

# Package ‘TSrepr’

July 13, 2020

**Type** Package

**Title** Time Series Representations

**Version** 1.1.0

**Date** 2020-07-12

**Description** Methods for representations (i.e. dimensionality reduction, preprocessing, feature extraction) of time series to help more accurate and effective time series data mining.

Non-data adaptive, data adaptive, model-based and data dictated (clipped) representation methods are implemented. Also various normalisation methods (min-max, z-score, Box-Cox, Yeo-Johnson), and forecasting accuracy measures are implemented.

**License** GPL-3 | file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 2.10)

**Imports** Rcpp (>= 0.12.12), MASS, quantreg, wavelets, mgcv, dtt

**LinkingTo** Rcpp

**RoxygenNote** 7.1.0

**URL** <https://petolau.github.io/package/>,  
<https://github.com/PetoLau/TSrepr/>

**BugReports** <https://github.com/PetoLau/TSrepr/issues>

**Suggests** knitr, rmarkdown, ggplot2, data.table, moments, testthat

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Peter Laurinec [aut, cre] (<<https://orcid.org/0000-0002-3501-8783>>)

**Maintainer** Peter Laurinec <[tsreprpackage@gmail.com](mailto:tsreprpackage@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-07-13 06:50:15 UTC

**R topics documented:**

clipping	3
coefComp	4
denorm_atan	5
denorm_boxcox	5
denorm_min_max	6
denorm_yj	7
denorm_z	8
elec_load	9
fast_stat	9
maape	10
mae	11
mape	11
mase	12
mdae	13
mse	13
norm_atan	14
norm_boxcox	15
norm_min_max	15
norm_min_max_list	16
norm_min_max_params	17
norm_yj	18
norm_z	18
norm_z_list	19
norm_z_params	20
repr_dct	21
repr_dft	22
repr_dwt	23
repr_exp	24
repr_feaclip	25
repr_feacliptrend	26
repr_featrend	27
repr_gam	28
repr_list	29
repr_lm	31
repr_matrix	32
repr_paa	34
repr_pip	35
repr_pla	36
repr_sax	37
repr_seas_profile	38
repr_sma	39
repr_windowing	39
rleC	41
rmse	41
smape	42
trending	43

<i>clipping</i>	3
TSrepr . . . . .	44
<b>Index</b>	<b>46</b>

`clipping`                      *Creates bit-level (clipped representation) from a vector*

**Description**

The `clipping` computes bit-level (clipped representation) from a vector.

**Usage**

`clipping(x)`

**Arguments**

`x`                      the numeric vector (time series)

**Details**

Clipping transforms time series to bit-level representation.

It is defined as follows:

$$repr_t = 1 \text{ if } x_t > \mu, 0 \text{ otherwise,}$$

where  $x_t$  is a value of a time series and  $\mu$  is average of a time series.

**Value**

the integer vector of zeros and ones

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Bagnