1],h=20)
trend <- fitted(fpoly$model)

residuals <- detrend(CATS[,1],trend)
x <- detrend.rev(residuals,trend)

```

Diff

*Differencing Transformation***Description**

The `Diff()` function returns a simple or seasonal differencing transformation of the provided time series. `Diff.rev()` reverses the transformation. Wrapper functions for `diff` and `diffinv` of the `stats` package, respectively.

**Usage**

```

Diff(
  x,
  lag = ifelse(type == "simple", 1, stats::frequency(x)),
  differences = NULL,
  type = c("simple", "seasonal"),
  ...
)

Diff.rev(
  x,
  lag = ifelse(type == "simple", 1, stats::frequency(x)),
  differences = 1,
  xi,
  type = c("simple", "seasonal"),
  addinit = TRUE
)

```

**Arguments**

<code>x</code>	A numeric vector or univariate time series containing the values to be differenced.
<code>lag</code>	Integer indicating the lag parameter. Default set to 1 if <code>type = "simple"</code> , or <code>frequency(x)</code> if <code>type = "seasonal"</code> .
<code>differences</code>	Integer representing the order of the difference. If <code>NULL</code> , the order of the difference is automatically selected using <code>ndiffs</code> (if <code>type = "simple"</code> ) or <code>nsdiffs</code> (if <code>type = "seasonal"</code> ) from the <code>forecast</code> package.

type	Character string. Indicates if the function should perform simple or seasonal differencing.
...	Additional arguments passed to <code>ndiffs</code> (if <code>type = "simple"</code> ) or <code>nsdiffs</code> (if <code>type = "seasonal"</code> ) from the forecast package.
xi	Numeric vector or time series containing the initial values for the integrals. If missing, zeros are used.
addinit	If FALSE, the reverse transformed time series does not contain xi. Default set to TRUE.

**Value**

x if differences is automatically selected, and is not set as greater than 0. Same as `diff` otherwise.

**Author(s)**

Rebecca Pontes Salles

**References**

- R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.  
 R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

Other transformation methods: `LogT()`, `WaveletT()`, `emd()`, `mas()`, `mlm_io()`, `outliers_bp()`, `pct()`, `train_test_subset()`

**Examples**

```
data(CATS)
d <- Diff(CATS[,1], differences = 1)
x <- Diff.rev(as.vector(d), differences = attributes(d)$differences, xi = attributes(d)$xi)
all(round(x,4)==round(CATS[,1],4))
```

---

 emd

*Automatic empirical mode decomposition*

---

**Description**

The function automatically applies an empirical mode decomposition to a provided univariate time series. Wrapper function for `emd` of the `Rlibeemd` package. It also allows the automatic selection of meaningful IMFs using `fittestEMD.emd.rev()` reverses the transformation.

**Usage**

```
emd(
  x,
  num_imfs = 0,
  S_number = 4L,
  num_siftings = 50L,
  meaningfulImfs = NULL,
  h = 1,
  ...
)

emd.rev(pred)
```

**Arguments**

**x** A numeric vector or univariate time series to be decomposed.

**num\_imfs** Number of Intrinsic Mode Functions (IMFs) to compute. See [emd](#).

**S\_number, num\_siftings** See [emd](#).

**meaningfulImfs** Vector indicating the indices of the meaningful IMFs according to the possible intervals  $i:num\_imfs$  for  $i=1, \dots, (num\_imfs-1)$ , where `num_imfs` is the number of IMFs in a decomposition. If `meaningfulImfs = NULL` (default), the function returns all IMF's produced by [emd](#) as meaningful. If `meaningfulImfs = 0` the function automatically selects the meaningful IMFs of a decomposition using [fittestEMD](#).

**h** See [fittestEMD](#). Passed to [fittestEMD](#) if `meaningfulImfs = 0`.

**...** Additional arguments passed to [fittestEMD](#).

**pred** A list containing IMFs produced by empirical mode decomposition.

**Value**

A list containing the meaningful IMFs of the empirical mode decomposition of `x`. A vector indicating the indices of the meaningful IMFs and the number of IMFs produced are passed as attributes named "meaningfulImfs" and "num\_imfs", respectively.

**Author(s)**

Rebecca Pontes Salles

**References**

Kim, D., Paek, S. H., & Oh, H. S. (2008). A Hilbert-Huang transform approach for predicting cyber-attacks. *Journal of the Korean Statistical Society*, 37(3), 277-283.

**See Also**

[fittestEMD](#), [fittestWavelet](#)

Other transformation methods: [Diff\(\)](#), [LogT\(\)](#), [WaveletT\(\)](#), [mas\(\)](#), [mlm\\_io\(\)](#), [outliers\\_bp\(\)](#), [pct\(\)](#), [train\\_test\\_subset\(\)](#)

**Examples**

```
data(CATS)
e <- emd(CATS[,1])
x <- emd.rev(e)
all(round(x,4)==round(CATS[,1],4))
```

---

EUNITE.Loads

*Electrical loads of the EUNITE Competition*


---

**Description**

The EUNITE Competition main dataset composed of a set of univariate time series of half-an-hour electrical loads measured between 1997 and 1998.

**Usage**

EUNITE.Loads

**Format**

A data frame with 730 observations on the following 48 variables.

**X00.30** a numeric vector with loads measured in the period 00:00-00:30 of 1997-1998.

**X01.00** a numeric vector with loads measured in the period 00:30-01:00 of 1997-1998.

**X01.30** a numeric vector with loads measured in the period 01:00-01:30 of 1997-1998.

**X02.00** a numeric vector with loads measured in the period 01:30-02:00 of 1997-1998.

**X02.30** a numeric vector with loads measured in the period 02:00-02:30 of 1997-1998.

**X03.00** a numeric vector with loads measured in the period 02:30-03:00 of 1997-1998.

**X03.30** a numeric vector with loads measured in the period 03:00-03:30 of 1997-1998.

**X04.00** a numeric vector with loads measured in the period 03:30-04:00 of 1997-1998.

**X04.30** a numeric vector with loads measured in the period 04:00-04:30 of 1997-1998.

**X05.00** a numeric vector with loads measured in the period 04:30-05:00 of 1997-1998.

**X05.30** a numeric vector with loads measured in the period 05:00-05:30 of 1997-1998.

**X06.00** a numeric vector with loads measured in the period 05:30-06:00 of 1997-1998.

**X06.30** a numeric vector with loads measured in the period 06:00-06:30 of 1997-1998.

- X07.00** a numeric vector with loads measured in the period 06:30-07:00 of 1997-1998.
- X07.30** a numeric vector with loads measured in the period 07:00-07:30 of 1997-1998.
- X08.00** a numeric vector with loads measured in the period 07:30-08:00 of 1997-1998.
- X08.30** a numeric vector with loads measured in the period 08:00-08:30 of 1997-1998.
- X09.00** a numeric vector with loads measured in the period 08:30-09:00 of 1997-1998.
- X09.30** a numeric vector with loads measured in the period 09:00-09:30 of 1997-1998.
- X10.00** a numeric vector with loads measured in the period 09:30-10:00 of 1997-1998.
- X10.30** a numeric vector with loads measured in the period 10:00-10:30 of 1997-1998.
- X11.00** a numeric vector with loads measured in the period 10:30-11:00 of 1997-1998.
- X11.30** a numeric vector with loads measured in the period 11:00-11:30 of 1997-1998.
- X12.00** a numeric vector with loads measured in the period 11:30-12:00 of 1997-1998.
- X12.30** a numeric vector with loads measured in the period 12:00-12:30 of 1997-1998.
- X13.00** a numeric vector with loads measured in the period 12:30-13:00 of 1997-1998.
- X13.30** a numeric vector with loads measured in the period 13:00-13:30 of 1997-1998.
- X14.00** a numeric vector with loads measured in the period 13:30-14:00 of 1997-1998.
- X14.30** a numeric vector with loads measured in the period 14:00-14:30 of 1997-1998.
- X15.00** a numeric vector with loads measured in the period 14:30-15:00 of 1997-1998.
- X15.30** a numeric vector with loads measured in the period 15:00-15:30 of 1997-1998.
- X16.00** a numeric vector with loads measured in the period 15:30-16:00 of 1997-1998.
- X16.30** a numeric vector with loads measured in the period 16:00-16:30 of 1997-1998.
- X17.00** a numeric vector with loads measured in the period 16:30-17:00 of 1997-1998.
- X17.30** a numeric vector with loads measured in the period 17:00-17:30 of 1997-1998.
- X18.00** a numeric vector with loads measured in the period 17:30-18:00 of 1997-1998.
- X18.30** a numeric vector with loads measured in the period 18:00-18:30 of 1997-1998.
- X19.00** a numeric vector with loads measured in the period 18:30-19:00 of 1997-1998.
- X19.30** a numeric vector with loads measured in the period 19:00-19:30 of 1997-1998.
- X20.00** a numeric vector with loads measured in the period 19:30-20:00 of 1997-1998.
- X20.30** a numeric vector with loads measured in the period 20:00-20:30 of 1997-1998.
- X21.00** a numeric vector with loads measured in the period 20:30-21:00 of 1997-1998.
- X21.30** a numeric vector with loads measured in the period 21:00-21:30 of 1997-1998.
- X22.00** a numeric vector with loads measured in the period 21:30-22:00 of 1997-1998.
- X22.30** a numeric vector with loads measured in the period 22:00-22:30 of 1997-1998.
- X23.00** a numeric vector with loads measured in the period 22:30-23:00 of 1997-1998.
- X23.30** a numeric vector with loads measured in the period 23:00-23:30 of 1997-1998.
- X24.00** a numeric vector with loads measured in the period 23:30-24:00 of 1997-1998.

## Details

The EUNITE Competition proposed the prediction of maximum daily electrical loads based on half-an-hour loads and average daily temperatures of 1997-1998 ([EUNITE.Temp](#)). The holidays with respect to this period were also provided ([EUNITE.Reg](#)) and the use of data on average daily temperatures of 1995-1996 was allowed. The dataset present considerable seasonality due to properties of electrical load demand, climate influence and holiday effects, among other reasons. Competitors were asked to predict the 31 values corresponding to the daily maximum electrical loads of January 1999 ([EUNITE.Loads.cont](#)). The performance evaluation done by the EUNITE Competition was based on the MAPE error and on the MAXIMAL error of prediction found by the competitors.

## Source

EUNITE 1999, Electricity Load Forecast using Intelligent Adaptive Technology: The EUNITE Network Competition. URL: [http://www.eunite.org/knowledge/Competitions/1st\\_competition/1st\\_competition.htm](http://www.eunite.org/knowledge/Competitions/1st_competition/1st_competition.htm).

## References

B.-J. Chen, M.-W. Chang, and C.-J. Lin, 2004, Load forecasting using support vector Machines: a study on EUNITE competition 2001, IEEE Transactions on Power Systems, v. 19, n. 4 (Nov.), p. 1821-1830.

## See Also

[EUNITE.Loads.cont](#), [EUNITE.Reg](#), [EUNITE.Temp](#)

## Examples

```
data(EUNITE.Loads)
str(EUNITE.Loads)
plot(ts(EUNITE.Loads["X24.00"]))
```

---

EUNITE.Loads.cont	<i>Continuation dataset of the electrical loads of the EUNITE Competition</i>
-------------------	---

---

## Description

A dataset of univariate time series providing 31 points beyond the end of the time series in [EUNITE.Loads](#) containing half-an-hour electrical loads measured in January 1999.

## Usage

EUNITE.Loads.cont



**Format**

A data frame with 31 observations on the following 48 variables.

- X00.30** a numeric vector containing further observations of X00.30 in [EUNITE.Loads](#) relative to January 1999.
- X01.00** a numeric vector containing further observations of X01.00 in [EUNITE.Loads](#) relative to January 1999.
- X01.30** a numeric vector containing further observations of X01.30 in [EUNITE.Loads](#) relative to January 1999.
- X02.00** a numeric vector containing further observations of X02.00 in [EUNITE.Loads](#) relative to January 1999.
- X02.30** a numeric vector containing further observations of X02.30 in [EUNITE.Loads](#) relative to January 1999.
- X03.00** a numeric vector containing further observations of X03.00 in [EUNITE.Loads](#) relative to January 1999.
- X03.30** a numeric vector containing further observations of X03.30 in [EUNITE.Loads](#) relative to January 1999.
- X04.00** a numeric vector containing further observations of X04.00 in [EUNITE.Loads](#) relative to January 1999.
- X04.30** a numeric vector containing further observations of X04.30 in [EUNITE.Loads](#) relative to January 1999.
- X05.00** a numeric vector containing further observations of X05.00 in [EUNITE.Loads](#) relative to January 1999.
- X05.30** a numeric vector containing further observations of X05.30 in [EUNITE.Loads](#) relative to January 1999.
- X06.00** a numeric vector containing further observations of X06.00 in [EUNITE.Loads](#) relative to January 1999.
- X06.30** a numeric vector containing further observations of X06.30 in [EUNITE.Loads](#) relative to January 1999.
- X07.00** a numeric vector containing further observations of X07.00 in [EUNITE.Loads](#) relative to January 1999.
- X07.30** a numeric vector containing further observations of X07.30 in [EUNITE.Loads](#) relative to January 1999.
- X08.00** a numeric vector containing further observations of X08.00 in [EUNITE.Loads](#) relative to January 1999.
- X08.30** a numeric vector containing further observations of X08.30 in [EUNITE.Loads](#) relative to January 1999.
- X09.00** a numeric vector containing further observations of X09.00 in [EUNITE.Loads](#) relative to January 1999.
- X09.30** a numeric vector containing further observations of X09.30 in [EUNITE.Loads](#) relative to January 1999.
- X10.00** a numeric vector containing further observations of X10.00 in [EUNITE.Loads](#) relative to January 1999.

- X10.30** a numeric vector containing further observations of X10.30 in [EUNITE.Loads](#) relative to January 1999.
- X11.00** a numeric vector containing further observations of X11.00 in [EUNITE.Loads](#) relative to January 1999.
- X11.30** a numeric vector containing further observations of X11.30 in [EUNITE.Loads](#) relative to January 1999.
- X12.00** a numeric vector containing further observations of X12.00 in [EUNITE.Loads](#) relative to January 1999.
- X12.30** a numeric vector containing further observations of X12.30 in [EUNITE.Loads](#) relative to January 1999.
- X13.00** a numeric vector containing further observations of X13.00 in [EUNITE.Loads](#) relative to January 1999.
- X13.30** a numeric vector containing further observations of X13.30 in [EUNITE.Loads](#) relative to January 1999.
- X14.00** a numeric vector containing further observations of X14.00 in [EUNITE.Loads](#) relative to January 1999.
- X14.30** a numeric vector containing further observations of X14.30 in [EUNITE.Loads](#) relative to January 1999.
- X15.00** a numeric vector containing further observations of X15.00 in [EUNITE.Loads](#) relative to January 1999.
- X15.30** a numeric vector containing further observations of X15.30 in [EUNITE.Loads](#) relative to January 1999.
- X16.00** a numeric vector containing further observations of X16.00 in [EUNITE.Loads](#) relative to January 1999.
- X16.30** a numeric vector containing further observations of X16.30 in [EUNITE.Loads](#) relative to January 1999.
- X17.00** a numeric vector containing further observations of X17.00 in [EUNITE.Loads](#) relative to January 1999.
- X17.30** a numeric vector containing further observations of X17.30 in [EUNITE.Loads](#) relative to January 1999.
- X18.00** a numeric vector containing further observations of X18.00 in [EUNITE.Loads](#) relative to January 1999.
- X18.30** a numeric vector containing further observations of X18.30 in [EUNITE.Loads](#) relative to January 1999.
- X19.00** a numeric vector containing further observations of X19.00 in [EUNITE.Loads](#) relative to January 1999.
- X19.30** a numeric vector containing further observations of X19.30 in [EUNITE.Loads](#) relative to January 1999.
- X20.00** a numeric vector containing further observations of X20.00 in [EUNITE.Loads](#) relative to January 1999.
- X20.30** a numeric vector containing further observations of X20.30 in [EUNITE.Loads](#) relative to January 1999.

- X21.00** a numeric vector containing further observations of X21.00 in [EUNITE.Loads](#) relative to January 1999.
- X21.30** a numeric vector containing further observations of X21.30 in [EUNITE.Loads](#) relative to January 1999.
- X22.00** a numeric vector containing further observations of X22.00 in [EUNITE.Loads](#) relative to January 1999.
- X22.30** a numeric vector containing further observations of X22.30 in [EUNITE.Loads](#) relative to January 1999.
- X23.00** a numeric vector containing further observations of X23.00 in [EUNITE.Loads](#) relative to January 1999.
- X23.30** a numeric vector containing further observations of X23.30 in [EUNITE.Loads](#) relative to January 1999.
- X24.00** a numeric vector containing further observations of X24.00 in [EUNITE.Loads](#) relative to January 1999.

### Details

Contains the 31 values corresponding to the daily maximum electrical loads of January 1999 which were to be predicted of [EUNITE.Loads](#) as demanded by the EUNITE Competition.

### Source

EUNITE 1999, Electricity Load Forecast using Intelligent Adaptive Technology: The EUNITE Network Competition. URL: [http://www.eunite.org/knowledge/Competitions/1st\\_competition/1st\\_competition.htm](http://www.eunite.org/knowledge/Competitions/1st_competition/1st_competition.htm).

### References

B.-J. Chen, M.-W. Chang, and C.-J. Lin, 2004, Load forecasting using support vector Machines: a study on EUNITE competition 2001, IEEE Transactions on Power Systems, v. 19, n. 4 (Nov.), p. 1821-1830.

### See Also

[EUNITE.Loads](#), [EUNITE.Reg](#), [EUNITE.Temp](#)

### Examples

```
data(EUNITE.Loads.cont)
str(EUNITE.Loads.cont)
plot(ts(EUNITE.Loads.cont["X24.00"]))
```

---

`EUNITE.Reg`*Electrical loads regressors of the EUNITE Competition*

---

**Description**

The EUNITE Competition dataset containing a set of variables serving as regressors for the electrical loads measured between 1997 and 1998 in [EUNITE.Loads](#).

**Usage**`EUNITE.Reg`**Format**

A data frame with 730 observations on the following 2 variables.

**Holiday** a numeric vector containing daily data on the holidays for the time period 1997-1998. Composed of binary values where 1 represents a holiday and 0 a common day.

**Weekday** a numeric vector containing daily data on the weekdays for the time period 1997-1998. Composed of integer values where 1 represents a Sunday, 2 a Monday, 3 a Tuesday, 4 a Wednesday, 5 a Thursday, 6 a Friday and 7 a Saturday.

**Details**

The EUNITE Competition proposed the prediction of maximum daily electrical loads based on half-an-hour loads ([EUNITE.Loads](#)) and average daily temperatures of 1997-1998 ([EUNITE.Temp](#)). Competitors were asked to predict the 31 values corresponding to the daily maximum electrical loads of January 1999 ([EUNITE.Loads.cont](#)). For the posed prediction problem, it is useful to consider as regressors the holidays and the weekdays with respect to this period in [EUNITE.Reg](#), which are expected to have a considerable impact on the electrical consumption.

**Source**

EUNITE 1999, Electricity Load Forecast using Intelligent Adaptive Technology: The EUNITE Network Competition. URL: [http://www.eunite.org/knowledge/Competitions/1st\\_competition/1st\\_competition.htm](http://www.eunite.org/knowledge/Competitions/1st_competition/1st_competition.htm).

**References**

B.-J. Chen, M.-W. Chang, and C.-J. Lin, 2004, Load forecasting using support vector Machines: a study on EUNITE competition 2001, IEEE Transactions on Power Systems, v. 19, n. 4 (Nov.), p. 1821-1830.

**See Also**

[EUNITE.Reg.cont](#), [EUNITE.Loads](#), [EUNITE.Temp](#)

## Examples

```
data(EUNITE.Reg)
str(EUNITE.Reg)
```

---

EUNITE.Reg.cont	<i>Continuation dataset of the electrical loads regressors of the EUNITE Competition</i>
-----------------	--

---

## Description

A dataset of regressor variables for electrical loads measured in January 1999, providing 31 points beyond the end of the data in [EUNITE.Reg](#).

## Usage

```
EUNITE.Reg.cont
```

## Format

A data frame with 31 observations on the following 2 variables.

**Holiday** a numeric vector containing further data of the variable Holiday in [EUNITE.Reg](#) relative to January 1999.

**Weekday** a numeric vector containing further data of the variable Weekday in [EUNITE.Reg](#) relative to January 1999.

## Details

Contains the 31 values of the regressors used for the prediction of the daily maximum electrical loads of January 1999 from [EUNITE.Loads](#) as demanded by the EUNITE Competition.

## Source

EUNITE 1999, Electricity Load Forecast using Intelligent Adaptive Technology: The EUNITE Network Competition. URL: [http://www.eunite.org/knowledge/Competitions/1st\\_competition/1st\\_competition.htm](http://www.eunite.org/knowledge/Competitions/1st_competition/1st_competition.htm).

## References

B.-J. Chen, M.-W. Chang, and C.-J. Lin, 2004, Load forecasting using support vector Machines: a study on EUNITE competition 2001, IEEE Transactions on Power Systems, v. 19, n. 4 (Nov.), p. 1821-1830.

## See Also

[EUNITE.Reg](#), [EUNITE.Loads](#), [EUNITE.Temp](#)

## Examples

```
data(EUNITE.Reg.cont)
str(EUNITE.Reg.cont)
```

---

EUNITE.Temp

*Temperatures of the EUNITE Competition*

---

## Description

The EUNITE Competition dataset composed of a univariate time series of average daily temperatures measured between 1995 and 1998.

## Usage

EUNITE.Temp

## Format

A data frame with 1461 observations on the following variable.

**Temperature** a numeric vector with average daily temperatures measured in the period 1995-1998.

## Details

The EUNITE Competition proposed the prediction of maximum daily electrical loads based on half-an-hour loads ([EUNITE.Loads](#)) and average daily temperatures of 1997-1998, where the latter is used as a regressor. Competitors were asked to predict the 31 values corresponding to the daily maximum electrical loads of January 1999 ([EUNITE.Loads.cont](#)). For the posed prediction problem, the average daily temperatures of January 1999 must also be predicted and for that, the use of data on average daily temperatures of 1995-1996 was allowed.

## Source

EUNITE 1999, Electricity Load Forecast using Intelligent Adaptive Technology: The EUNITE Network Competition. URL: [http://www.eunite.org/knowledge/Competitions/1st\\_competition/1st\\_competition.htm](http://www.eunite.org/knowledge/Competitions/1st_competition/1st_competition.htm).

## References

B.-J. Chen, M.-W. Chang, and C.-J. Lin, 2004, Load forecasting using support vector Machines: a study on EUNITE competition 2001, IEEE Transactions on Power Systems, v. 19, n. 4 (Nov.), p. 1821-1830.

## See Also

[EUNITE.Temp.cont](#), [EUNITE.Loads](#), [EUNITE.Reg](#)

## Examples

```
data(EUNITE.Temp)
str(EUNITE.Temp)
plot(ts(EUNITE.Temp))
```

---

EUNITE.Temp.cont	<i>Continuation dataset of the temperatures of the EUNITE Competition</i>
------------------	---

---

## Description

A dataset with a univariate time series providing 31 points beyond the end of the time series in [EUNITE.Temp](#) containing average daily temperatures measured in January 1999.

## Usage

```
EUNITE.Temp.cont
```

## Format

A data frame with 31 observations on the following variable.

**Temperature** a numeric vector containing further observations of Temperature in [EUNITE.Temp](#) relative to January 1999.

## Details

Contains the 31 values corresponding to the average daily temperatures of January 1999 which were to be predicted of [EUNITE.Temp](#) as demanded by the EUNITE Competition.

## Source

EUNITE 1999, Electricity Load Forecast using Intelligent Adaptive Technology: The EUNITE Network Competition. URL: [http://www.eunite.org/knowledge/Competitions/1st\\_competition/1st\\_competition.htm](http://www.eunite.org/knowledge/Competitions/1st_competition/1st_competition.htm).

## References

B.-J. Chen, M.-W. Chang, and C.-J. Lin, 2004, Load forecasting using support vector Machines: a study on EUNITE competition 2001, IEEE Transactions on Power Systems, v. 19, n. 4 (Nov.), p. 1821-1830.

## See Also

[EUNITE.Temp](#), [EUNITE.Loads](#), [EUNITE.Reg](#)

**Examples**

```
data(EUNITE.Temp.cont)
str(EUNITE.Temp.cont)
plot(ts(EUNITE.Temp.cont))
```

---

 evaluate

*Evaluating prediction/modeling quality*


---

**Description**

evaluate is a generic function for evaluating the quality of time series prediction or modeling fitness based on a particular metric defined in an [evaluating](#) object. The function invokes particular *methods* which depend on the class of the first argument.

**Usage**

```
evaluate(obj, ...)

## S3 method for class 'evaluating'
evaluate(obj, test, pred, ...)

## S3 method for class 'fitness'
evaluate(obj, mdl, test = NULL, pred = NULL, ...)

## S3 method for class 'error'
evaluate(obj, mdl = NULL, test = NULL, pred = NULL, ..., fitness = FALSE)
```

**Arguments**

obj	An object of class <a href="#">evaluating</a> defining a particular metric.
...	Other parameters passed to eval_func of obj.
test	A vector or univariate time series containing actual values for a time series that are to be compared against pred.
pred	A vector or univariate time series containing time series predictions that are to be compared against the values in test.
mdl	A time series model object for which fitness is to be evaluated.
fitness	Should the function compute the fitness quality? If TRUE the function uses mdl to compute fitness error, otherwise, it uses test and pred to compute prediction error.  For evaluate.fitness, test and pred are ignored and can be set to NULL. For evaluate.error, mdl is ignored if fitness is FALSE, otherwise, test and pred are ignored and can be set to NULL.



**Value**

A list containing obj and the computed metric values.

**Author(s)**

Rebecca Pontes Salles

**See Also**

Other evaluate: [evaluate.tspred\(\)](#)

**Examples**

```
data(CATS,CATS.cont)
mdl <- forecast::auto.arima(CATS[,1])
pred <- forecast::forecast(mdl, h=length(CATS.cont[,1]))

evaluate(MSE_eval(), test=CATS.cont[,1], pred=pred$mean)
evaluate(MSE_eval(), mdl, fitness=TRUE)
evaluate(AIC_eval(), mdl)
```

---

evaluate.tspred

*Evaluate method for [tspred](#) objects*

---

**Description**

Evaluates the modeling fitness and quality of time series prediction of the trained models and predicted time series data contained in a [tspred](#) class object, respectively, based on a particular metric. Each metric is defined by an [evaluating](#) object in the list contained in the [tspred](#) class object.

**Usage**

```
## S3 method for class 'tspred'
evaluate(obj, fitness = TRUE, ...)
```

**Arguments**

obj	An object of class <a href="#">tspred</a> defining a particular time series prediction process.
fitness	Should the function compute fitness quality metrics?
...	Other parameters passed to the method <a href="#">evaluate</a> of the <a href="#">evaluating</a> objects from obj.

**Details**

The function [evaluate.tspred](#) calls the method [evaluate](#) on each [evaluating](#) object contained in obj. It uses each trained model, the testing set and the time series predictions contained in obj to compute the metrics. Finally, the produced quality metrics are introduced in the structure of the [tspred](#) class object in obj.

**Value**

An object of class `tspred` with updated structure containing computed quality metric values.

**Author(s)**

Rebecca Pontes Salles

**See Also**

[`tspred()`] for defining a particular time series prediction process, and [`MSE_eval()`] for defining a time series prediction/modeling quality metric.

Other evaluate: [evaluate\(\)](#)

**Examples**

```
data(CATS)

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class
mod1 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod1,
                  evaluating=list(MSE=eval1)
)
summary(tspred_1)

tspred_1 <- subset(tspred_1, data=CATS[3])
tspred_1 <- preprocess(tspred_1, prep_test=FALSE)
tspred_1 <- train(tspred_1)
tspred_1 <- predict(tspred_1, onestep=TRUE)
tspred_1 <- postprocess(tspred_1)
tspred_1 <- evaluate(tspred_1)
```

evaluating

*Prediction/modeling quality evaluation***Description**

Constructor for the `evaluating` class representing a time series prediction or modeling fitness quality evaluation based on a particular metric. The `evaluating` class has two specialized subclasses `fitness` and `error` regarding fitness criteria and prediction/modeling error metrics, respectively.

**Usage**

```
evaluating(eval_func, eval_par = NULL, ..., subclass = NULL)
```

```
fitness(eval_func, eval_par = NULL, ..., subclass = NULL)
```

```
error(eval_func, eval_par = NULL, ..., subclass = NULL)
```

**Arguments**

<code>eval_func</code>	A function for computing a particular metric.
<code>eval_par</code>	List of named parameters required by <code>eval_func</code> .
<code>...</code>	Other parameters to be encapsulated in the class object.
<code>subclass</code>	Name of new specialized subclass object created in case it is provided.

**Value**

An object of class `evaluating`. A list usually containing at least the following elements:

<code>func</code>	A function for computing a particular metric.
<code>par</code>	Particular parameters required by <code>func</code> .

**Author(s)**

Rebecca Pontes Salles

**See Also**

Other constructors: [ARIMA\(\)](#), [LT\(\)](#), [MSE\\_eval\(\)](#), [modeling\(\)](#), [processing\(\)](#), [tspred\(\)](#)

**Examples**

```
e <- error(eval_func=TSPred::NMSE, eval_par=list(train.actual=NULL),
           method="Normalised Mean Squared Error", subclass="NMSE")
summary(e)

f <- fitness(eval_func=stats::AIC, method="Akaike's Information Criterion", subclass="AIC")
summary(f)
```

**Description**

The function predicts and returns the next  $n$  consecutive values of a univariate time series using an automatically best fitted ARIMA model. It also evaluates the fitness of the produced model, using AICc, AIC, BIC and logLik criteria, and its prediction accuracy, using the MSE, NMSE, MAPE, sMAPE and maximal error accuracy measures.

**Usage**

```
fittestArima(
  timeseries,
  timeseries.test = NULL,
  h = NULL,
  na.action = stats::na.omit,
  level = c(80, 95),
  ...
)
```

**Arguments**

<code>timeseries</code>	A vector or univariate time series which contains the values used for fitting an ARIMA model.
<code>timeseries.test</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if NULL.
<code>h</code>	Number of consecutive values of the time series to be predicted. If <code>h</code> is NULL, the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.test</code> . Required when <code>timeseries.test</code> is NULL.
<code>na.action</code>	A function for treating missing values in <code>timeseries</code> and <code>timeseries.test</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.test</code> .
<code>level</code>	Confidence level for prediction intervals.
<code>...</code>	Additional arguments passed to the <code>auto.arima</code> modelling function.

**Details**

The ARIMA model is automatically fitted by the `auto.arima` function and it is used for prediction by the `forecast` function both in the forecast package.

The fitness criteria AICc, AIC (AIC), BIC (BIC) and log-likelihood (logLik) are extracted from the fitted ARIMA model. Also, the prediction accuracy of the model is computed by means of MSE (MSE), NMSE (NMSE), MAPE (MAPE), sMAPE (sMAPE) and maximal error (MAXError) measures.

**Value**

A list with components:

model	A list of class "ARIMA" containing the best fitted ARIMA model. See the <a href="#">auto.arima</a> function in the forecast package.
parameters	A list containing the parameters of the best fitted ARIMA model. See the <a href="#">arimaparameters</a> function.
AICc	Numeric value of the computed AICc criterion of the fitted model.
AIC	Numeric value of the computed AIC criterion of the fitted model.
BIC	Numeric value of the computed BIC criterion of the fitted model.
logLik	Numeric value of the computed log-likelihood of the fitted model.
pred	A list with the components mean, lower and upper, containing the predictions and the lower and upper limits for prediction intervals, respectively. All components are time series. See the <a href="#">forecast</a> function in the forecast package.
MSE	Numeric value of the resulting MSE error of prediction.
NMSE	Numeric value of the resulting NMSE error of prediction.
MAPE	Numeric value of the resulting MAPE error of prediction.
sMAPE	Numeric value of the resulting sMAPE error of prediction.
MaxError	Numeric value of the maximal error of prediction.

**Author(s)**

Rebecca Pontes Salles

**References**

- R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.
- R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

[fittestArimaKF](#), [fittestLM](#), [marimapred](#)

**Examples**

```
data(CATS,CATS.cont)
fArima <- fittestArima(CATS[,1],CATS.cont[,1])
#predicted values
pred <- fArima$pred$mean
#model information
cbind(AICc=fArima$AICc, AIC=fArima$AIC, BIC=fArima$BIC,
      logLik=fArima$logLik, MSE=fArima$MSE, NMSE=fArima$NMSE,
      MAPE=fArima$MAPE, sMAPE=fArima$sMAPE, MaxError=fArima$MaxError)
```

```

#plotting the time series data
plot(c(CATS[,1],CATS.cont[,1]),type='o',lwd=2,xlim=c(960,1000),ylim=c(0,200),
      xlab="Time",ylab="ARIMA")
#plotting the predicted values
lines(ts(pred,start=981),lwd=2,col='blue')
#plotting prediction intervals
lines(ts(fArima$pred$upper[,2],start=981),lwd=2,col='light blue')
lines(ts(fArima$pred$lower[,2],start=981),lwd=2,col='light blue')

```

---

fittestArimaKF

*Automatic ARIMA fitting and prediction with Kalman filter*


---

## Description

The function predicts and returns the next  $n$  consecutive values of a univariate time series using the best evaluated ARIMA model automatically fitted with Kalman filter. It also evaluates the fitness of the produced model, using AICc, AIC, BIC and logLik criteria, and its prediction accuracy, using the MSE, NMSE, MAPE, sMAPE and maximal error accuracy measures.

## Usage

```

fittestArimaKF(
  timeseries,
  timeseries.test = NULL,
  h = NULL,
  na.action = stats::na.omit,
  level = 0.9,
  filtered = TRUE,
  initQ = NULL,
  rank.by = c("MSE", "NMSE", "MAPE", "sMAPE", "MaxError", "AIC", "AICc", "BIC",
             "logLik", "errors", "fitness"),
  ...
)

```

## Arguments

<code>timeseries</code>	A vector or univariate time series which contains the values used for fitting an ARIMA model with Kalman filter.
<code>timeseries.test</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if NULL.
<code>h</code>	Number of consecutive values of the time series to be predicted. If <code>h</code> is NULL, the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.test</code> . Required when <code>timeseries.test</code> is NULL.

<code>na.action</code>	A function for treating missing values in <code>timeseries</code> and <code>timeseries.test</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.test</code> .
<code>level</code>	Confidence level for prediction intervals. See the <code>predict.SSModel</code> function in the KFAS package.
<code>filtered</code>	If <code>filtered</code> is TRUE, Kalman filtered time series observations are used for prediction, otherwise, Kalman smoothed observations are used for prediction.
<code>initQ</code>	Numeric argument regarding the initial value for the covariance of disturbances parameter to be optimized over. The initial value to be optimized is set to <code>exp(initQ)</code> . See the <code>Q</code> argument of the <code>SSMarma</code> function in the KFAS package and the examples in <code>KFAS</code> . If NULL, <code>initQ</code> is automatically set. See 'Details'.
<code>rank.by</code>	Character string. Criteria used for ranking candidate models generated using different options of values for <code>initQ</code> . Only used if <code>initQ</code> is NULL. Ignored otherwise. See 'Details'.
<code>...</code>	Additional arguments passed to the <code>auto.arima</code> modelling function.

### Details

A best ARIMA model is automatically fitted by the `auto.arima` function in the `forecast` package. The coefficients of this model are then used as initial parameters for optimization of a state space model (`SSModel`) using the Kalman filter and functions of the KFAS package (see `SSMarma` and `artransform`).

If `initQ` is NULL, it is automatically set as either `log(var(timeseries))` or  $\emptyset$ . For that, a set of candidate ARIMA state space models is generated by different initial parameterization of `initQ` during the model optimization process. The value option which generates the best ranked candidate ARIMA model according to the criteria in `rank.by` is selected.

The ranking criteria in `rank.by` may be set as a prediction error measure (such as `MSE`, `NMSE`, `MAPE`, `sMAPE` or `MAXError`), or as a fitness criteria (such as `AIC`, `AICc`, `BIC` or `logLik`). In the former case, the candidate models are used for time series prediction and the error measures are calculated by means of a cross-validation process. In the latter case, the candidate models are fitted and fitness criteria are calculated based on all observations in `timeseries`.

If `rank.by` is set as "errors" or "fitness", the candidate models are ranked by all the mentioned prediction error measures or fitness criteria, respectively. The weight of the ranking criteria is equally distributed. In this case, a `rank.position.sum` criterion is produced for ranking the candidate models. The `rank.position.sum` criterion is calculated as the sum of the rank positions of a model (1 = 1st position = better ranked model, 2 = 2nd position, etc.) on each calculated ranking criteria.

### Value

A list with components:

<code>model</code>	An object of class "SSModel" containing the best evaluated ARIMA model fitted with Kalman Filter.
<code>initQ</code>	The <code>initQ</code> argument provided (or automatically selected) for optimization of the best evaluated ARIMA model fitted with Kalman Filter.

AICc	Numeric value of the computed AICc criterion of the best evaluated model.
AIC	Numeric value of the computed AIC criterion of the best evaluated model.
BIC	Numeric value of the computed BIC criterion of the best evaluated model.
logLik	Numeric value of the computed log-likelihood of the best evaluated model.
pred	A list with the components mean, lower and upper, containing the predictions of the best evaluated model and the lower and upper limits for prediction intervals, respectively. All components are time series. See <a href="#">predict.SSModel</a> .
MSE	Numeric value of the resulting MSE error of prediction. Require <code>timeseries.test</code> .
NMSE	Numeric value of the resulting NMSE error of prediction. Require <code>timeseries.test</code> .
MAPE	Numeric value of the resulting MAPE error of prediction. Require <code>timeseries.test</code> .
sMAPE	Numeric value of the resulting sMAPE error of prediction. Require <code>timeseries.test</code> .
MaxError	Numeric value of the maximal error of prediction. Require <code>timeseries.test</code> .
rank.val	Data.frame with the fitness or prediction accuracy criteria computed for all candidate ARIMA with Kalman filter models ranked by <code>rank.by</code> . It has the attribute <code>"ranked.models"</code> , which is a list of objects of class <code>"SSModel"</code> containing all the candidate ARIMA models fitted with Kalman Filter, also ranked by <code>rank.by</code> . Only provided if <code>initQ</code> was automatically selected.
rank.by	Ranking criteria used for ranking candidate models and producing <code>rank.val</code> .

**Author(s)**

Rebecca Pontes Salles

**References**

- R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.
- R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

[fittestArima](#), [fittestLM](#), [marimapred](#)

**Examples**

```
data(CATS,CATS.cont)
fArimaKF <- fittestArimaKF(CATS[,2],CATS.cont[,2])
#predicted values
pred <- fArimaKF$pred

#extracting Kalman filtered and smoothed time series from the best fitted model
fs <- KFAS::KFS(fArimaKF$model,filtering=c("state","mean"),smoothing=c("state","mean"))
f <- fitted(fs, filtered = TRUE) #Kalman filtered time series
s <- fitted(fs) #Kalman smoothed time series
#plotting the time series data
```



```

plot(c(CATS[,2],CATS.cont[,2]),type='o',lwd=2,xlim=c(960,1000),ylim=c(200,600),
     xlab="Time",ylab="ARIMAKF")
#plotting the Kalman filtered time series
lines(f,col='red',lty=2,lwd=2)
#plotting the Kalman smoothed time series
lines(s,col='green',lty=2,lwd=2)
#plotting predicted values
lines(ts(pred$mean,start=981),lwd=2,col='blue')
#plotting prediction intervals
lines(ts(pred$upper,start=981),lwd=2,col='light blue')
lines(ts(pred$lower,start=981),lwd=2,col='light blue')

```

---

fittestEMD

*Automatic prediction with empirical mode decomposition*


---

## Description

The function automatically applies an empirical mode decomposition to a provided univariate time series. The resulting components of the decomposed series are used as base for predicting and returning the next  $n$  consecutive values of the provided univariate time series using also automatically fitted models. It also evaluates fitness and prediction accuracy of the produced models.

## Usage

```

fittestEMD(
  timeseries,
  timeseries.test = NULL,
  h = NULL,
  num_imfs = 0,
  S_number = 4L,
  num_siftings = 50L,
  level = 0.95,
  na.action = stats::na.omit,
  model = c("ets", "arima"),
  rank.by = c("MSE", "NMSE", "MAPE", "sMAPE", "MaxError", "errors")
)

```

## Arguments

<code>timeseries</code>	A vector or univariate time series.
<code>timeseries.test</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if <code>NULL</code> .
<code>h</code>	Number of consecutive values of the time series to be predicted. If <code>h</code> is <code>NULL</code> , the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.test</code> . Required when <code>timeseries.test</code> is <code>NULL</code> .

<code>num_imfs</code>	Number of Intrinsic Mode Functions (IMFs) to compute. See <a href="#">emd</a> .
<code>S_number, num_siftings</code>	See <a href="#">emd</a> .
<code>level</code>	Confidence level for prediction intervals. See <a href="#">predict.lm</a> and <a href="#">predict</a> .
<code>na.action</code>	A function for treating missing values in <code>timeseries</code> and <code>timeseries.test</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.test</code> .
<code>model</code>	Character string. Indicates which model is to be used for fitting and prediction of the components of the decomposed series.
<code>rank.by</code>	Character string. Criteria used for ranking candidate decompositions/models/predictions generated during parameter selection. See 'Details'.

## Details

The function produces an empirical mode decomposition of `timeseries`. See the [emd](#) function. The Intrinsic Mode Functions (IMFs) and residue series resulting from the decomposition are separately used as base for model fitting and prediction. The set of predictions for all IMFs and residue series are then reversed transformed in order to produce the next `h` consecutive values of the provided univariate time series in `timeseries`. See the [emd.rev](#) function.

The function automatically selects the meaningful IMFs of a decomposition. For that, the function produces models for different selections of meaningful IMFs according to the possible intervals  $i: \text{num\_imfs}$  for  $i=1, \dots, (\text{num\_imfs}-1)$ , where `num_imfs` is the number of IMFs in a decomposition. The options of meaningful IMFs of a decomposition which generate the best ranked model fitness/predictions according to the criteria in `rank.by` are selected.

The ranking criteria in `rank.by` may be set as a prediction error measure (such as [MSE](#), [NMSE](#), [MAPE](#), [sMAPE](#) or [MAXError](#)), or as a fitness criteria (such as [AIC](#), [AICc](#), [BIC](#) or [logLik](#)). In the former case, the candidate empirical mode decompositions are used for time series prediction and the error measures are calculated by means of a cross-validation process. In the latter case, the component series of the candidate decompositions are modeled and model fitness criteria are calculated based on all observations in `timeseries`. In particular, the fitness criteria calculated for ranking the candidate decompositions correspond to the models produced for the IMFs.

If `rank.by` is set as "errors" or "fitness", the candidate decompositions are ranked by all the mentioned prediction error measures or fitness criteria, respectively. The weight of the ranking criteria is equally distributed. In this case, a `rank.position.sum` criterion is produced for ranking the candidate decompositions. The `rank.position.sum` criterion is calculated as the sum of the rank positions of a decomposition (1 = 1st position = better ranked model, 2 = 2nd position, etc.) on each calculated ranking criteria.

## Value

A list with components:

<code>emd</code>	Same as <a href="#">emd</a> . Contains the empirical mode decomposition of <code>timeseries</code> .
<code>meaningfulImfs</code>	Character string indicating the automatically selected meaningful IMFs of the decomposition.

pred	A list with the components mean, lower and upper, containing the predictions based on the best evaluated decomposition and the lower and upper limits for prediction intervals, respectively. All components are time series.
MSE	Numeric value of the resulting MSE error of prediction. Require <code>timeseries.test</code> .
NMSE	Numeric value of the resulting NMSE error of prediction. Require <code>timeseries.test</code> .
MAPE	Numeric value of the resulting MAPE error of prediction. Require <code>timeseries.test</code> .
sMAPE	Numeric value of the resulting sMAPE error of prediction. Require <code>timeseries.test</code> .
MaxError	Numeric value of the maximal error of prediction. Require <code>timeseries.test</code> .
rank.val	Data.frame with the fitness or prediction accuracy criteria computed based on all candidate decompositions ranked by <code>rank.by</code> .
rank.by	Ranking criteria used for ranking candidate decompositions and producing <code>rank.val</code> .

**Author(s)**

Rebecca Pontes Salles

**References**

Kim, D., Paek, S. H., & Oh, H. S. (2008). A Hilbert-Huang transform approach for predicting cyber-attacks. *Journal of the Korean Statistical Society*, 37(3), 277-283.

**See Also**

[fittestWavelet](#), [fittestMAS](#)

**Examples**

```
data(CATS)
femd <- fittestEMD(CATS[,1],h=20)
```

---

fittestLM

*Automatically finding fittest linear model for prediction*


---

**Description**

The function automatically evaluates and returns the fittest linear model among ARIMA and polynomial regression, with and without Kalman filtering, for prediction of a given univariate time series. Wrapper for the [fittestArima](#), [fittestArimaKF](#), [fittestPolyR](#) and [fittestPolyRKF](#) functions for automatic time series prediction, whose results are also returned.

**Usage**

```
fittestLM(
  timeseries,
  timeseries.test = NULL,
  h = NULL,
  level = 0.95,
  na.action = stats::na.omit,
  filtered = TRUE,
  order = NULL,
  minorder = 0,
  maxorder = 5,
  raw = FALSE,
  initQ = NULL,
  rank.by = c("MSE", "NMSE", "MAPE", "sMAPE", "MaxError", "AIC", "AICc", "BIC",
    "logLik", "errors", "fitness"),
  ...
)
```

**Arguments**

<code>timeseries</code>	A vector or univariate time series which contains the values used for fitting the models.
<code>timeseries.test</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if <code>NULL</code> .
<code>h</code>	Number of consecutive values of the time series to be predicted. If <code>h</code> is <code>NULL</code> , the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.test</code> . Required when <code>timeseries.test</code> is <code>NULL</code> .
<code>level</code>	Confidence level for prediction intervals.
<code>na.action</code>	A function for treating missing values in <code>timeseries</code> and <code>timeseries.test</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.test</code> .
<code>filtered</code>	See <a href="#">fittestArimaKF</a> and <a href="#">fittestPolyRKF</a> .
<code>order</code>	See <a href="#">fittestPolyR</a> and <a href="#">fittestPolyRKF</a> .
<code>minorder</code>	See <a href="#">fittestPolyR</a> and <a href="#">fittestPolyRKF</a> .
<code>maxorder</code>	See <a href="#">fittestPolyR</a> and <a href="#">fittestPolyRKF</a> .
<code>raw</code>	See <a href="#">fittestPolyR</a> .
<code>initQ</code>	See <a href="#">fittestArimaKF</a> and <a href="#">fittestPolyRKF</a> .
<code>rank.by</code>	Character string. Criteria used for ranking candidate models. See 'Details'.
<code>...</code>	See <a href="#">fittestArima</a> and <a href="#">fittestArimaKF</a> .

## Details

The results of the best evaluated models returned by `fittestArima`, `fittestArimaKF`, `fittestPolyR` and `fittestPolyRKF` are ranked and the fittest linear model for prediction of the given univariate time series is selected based on the criteria in `rank.by`.

The ranking criteria in `rank.by` may be set as a prediction error measure (such as `MSE`, `NMSE`, `MAPE`, `sMAPE` or `MAXError`), or as a fitness criteria (such as `AIC`, `AICc`, `BIC` or `logLik`). See `fittestArima`, `fittestArimaKF`, `fittestPolyR` or `fittestPolyRKF`.

If `rank.by` is set as "errors" or "fitness", the candidate models are ranked by all the mentioned prediction error measures or fitness criteria, respectively. The weight of the ranking criteria is equally distributed. In this case, a `rank.position.sum` criterion is produced for ranking the candidate models. The `rank.position.sum` criterion is calculated as the sum of the rank positions of a model (1 = 1st position = better ranked model, 2 = 2nd position, etc.) on each calculated ranking criteria.

## Value

A list with components:

<code>model</code>	An object containing the fittest evaluated linear model. The class of the model object is dependent on the results of the evaluation (ranking). See <code>fittestArima</code> , <code>fittestArimaKF</code> , <code>fittestPolyR</code> and <code>fittestPolyRKF</code> .
<code>rank</code>	Data.frame with the fitness and/or prediction accuracy criteria computed for all models considered, ranked by <code>rank.by</code> .
<code>ranked.results</code>	A list of lists containing the ranked results of the functions <code>fittestArima</code> , <code>fittestArimaKF</code> , <code>fittestPolyR</code> and <code>fittestPolyRKF</code> . Also ranked by <code>rank.by</code> .

## Author(s)

Rebecca Pontes Salles

## See Also

`fittestArima`, `fittestArimaKF`, `fittestPolyR`, `fittestPolyRKF`

## Examples

```
data(CATS,CATS.cont)
fittest <- fittestLM(CATS[,1],CATS.cont[,1])

#fittest model information
fittest$rank[1,]

#predictions of the fittest model
fittest$ranked.results[[1]]$pred
```

**Description**

The function uses an automatically produced moving average smoother as base for predicting and returning the next  $n$  consecutive values of the provided univariate time series using an also automatically fitted model ([ets/stlf](#) or [arima](#)). It also evaluates the fitness and prediction accuracy of the produced model.

**Usage**

```
fittestMAS(
  timeseries,
  timeseries.test = NULL,
  h = NULL,
  order = NULL,
  minorder = 1,
  maxorder = min(36, length(ts(na.action(timeseries)))/2),
  model = c("ets", "arima"),
  level = 0.95,
  na.action = stats::na.omit,
  rank.by = c("MSE", "NMSE", "MAPE", "sMAPE", "MaxError", "AIC", "AICc", "BIC",
    "logLik", "errors", "fitness"),
  ...
)
```

**Arguments**

<code>timeseries</code>	A vector or univariate time series.
<code>timeseries.test</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if NULL.
<code>h</code>	Number of consecutive values of the time series to be predicted. If <code>h</code> is NULL, the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.test</code> . Required when <code>timeseries.test</code> is NULL.
<code>order</code>	A numeric integer value corresponding to the order of moving average smoother to be produced. If NULL, the order of the moving average smoother returned by the function is automatically selected within the interval <code>minorder:maxorder</code> . See 'Details'.
<code>minorder</code>	A numeric integer value corresponding to the minimum order of candidate moving average smoothers to be produced and evaluated. Ignored if <code>order</code> is provided. See 'Details'.

maxorder	A numeric integer value corresponding to the maximal order of candidate moving average smoothers to be produced and evaluated. Ignored if order is provided. See 'Details'.
model	Character string. Indicates which model is to be used for fitting and prediction of the moving average smoothed series.
level	Confidence level for prediction intervals. See the <a href="#">forecast</a> function of the forecast package.
na.action	A function for treating missing values in <code>timeseries</code> and <code>timeseries.test</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.test</code> .
rank.by	Character string. Criteria used for ranking candidate models generated. See 'Details'.
...	Additional arguments passed to the modeling functions.

### Details

The function produces a moving average smoother of `timeseries` with order `order` and uses it as base for model fitting and prediction. If `model="arima"`, an arima model is used and automatically fitted using the [auto.arima](#) function. If `model="ets"`, the function fits an `[forecast]ets` model (if `timeseries` is non-seasonal or the seasonal period is 12 or less) or `stlf` model (if the seasonal period is 13 or more).

For producing the prediction of the next `h` consecutive values of the provided univariate time series, the function [mas.rev](#) is used.

If `order` is NULL, it is automatically selected. For that, a set with candidate models constructed for moving average smoothers of orders from `minorder` to `maxorder` is generated. The default value of `maxorder` is set based on code from the `sma` function of `smooth` package. The value option of `order` which generate the best ranked candidate model according to the criteria in `rank.by` is selected.

The ranking criteria in `rank.by` may be set as a prediction error measure (such as [MSE](#), [NMSE](#), [MAPE](#), [sMAPE](#) or [MAXError](#)), or as a fitness criteria (such as [AIC](#), [AICc](#), [BIC](#) or [logLik](#)). In the former case, the candidate models are used for time series prediction and the error measures are calculated by means of a cross-validation process. In the latter case, the candidate models are fitted and fitness criteria are calculated based on all observations in `timeseries`.

If `rank.by` is set as "errors" or "fitness", the candidate models are ranked by all the mentioned prediction error measures or fitness criteria, respectively. The weight of the ranking criteria is equally distributed. In this case, a `rank.position.sum` criterion is produced for ranking the candidate models. The `rank.position.sum` criterion is calculated as the sum of the rank positions of a model (1 = 1st position = better ranked model, 2 = 2nd position, etc.) on each calculated ranking criteria.

### Value

A list with components:

model	A list containing information about the best evaluated model.
order	The order of moving average smoother provided or automatically selected.
ma	The simple moving average smoother of order <code>order</code> of the provided time series.

AICc	Numeric value of the computed AICc criterion of the best evaluated model.
AIC	Numeric value of the computed AIC criterion of the best evaluated model.
BIC	Numeric value of the computed BIC criterion of the best evaluated model.
logLik	Numeric value of the computed log-likelihood of the best evaluated model.
pred	A list with the components mean, lower and upper, containing the predictions of the best evaluated model and the lower and upper limits for prediction intervals, respectively. All components are time series. See the <a href="#">forecast</a> function in the forecast package.
MSE	Numeric value of the resulting MSE error of prediction. Require <code>timeseries.test</code> .
NMSE	Numeric value of the resulting NMSE error of prediction. Require <code>timeseries.test</code> .
MAPE	Numeric value of the resulting MAPE error of prediction. Require <code>timeseries.test</code> .
sMAPE	Numeric value of the resulting sMAPE error of prediction. Require <code>timeseries.test</code> .
MaxError	Numeric value of the maximal error of prediction. Require <code>timeseries.test</code> .
rank.val	Data.frame with the fitness or prediction accuracy criteria computed for all candidate models ranked by <code>rank.by</code> . It has the attribute "ranked.models", which is a list of objects containing all the candidate models, also ranked by <code>rank.by</code> .
rank.by	Ranking criteria used for ranking candidate models and producing <code>rank.val</code> .

**Author(s)**

Rebecca Pontes Salles

**References**

- R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.
- R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

[fittestEMD](#), [fittestWavelet](#)

**Examples**

```
data(CATS)

fMAS <- fittestMAS(CATS[,1],h=20,model="arima")

#automatically selected order of moving average
mas.order <- fMAS$order
```



**Description**

The function predicts and returns the next  $n$  consecutive values of a univariate time series using the best evaluated automatically fitted polynomial regression model. It also evaluates the fitness of the produced model, using AICc, AIC, BIC and logLik criteria, and its prediction accuracy, using the MSE, NMSE, MAPE, sMAPE and maximal error accuracy measures.

**Usage**

```
fittestPolyR(
  timeseries,
  timeseries.test = NULL,
  h = NULL,
  order = NULL,
  minorder = 0,
  maxorder = 5,
  raw = FALSE,
  na.action = stats::na.omit,
  level = 0.95,
  rank.by = c("MSE", "NMSE", "MAPE", "sMAPE", "MaxError", "AIC", "AICc", "BIC",
             "logLik", "errors", "fitness")
)
```

**Arguments**

<code>timeseries</code>	A vector or univariate time series which contains the values used for fitting a polynomial regression model.
<code>timeseries.test</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if NULL.
<code>h</code>	Number of consecutive values of the time series to be predicted. If <code>h</code> is NULL, the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.test</code> . Required when <code>timeseries.test</code> is NULL.
<code>order</code>	A numeric integer value corresponding to the order of polynomial regression to be fitted. If NULL, the order of the polynomial regression returned by the function is automatically selected within the interval <code>minorder:maxorder</code> . See 'Details'.
<code>minorder</code>	A numeric integer value corresponding to the minimum order of candidate polynomial regression to be fitted and evaluated. Ignored if <code>order</code> is provided. See 'Details'.
<code>maxorder</code>	A numeric integer value corresponding to the maximal order of candidate polynomial regression to be fitted and evaluated. Ignored if <code>order</code> is provided. See 'Details'.

<code>raw</code>	If TRUE, use raw and not orthogonal polynomials. Orthogonal polynomials help avoid correlation between variables. Default is FALSE. See <code>poly</code> of the stats package.
<code>na.action</code>	A function for treating missing values in <code>timeseries</code> and <code>timeseries.test</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.test</code> .
<code>level</code>	Confidence level for prediction intervals. See the <code>predict.lm</code> function in the stats package.
<code>rank.by</code>	Character string. Criteria used for ranking candidate models generated. See 'Details'.

### Details

A set with candidate polynomial regression models of order `order` is generated with help from the `dredge` function from the MuMIn package. The candidate models are ranked according to the criteria in `rank.by` and the best ranked model is returned by the function.

If `order` is NULL, it is automatically selected. For that, the candidate polynomial regression models generated receive orders from `minororder` to `maxorder`. The value option of `order` which generate the best ranked candidate polynomial regression model according to the criteria in `rank.by` is selected.

The ranking criteria in `rank.by` may be set as a prediction error measure (such as `MSE`, `NMSE`, `MAPE`, `SMAPE` or `MAXError`), or as a fitness criteria (such as `AIC`, `AICc`, `BIC` or `logLik`). In the former case, the candidate models are used for time series prediction and the error measures are calculated by means of a cross-validation process. In the latter case, the candidate models are fitted and fitness criteria are calculated based on all observations in `timeseries`.

If `rank.by` is set as "errors" or "fitness", the candidate models are ranked by all the mentioned prediction error measures or fitness criteria, respectively. The weight of the ranking criteria is equally distributed. In this case, a `rank.position.sum` criterion is produced for ranking the candidate models. The `rank.position.sum` criterion is calculated as the sum of the rank positions of a model (1 = 1st position = better ranked model, 2 = 2nd position, etc.) on each calculated ranking criteria.

### Value

A list with components:

<code>model</code>	An object of class "stats::lm" containing the best evaluated polynomial regression model.
<code>order</code>	The order argument provided (or automatically selected) for the best evaluated polynomial regression model.
<code>AICc</code>	Numeric value of the computed AICc criterion of the best evaluated model.
<code>AIC</code>	Numeric value of the computed AIC criterion of the best evaluated model.
<code>BIC</code>	Numeric value of the computed BIC criterion of the best evaluated model.
<code>logLik</code>	Numeric value of the computed log-likelihood of the best evaluated model.

pred	A list with the components mean, lower and upper, containing the predictions of the best evaluated model and the lower and upper limits for prediction intervals, respectively. All components are time series. See <a href="#">predict.lm</a> .
MSE	Numeric value of the resulting MSE error of prediction. Require <code>timeseries.test</code> .
NMSE	Numeric value of the resulting NMSE error of prediction. Require <code>timeseries.test</code> .
MAPE	Numeric value of the resulting MAPE error of prediction. Require <code>timeseries.test</code> .
sMAPE	Numeric value of the resulting sMAPE error of prediction. Require <code>timeseries.test</code> .
MaxError	Numeric value of the maximal error of prediction. Require <code>timeseries.test</code> .
rank.val	Data.frame with the coefficients and the fitness or prediction accuracy criteria computed for all candidate polynomial regression models ranked by <code>rank.by</code> . It has the attribute "model.calls", which is a list of objects of class "expression" containing the calls of all the candidate polynomial regression models, also ranked by <code>rank.by</code> .
rank.by	Ranking criteria used for ranking candidate models and producing <code>rank.val</code> .

**Author(s)**

Rebecca Pontes Salles

**References**

- R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.
- R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

[fittestPolyRKF](#), [fittestLM](#)

**Examples**

```
data(CATS, CATS.cont)
fPolyR <- fittestPolyR(CATS[,3], CATS.cont[,3])
#predicted values
pred <- fPolyR$pred

#plotting the time series data
plot(c(CATS[,3], CATS.cont[,3]), type='o', lwd=2, xlim=c(960, 1000), ylim=c(-100, 300),
      xlab="Time", ylab="PR")
#plotting predicted values
lines(ts(pred$mean, start=981), lwd=2, col='blue')
#plotting prediction intervals
lines(ts(pred$lower, start=981), lwd=2, col='light blue')
lines(ts(pred$upper, start=981), lwd=2, col='light blue')
```

---

fittestPolyRKF	<i>Automatic fitting and prediction of polynomial regression with Kalman filter</i>
----------------	---

---

### Description

The function predicts and returns the next  $n$  consecutive values of a univariate time series using the best evaluated polynomial regression model automatically fitted with Kalman filter. It also evaluates the fitness of the produced model, using AICc, AIC, BIC and logLik criteria, and its prediction accuracy, using the MSE, NMSE, MAPE, sMAPE and maximal error accuracy measures.

### Usage

```
fittestPolyRKF(
  timeseries,
  timeseries.test = NULL,
  h = NULL,
  na.action = stats::na.omit,
  level = 0.9,
  order = NULL,
  minorder = 0,
  maxorder = 5,
  initQ = NULL,
  filtered = TRUE,
  rank.by = c("MSE", "NMSE", "MAPE", "sMAPE", "MaxError", "AIC", "AICc", "BIC",
             "logLik", "errors", "fitness")
)
```

### Arguments

timeseries	A vector or univariate time series which contains the values used for fitting a polynomial regression model with Kalman filter. <i>~~Describe timeseries here~~</i>
timeseries.test	A vector or univariate time series containing a continuation for timeseries with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if NULL.
h	Number of consecutive values of the time series to be predicted. If $h$ is NULL, the number of consecutive values to be predicted is assumed to be equal to the length of timeseries.test. Required when timeseries.test is NULL.
na.action	A function for treating missing values in timeseries and timeseries.test. The default function is <code>na.omit</code> , which omits any missing values found in timeseries or timeseries.test.
level	Confidence level for prediction intervals. See the <code>predict.SSModel</code> function in the KFAS package. <i>~~Describe na.action here~~</i>

order	A numeric integer value corresponding to the order of polynomial regression to be fitted. If NULL, the order of the polynomial regression returned by the function is automatically selected within the interval <code>minororder:maxorder</code> . See 'Details'.
minororder	A numeric integer value corresponding to the minimum order of candidate polynomial regression to be fitted and evaluated. Ignored if <code>order</code> is provided. See 'Details'.
maxorder	A numeric integer value corresponding to the maximal order of candidate polynomial regression to be fitted and evaluated. Ignored if <code>order</code> is provided. See 'Details'.
initQ	Numeric argument regarding the initial values for the covariance of disturbances parameter to be optimized over. The initial values to be optimized are set to <code>rep(initQ, (order+1))</code> . See the <code>Q</code> argument of the <code>SSModel</code> function in the <code>KFAS</code> package and the examples in <code>KFAS</code> . If NULL, <code>initQ</code> is automatically set. See 'Details'.
filtered	If <code>filtered</code> is TRUE, Kalman filtered time series observations are used for prediction, otherwise, Kalman smoothed observations are used for prediction.
rank.by	Character string. Criteria used for ranking candidate models generated using different options of values for <code>order</code> and/or <code>initQ</code> . Ignored if both <code>order</code> and <code>initQ</code> are provided. See 'Details'.

## Details

The polynomial regression model produced and returned by the function is generated and represented as state space model (`SSModel`) based on code from the `dlmodeler` package. See `dlmodeler.polynomial`. The model is optimized using the Kalman filter and functions of the `KFAS` package (see `fitSSM`).

If `order` is NULL, it is automatically selected. For that, a set of candidate polynomial regression state space models of orders from `minororder` to `maxorder` is generated and evaluated. Also, if `initQ` is NULL, it is automatically set as either `log(stats::var(timeseries))` or `0`. For that, candidate models receive different initial parameterization of `initQ` during the model optimization process. The value options of `order` and/or `initQ` which generate the best ranked candidate polynomial regression model according to the criteria in `rank.by` are selected.

The ranking criteria in `rank.by` may be set as a prediction error measure (such as `MSE`, `NMSE`, `MAPE`, `sMAPE` or `MAXError`), or as a fitness criteria (such as `AIC`, `AICc`, `BIC` or `logLik`). In the former case, the candidate models are used for time series prediction and the error measures are calculated by means of a cross-validation process. In the latter case, the candidate models are fitted and fitness criteria are calculated based on all observations in `timeseries`.

If `rank.by` is set as "errors" or "fitness", the candidate models are ranked by all the mentioned prediction error measures or fitness criteria, respectively. The weight of the ranking criteria is equally distributed. In this case, a `rank.position.sum` criterion is produced for ranking the candidate models. The `rank.position.sum` criterion is calculated as the sum of the rank positions of a model (1 = 1st position = better ranked model, 2 = 2nd position, etc.) on each calculated ranking criteria.

## Value

A list with components:

<code>model</code>	An object of class "SSModel" containing the best evaluated polynomial regression model fitted with Kalman Filter.
<code>order</code>	The order argument provided (or automatically selected) for the best evaluated polynomial regression model fitted with Kalman Filter.
<code>initQ</code>	The <code>initQ</code> argument provided (or automatically selected) for optimization of the best evaluated polynomial regression model fitted with Kalman Filter.
<code>AICc</code>	Numeric value of the computed AICc criterion of the best evaluated model.
<code>AIC</code>	Numeric value of the computed AIC criterion of the best evaluated model.
<code>BIC</code>	Numeric value of the computed BIC criterion of the best evaluated model.
<code>logLik</code>	Numeric value of the computed log-likelihood of the best evaluated model.
<code>pred</code>	A list with the components <code>mean</code> , <code>lower</code> and <code>upper</code> , containing the predictions of the best evaluated model and the lower and upper limits for prediction intervals, respectively. All components are time series. See <code>predict.SSModel</code> .
<code>MSE</code>	Numeric value of the resulting MSE error of prediction. Require <code>timeseries.test</code> .
<code>NMSE</code>	Numeric value of the resulting NMSE error of prediction. Require <code>timeseries.test</code> .
<code>MAPE</code>	Numeric value of the resulting MAPE error of prediction. Require <code>timeseries.test</code> .
<code>sMAPE</code>	Numeric value of the resulting sMAPE error of prediction. Require <code>timeseries.test</code> .
<code>MaxError</code>	Numeric value of the maximal error of prediction. Require <code>timeseries.test</code> .
<code>rank.val</code>	Data.frame with the fitness or prediction accuracy criteria computed for all candidate polynomial regression with Kalman filter models ranked by <code>rank.by</code> . It has the attribute <code>"ranked.models"</code> , which is a list of objects of class "SS-Model" containing all the candidate polynomial regression models fitted with Kalman Filter, also ranked by <code>rank.by</code> . Only provided if <code>order</code> or <code>initQ</code> were automatically selected.
<code>rank.by</code>	Ranking criteria used for ranking candidate models and producing <code>rank.val</code> .

**Author(s)**

Rebecca Pontes Salles

**References**

- R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.
- R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

[fittestPolyR](#), [fittestLM](#) ~

## Examples

```

data(CATS,CATS.cont)
fPolyRKF <- fittestPolyRKF(CATS[,1],CATS.cont[,1])
#predicted values
pred <- fPolyRKF$pred

#extracting Kalman filtered and smoothed time series from the best fitted model
fs <- KFAS::KFS(fPolyRKF$model,filtering=c("state","mean"),smoothing=c("state","mean"))
f <- fitted(fs, filtered = TRUE) #Kalman filtered time series
s <- fitted(fs) #Kalman smoothed time series
#plotting the time series data
plot(c(CATS[,1],CATS.cont[,1]),type='o',lwd=2,xlim=c(960,1000),ylim=c(0,200),
      xlab="Time",ylab="PRKF")
#plotting the Kalman filtered time series
lines(f,col='red',lty=2,lwd=2)
#plotting the Kalman smoothed time series
lines(s,col='green',lty=2,lwd=2)
#plotting predicted values
lines(ts(pred$mean,start=981),lwd=2,col='blue')
#plotting prediction intervals
lines(ts(pred$lower,start=981),lwd=2,col='light blue')
lines(ts(pred$upper,start=981),lwd=2,col='light blue')

```

---

fittestWavelet

*Automatic prediction with wavelet transform*


---

## Description

The function automatically applies a maximal overlap discrete wavelet transform to a provided univariate time series. The resulting components of the decomposed series are used as base for predicting and returning the next  $n$  consecutive values of the provided univariate time series using also automatically fitted models ([ets](#) or [arima](#)). It also evaluates fitness and prediction accuracy of the produced models.

## Usage

```

fittestWavelet(
  timeseries,
  timeseries.test = NULL,
  h = 1,
  filters = c("haar", "d4", "la8", "b114", "c6"),
  n.levels = NULL,
  maxlevel = NULL,
  boundary = "periodic",

```

```

model = c("ets", "arima"),
conf.level = 0.95,
na.action = stats::na.omit,
rank.by = c("MSE", "NMSE", "MAPE", "sMAPE", "MaxError", "AIC", "AICc", "BIC",
  "logLik", "errors", "fitness"),
...
)

```

## Arguments

<code>timeseries</code>	A vector or univariate time series.
<code>timeseries.test</code>	A vector or univariate time series containing a continuation for <code>timeseries</code> with actual values. It is used as a testing set and base for calculation of prediction error measures. Ignored if NULL.
<code>h</code>	Number of consecutive values of the time series to be predicted. If <code>h</code> is NULL, the number of consecutive values to be predicted is assumed to be equal to the length of <code>timeseries.test</code> . Required when <code>timeseries.test</code> is NULL.
<code>filters</code>	A vector containing character strings indicating which wavelet filter to use in the decomposition. If <code>length(filters)&gt;1</code> , the wavelet transform filter used for generating the return of the function is automatically selected. If NULL, all supported filters are considered for automatic selection. See 'Details'. For more details on all the supported filters and corresponding character strings see <a href="#">wt.filter</a> .
<code>n.levels</code>	An integer specifying the level of the decomposition. If NULL, the level of the wavelet decomposition returned by the function is automatically selected within the interval <code>1:maxlevel</code> . See 'Details'.
<code>maxlevel</code>	A numeric integer value corresponding to the maximal level of candidate wavelet decompositions to be produced and evaluated. If NULL, <code>maxlevel</code> is set as <code>floor(log(((nobs-1)/(L-1))+1)/log(2))</code> , where <code>nobs=length(timeseries)</code> and <code>L</code> is the length of the wavelet and scaling filters. See <a href="#">modwt</a> and <a href="#">wt.filter</a> . Ignored if <code>n.levels</code> is provided. See 'Details'.
<code>boundary</code>	Character string. Indicates which boundary method to use. See <a href="#">modwt</a> .
<code>model</code>	Character string. Indicates which model is to be used for fitting and prediction of the components of the decomposed series.
<code>conf.level</code>	Confidence level for prediction intervals. See the <a href="#">forecast</a> function of the forecast package. ~~Describe <code>na.action</code> here~~
<code>na.action</code>	A function for treating missing values in <code>timeseries</code> and <code>timeseries.test</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> or <code>timeseries.test</code> .
<code>rank.by</code>	Character string. Criteria used for ranking candidate decompositions/models/predictions generated during parameter selection. See 'Details'.
<code>...</code>	Additional arguments passed to the modeling functions. ~~Describe <code>na.action</code> here~~



## Details

The function produces a maximal overlap discrete wavelet transform of `timeseries`. It performs a time series decomposition of level `n.levels` using the wavelet filter `filters`. See the `modwt` function. Each component series resulting from the decomposition (`n.levels` wavelet coefficients series and `n.levels` scaling coefficients series) is separately used as base for model fitting and prediction. If `model="arima"`, arima models are used and automatically fitted using the `auto.arima` function. If `model="ets"`, the function fits `[forecast]ets` models. The set of predictions for all component series are then reversed transformed in order to produce the next `h` consecutive values of the provided univariate time series in `timeseries`. See the `imodwt` function.

If `length(filters)>1` or `filters=NULL`, it is automatically selected. For that, a set of candidate wavelet decompositions with different options of filters is generated and used for model fitting and prediction. Also, if `n.levels` is `NULL`, it is automatically set as a value within the interval `1:maxlevel` (if `maxlevel` is not provided, it is calculated according to the wavelet filter based on code from `modwt`). For that, candidate decompositions are specified with different levels. The options of filter and/or level of decomposition which generate the best ranked model fitness/predictions according to the criteria in `rank.by` are selected.

The ranking criteria in `rank.by` may be set as a prediction error measure (such as `MSE`, `NMSE`, `MAPE`, `SMAPE` or `MAXError`), or as a fitness criteria (such as `AIC`, `AICc`, `BIC` or `logLik`). In the former case, the candidate wavelet decompositions are used for time series prediction and the error measures are calculated by means of a cross-validation process. In the latter case, the component series of the candidate decompositions are modeled and model fitness criteria are calculated based on all observations in `timeseries`. In particular, the fitness criteria calculated for ranking the candidate decomposition correspond to the model produced for the `n.level`th scaling coefficients series as it can be considered the main component of a decomposition of level `n.levels` (Conejo,2005).

If `rank.by` is set as `"errors"` or `"fitness"`, the candidate decompositions are ranked by all the mentioned prediction error measures or fitness criteria, respectively. The weight of the ranking criteria is equally distributed. In this case, a `rank.position.sum` criterion is produced for ranking the candidate decompositions. The `rank.position.sum` criterion is calculated as the sum of the rank positions of a decomposition (`1 = 1st position = better ranked model`, `2 = 2nd position`, etc.) on each calculated ranking criteria.

## Value

A list with components:

<code>WT</code>	An object of class <code>modwt</code> containing the wavelet transformed/decomposed time series.
<code>level</code>	The level of wavelet decomposition provided or automatically selected.
<code>filter</code>	A character string indicating the (provided or automatically selected) wavelet filter used in the decomposition.
<code>AICc</code>	Numeric value of the computed AICc criterion of the fitted model for the <code>level</code> th scaling coefficients series.
<code>AIC</code>	Numeric value of the computed AIC criterion of the fitted model for the <code>level</code> th scaling coefficients series.
<code>BIC</code>	Numeric value of the computed BIC criterion of the fitted model for the <code>level</code> th scaling coefficients series.

logLik	Numeric value of the computed log-likelihood of the fitted model for the levelth scaling coefficients series.
pred	A list with the components mean, lower and upper, containing the predictions based on the best evaluated decomposition and the lower and upper limits for prediction intervals, respectively. All components are time series. See the <a href="#">forecast</a> function in the forecast package.
MSE	Numeric value of the resulting MSE error of prediction. Require timeseries.test.
NMSE	Numeric value of the resulting NMSE error of prediction. Require timeseries.test.
MAPE	Numeric value of the resulting MAPE error of prediction. Require timeseries.test.
sMAPE	Numeric value of the resulting sMAPE error of prediction. Require timeseries.test.
MaxError	Numeric value of the maximal error of prediction. Require timeseries.test.
rank.val	Data.frame with the fitness or prediction accuracy criteria computed based on all candidate decompositions ranked by rank.by. It has the attribute "ranked.wt", which is a list of <a href="#">modwt</a> objects containing all the candidate decompositions, also ranked by rank.by. Only provided if filters or n.levels were automatically selected.
rank.by	Ranking criteria used for ranking candidate decompositions and producing rank.val.

**Author(s)**

Rebecca Pontes Salles

**References**

- A. J. Conejo, M. A. Plazas, R. Espinola, A. B. Molina, Day-ahead electricity price forecasting using the wavelet transform and ARIMA models, IEEE Transactions on Power Systems 20 (2005) 1035-1042.
- T. Joo, S. Kim, Time series forecasting based on wavelet filtering, Expert Systems with Applications 42 (2015) 3868-3874.
- C. Stojescu, I. Railean, S. M. P. Lenca, A. Isar, A wavelet based prediction method for time series. In Proceedings of the 2010 International Conference Stochastic Modeling Techniques and Data Analysis, Chania, Greece (pp. 8-11) (2010).

**See Also**

[fittestEMD](#), [fittestMAS](#) ~

**Examples**

```
data(CATS)

fW <- fittestWavelet(CATS[,1],h=20,model="arima")

#plot wavelet transform/decomposition
plot(fW$WT)
```

ipeadata\_d

*The Ipea Most Requested Dataset (daily)***Description**

The Institute of Applied Economic Research of Brazil (Ipea) (Ipea, 2017) is a public institution of Brazil that provides support to the federal government with regard to public policies: fiscal, social, and economic. Ipea provides public datasets derived from real economic and financial data of the world.

**Usage**

ipeadata\_d

**Format**

The ipeadata\_d dataset contains 12 time series of 901 to 8154 observations. The 12 time series are provided as the following variables of a data frame.

**GM366\_IBVSP366** Stock Index: Sao Paulo Stock Exchange - closed - BM&FBovespa.

**GM366\_ERC366** Exchange rate - R\$ / US\$ - commercial - purchase - mean - R\$ - Bacen Out-ras/SGS.

**GM366\_EREURO366** Euro area - exchange rate - euro / US\$ - mean - Euro - Bacen Out-ras/SGS.

**GM366\_ERPV366** Exchange rate - R\$ / US\$ - parallel - selling - mean - R\$ - Economic value.

**GM366\_ERV366** Exchange rate - R\$ / US\$ - commercial - selling - mean - R\$ - Bacen Out-ras/SGS.

**GM366\_TJOVER366** Interest Rate: Over / Selic - (% p.a.) - Bacen Out-ras/SGS.

**GM366\_TJTR366** Interest rate - TR - (% p.m.) - Bacen Out-ras/SGS.

**SECEX366\_MVTOT366** Imports - weekly mean - US\$ - MDIC/Secex.

**SECEX366\_XVTOT366** Exports - weekly mean - US\$ - MDIC/Secex.

**JPM366\_EMBI366** EMBI + Risco-Brasil - JP Morgan.

**BM366\_TJOVER366** Interest rate - Selic - fixed by Copom - (% p.a.) - Bacen/Boletim/M. Finan..

**GM366\_TJOVERV366** Interest Rate: Over / Selic - Ipea.

**Details**

The ipeadata\_d dataset is provided by Ipea. It comprehends the most requested time series collected in daily rates. The ipeadata\_d dataset comprehend observations of exchange rates (R\$/US\$), exports/imports prices, interest rates, and more, measured from 1962 to September of 2017.

The data had missing data removed by the function `na.omit`.

ipeadata\_d.cont provide 30 points beyond the end of the time series in ipeadata\_d. Intended for use as testing set.

**Source**

Ipea, Ipeadata. Macroeconomic and regional data, Technical Report, <http://www.ipeadata.gov.br>, 2017. The "Most request series" section and filtered by "Frequency" equal to "Daily".

**References**

Ipea, Ipeadata. Macroeconomic and regional data, Technical Report, <http://www.ipeadata.gov.br>, 2017.

**See Also**

[ipeadata\\_m](#) ~

**Examples**

```
data(ipeadata_d)
str(ipeadata_d)
plot(ts(ipeadata_d[1]))
```

---

ipeadata\_m

---

*The Ipea Most Requested Dataset (monthly)*


---

**Description**

The Institute of Applied Economic Research of Brazil (Ipea) (Ipea, 2017) is a public institution of Brazil that provides support to the federal government with regard to public policies: fiscal, social, and economic. Ipea provides public datasets derived from real economic and financial data of the world.

**Usage**

```
ipeadata_m
```

**Format**

The `ipeadata_m` dataset contains 23 time series of 156 to 1019 observations. The 23 time series are provided as the following variables of a data frame.

**BM12\_ERC12** Exchange rate - Brazilian real (R\$) / US dollar (US\$) - purchase - average - R\$ - Bacen / Boletim / BP.

**BM12\_ERV12** Exchange rate - Brazilian real (R\$) / US dollar (US\$) - selling - average - R\$ - Bacen / Boletim / BP.

**IGP12\_IGPDI12** IGP-DI - general price index domestic supply (aug 1994 = 100) - FGV/Conj. Econ. - IGP.

**FUNCEX12\_MDPT12** Imports - prices - index (average 2006 = 100) - Funcex.

**FUNCX12\_XPT12** Exports - prices - index (average 2006 = 100) - Funcex.

**PRECOS12\_INPC12** INPC - national consumer price index (dec 1993 = 100) - IBGE/SNIPC.

**PRECOS12\_INPCBR12** INPC - national consumer price index - growth rate - (% p.m.) - IBGE/SNIPC.

**PRECOS12\_IPCA12** IPCA - extended consumer price index (dec 1993 = 100) - IBGE/SNIPC.

**SEADE12\_TDAGSP12** Unemployment rate - open - RMSP - (%) - Seade/PED.

**SEADE12\_TDOTSP12** Unemployment rate - hidden - RMSP - (%) - Seade/PED.

**SEADE12\_TDOPSP12** Unemployment rate - hidden - precarious - RMSP - (%) - Seade/PED.

**GAC12\_SALMINRE12** Real minimum wage - R\$ - Ipea.

**IGP12\_IGPM12** IGP-M - general price index at market prices (aug 1994 = 100) - FGV/Conj. Econ. - IGP.

**PRECOS12\_IPCAG12** IPCA - extended consumer price index - growth rate - (% p.m.) - IBGE/SNIPC.

**IGP12\_IGPDIG12** IGP-DI - general price index domestic supply - growth rate - (% p.m.) - FGV/Conj. Econ. - IGP.

**IGP12\_IGPMG12** IGP-M - general price index at market prices - growth rate - (% p.m.) - FGV/Conj. Econ. - IGP.

**IGP12\_IGPOGG12** IGP-OG - general price index overall supply - growth rate - (% p.m.) - FGV/Conj. Econ. - IGP.

**PRECOS12\_IPCA15G12** IPCA 15 - extended consumer price index - growth rate - (% p.m.) - IBGE/SNIPC.

**[BM12\_PIB12** GDP - R\$ - Bacen / Boletim / Ativ. Ec..

**MTE12\_SALMIN12** Minimum wage - R\$ - MTE.

**BM12\_TJOVER12** Interest rate - Overnight/Selic - (% p.m.) - Bacen/Boletim/M. Finan..

**SEADE12\_TDTGSP12** Unemployment rate - Sao Paulo - (%) - Seade/PED.

**PMEN12\_TD12** Unemployment rate - reference: 30 days - RMs - (%) - IBGE/PME - obs: PME closed in 2016-mar.

## Details

The `ipeadata_m` dataset is provided by Ipea. It comprehends the most requested time series collected in monthly rates. The `ipeadata_m` dataset comprehend observations of exchange rates (R\$/US\$), exports/imports prices, interest rates, minimum wage, unemployment rate, and more, measured from 1930 to September of 2017.

The data had missing data removed by the function `na.omit`.

`ipeadata_m.cont` provide 12 points beyond the end of the time series in `ipeadata_m`. Intended for use as testing set.

## Source

Ipea, Ipeadata. Macroeconomic and regional data, Technical Report, <http://www.ipeadata.gov.br>, 2017. The "Most request series" section and filtered by "Frequency" equal to "Monthly".

## References

Ipea, Ipeadata. Macroeconomic and regional data, Technical Report, <http://www.ipeadata.gov.br>, 2017.

## See Also

[ipeadata\\_d](#) ~

## Examples

```
data(ipeadata_m)
str(ipeadata_m)
plot(ts(ipeadata_m[1]))
```

---

LogT

*Logarithmic Transformation*

---

## Description

The `LogT()` function returns a logarithmic transformation of the provided time series. A natural log is returned by default. `LogT.rev()` reverses the transformation.

## Usage

```
LogT(x, base = exp(1))
```

```
LogT.rev(x, base = exp(1))
```

## Arguments

`x` A numeric vector or univariate time series of class `ts`.

`base` A numeric value corresponding to the base with respect to which logarithms are computed. Default: `exp(1)`.

## Value

A vector of the same length as `x` containing the transformed values.

## Author(s)

Rebecca Pontes Salles

## References

R. H. Shumway, D. S. Stoffer, Time Series Analysis and Its Applications: With R Examples, Springer, New York, NY, 4 edition, 2017.

**See Also**

Other transformation methods: [Diff\(\)](#), [WaveletT\(\)](#), [emd\(\)](#), [mas\(\)](#), [mlm\\_io\(\)](#), [outliers\\_bp\(\)](#), [pct\(\)](#), [train\\_test\\_subset\(\)](#)

**Examples**

```
data(NN5.A)
LogT(NN5.A[,10])
```

---

LT

*Time series transformation methods*

---

**Description**

Constructors for the processing class representing a time series processing method based on a particular time series transformation.

**Usage**

```
LT(base = exp(1))
```

```
BoxCoxT(lambda = NULL, prep_par = NULL, postp_par = NULL, ...)
```

```
WT(
  level = NULL,
  filter = NULL,
  boundary = "periodic",
  prep_par = NULL,
  postp_par = NULL,
  ...
)
```

```
subsetting(train_perc = 0.8, test_len = NULL)
```

```
SW(window_len = NULL)
```

```
NAS(na.action = stats::na.omit, prep_par = NULL)
```

```
MinMax(min = NULL, max = NULL, byRow = TRUE)
```

```
AN(min = NULL, max = NULL, byRow = TRUE, outlier.rm = TRUE, alpha = 1.5)
```

```
DIFF(
  lag = NULL,
  differences = NULL,
```

```

    type = "simple",
    postp_par = list(addinit = FALSE)
)

MAS(order = NULL, prep_par = NULL, postp_par = list(addinit = FALSE))

PCT(postp_par = NULL)

EMD(num_imfs = 0, meaningfulImfs = NULL, prep_par = NULL)

```

### Arguments

base	See <a href="#">LogT</a>
lambda	See <a href="#">BCT</a>
prep_par	List of named parameters required by prep_func.
postp_par	List of named parameters required by postp_func.
...	Other parameters to be encapsulated in the class object.
level	See <a href="#">WaveletT</a>
filter	See <a href="#">WaveletT</a>
boundary	See <a href="#">WaveletT</a>
train_perc	See <a href="#">train_test_subset</a>
test_len	See <a href="#">train_test_subset</a>
window_len	See <a href="#">sw</a>
na.action	Function for handling missing values in time series data
min	See <a href="#">an</a>
max	See <a href="#">an</a>
byRow	See <a href="#">an</a>
outlier.rm	See <a href="#">an</a>
alpha	See <a href="#">an</a>
lag	See <a href="#">Diff</a>
differences	See <a href="#">Diff</a>
type	See <a href="#">Diff</a>
order	See <a href="#">mas</a>
num_imfs	See <a href="#">emd</a>
meaningfulImfs	See <a href="#">emd</a>

### Value

An object of class processing.



**Mapping-based nonstationary transformation methods**

LT: Logarithmic transform. `prep_func` set as `LogT` and `postp_func` set as `LogT.rev`.

BoxCoxT: Box-Cox transform. `prep_func` set as `BCT` and `postp_func` set as `BCT.rev`.

DIFF: Differencing. `prep_func` set as `Diff` and `postp_func` set as `Diff.rev`.

MAS: Moving average smoothing. `prep_func` set as `mas` and `postp_func` set as `mas.rev`.

PCT: Percentage change transform. `prep_func` set as `pct` and `postp_func` set as `pct.rev`.

**Splitting-based nonstationary transformation methods**

WT: Wavelet transform. `prep_func` set as `WaveletT` and `postp_func` set as `WaveletT.rev`.

EMD: Empirical mode decomposition. `prep_func` set as `emd` and `postp_func` set as `emd.rev`.

**Data subsetting methods**

subsetting: Subsetting data into training and testing sets. `prep_func` set as `train_test_subset` and `postp_func` set to `NULL`.

SW: Sliding windows. `prep_func` set as `sw` and `postp_func` set to `NULL`.

**Methods for handling missing values**

NAS: Missing values treatment. `prep_func` set as parameter `na.action` and `postp_func` set to `NULL`.

**Normalization methods**

MinMax: MinMax normalization. `prep_func` set as `minmax` and `postp_func` set to `minmax.rev`.

AN: Adaptive normalization. `prep_func` set as `an` and `postp_func` set to `an.rev`.

**Author(s)**

Rebecca Pontes Salles

**References**

R. Salles, K. Belloze, F. Porto, P.H. Gonzalez, and E. Ogasawara. Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274-291, 2019.

**See Also**

Other constructors: `ARIMA()`, `MSE_eval()`, `evaluating()`, `modeling()`, `processing()`, `tspred()`

---

MAPE

*MAPE error of prediction*

---

### Description

The function calculates the MAPE error between actual and predicted values.

### Usage

```
MAPE(actual, prediction)
```

### Arguments

actual	A vector or univariate time series containing actual values for a time series that are to be compared against its respective predictions.
prediction	A vector or univariate time series containing time series predictions that are to be compared against the values in actual.

### Value

A numeric value of the MAPE error of prediction.

### Author(s)

Rebecca Pontes Salles

### References

Z. Chen and Y. Yang, 2004, Assessing forecast accuracy measures, Preprint Series, n. 2004-2010, p. 2004-10.

### See Also

[sMAPE](#), [MSE](#), [NMSE](#), [MAXError](#)

### Examples

```
data(SantaFe.A,SantaFe.A.cont)
pred <- marimapred(SantaFe.A,n.ahead=100)
MAPE(SantaFe.A.cont[,1], pred)
```

---

`marimapar`*Get parameters of multiple ARIMA models.*

---

### Description

The function returns the parameters of a set of automatically fitted ARIMA models, including non-seasonal and seasonal orders and drift. Based on multiple application of the `arimapar` function.

### Usage

```
marimapar(timeseries, na.action = stats::na.omit, xreg = NULL)
```

### Arguments

<code>timeseries</code>	A vector, matrix, or data frame which contains a set of time series used for fitting ARIMA models. Each column corresponds to one time series.
<code>na.action</code>	A function for treating missing values in <code>timeseries</code> . The default function is <code>na.omit</code> , which omits any missing values found in <code>timeseries</code> .
<code>xreg</code>	A vector, matrix, data frame or times series of external regressors used for fitting all the ARIMA models. It must have the same number of rows as <code>TimeSeries</code> . Ignored if <code>NULL</code> .

### Details

See the `arimapar` function.

### Value

A list of numeric vectors, each one giving the number of AR, MA, seasonal AR and seasonal MA coefficients, plus the period and the number of non-seasonal and seasonal differences of the automatically fitted ARIMA models. It is also presented the value of the fitted drift constants.

### References

See the `arimapar` function.

### See Also

`arimapar`, [arimapred](#), [marimapred](#)

marimapred

*Multiple time series automatic ARIMA fitting and prediction***Description**

The function predicts and returns the next  $n$  consecutive values of a set of time series using automatically fitted ARIMA models. Based on multiple application of the [arimapred](#) function.

**Usage**

```
marimapred(
  TimeSeries,
  TimeSeriesCont = NULL,
  n.ahead = NULL,
  na.action = stats::na.omit,
  xreg = NULL,
  newxreg = NULL,
  se.fit = FALSE,
  plot = FALSE,
  range.p = 0.2,
  ylab = NULL,
  xlab = NULL,
  main = NULL
)
```

**Arguments**

TimeSeries	A vector, matrix, or data frame which contains a set of time series used for fitting ARIMA models. Each column corresponds to one time series.
TimeSeriesCont	A vector, matrix, or data frame containing continuation points for TimeSeries with actual values. Each column corresponds to one time series. Ignored if NULL.
n.ahead	A numeric vector (or a single numeric value) with the number of consecutive values which are to be predicted of each respective time series in TimeSeries. If n.ahead is NULL, the number of values to be predicted of each time series in TimeSeries is assumed to be equal to the number of rows in each respective time series in TimeSeriesCont. Required when TimeSeriesCont is NULL.
na.action	A function for treating missing values in TimeSeries and TimeSeriesCont. The default function is <a href="#">na.omit</a> , which omits any missing values found in TimeSeries or TimeSeriesCont.
xreg	A list of vectors, matrices, data frames or times series of external regressors used for fitting the ARIMA models. The first component of the list contains external regressors for the first time series in TimeSeries and therefore must have the same number of rows as this respective time series. This is also valid for the second component, and so forth. Ignored if NULL.

<code>newxreg</code>	A list of vectors, matrices, data frames or times series with new values of <code>xreg</code> to be used for prediction. The first component of the list must have at least the same number of rows as the respective first value in <code>n.ahead</code> or, if <code>n.ahead</code> is <code>NULL</code> , the number of continuation points in the respective first time series in <code>TimeSeriesCont</code> . This is also valid for the second component, and so forth. Ignored if <code>NULL</code> .
<code>se.fit</code>	If <code>se.fit</code> is <code>TRUE</code> , the standard errors of the predictions are returned.
<code>plot</code>	A Boolean parameter which defines whether the function <code>arimapred</code> will generate a graphic. If <code>plot</code> is <code>TRUE</code> , graphics will be generated for each time series in <code>TimeSeries</code> .
<code>range.p</code>	A percentage which defines how much the range of the graphics' y-axis will be increased from the minimum limits imposed by data.
<code>ylab</code>	A title for the graphics' y-axis. Ignored if <code>NULL</code> . <code>~~Describe ylab here~~</code>
<code>xlab</code>	A title for the graphics' x-axis. Ignored if <code>NULL</code> . <code>~~Describe xlab here~~</code>
<code>main</code>	An overall title for the graphics. Ignored if <code>NULL</code> . <code>~~Describe main here~~</code>

### Details

See the [arimapred](#) function.

### Value

A vector of time series of predictions, if the number of consecutive values predicted of each time series in `TimeSeries` is the same, otherwise a list of time series of predictions.

If `se.fit` is `TRUE`, a vector of lists, each one with the components `pred`, the predictions, and `se`, the estimated standard errors. Both components are time series. See the [predict.Arima](#) function in the `stats` package and the function [arimapred](#).

### Author(s)

Rebecca Pontes Salles

### References

See the [arimapred](#) function. the literature/web site here ~

### See Also

[arimapred](#) ~

### Examples

```
data(SantaFe.A,SantaFe.A.cont)
marimapred(SantaFe.A,SantaFe.A.cont)
```

---

mas *Moving average smoothing*

---

**Description**

The `mas()` function returns a simple moving average smoother of the provided time series. `mas.rev()` reverses the transformation(smoothing) process.

**Usage**

```
mas(x, order, ...)
```

```
mas.rev(xm, xi, order, addinit = TRUE)
```

**Arguments**

<code>x</code>	A numeric vector or univariate time series.
<code>order</code>	Order of moving average smoother. If NULL, it is automatically selected by <a href="#">fittestMAS</a> .
<code>...</code>	Additional arguments passed to <a href="#">fittestMAS</a> .
<code>xm</code>	A numeric vector or univariate time series that has been moving average smoothed. Possibly returned by <code>mas()</code> .
<code>xi</code>	Initial order-1 values/observations used for reverse smoothing. First order-1 known non-transformed values used to recursively obtain the original series. By default, <code>mas()</code> returns <code>xi</code> as an attribute.
<code>addinit</code>	If TRUE, <code>xi</code> is included in the return.

**Details**

The moving average smoother transformation is given by

$$(1/k) * (x[t] + x[t + 1] + \dots + x[t + k - 1])$$

where  $k=order$ ,  $t$  assume values in the range  $1:(n-k+1)$ , and  $n=length(x)$ . See also the [ma](#) of the forecast package.

**Value**

Numerical time series of length  $length(x)-order+1$  containing the simple moving average smoothed values.

**Author(s)**

Rebecca Pontes Salles

## References

R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

## See Also

Other transformation methods: [Diff\(\)](#), [LogT\(\)](#), [WaveletT\(\)](#), [emd\(\)](#), [mlm\\_io\(\)](#), [outliers\\_bp\(\)](#), [pct\(\)](#), [train\\_test\\_subset\(\)](#)

## Examples

```
data(CATS)

m <- mas(CATS[,1],order=5)

#automatically select order of moving average
m <- mas(CATS[,1],order=NULL,h=20)

x <- mas.rev(m, attributes(m)$xi, attributes(m)$order)

all(round(x,4)==round(CATS[,1],4))
```

---

MAXError

*Maximal error of prediction*

---

## Description

The function calculates the maximal error between actual and predicted values.

## Usage

```
MAXError(actual, prediction)
```

## Arguments

actual	A vector or univariate time series containing actual values for a time series that are to be compared against its respective predictions.
prediction	A vector or univariate time series containing time series predictions that are to be compared against the values in actual.

## Value

A numeric value of the maximal error of prediction.

**Author(s)**

Rebecca Pontes Salles

**See Also**

[sMAPE](#), [MAPE](#)

**Examples**

```
data(SantaFe.A,SantaFe.A.cont)
pred <- marimapred(SantaFe.A,n.ahead=100)
MAXError(SantaFe.A.cont[,1], pred)
```

---

 minmax

*Minmax Data Normalization*


---

**Description**

The `minmax()` function normalizes data of the provided time series to bring values into the range [0,1]. `minmax.rev()` reverses the normalization.

**Usage**

```
minmax(data, max = NULL, min = NULL, byRow = FALSE)
```

```
minmax.rev(data, max, min)
```

**Arguments**

<code>data</code>	A numeric vector, a univariate time series containing the values to be normalized, or a matrix with sliding windows as returned by <a href="#">sw</a> .
<code>max</code>	Integer indicating the maximal value in <code>data</code> , or a vector with the maximal values of each row (sliding window) in <code>data</code> . If <code>NULL</code> it is automatically computed.
<code>min</code>	Integer indicating the minimum value in <code>data</code> , or a vector with the minimum values of each row (sliding window) in <code>data</code> . If <code>NULL</code> it is automatically computed.
<code>byRow</code>	If <code>TRUE</code> , the normalization is performed by rows (sliding windows). Default set to <code>FALSE</code> .

**Details**

Ranging is done by using:

$$X' = \frac{(x - x_{min})}{(x_{max} - x_{min})}$$



**Value**

data normalized between 0 and 1. If `byRow` is `TRUE`, the function returns data normalized by rows (sliding windows). `max` and `min` are returned as attributes.

**Author(s)**

Rebecca Pontes Salles

**References**

R.J. Hyndman and G. Athanasopoulos, 2013, Forecasting: principles and practice. OTexts.  
E. Ogasawara, L. C. Martinez, D. De Oliveira, G. Zimbrao, G. L. Pappa, and M. Mattoso, 2010, Adaptive Normalization: A novel data normalization approach for non-stationary time series, Proceedings of the International Joint Conference on Neural Networks.

**See Also**

Other normalization methods: [an\(\)](#)

**Examples**

```
data(CATS)
d <- minmax(CATS[,1])
x <- minmax.rev(d, max = attributes(d)$max, min = attributes(d)$min)
all(round(x,4)==round(CATS[,1],4))

d <- minmax(sw(CATS[,1],5), byRow = TRUE)
x <- minmax.rev(d, max = attributes(d)$max, min = attributes(d)$min)
all(round(x,4)==round(sw(CATS[,1],5),4))
```

**Description**

Constructor for the modeling class representing a time series modeling and prediction method based on a particular model. The modeling class has two specialized subclasses `linear` and `MLM` regarding linear models and machine learning based models, respectively.

**Usage**

```
modeling(  
  train_func,  
  train_par = NULL,  
  pred_func = NULL,  
  pred_par = NULL,
```

```

    ...,
    subclass = NULL
)

MLM(
  train_func,
  train_par = NULL,
  pred_func = NULL,
  pred_par = NULL,
  sw = NULL,
  proc = NULL,
  ...,
  subclass = NULL
)

linear(
  train_func,
  train_par = NULL,
  pred_func = NULL,
  pred_par = NULL,
  ...,
  subclass = NULL
)

```

### Arguments

<code>train_func</code>	A function for training a particular model.
<code>train_par</code>	List of named parameters required by <code>train_func</code> .
<code>pred_func</code>	A function for prediction based on the model trained by <code>train_func</code> .
<code>pred_par</code>	List of named parameters required by <code>pred_func</code> .
<code>...</code>	Other parameters to be encapsulated in the class object.
<code>subclass</code>	Name of new specialized subclass object created in case it is provided.
<code>sw</code>	A <a href="#">SW</a> object regarding sliding windows processing. Optional.
<code>proc</code>	A list of <a href="#">processing</a> objects regarding any pre(post)processing needed during training or prediction. Optional.

### Value

An object of class `modeling`.

### Author(s)

Rebecca Pontes Salles

### See Also

Other constructors: [ARIMA\(\)](#), [LT\(\)](#), [MSE\\_eval\(\)](#), [evaluating\(\)](#), [processing\(\)](#), [tspred\(\)](#)

**Examples**

```
forecast_mean <- function(...){
  do.call(forecast::forecast,c(list(...)))$mean
}

l <- linear(train_func = forecast::auto.arima, pred_func = forecast_mean,
           method="ARIMA model", subclass="ARIMA")
summary(l)

m <- MLM(train_func = nnet::nnet, train_par=list(size=5),
         pred_func = predict, sw=SW(window_len = 6), proc=list(MM=MinMax()),
         method="Artificial Neural Network model", subclass="NNET")
summary(m)
```

---

MSE

*MSE error of prediction*

---

**Description**

The function calculates the MSE error between actual and predicted values.

**Usage**

```
MSE(actual, prediction)
```

**Arguments**

actual	A vector or univariate time series containing actual values for a time series that are to be compared against its respective predictions.
prediction	A vector or univariate time series containing time series predictions that are to be compared against the values in actual.

**Value**

A numeric value of the MSE error of prediction.

**Author(s)**

Rebecca Pontes Salles

**References**

Z. Chen and Y. Yang, 2004, Assessing forecast accuracy measures, Preprint Series, n. 2004-2010, p. 2004-10.

**See Also**

[NMSE](#), [MAPE](#), [sMAPE](#), [MAXError](#)

**Examples**

```
data(SantaFe.A, SantaFe.A.cont)
pred <- marimapred(SantaFe.A, n.ahead=100)
MSE(SantaFe.A.cont[,1], pred)
```

---

MSE\_eval

*Prediction/modeling quality metrics*

---

**Description**

Constructors for the evaluating class representing a time series prediction or modeling fitness quality evaluation based on particular metrics.

**Usage**

```
MSE_eval()
NMSE_eval(eval_par = list(train.actual = NULL))
RMSE_eval()
MAPE_eval()
sMAPE_eval()
MAXError_eval()
AIC_eval()
BIC_eval()
AICc_eval()
LogLik_eval()
```

**Arguments**

`eval_par` List of named parameters required by [NMSE](#) such as `train.actual`.

**Value**

An object of class `evaluating`.

**Error metrics**

MSE\_eval: Mean Squared Error.  
 NMSE\_eval: Normalised Mean Squared Error.  
 RMSE\_eval: Root Mean Squared Error.  
 MAPE\_eval: Mean Absolute Percentage Error.  
 sMAPE\_eval: Symmetric Mean Absolute Percentage Error.  
 MAXError\_eval: Maximal Error.

**Fitness criteria**

AIC\_eval: Akaike's Information Criterion.  
 BIC\_eval: Schwarz's Bayesian Information Criterion.  
 AICc\_eval: Second-order Akaike's Information Criterion.  
 LogLik\_eval: Log-Likelihood.

**Author(s)**

Rebecca Pontes Salles

**See Also**

Other constructors: [ARIMA\(\)](#), [LT\(\)](#), [evaluating\(\)](#), [modeling\(\)](#), [processing\(\)](#), [tspred\(\)](#)

---

 NMSE

---

*NMSE error of prediction*


---

**Description**

The function calculates the NMSE error between actual and predicted values.

**Usage**

```
NMSE(actual, prediction, train.actual)
```

**Arguments**

actual	A vector or univariate time series containing actual values for a time series that are to be compared against its respective predictions.
prediction	A vector or univariate time series containing time series predictions that are to be compared against the values in actual.
train.actual	A vector or univariate time series that was used to train the model that produced the predictions in prediction.

**Value**

A numeric value of the NMSE error of prediction.

**Author(s)**

Rebecca Pontes Salles

**References**

Z. Chen and Y. Yang, 2004, Assessing forecast accuracy measures, Preprint Series, n. 2004-2010, p. 2004-10.

**See Also**

[MSE](#), [MAPE](#), [sMAPE](#), [MAXError](#)

**Examples**

```
data(SantaFe.A, SantaFe.A.cont)
pred <- marimapred(SantaFe.A, n.ahead=100)
NMSE(SantaFe.A.cont[,1], pred, SantaFe.A[,1])
```

---

NN3.A

*Dataset A of the NN3 Competition*

---

**Description**

The NN3 Competition dataset composed of monthly time series drawn from homogeneous population of real empirical business time series.

**Usage**

NN3.A

**Format**

A data frame with 126 observations on the following 111 variables.

**NN3.001** a numeric vector containing the 51 observations of a univariate time series.

**NN3.002** a numeric vector containing the 51 observations of a univariate time series.

**NN3.003** a numeric vector containing the 51 observations of a univariate time series.

**NN3.004** a numeric vector containing the 51 observations of a univariate time series.

**NN3.005** a numeric vector containing the 51 observations of a univariate time series.

**NN3.006** a numeric vector containing the 51 observations of a univariate time series.







- NN3.081 a numeric vector containing the 126 observations of a univariate time series.
- NN3.082 a numeric vector containing the 116 observations of a univariate time series.
- NN3.083 a numeric vector containing the 126 observations of a univariate time series.
- NN3.084 a numeric vector containing the 122 observations of a univariate time series.
- NN3.085 a numeric vector containing the 126 observations of a univariate time series.
- NN3.086 a numeric vector containing the 126 observations of a univariate time series.
- NN3.087 a numeric vector containing the 121 observations of a univariate time series.
- NN3.088 a numeric vector containing the 78 observations of a univariate time series.
- NN3.089 a numeric vector containing the 126 observations of a univariate time series.
- NN3.090 a numeric vector containing the 126 observations of a univariate time series.
- NN3.091 a numeric vector containing the 116 observations of a univariate time series.
- NN3.092 a numeric vector containing the 115 observations of a univariate time series.
- NN3.093 a numeric vector containing the 126 observations of a univariate time series.
- NN3.094 a numeric vector containing the 116 observations of a univariate time series.
- NN3.095 a numeric vector containing the 126 observations of a univariate time series.
- NN3.096 a numeric vector containing the 126 observations of a univariate time series.
- NN3.097 a numeric vector containing the 126 observations of a univariate time series.
- NN3.098 a numeric vector containing the 126 observations of a univariate time series.
- NN3.099 a numeric vector containing the 115 observations of a univariate time series.
- NN3.100 a numeric vector containing the 116 observations of a univariate time series.
- NN3\_101 a numeric vector containing the 126 observations of a univariate time series.
- NN3\_102 a numeric vector containing the 126 observations of a univariate time series.
- NN3\_103 a numeric vector containing the 126 observations of a univariate time series.
- NN3\_104 a numeric vector containing the 115 observations of a univariate time series.
- NN3\_105 a numeric vector containing the 126 observations of a univariate time series.
- NN3\_106 a numeric vector containing the 126 observations of a univariate time series.
- NN3\_107 a numeric vector containing the 126 observations of a univariate time series.
- NN3\_108 a numeric vector containing the 115 observations of a univariate time series.
- NN3\_109 a numeric vector containing the 123 observations of a univariate time series.
- NN3\_110 a numeric vector containing the 126 observations of a univariate time series.
- NN3\_111 a numeric vector containing the 126 observations of a univariate time series.

### Details

The NN3 Competition's Dataset A contains 111 different monthly time series. Each of this time series possess from 50 to 126 observations. Each competitor in NN3 was asked to predict the next 18 corresponding observations of each times series ([NN3.A.cont](#)). The performance evaluation done by NN3 Competition was based on the mean SMAPE error of prediction found by the competitors across all time series.

**Source**

NN3 2007, The NN3 Competition: Forecasting competition for artificial neural networks and computational intelligence. URL: <http://www.neural-forecasting-competition.com/NN3/index.htm>.

**References**

S.F. Crone, M. Hibon, and K. Nikolopoulos, 2011, Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction, International Journal of Forecasting, v. 27, n. 3 (Jul.), p. 635-660.

**See Also**

[NN3.A.cont](#) ~

**Examples**

```
data(NN3.A)
str(NN3.A)
plot(ts(NN3.A["NN3_111"]))
```

---

NN3.A.cont

*Continuation dataset of the Dataset A of the NN3 Competition*

---

**Description**

A dataset of univariate time series providing 18 points beyond the end of the time series in [NN3.A](#).

**Usage**

NN3.A.cont

**Format**

A data frame with 18 observations on the following 111 variables.

**NN3.001** a numeric vector containing further observations of NN3.001 in [NN3.A](#).

**NN3.002** a numeric vector containing further observations of NN3.002 in [NN3.A](#).

**NN3.003** a numeric vector containing further observations of NN3.003 in [NN3.A](#).

**NN3.004** a numeric vector containing further observations of NN3.004 in [NN3.A](#).

**NN3.005** a numeric vector containing further observations of NN3.005 in [NN3.A](#).

**NN3.006** a numeric vector containing further observations of NN3.006 in [NN3.A](#).

**NN3.007** a numeric vector containing further observations of NN3.007 in [NN3.A](#).

**NN3.008** a numeric vector containing further observations of NN3.008 in [NN3.A](#).





**NN3.083** a numeric vector containing further observations of NN3.083 in [NN3.A.](#)  
**NN3.084** a numeric vector containing further observations of NN3.084 in [NN3.A.](#)  
**NN3.085** a numeric vector containing further observations of NN3.085 in [NN3.A.](#)  
**NN3.086** a numeric vector containing further observations of NN3.086 in [NN3.A.](#)  
**NN3.087** a numeric vector containing further observations of NN3.087 in [NN3.A.](#)  
**NN3.088** a numeric vector containing further observations of NN3.088 in [NN3.A.](#)  
**NN3.089** a numeric vector containing further observations of NN3.089 in [NN3.A.](#)  
**NN3.090** a numeric vector containing further observations of NN3.090 in [NN3.A.](#)  
**NN3.091** a numeric vector containing further observations of NN3.091 in [NN3.A.](#)  
**NN3.092** a numeric vector containing further observations of NN3.092 in [NN3.A.](#)  
**NN3.093** a numeric vector containing further observations of NN3.093 in [NN3.A.](#)  
**NN3.094** a numeric vector containing further observations of NN3.094 in [NN3.A.](#)  
**NN3.095** a numeric vector containing further observations of NN3.095 in [NN3.A.](#)  
**NN3.096** a numeric vector containing further observations of NN3.096 in [NN3.A.](#)  
**NN3.097** a numeric vector containing further observations of NN3.097 in [NN3.A.](#)  
**NN3.098** a numeric vector containing further observations of NN3.098 in [NN3.A.](#)  
**NN3.099** a numeric vector containing further observations of NN3.099 in [NN3.A.](#)  
**NN3.100** a numeric vector containing further observations of NN3.100 in [NN3.A.](#)  
**NN3\_101** a numeric vector containing further observations of NN3\_101 in [NN3.A.](#)  
**NN3\_102** a numeric vector containing further observations of NN3\_102 in [NN3.A.](#)  
**NN3\_103** a numeric vector containing further observations of NN3\_103 in [NN3.A.](#)  
**NN3\_104** a numeric vector containing further observations of NN3\_104 in [NN3.A.](#)  
**NN3\_105** a numeric vector containing further observations of NN3\_105 in [NN3.A.](#)  
**NN3\_106** a numeric vector containing further observations of NN3\_106 in [NN3.A.](#)  
**NN3\_107** a numeric vector containing further observations of NN3\_107 in [NN3.A.](#)  
**NN3\_108** a numeric vector containing further observations of NN3\_108 in [NN3.A.](#)  
**NN3\_109** a numeric vector containing further observations of NN3\_109 in [NN3.A.](#)  
**NN3\_110** a numeric vector containing further observations of NN3\_110 in [NN3.A.](#)  
**NN3\_111** a numeric vector containing further observations of NN3\_111 in [NN3.A.](#)

### Details

Contains the 18 observations which were to be predicted of each time series in Dataset A ([NN3.A](#)) as demanded by the NN3 Competition.

### Source

NN3 2007, The NN3 Competition: Forecasting competition for artificial neural networks and computational intelligence. URL: <http://www.neural-forecasting-competition.com/NN3/index.htm>.

## References

S.F. Crone, M. Hibon, and K. Nikolopoulos, 2011, Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction, *International Journal of Forecasting*, v. 27, n. 3 (Jul.), p. 635-660.

## See Also

[NN3.A](#) ~

## Examples

```
data(NN3.A.cont)
str(NN3.A.cont)
plot(ts(NN3.A.cont["NN3_111"]))
```

---

NN5.A

*Dataset A of the NN5 Competition*

---

## Description

The NN5 Competition dataset composed of daily time series originated from the observation of daily withdrawals at 111 randomly selected different cash machines at different locations within England.

## Usage

NN5.A

## Format

A data frame with 735 observations on the following 111 variables.

**NN5.001** a numeric vector containing observations of a univariate time series.

**NN5.002** a numeric vector containing observations of a univariate time series.

**NN5.003** a numeric vector containing observations of a univariate time series.

**NN5.004** a numeric vector containing observations of a univariate time series.

**NN5.005** a numeric vector containing observations of a univariate time series.

**NN5.006** a numeric vector containing observations of a univariate time series.

**NN5.007** a numeric vector containing observations of a univariate time series.

**NN5.008** a numeric vector containing observations of a univariate time series.

**NN5.009** a numeric vector containing observations of a univariate time series.

**NN5.010** a numeric vector containing observations of a univariate time series.

**NN5.011** a numeric vector containing observations of a univariate time series.







- NN5.086 a numeric vector containing observations of a univariate time series.
- NN5.087 a numeric vector containing observations of a univariate time series.
- NN5.088 a numeric vector containing observations of a univariate time series.
- NN5.089 a numeric vector containing observations of a univariate time series.
- NN5.090 a numeric vector containing observations of a univariate time series.
- NN5.091 a numeric vector containing observations of a univariate time series.
- NN5.092 a numeric vector containing observations of a univariate time series.
- NN5.093 a numeric vector containing observations of a univariate time series.
- NN5.094 a numeric vector containing observations of a univariate time series.
- NN5.095 a numeric vector containing observations of a univariate time series.
- NN5.096 a numeric vector containing observations of a univariate time series.
- NN5.097 a numeric vector containing observations of a univariate time series.
- NN5.098 a numeric vector containing observations of a univariate time series.
- NN5.099 a numeric vector containing observations of a univariate time series.
- NN5.100 a numeric vector containing observations of a univariate time series.
- NN5.101 a numeric vector containing observations of a univariate time series.
- NN5.102 a numeric vector containing observations of a univariate time series.
- NN5.103 a numeric vector containing observations of a univariate time series.
- NN5.104 a numeric vector containing observations of a univariate time series.
- NN5.105 a numeric vector containing observations of a univariate time series.
- NN5.106 a numeric vector containing observations of a univariate time series.
- NN5.107 a numeric vector containing observations of a univariate time series.
- NN5.108 a numeric vector containing observations of a univariate time series.
- NN5.109 a numeric vector containing observations of a univariate time series.
- NN5.110 a numeric vector containing observations of a univariate time series.
- NN5.111 a numeric vector containing observations of a univariate time series.

### Details

The NN5 Competition's Dataset A contains 111 different daily time series. Each of these time series possesses 735 observations, and may present missing data. The time series also show different patterns of single or multiple overlying seasonal properties. Each competitor in NN5 was asked to predict the next 56 corresponding observations of each time series (NN5.A.cont). The performance evaluation done by NN5 Competition was based on the mean SMAPE error of prediction found by the competitors across all time series.

### Source

NN5 2008, The NN5 Competition: Forecasting competition for artificial neural networks and computational intelligence. URL: <http://www.neural-forecasting-competition.com/NN5/index.htm>.

## References

S.F. Crone, 2008, Results of the NN5 time series forecasting competition. Hong Kong, Presentation at the IEEE world congress on computational intelligence. WCCI'2008.

## See Also

[NN5.A.cont](#) ~

## Examples

```
data(NN5.A)
str(NN5.A)
plot(ts(NN5.A["NN5.111"]))
```

---

NN5.A.cont

*Continuation dataset of the Dataset A of the NN5 Competition*

---

## Description

A dataset of univariate time series providing 56 points beyond the end of the time series in [NN5.A](#).

## Usage

```
NN5.A.cont
```

## Format

A data frame with 56 observations on the following 111 variables.

**NN5.001** a numeric vector containing further observations of NN5.001 in [NN5.A](#).

**NN5.002** a numeric vector containing further observations of NN5.002 in [NN5.A](#).

**NN5.003** a numeric vector containing further observations of NN5.003 in [NN5.A](#).

**NN5.004** a numeric vector containing further observations of NN5.004 in [NN5.A](#).

**NN5.005** a numeric vector containing further observations of NN5.005 in [NN5.A](#).

**NN5.006** a numeric vector containing further observations of NN5.006 in [NN5.A](#).

**NN5.007** a numeric vector containing further observations of NN5.007 in [NN5.A](#).

**NN5.008** a numeric vector containing further observations of NN5.008 in [NN5.A](#).

**NN5.009** a numeric vector containing further observations of NN5.009 in [NN5.A](#).

**NN5.010** a numeric vector containing further observations of NN5.010 in [NN5.A](#).

**NN5.011** a numeric vector containing further observations of NN5.011 in [NN5.A](#).

**NN5.012** a numeric vector containing further observations of NN5.012 in [NN5.A](#).

**NN5.013** a numeric vector containing further observations of NN5.013 in [NN5.A](#).





**NN5.088** a numeric vector containing further observations of NN5.088 in [NN5.A](#).  
**NN5.089** a numeric vector containing further observations of NN5.089 in [NN5.A](#).  
**NN5.090** a numeric vector containing further observations of NN5.090 in [NN5.A](#).  
**NN5.091** a numeric vector containing further observations of NN5.091 in [NN5.A](#).  
**NN5.092** a numeric vector containing further observations of NN5.092 in [NN5.A](#).  
**NN5.093** a numeric vector containing further observations of NN5.093 in [NN5.A](#).  
**NN5.094** a numeric vector containing further observations of NN5.094 in [NN5.A](#).  
**NN5.095** a numeric vector containing further observations of NN5.095 in [NN5.A](#).  
**NN5.096** a numeric vector containing further observations of NN5.096 in [NN5.A](#).  
**NN5.097** a numeric vector containing further observations of NN5.097 in [NN5.A](#).  
**NN5.098** a numeric vector containing further observations of NN5.098 in [NN5.A](#).  
**NN5.099** a numeric vector containing further observations of NN5.099 in [NN5.A](#).  
**NN5.100** a numeric vector containing further observations of NN5.100 in [NN5.A](#).  
**NN5.101** a numeric vector containing further observations of NN5.101 in [NN5.A](#).  
**NN5.102** a numeric vector containing further observations of NN5.102 in [NN5.A](#).  
**NN5.103** a numeric vector containing further observations of NN5.103 in [NN5.A](#).  
**NN5.104** a numeric vector containing further observations of NN5.104 in [NN5.A](#).  
**NN5.105** a numeric vector containing further observations of NN5.105 in [NN5.A](#).  
**NN5.106** a numeric vector containing further observations of NN5.106 in [NN5.A](#).  
**NN5.107** a numeric vector containing further observations of NN5.107 in [NN5.A](#).  
**NN5.108** a numeric vector containing further observations of NN5.108 in [NN5.A](#).  
**NN5.109** a numeric vector containing further observations of NN5.109 in [NN5.A](#).  
**NN5.110** a numeric vector containing further observations of NN5.110 in [NN5.A](#).  
**NN5.111** a numeric vector containing further observations of NN5.111 in [NN5.A](#).

### Details

Contains the 56 observations which were to be predicted of each time series in Dataset A ([NN5.A](#)) as demanded by the NN5 Competition.

### Source

NN5 2008, The NN5 Competition: Forecasting competition for artificial neural networks and computational intelligence. URL: <http://www.neural-forecasting-competition.com/NN5/index.htm>.

### References

S.F. Crone, 2008, Results of the NN5 time series forecasting competition. Hong Kong, Presentation at the IEEE world congress on computational intelligence. WCCI'2008.

**See Also**[NN5.A ~](#)**Examples**

```
data(NN5.A.cont)
str(NN5.A.cont)
plot(ts(NN5.A.cont["NN5.111"]))
```

---

`outliers_bp`*Outlier removal from sliding windows of data*

---

**Description**

Function to perform outlier removal from sliding windows of data. The `outliers_bp()` function removes windows with extreme values using a method based on Box plots for detecting outliers.

**Usage**

```
outliers_bp(data, alpha = 1.5)
```

**Arguments**

<code>data</code>	A numeric matrix with sliding windows of time series data as returned by <a href="#">sw</a> .
<code>alpha</code>	The multiplier for the interquartile range used as base for outlier removal. The default is set to 1.5. The value 3.0 is also commonly used to remove only the extreme outliers.

**Details**

The method applied prune any value smaller than the first quartile minus 1.5 times the interquartile range, and also any value larger than the third quartile plus 1.5 times the interquartile range, that is, all the values that are not in the range  $[Q1-1.5 \times IQR, Q3+1.5 \times IQR]$  are considered outliers and are consequently removed.

**Value**

Same as `data` with outliers removed.

**Author(s)**

Rebecca Pontes Salles

## References

E. Ogasawara, L. C. Martinez, D. De Oliveira, G. Zimbrao, G. L. Pappa, and M. Mattoso, 2010, Adaptive Normalization: A novel data normalization approach for non-stationary time series, Proceedings of the International Joint Conference on Neural Networks.

## See Also

Other transformation methods: [Diff\(\)](#), [LogT\(\)](#), [WaveletT\(\)](#), [emd\(\)](#), [mas\(\)](#), [mlm\\_io\(\)](#), [pct\(\)](#), [train\\_test\\_subset\(\)](#)

## Examples

```
data(CATS)
swin <- sw(CATS[,1],5)
d <- outliers_bp(swin)
```

---

pct	<i>Percentage Change Transformation</i>
-----	---

---

## Description

The `pct()` function returns a transformation of the provided time series using a Percentage Change transformation. `pct.rev()` reverses the transformation.

## Usage

```
pct(x)

pct.rev(p, xi, addinit = TRUE)
```

## Arguments

x	A numeric vector or univariate time series of class <code>ts</code> .
p	A numeric vector or univariate time series of percentage changes. Possibly returned by <code>pct()</code> .
xi	Initial value/observation of x ( <code>x[1]</code> ). First known non-transformed value used to recursively obtain the original series.
addinit	If <code>TRUE</code> , <code>xi</code> is included in the return.

## Details

The Percentage Change transformation is given approximately by

$$\log(x[2 : n]/x[1 : (n - 1)]) = \log(x[2 : n]) - \log(x[1 : (n - 1)])$$

where `n=length(x)`.

**Value**

A vector of length  $\text{length}(x)-1$  containing the transformed values.

**Author(s)**

Rebecca Pontes Salles

**References**

R.H. Shumway and D.S. Stoffer, 2010, Time Series Analysis and Its Applications: With R Examples. 3rd ed. 2011 edition ed. New York, Springer.

**See Also**

Other transformation methods: [Diff\(\)](#), [LogT\(\)](#), [WaveletT\(\)](#), [emd\(\)](#), [mas\(\)](#), [mlm\\_io\(\)](#), [outliers\\_bp\(\)](#), [train\\_test\\_subset\(\)](#)

**Examples**

```
data(NN5.A)
ts <- na.omit(NN5.A[,10])
length(ts)

p <- pct(ts)
length(p)

p_rev <- pct.rev(p, attributes(p)$xi)

all(round(p_rev,4)==round(ts,4))
```

---

plotarimapred

*Plot ARIMA predictions against actual values*

---

**Description**

The function plots ARIMA predictions against its actual values with prediction intervals.

**Usage**

```
plotarimapred(
  ts.cont,
  fit.arima,
  xlim,
  range.percent = 0.2,
  xreg = NULL,
  ylab = NULL,
```



```

    xlab = NULL,
    main = NULL
  )

```

### Arguments

<code>ts.cont</code>	A vector or univariate time series containing actual values for a time series that are to be plotted against its respective predictions. The number of consecutive values to be predicted is assumed to be equal to the number of rows in <code>ts.cont</code> . If <code>xreg</code> is used, the number of values to be predicted is set to the number of rows of <code>xreg</code> . <i>~~Describe ts.cont here~~</i>
<code>fit.arma</code>	A fitted ARIMA model for the time series that is to be predicted. An object of class "Arima", "ar" or "fracdiff". See the object argument of the <a href="#">forecast.Arima</a> function in the forecast package.
<code>xlim</code>	Numeric vector containing the initial and final limits of the x-axis to be plotted, respectively.
<code>range.percent</code>	A percentage which defines how much the range of the graphic's y-axis will be increased from the minimum limits imposed by data.
<code>xreg</code>	A vector, matrix, data frame or times series with new values of external regressors to be used for prediction (for class Arima objects only). See the <code>xreg</code> argument of the <a href="#">forecast.Arima</a> function in the forecast package.
<code>ylab</code>	A title for the graphic's y-axis. Ignored if NULL. <i>~~Describe ylab here~~</i>
<code>xlab</code>	A title for the graphic's x-axis. Ignored if NULL. <i>~~Describe xlab here~~</i>
<code>main</code>	An overall title for the graphic. Ignored if NULL. <i>~~Describe main here~~</i>

### Details

The model in `fit.arma` is used for prediction by the [forecast.Arima](#) function in the forecast package. The resulting forecast object is then used for plotting the predictions and their intervals by the [plot.forecast](#) function also in the forecast package. For more details, see the [forecast.Arima](#) and the [plot.forecast](#) functions in the forecast package.

### Value

None.

### Author(s)

Rebecca Pontes Salles

### References

See the [forecast.Arima](#) and the [plot.forecast](#) functions in the forecast package. references to the literature/web site here ~

### See Also

[forecast.Arima](#), [plot.forecast](#), [arimapred](#) ~

## Examples

```
data(SantaFe.A,SantaFe.A.cont)
fit <- forecast::auto.arima(SantaFe.A)
ts.cont <- ts(SantaFe.A.cont,start=1001)
plotarimapred(ts.cont, fit, xlim=c(1001,1100))
```

---

postprocess.tspred      *Postprocess method for [tspred](#) objects*

---

## Description

Performs postprocessing of the predicted time series data contained in a [tspred](#) class object reversing a particular set of transformation methods. Each transformation method is defined by a [processing](#) object in the list contained in the [tspred](#) class object.

## Usage

```
## S3 method for class 'tspred'
postprocess(obj, ...)
```

## Arguments

`obj`                    An object of class [tspred](#) defining a particular time series prediction process.

`...`                    Other parameters passed to the method [postprocess](#) of the [processing](#) objects from `obj`.

## Details

The function [postprocess.tspred](#) recursively calls the method [postprocess](#) on each [processing](#) object contained in `obj` in the inverse order as done by [preprocess.tspred](#). The postprocessed predictions resulting from each of these calls is used as input to the next call. Finally, the produced postprocessed time series predictions are introduced in the structure of the [tspred](#) class object in `obj`.

The same transformation method parameters used/computed during preprocessing, duly saved in the structure of the [tspred](#) class object in `obj`, are used for reversing the transformations during postprocessing.

## Value

An object of class `tspred` with updated structure containing postprocessed time series predictions.

## Author(s)

Rebecca Pontes Salles

**See Also**

[[tspred\(\)](#)] for defining a particular time series prediction process, and [[LT\(\)](#)] for defining a time series transformation method.

Other preprocess: [preprocess.tspred\(\)](#), [subset\(\)](#)

**Examples**

```
data(CATS)

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class
mod11 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod11,
                  evaluating=list(MSE=eval1)
)
summary(tspred_1)

tspred_1 <- subset(tspred_1, data=CATS[3])
tspred_1 <- preprocess(tspred_1,prep_test=FALSE)
tspred_1 <- train(tspred_1)
tspred_1 <- predict(tspred_1, onestep=TRUE)
tspred_1 <- postprocess(tspred_1)
```

---

predict

*Predict method for [modeling](#) objects*

---

**Description**

Obtains time series predictions based on a trained model and a particular prediction function defined in a [modeling](#) object.

**Usage**

```
## S3 method for class 'MLM'
predict(object, mdl, data, n.ahead, ..., onestep = TRUE)

## S3 method for class 'linear'
predict(object, mdl, data, n.ahead, ..., onestep = TRUE)
```

**Arguments**

object	An object of class <code>modeling</code> defining a particular model.
mdl	A time series model object used for prediction.
data	A list of time series data input for prediction.
n.ahead	Integer defining the number of observations to be predicted.
...	Other parameters passed to <code>pred_func</code> of object.
onestep	Should the function produce one-step ahead predictions? If FALSE, a multi-step ahead prediction approach is adopted.  For <code>predict.MLM</code> , <code>sw</code> of object may be used to transform the time series input in data into sliding windows used during prediction. Also, <code>proc</code> of object may be used to preprocess/postprocess the input during prediction.

**Value**

A list containing object and the produced predictions.

**Author(s)**

Rebecca Pontes Salles

**See Also**

Other predict: `predict.tspred()`

**Examples**

```
data(CATS,CATS.cont)

a <- ARIMA()
model <- train(a,list(CATS[,1]))$results[[1]]$res
pred_data <- predict(a,model,data=NULL,n.ahead=20,onestep=FALSE)

n <- NNET(size=5, sw=SW(window_len = 5+1), proc=list(MM=MinMax()))
model <- train(n,list(CATS[,1]))$results[[1]]$res
pred_data <- predict(n,model,data=list(CATS.cont[,1]),n.ahead=20)
```

---

predict.tspred      *Predict method for `tspred` objects*

---

### Description

Obtains predictions for the time series data contained in a `tspred` class object based on a particular trained model and a prediction method. The model training and prediction method is defined by a `modeling` object contained in the `tspred` class object.

### Usage

```
## S3 method for class 'tspred'  
predict(object, onestep = obj$one_step, ...)
```

### Arguments

<code>object</code>	An object of class <code>tspred</code> defining a particular time series prediction process.
<code>onestep</code>	Should the function produce one-step ahead predictions? If FALSE, a multi-step ahead prediction approach is adopted.
<code>...</code>	Other parameters passed to the method <code>predict</code> of the <code>modeling</code> object from <code>object</code> .

### Details

The function `predict.tspred` calls the method `predict` on the `modeling` objects for each trained model and time series contained in `object`. Finally, the produced time series predictions are introduced in the structure of the `tspred` class object in `object`.

### Value

An object of class `tspred` with updated structure containing the produced time series predictions.

### Author(s)

Rebecca Pontes Salles

### See Also

[`tspred()`] for defining a particular time series prediction process, and [`ARIMA()`] for defining a time series modeling and prediction method.

Other predict: `predict()`

**Examples**

```

data(CATS)

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class
mod1 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod1,
                  evaluating=list(MSE=eval1)
)
summary(tspred_1)

tspred_1 <- subset(tspred_1, data=CATS[3])
tspred_1 <- preprocess(tspred_1,prep_test=FALSE)
tspred_1 <- train(tspred_1)
tspred_1 <- predict(tspred_1, onestep=TRUE)

```

---

preprocess

*Preprocessing/Postprocessing time series data*


---

**Description**

preprocess and postprocess are generic functions for preprocessing and postprocessing time series data, respectively, based on a particular transformation method defined in a [processing](#) object. Generally, postprocessing reverses the transformation performed during preprocessing.

**Usage**

```

preprocess(obj, ...)

## S3 method for class 'processing'
preprocess(obj, data, ..., map = TRUE)

postprocess(obj, ...)

```

```
## S3 method for class 'processing'
postprocess(obj, data, ..., map = TRUE)
```

### Arguments

obj	An object of class <code>processing</code> defining a particular transformation method.
...	Other parameters passed to <code>prep_func/postp_func</code> of obj.
data	A list of time series to be transformed.
map	Should the transformation be performed in each individual time series? If FALSE the function processes the provided set of time series as a whole.

### Value

A list containing obj and the transformed time series.

### Author(s)

Rebecca Pontes Salles

### Examples

```
data(NN5.A)
t <- LT(base = exp(1))
prep_ts <- preprocess(t, list(NN5.A[,10]))$results[[1]]$res
postp_ts <- postprocess(t, list(prep_ts))$results[[1]]$res
```

---

preprocess.tspred      *Preprocess method for `tspred` objects*

---

### Description

Performs preprocessing of the time series data contained in a `tspred` class object based on a particular set of transformation methods. Each transformation method is defined by a `processing` object in the list contained in the `tspred` class object.

### Usage

```
## S3 method for class 'tspred'
preprocess(obj, prep_test = FALSE, ...)
```

### Arguments

obj	An object of class <code>tspred</code> defining a particular time series prediction process.
prep_test	Should the testing set of data be preprocessed as well?
...	Other parameters passed to the method <code>preprocess</code> of the <code>processing</code> objects from obj.

## Details

The function `preprocess.tspred` recursively calls the method `preprocess` on each `processing` object contained in `obj`. The preprocessed time series resulting from each of these calls is used as input to the next call. Thus, the order of the list of `processing` objects in `obj` becomes important. Finally, the produced preprocessed time series data are introduced in the structure of the `tspred` class object in `obj`.

If any transformation method parameters are computed during preprocessing, they are duly updated in the structure of the `tspred` class object in `obj`. This is important not only for provenance and reproducibility of the prediction process, but it is also crucial for the postprocessing step, since the same parameters must be used for reversing any transformations. Furthermore, if `prep_test` is `TRUE`, testing sets are preprocessed with the same parameters saved from preprocessing the training set.

## Value

An object of class `tspred` with updated structure containing preprocessed time series data.

## Author(s)

Rebecca Pontes Salles

## See Also

[`tspred()`] for defining a particular time series prediction process, and [`LT()`] for defining a time series transformation method.

Other preprocess: `postprocess.tspred()`, `subset()`

## Examples

```
data(CATS)

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class
mod11 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod11,
                  evaluating=list(MSE=eval1)
)
summary(tspred_1)
```



```

tspred_1 <- subset(tspred_1, data=CATS[3])
tspred_1 <- preprocess(tspred_1, prep_test=FALSE)

```

---

processing

*Time series data processing*


---

### Description

Constructor for the processing class representing a time series processing method based on a particular time series transformation.

### Usage

```

processing(
  prep_func,
  prep_par = NULL,
  postp_func = NULL,
  postp_par = NULL,
  ...,
  subclass = NULL
)

```

### Arguments

prep_func	A function for preprocessing the time series data.
prep_par	List of named parameters required by prep_func.
postp_func	A function for postprocessing the time series data. Generally reverses the transformation performed by prep_func.
postp_par	List of named parameters required by postp_func.
...	Other parameters to be encapsulated in the class object.
subclass	Name of new specialized subclass object created in case it is provided.

### Value

An object of class processing.

### Author(s)

Rebecca Pontes Salles

### See Also

Other constructors: [ARIMA\(\)](#), [LT\(\)](#), [MSE\\_eval\(\)](#), [evaluating\(\)](#), [modeling\(\)](#), [tspred\(\)](#)

## Examples

```
base <- exp(1)
lt <- processing(prepare_func=TSPred::LogT, prep_par=list(base=base),
                 postp_func=TSPred::LogT.rev, postp_par=list(base=base),
                 method="Logarithmic transform", subclass="LT")
summary(lt)
```

---

SantaFe.A

*Time series A of the Santa Fe Time Series Competition*

---

## Description

A univariate time series derived from laser-generated data recorded from a Far-Infrared-Laser in a chaotic state.

## Usage

SantaFe.A

## Format

A data frame with 1000 observations on the following variable.

**V1** a numeric vector containing the observations of the univariate time series A of the Santa Fe Time Series Competition.

## Details

The main benchmark of the Santa Fe Time Series Competition, time series A, is composed of a clean low-dimensional nonlinear and stationary time series with 1,000 observations. Competitors were asked to correctly predict the next 100 observations ([SantaFe.A.cont](#)). The performance evaluation done by the Santa Fe Competition was based on the NMSE errors of prediction found by the competitors.

## References

A.S. Weigend, 1993, Time Series Prediction: Forecasting The Future And Understanding The Past. Reading, MA, Westview Press.

## See Also

[SantaFe.A.cont](#), [SantaFe.D](#), [SantaFe.D.cont](#) ~

## Examples

```
data(SantaFe.A)
str(SantaFe.A)
plot(ts(SantaFe.A))
```

---

SantaFe.A.cont	<i>Continuation dataset of the time series A of the Santa Fe Time Series Competition</i>
----------------	--

---

## Description

A univariate time series providing 100 points beyond the end of the time series A in [SantaFe.A](#).

## Usage

```
SantaFe.A.cont
```

## Format

A data frame with 100 observations on the following variable.

**V1** a numeric vector containing further observations of the univariate time series A of the Santa Fe Time Series Competition in [SantaFe.A](#).

## Details

Contains the 100 observations which were to be predicted of the time series A ([SantaFe.A](#)) as demanded by the Santa Fe Time Series Competition.

## References

A.S. Weigend, 1993, Time Series Prediction: Forecasting The Future And Understanding The Past. Reading, MA, Westview Press.

## See Also

[SantaFe.A](#), [SantaFe.D](#), [SantaFe.D.cont](#) ~

## Examples

```
data(SantaFe.A.cont)
str(SantaFe.A.cont)
plot(ts(SantaFe.A.cont))
```

---

SantaFe.D

*Time series D of the Santa Fe Time Series Competition*

---

**Description**

A univariate computer-generated time series.

**Usage**

SantaFe.D

**Format**

A data frame with 100000 observations on the following variable.

**V1** a numeric vector containing the observations of the univariate time series D of the Santa Fe Time Series Competition.

**Details**

One of the benchmarks of the Santa Fe Time Series Competition, time series D, is composed of a four-dimensional nonlinear time series with non-stationary properties and 100,000 observations. Competitors were asked to correctly predict the next 500 observations of this time series ([SantaFe.D.cont](#)). The performance evaluation done by the Santa Fe Competition was based on the NMSE errors of prediction found by the competitors.

**References**

A.S. Weigend, 1993, Time Series Prediction: Forecasting The Future And Understanding The Past. Reading, MA, Westview Press.

**See Also**

[SantaFe.D.cont](#), [SantaFe.A](#), [SantaFe.A.cont](#) ~

**Examples**

```
data(SantaFe.D)
str(SantaFe.D)
plot(ts(SantaFe.D), xlim=c(1,2000))
```

---

SantaFe.D.cont	<i>Continuation dataset of the time series D of the Santa Fe Time Series Competition</i>
----------------	--

---

### Description

A univariate time series providing 500 points beyond the end of the time series D in [SantaFe.D](#).

### Usage

SantaFe.D.cont

### Format

A data frame with 500 observations on the following variable.

**V1** a numeric vector containing further observations of the univariate time series D of the Santa Fe Time Series Competition in [SantaFe.D](#).

### Details

Contains the 500 observations which were to be predicted of the time series D ([SantaFe.D](#)) as demanded by the Santa Fe Time Series Competition.

### References

A.S. Weigend, 1993, Time Series Prediction: Forecasting The Future And Understanding The Past. Reading, MA, Westview Press.

### See Also

[SantaFe.D](#), [SantaFe.A](#), [SantaFe.A.cont](#) ~

### Examples

```
data(SantaFe.D.cont)
str(SantaFe.D.cont)
plot(ts(SantaFe.D.cont))
```

---

sMAPE

*sMAPE error of prediction*

---

### Description

The function calculates the sMAPE error between actual and predicted values.

### Usage

```
sMAPE(actual, prediction)
```

### Arguments

actual	A vector or univariate time series containing actual values for a time series that are to be compared against its respective predictions.
prediction	A vector or univariate time series containing time series predictions that are to be compared against the values in actual. ~~Describe forecast here~~

### Value

A numeric value of the sMAPE error of prediction.

### Author(s)

Rebecca Pontes Salles

### References

Z. Chen and Y. Yang, 2004, Assessing forecast accuracy measures, Preprint Series, n. 2004-2010, p. 2004-10. literature/web site here ~

### See Also

[MAPE](#), [MSE](#), [NMSE](#), [MAXError](#) ~

### Examples

```
data(SantaFe.A,SantaFe.A.cont)
pred <- marimaped(SantaFe.A,n.ahead=100)
sMAPE(SantaFe.A.cont[,1], pred)
```

---

`subset`*Subsetting data into training and testing sets*

---

### Description

`subset` is a generic function for subsetting time series data into training and testing sets. The function invokes particular *methods* which depend on the class of the first argument.

### Usage

```
subset(obj, ...)
```

```
## S3 method for class 'tspred'  
subset(obj, data = NULL, ...)
```

### Arguments

<code>obj</code>	An object of class <code>tspred</code> defining a particular time series prediction process.
<code>...</code>	Other parameters passed to the method <code>preprocess</code> of object <code>subsetting</code> from <code>obj</code> .
<code>data</code>	A list of time series to be transformed.

### Details

The function `subset.tspred` calls the method `preprocess` on the `subsetting` object from `obj`. The produced training and testing sets of time series data are introduced in the structure of the `tspred` class object in `obj`.

### Value

An object of class `tspred` with updated structure containing the produced training and testing sets of time series data.

### Author(s)

Rebecca Pontes Salles

### See Also

[`tspred()`] for defining a particular time series prediction process, and [`subsetting()`] for defining a time series subsetting transformation.

Other preprocess: `postprocess.tspred()`, `preprocess.tspred()`

## Examples

```
data(CATS)

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)

#Obtaining objects of the modeling class
mod11 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2),
                  modeling=mod11,
                  evaluating=list(MSE=eval1)
)
summary(tspred_1)

tspred_1_subset <- subset(tspred_1, data=CATS[3])
```

---

sw

*Generating sliding windows of data*

---

## Description

The function extracts all possible subsequences (of the same length) of a time series (or numeric vector), generating a set of sliding windows of data, often used to train machine learning methods.

## Usage

```
sw(x, k)
```

## Arguments

x	A vector or univariate time series from which the sliding windows are to be extracted.
k	Numeric value corresponding to the required size (length) of each sliding window.

## Details

The function returns all (overlapping) subsequences of size `swSize` of `timeseries`.



**Value**

A numeric matrix of size  $(\text{length}(x)-k+1)$  by  $k$ , where each line is a sliding window.

**Author(s)**

Rebecca Pontes Salles

**References**

Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1-8. IEEE.

Keogh, E. and Lin, J. (2005). Clustering of time series subsequences is meaningless: Implications for previous and future research. *Knowledge and Information Systems*, 8(2):154-177.

**Examples**

```
data("CATS")
s <- sw(CATS[,1],4)
```

---

train

*Training a time series model*


---

**Description**

`train` is a generic function for training a time series model based on a particular training function defined in a `modeling` object. The function invokes particular *methods* which depend on the class of the first argument.

**Usage**

```
train(obj, ...)

## S3 method for class 'MLM'
train(obj, data, ...)

## S3 method for class 'linear'
train(obj, data, ...)
```

**Arguments**

`obj` An object of class `modeling` defining a particular model.

`...` Other parameters passed to `train_func` of `obj`.  
For `train.MLM`, `sw` of `obj` may be used to transform the time series in `data` into sliding windows used during training. Also, `proc` of `obj` may be used to preprocess the time series before training.

data            A list of time series to be modelled.

**Value**

A list containing obj and the trained models.

**Author(s)**

Rebecca Pontes Salles

**See Also**

Other train: [train.tspred\(\)](#)

**Examples**

```
data(CATS,CATS.cont)

a <- ARIMA()
model <- train(a,list(CATS[,1]))

n <- NNET(size=5, sw=SW(window_len = 5+1), proc=list(MM=MinMax()))
model <- train(n,list(CATS[,1]))
```

---

train.tspred            *Train method for [tspred](#) objects*

---

**Description**

Fits a model to the time series data contained in a [tspred](#) class object based on a particular model training method. The model training method is defined by a [modeling](#) object contained in the [tspred](#) class object.

**Usage**

```
## S3 method for class 'tspred'
train(obj, ...)
```

**Arguments**

obj            An object of class [tspred](#) defining a particular time series prediction process.  
...            Ignored

## Details

The function `train.tspred` calls the method `train` on the `modeling` object for each time series contained in `obj`. Finally, the produced time series model is introduced in the structure of the `tspred` class object in `obj`.

If any modeling parameters are computed during training, they are duly updated in the structure of the `tspred` class object in `obj`. This is important for provenance and reproducibility of the training process.

## Value

An object of class `tspred` with updated structure containing the produced trained time series models.

## Author(s)

Rebecca Pontes Salles

## See Also

[`tspred()`] for defining a particular time series prediction process, and [`ARIMA()`] for defining a time series modeling and prediction method.

Other train: `train()`

## Examples

```
data(CATS)

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class
mod11 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod11,
                  evaluating=list(MSE=eval1)
)
summary(tspred_1)

tspred_1 <- subset(tspred_1, data=CATS[3])
tspred_1 <- preprocess(tspred_1,prep_test=FALSE)
tspred_1 <- train(tspred_1)
```

---

train\_test\_subset      *Get training and testing subsets of data*

---

### Description

Function subsets data into training and testing datasets.

### Usage

```
train_test_subset(data, train_perc = 0.8, test_len = NULL)
```

### Arguments

data	A numeric vector, time series, data.frame or matrix containing data to be subsetted.
train_perc	Percentage of data observations to compose the training dataset. Ignored if test_len is given.
test_len	Required length of testing dataset. If NULL, 1-train_perc is used for computing the number of data observations in the testing dataset.

### Value

A list with train and test subsets of data.

### Author(s)

Rebecca Pontes Salles

### See Also

Other transformation methods: [Diff\(\)](#), [LogT\(\)](#), [WaveletT\(\)](#), [emd\(\)](#), [mas\(\)](#), [mlm\\_io\(\)](#), [outliers\\_bp\(\)](#), [pct\(\)](#)

### Examples

```
data(CATS)
d <- train_test_subset(CATS[,1])

swin <- sw(CATS[,1],5)
d_sw <- train_test_subset(swin)
```

---

tspred	<i>Time series prediction process</i>
--------	---------------------------------------

---

### Description

Constructor for the `tspred` class representing a time series prediction process. This process may involve subsetting the time series data into training and testing sets, preprocessing/postprocessing the data, modeling, prediction and finally an evaluation of modeling fitness and prediction quality. All these process steps should be based on particular time series transformation methods, a modeling and prediction method, and quality metrics which are defined in a `tspred` class object.

### Usage

```
tspred(
  subsetting = NULL,
  processing = NULL,
  modeling = NULL,
  evaluating = NULL,
  data = NULL,
  n.ahead = NULL,
  one_step = FALSE,
  ...,
  subclass = NULL
)
```

### Arguments

<code>subsetting</code>	A <code>subsetting</code> object regarding subsetting processing.
<code>processing</code>	List of named <code>processing</code> objects used for pre(post)processing the data.
<code>modeling</code>	A <code>modeling</code> object used for time series modeling and prediction.
<code>evaluating</code>	List of named <code>evaluating</code> objects used for prediction/modeling quality evaluation.
<code>data</code>	A list of time series to be pre(post)processed, modelled and/or predicted.
<code>n.ahead</code>	Integer defining the number of observations to be predicted.
<code>one_step</code>	Should the function produce one-step ahead predictions? If FALSE, a multi-step ahead prediction approach is adopted.
<code>...</code>	Other parameters to be encapsulated in the class object.
<code>subclass</code>	Name of new specialized subclass object created in case it is provided.

### Value

An object of class `tspred`.

### Author(s)

Rebecca Pontes Salles

**See Also**

Other constructors: [ARIMA\(\)](#), [LT\(\)](#), [MSE\\_eval\(\)](#), [evaluating\(\)](#), [modeling\(\)](#), [processing\(\)](#)

**Examples**

```
#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class
mod1 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()
eval2 <- MAPE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod1,
                  evaluating=list(MSE=eval1,
                                MAPE=eval2)
                  )
summary(tspred_1)

#Obtaining objects of the processing class
proc4 <- SW(window_len = 6)
proc5 <- MinMax()

#Obtaining objects of the modeling class
mod2 <- NNET(size=5,sw=proc4,proc=list(MM=proc5))

#Defining a time series prediction process
tspred_2 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                WT=proc3),
                  modeling=mod2,
                  evaluating=list(MSE=eval1,
                                MAPE=eval2)
                  )
summary(tspred_2)
```

**Description**

The function automatically applies a maximal overlap discrete wavelet transform to a provided univariate time series. Wrapper function for `modwt` of the `wavelets` package. It also allows the automatic selection of the level and filter of the transform using `fittestWavelet`. `WaveletT.rev()` reverses the transformation based on the `imodwt` function.

**Usage**

```
WaveletT(
  x,
  level = NULL,
  filter = c("haar", "d4", "la8", "b114", "c6"),
  boundary = "periodic",
  ...
)
```

```
WaveletT.rev(pred = NULL, wt_obj)
```

**Arguments**

<code>x</code>	A numeric vector or univariate time series to be decomposed.
<code>level</code>	An integer specifying the level of the decomposition. If <code>NULL</code> , it is automatically selected using <code>fittestWavelet</code> .
<code>filter</code>	A character string indicating which wavelet filter to use in the decomposition. If <code>NULL</code> , or a vector and <code>length(filters)&gt;1</code> , the wavelet transform filter is automatically selected using <code>fittestWavelet</code> .
<code>boundary</code>	See <code>modwt</code> .
<code>...</code>	Additional arguments passed to <code>fittestWavelet</code> .
<code>pred</code>	A list containing component series (such as) resulting from wavelet transform ( <code>WaveletT()</code> ).
<code>wt_obj</code>	Object of class <code>modwt</code> containing the wavelet transformed series.

**Value**

A list containing each component series resulting from the decomposition of `x` (`level` wavelet coefficients series and `level` scaling coefficients series). An object of class `modwt` containing the wavelet transformed/decomposed time series is passed as an attribute named `"wt_obj"`. This attribute is passed to `wt_obj` in `WaveletT.rev()`.

**Author(s)**

Rebecca Pontes Salles

## References

- A. J. Conejo, M. A. Plazas, R. Espinola, A. B. Molina, Day-ahead electricity price forecasting using the wavelet transform and ARIMA models, *IEEE Transactions on Power Systems* 20 (2005) 1035-1042.
- T. Joo, S. Kim, Time series forecasting based on wavelet filtering, *Expert Systems with Applications* 42 (2015) 3868-3874.
- C. Stolojescu, I. Railean, S. M. P. Lenca, A. Isar, A wavelet based prediction method for time series. In *Proceedings of the 2010 International Conference Stochastic Modeling Techniques and Data Analysis*, Chania, Greece (pp. 8-11) (2010).

## See Also

[fittestWavelet](#), [fittestEMD](#)

Other transformation methods: [Diff\(\)](#), [LogT\(\)](#), [emd\(\)](#), [mas\(\)](#), [mlm\\_io\(\)](#), [outliers\\_bp\(\)](#), [pct\(\)](#), [train\\_test\\_subset\(\)](#)

## Examples

```
data(CATS)

w <- WaveletT(CATS[,1])

#plot wavelet transform/decomposition
plot(attr(w,"wt_obj"))

x <- WaveletT.rev(pred=NULL, attr(w,"wt_obj"))

all(round(x,4)==round(CATS[,1],4))
```

---

workflow

*Executing a time series prediction process*

---

## Description

workflow is a generic function for executing the steps of a particular data workflow. The function invokes particular *methods* which depend on the class of the first argument.

## Usage

```
workflow(obj, ...)

## S3 method for class 'tspred'
workflow(
  obj,
```



```

    data = NULL,
    prep_test = FALSE,
    onestep = obj$one_step,
    eval_fitness = TRUE,
    seed = 1234,
    ...
  )

```

### Arguments

obj	An object of class <code>tspred</code> defining a particular time series prediction process.
...	Ignored
data	See <code>subset.tspred</code>
prep_test	See <code>preprocess.tspred</code>
onestep	See <code>predict.tspred</code>
eval_fitness	See <code>evaluate.tspred</code>
seed	See <code>set.seed</code>

### Details

The function `workflow.tspred` executes a time series prediction process defined by a `tspred` object. It is a wrapper for the methods `subset.preprocess`, `train`, `predict`, `postprocess`, and `evaluate`, which are called in this order. The artifacts generated by the execution of the time series prediction process are introduced in the structure of the `tspred` class object in `obj`.

### Value

An object of class `tspred` with updated structure containing all artifacts generated by the execution of the time series prediction process.

### Author(s)

Rebecca Pontes Salles

### See Also

`[tspred()]` for defining a particular time series prediction process.

### Examples

```

data(CATS)

#Obtaining objects of the processing class
proc1 <- subsetting(test_len=20)
proc2 <- BoxCoxT(lambda=NULL)
proc3 <- WT(level=1, filter="b114")

#Obtaining objects of the modeling class

```

```
mod11 <- ARIMA()

#Obtaining objects of the evaluating class
eval1 <- MSE_eval()

#Defining a time series prediction process
tspred_1 <- tspred(subsetting=proc1,
                  processing=list(BCT=proc2,
                                 WT=proc3),
                  modeling=mod11,
                  evaluating=list(MSE=eval1)
)
summary(tspred_1)

tspred_1 <- workflow(tspred_1,data=CATS[3],onestep=TRUE)
```

# Index

## \* ARIMA

- arimainterp, 8
- arimaparameters, 9
- arimapred, 10
- fittestArima, 36
- fittestArimaKF, 38
- fittestLM, 43
- marimapar, 67
- marimapred, 68
- plotarimapred, 96

## \* Box-Cox

- BCT, 12

## \* CATS

- CATS, 16
- CATS.cont, 17

## \* Competition

- CATS, 16
- CATS.cont, 17
- EUNITE.Loads, 22
- EUNITE.Loads.cont, 24
- EUNITE.Reg, 28
- EUNITE.Reg.cont, 29
- EUNITE.Temp, 30
- EUNITE.Temp.cont, 31
- NN3.A, 78
- NN3.A.cont, 82
- NN5.A, 86
- NN5.A.cont, 90
- SantaFe.A, 106
- SantaFe.A.cont, 107
- SantaFe.D, 108
- SantaFe.D.cont, 109

## \* EUNITE

- EUNITE.Loads, 22
- EUNITE.Loads.cont, 24
- EUNITE.Reg, 28
- EUNITE.Reg.cont, 29
- EUNITE.Temp, 30
- EUNITE.Temp.cont, 31

## \* Fe

- SantaFe.A, 106
- SantaFe.A.cont, 107
- SantaFe.D, 108
- SantaFe.D.cont, 109

## \* Imfs

- emd, 20

## \* Kalman

- fittestArimaKF, 38
- fittestLM, 43
- fittestPolyRKF, 52

## \* MAPE

- MAPE, 66

## \* MSE

- MSE, 75

## \* NMSE

- NMSE, 77

## \* NN3

- NN3.A, 78
- NN3.A.cont, 82

## \* NN5

- NN5.A, 86
- NN5.A.cont, 90

## \* SMAPE

- sMAPE, 110

## \* Santa

- SantaFe.A, 106
- SantaFe.A.cont, 107
- SantaFe.D, 108
- SantaFe.D.cont, 109

## \* Series

- CATS, 16
- CATS.cont, 17
- EUNITE.Loads, 22
- EUNITE.Loads.cont, 24
- EUNITE.Reg, 28
- EUNITE.Reg.cont, 29
- EUNITE.Temp, 30
- EUNITE.Temp.cont, 31

- NN3.A, 78
  - NN3.A.cont, 82
  - NN5.A, 86
  - NN5.A.cont, 90
  - SantaFe.A, 106
  - SantaFe.A.cont, 107
  - SantaFe.D, 108
  - SantaFe.D.cont, 109
- \* **Time**
  - CATS, 16
  - CATS.cont, 17
  - EUNITE.Loads, 22
  - EUNITE.Loads.cont, 24
  - EUNITE.Reg, 28
  - EUNITE.Reg.cont, 29
  - EUNITE.Temp, 30
  - EUNITE.Temp.cont, 31
  - NN3.A, 78
  - NN3.A.cont, 82
  - NN5.A, 86
  - NN5.A.cont, 90
  - SantaFe.A, 106
  - SantaFe.A.cont, 107
  - SantaFe.D, 108
  - SantaFe.D.cont, 109
- \* **adjustment**
  - arimainterp, 8
  - arimapred, 10
  - fittestArima, 36
  - fittestArimaKF, 38
  - fittestEMD, 41
  - fittestLM, 43
  - fittestMAS, 46
  - fittestPolyR, 49
  - fittestPolyRKF, 52
  - fittestWavelet, 55
  - marimapar, 67
  - marimapred, 68
- \* **automatic**
  - arimainterp, 8
  - arimapred, 10
  - fittestArima, 36
  - fittestArimaKF, 38
  - fittestEMD, 41
  - fittestLM, 43
  - fittestMAS, 46
  - fittestPolyR, 49
  - fittestPolyRKF, 52
- fittestWavelet, 55
  - marimapar, 67
  - marimapred, 68
- \* **average**
  - fittestMAS, 46
  - mas, 70
- \* **benchmark**
  - benchmark, 14
- \* **change**
  - pct, 95
- \* **constructors**
  - ARIMA, 5
  - evaluating, 35
  - LT, 63
  - modeling, 73
  - MSE\_eval, 76
  - processing, 105
  - tspred, 117
- \* **criterion**
  - fittestArima, 36
  - fittestArimaKF, 38
  - fittestEMD, 41
  - fittestLM, 43
  - fittestMAS, 46
  - fittestPolyR, 49
  - fittestPolyRKF, 52
  - fittestWavelet, 55
- \* **datasets**
  - CATS, 16
  - CATS.cont, 17
  - EUNITE.Loads, 22
  - EUNITE.Loads.cont, 24
  - EUNITE.Reg, 28
  - EUNITE.Reg.cont, 29
  - EUNITE.Temp, 30
  - EUNITE.Temp.cont, 31
  - ipeadata\_d, 59
  - ipeadata\_m, 60
  - NN3.A, 78
  - NN3.A.cont, 82
  - NN5.A, 86
  - NN5.A.cont, 90
  - SantaFe.A, 106
  - SantaFe.A.cont, 107
  - SantaFe.D, 108
  - SantaFe.D.cont, 109
- \* **data**
  - train\_test\_subset, 116

- \* **decomposition**
  - emd, 20
  - fittestEMD, 41
  - fittestWavelet, 55
  - WaveletT, 118
- \* **detrending**
  - detrend, 18
- \* **differencing**
  - Diff, 19
- \* **emd**
  - emd, 20
  - fittestEMD, 41
- \* **errors**
  - fittestArima, 36
  - fittestArimaKF, 38
  - fittestEMD, 41
  - fittestLM, 43
  - fittestMAS, 46
  - fittestPolyR, 49
  - fittestPolyRKF, 52
  - fittestWavelet, 55
- \* **error**
  - MAPE, 66
  - MAXError, 71
  - MSE, 75
  - NMSE, 77
  - sMAPE, 110
- \* **evaluate**
  - benchmark, 14
  - evaluate, 32
  - evaluate.tspred, 33
  - tspred, 117
  - workflow, 120
- \* **evaluation**
  - evaluate, 32
  - evaluate.tspred, 33
  - evaluating, 35
  - fittestArima, 36
  - fittestArimaKF, 38
  - fittestEMD, 41
  - fittestLM, 43
  - fittestMAS, 46
  - fittestPolyR, 49
  - fittestPolyRKF, 52
  - fittestWavelet, 55
  - MSE\_eval, 76
- \* **filter**
  - fittestArimaKF, 38
  - fittestLM, 43
  - fittestPolyRKF, 52
- \* **fitting**
  - arimainterp, 8
  - arimapred, 10
  - fittestArima, 36
  - fittestArimaKF, 38
  - fittestEMD, 41
  - fittestLM, 43
  - fittestMAS, 46
  - fittestPolyR, 49
  - fittestPolyRKF, 52
  - fittestWavelet, 55
  - marimapar, 67
  - marimapred, 68
- \* **interpolation**
  - arimainterp, 8
- \* **ipeadata**
  - ipeadata\_d, 59
  - ipeadata\_m, 60
- \* **linear**
  - fittestLM, 43
- \* **logarithm**
  - LogT, 62
- \* **maximal**
  - MAXError, 71
- \* **meaningful**
  - emd, 20
- \* **method**
  - ARIMA, 5
  - modeling, 73
- \* **metric**
  - evaluate, 32
  - evaluate.tspred, 33
  - evaluating, 35
  - MSE\_eval, 76
- \* **modeling**
  - ARIMA, 5
  - modeling, 73
- \* **model**
  - ARIMA, 5
  - benchmark, 14
  - fittestLM, 43
  - modeling, 73
  - train, 113
  - train.tspred, 114

- tspred, 117
  - workflow, 120
- \* **moving**
  - fittestMAS, 46
  - mas, 70
- \* **normalization methods**
  - an, 4
  - minmax, 72
- \* **normalization**
  - an, 4
  - minmax, 72
- \* **outlier**
  - outliers\_bp, 94
- \* **package**
  - TSPred-package, 3
- \* **parameters**
  - arimaparameters, 9
  - marimapar, 67
- \* **percentage**
  - pct, 95
- \* **plot**
  - plotarimapred, 96
- \* **polynomial**
  - fittestLM, 43
  - fittestPolyR, 49
  - fittestPolyRKF, 52
- \* **postprocessing**
  - LT, 63
  - postprocess.tspred, 98
  - preprocess, 102
  - preprocess.tspred, 103
  - processing, 105
  - subset, 111
- \* **prediction**
  - ARIMA, 5
  - arimainterp, 8
  - arimapred, 10
  - benchmark, 14
  - fittestArima, 36
  - fittestArimaKF, 38
  - fittestEMD, 41
  - fittestLM, 43
  - fittestMAS, 46
  - fittestPolyR, 49
  - fittestPolyRKF, 52
  - fittestWavelet, 55
  - MAPE, 66
  - marimapred, 68
  - MAXError, 71
  - modeling, 73
  - MSE, 75
  - NMSE, 77
  - plotarimapred, 96
  - predict, 99
  - predict.tspred, 101
  - sMAPE, 110
  - tspred, 117
  - workflow, 120
- \* **predict**
  - predict, 99
  - predict.tspred, 101
- \* **preprocessing**
  - LT, 63
  - postprocess.tspred, 98
  - preprocess, 102
  - preprocess.tspred, 103
  - processing, 105
  - subset, 111
- \* **preprocess**
  - benchmark, 14
  - postprocess.tspred, 98
  - preprocess.tspred, 103
  - subset, 111
  - tspred, 117
  - workflow, 120
- \* **processing**
  - LT, 63
  - postprocess.tspred, 98
  - preprocess, 102
  - preprocess.tspred, 103
  - processing, 105
- \* **quality**
  - evaluate, 32
  - evaluate.tspred, 33
  - evaluating, 35
  - MSE\_eval, 76
- \* **regression**
  - fittestLM, 43
  - fittestPolyR, 49
  - fittestPolyRKF, 52
- \* **removal**
  - outliers\_bp, 94
- \* **series**
  - an, 4
  - BCT, 12
  - detrend, 18

- Diff, 19
  - fittestEMD, 41
  - fittestMAS, 46
  - fittestWavelet, 55
  - LogT, 62
  - mas, 70
  - minmax, 72
  - outliers\_bp, 94
  - pct, 95
  - predict, 99
  - predict.tspred, 101
  - sw, 112
  - \* **sliding**
    - sw, 112
  - \* **smoother**
    - fittestMAS, 46
    - mas, 70
  - \* **subsets**
    - train\_test\_subset, 116
  - \* **subsetting**
    - subset, 111
  - \* **teries**
    - ipeadata\_d, 59
    - ipeadata\_m, 60
  - \* **test**
    - train\_test\_subset, 116
  - \* **time**
    - an, 4
    - BCT, 12
    - detrend, 18
    - Diff, 19
    - fittestEMD, 41
    - fittestMAS, 46
    - fittestWavelet, 55
    - ipeadata\_d, 59
    - ipeadata\_m, 60
    - LogT, 62
    - mas, 70
    - minmax, 72
    - outliers\_bp, 94
    - pct, 95
    - predict, 99
    - predict.tspred, 101
    - sw, 112
  - \* **training**
    - train, 113
    - train.tspred, 114
  - \* **train**
    - train, 113
    - train.tspred, 114
    - train\_test\_subset, 116
  - \* **transformation methods**
    - Diff, 19
    - emd, 20
    - LogT, 62
    - mas, 70
    - outliers\_bp, 94
    - pct, 95
    - train\_test\_subset, 116
    - WaveletT, 118
  - \* **transformation**
    - LT, 63
    - postprocess.tspred, 98
    - preprocess, 102
    - preprocess.tspred, 103
    - processing, 105
  - \* **transform**
    - BCT, 12
    - detrend, 18
    - Diff, 19
    - emd, 20
    - fittestEMD, 41
    - fittestWavelet, 55
    - LogT, 62
    - mas, 70
    - pct, 95
    - WaveletT, 118
  - \* **trend**
    - detrend, 18
  - \* **tspred**
    - benchmark, 14
    - subset, 111
    - tspred, 117
    - workflow, 120
  - \* **wavelet**
    - fittestWavelet, 55
    - WaveletT, 118
  - \* **windows**
    - sw, 112
  - \* **workflow**
    - workflow, 120
- AIC, 36, 39, 42, 45, 47, 50, 53, 57
- AIC\_eval (MSE\_eval), 76
- AICc, 39, 42, 45, 47, 50, 53, 57
- AICc\_eval (MSE\_eval), 76
- AN (LT), 63

- an, [4](#), [64](#), [65](#), [73](#)
- an.rev, [65](#)
- ARIMA, [5](#), [35](#), [65](#), [74](#), [77](#), [105](#), [118](#)
- Arima, [10](#)
- arima, [10](#), [46](#), [55](#)
- arimainterp, [8](#)
- arimaparameters, [9](#), [37](#)
- arimapred, [8–10](#), [10](#), [67–69](#), [97](#)
- artransform, [39](#)
- auto.arima, [7](#), [10](#), [12](#), [36](#), [37](#), [39](#), [47](#), [57](#)
  
- BCT, [12](#), [18](#), [64](#), [65](#)
- BCT.rev, [65](#)
- benchmark, [14](#)
- benchmark.tspred, [14](#)
- BIC, [36](#), [39](#), [42](#), [45](#), [47](#), [50](#), [53](#), [57](#)
- BIC\_eval (MSE\_eval), [76](#)
- BoxCox, [12](#)
- BoxCox.lambda, [13](#)
- BoxCoxT (LT), [63](#)
  
- CATS, [16](#), [17](#)
- CATS.cont, [16](#), [17](#)
  
- detrend, [13](#), [18](#)
- DIF, [13](#), [18](#)
- DIFF (LT), [63](#)
- Diff, [19](#), [22](#), [63–65](#), [71](#), [95](#), [96](#), [116](#), [120](#)
- diff, [19](#), [20](#)
- Diff.rev, [65](#)
- diffinv, [19](#)
- dredge, [50](#)
  
- ELM (ARIMA), [5](#)
- elm\_predict, [7](#)
- elm\_train, [7](#)
- EMD (LT), [63](#)
- emd, [20](#), [20](#), [21](#), [42](#), [63–65](#), [71](#), [95](#), [96](#), [116](#), [120](#)
- emd.rev, [42](#), [65](#)
- error (evaluating), [35](#)
- ETS (ARIMA), [5](#)
- ets, [7](#), [46](#), [47](#), [55](#), [57](#)
- EUNITE.Loads, [22](#), [24–31](#)
- EUNITE.Loads.cont, [24](#), [24](#), [28](#), [30](#)
- EUNITE.Reg, [24](#), [27](#), [28](#), [28](#), [29–31](#)
- EUNITE.Reg.cont, [28](#), [29](#)
- EUNITE.Temp, [24](#), [27–29](#), [30](#), [31](#)
- EUNITE.Temp.cont, [30](#), [31](#)
- evaluate, [32](#), [33](#), [34](#), [121](#)
- evaluate.tspred, [33](#), [33](#), [121](#)
- evaluating, [7](#), [32](#), [33](#), [35](#), [65](#), [74](#), [77](#), [105](#), [117](#), [118](#)
  
- fitness (evaluating), [35](#)
- fitSSM, [53](#)
- fittestArima, [10](#), [36](#), [40](#), [43–45](#)
- fittestArimaKF, [37](#), [38](#), [43–45](#)
- fittestEMD, [20–22](#), [41](#), [48](#), [58](#), [120](#)
- fittestLM, [37](#), [40](#), [43](#), [51](#), [54](#)
- fittestMAS, [43](#), [46](#), [58](#), [70](#)
- fittestPolyR, [43–45](#), [49](#), [54](#)
- fittestPolyRKF, [43–45](#), [51](#), [52](#)
- fittestWavelet, [22](#), [43](#), [48](#), [55](#), [119](#), [120](#)
- forecast, [7](#), [36](#), [37](#), [47](#), [48](#), [56](#), [58](#)
- forecast.Arima, [97](#)
  
- HW (ARIMA), [5](#)
- hw, [7](#)
  
- imodwt, [57](#), [119](#)
- InvBoxCox, [12](#), [13](#)
- ipeadata\_d, [59](#), [62](#)
- ipeadata\_m, [60](#), [60](#)
  
- KFAS, [39](#), [53](#)
  
- linear (modeling), [73](#)
- logLik, [36](#), [39](#), [42](#), [45](#), [47](#), [50](#), [53](#), [57](#)
- LogLik\_eval (MSE\_eval), [76](#)
- LogT, [20](#), [22](#), [62](#), [64](#), [65](#), [71](#), [95](#), [96](#), [116](#), [120](#)
- LogT.rev, [65](#)
- LT, [7](#), [13](#), [18](#), [35](#), [63](#), [74](#), [77](#), [105](#), [118](#)
  
- ma, [70](#)
- MAPE, [36](#), [39](#), [42](#), [45](#), [47](#), [50](#), [53](#), [57](#), [66](#), [72](#), [76](#), [78](#), [110](#)
- MAPE\_eval (MSE\_eval), [76](#)
- marimapar, [67](#)
- marimapred, [9](#), [12](#), [37](#), [40](#), [67](#), [68](#)
- MAS, [13](#), [18](#)
- MAS (LT), [63](#)
- mas, [20](#), [22](#), [63–65](#), [70](#), [95](#), [96](#), [116](#), [120](#)
- mas.rev, [47](#), [65](#)
- MAXError, [36](#), [39](#), [42](#), [45](#), [47](#), [50](#), [53](#), [57](#), [66](#), [71](#), [76](#), [78](#), [110](#)
- MAXError\_eval (MSE\_eval), [76](#)
- MinMax (LT), [63](#)
- minmax, [5](#), [65](#), [72](#)
- minmax.rev, [65](#)



- MLM (modeling), 73
- mIm\_io, 20, 22, 63, 71, 95, 96, 116, 120
- MLP (ARIMA), 5
- mIp, 7
- modeling, 7, 35, 65, 73, 77, 99–101, 105, 113–115, 117, 118
- modwt, 56–58, 119
- MSE, 36, 39, 42, 45, 47, 50, 53, 57, 66, 75, 78, 110
- MSE\_eval, 7, 35, 65, 74, 76, 105, 118
- na.omit, 11, 36, 39, 42, 44, 47, 50, 52, 56, 59, 61, 67, 68
- NAS (LT), 63
- ndiffs, 19, 20
- NMSE, 36, 39, 42, 45, 47, 50, 53, 57, 66, 76, 77, 110
- NMSE\_eval (MSE\_eval), 76
- NN3.A, 78, 82–86
- NN3.A.cont, 81, 82, 82
- NN5.A, 86, 90–94
- NN5.A.cont, 89, 90, 90
- NNET (ARIMA), 5
- nnet, 7
- nsdiffs, 19, 20
- outliers\_bp, 20, 22, 63, 71, 94, 96, 116, 120
- PCT, 13, 18
- PCT (LT), 63
- pct, 20, 22, 63, 65, 71, 95, 95, 116, 120
- pct.rev, 65
- plot.forecast, 97
- plotarimapred, 10, 12, 96
- poly, 50
- postprocess, 98, 121
- postprocess (preprocess), 102
- postprocess.tspred, 98, 98, 104, 111
- predict, 7, 42, 99, 101, 121
- predict.Arima, 9, 12, 69
- predict.lm, 42, 50, 51
- predict.SSModel, 39, 40, 52, 54
- predict.tspred, 100, 101, 101, 121
- preprocess, 102, 103, 104, 111, 121
- preprocess.tspred, 98, 99, 103, 104, 111, 121
- processing, 7, 35, 65, 74, 77, 98, 102–104, 105, 117, 118
- randomForest, 7
- RBF (ARIMA), 5
- rbf, 7
- RFrst (ARIMA), 5
- RMSE\_eval (MSE\_eval), 76
- SantaFe.A, 106, 107–109
- SantaFe.A.cont, 106, 107, 108, 109
- SantaFe.D, 106, 107, 108, 109
- SantaFe.D.cont, 106–108, 109
- set.seed, 121
- sMAPE, 36, 39, 42, 45, 47, 50, 53, 57, 66, 72, 76, 78, 110
- sMAPE\_eval (MSE\_eval), 76
- SSMarima, 39
- SSModel, 39, 53
- stlf, 46, 47
- subset, 99, 104, 111, 121
- subset.tspred, 111, 121
- subsetting, 111, 117
- subsetting (LT), 63
- SVM (ARIMA), 5
- SW, 7, 74
- SW (LT), 63
- sw, 4, 64, 65, 72, 94, 112
- Tensor\_CNN (ARIMA), 5
- Tensor\_LSTM (ARIMA), 5
- TF (ARIMA), 5
- thetaf, 7
- train, 113, 115, 121
- train.tspred, 114, 114, 115
- train\_test\_subset, 20, 22, 63–65, 71, 95, 96, 116, 120
- tspred, 7, 14, 33, 35, 65, 74, 77, 98, 101, 103–105, 111, 114, 115, 117, 121
- TSPred-package, 3
- tune.svm, 7
- WaveletT, 20, 22, 63–65, 71, 95, 96, 116, 118
- WaveletT.rev, 65
- workflow, 120
- workflow.tspred, 121
- WT (LT), 63
- wt.filter, 56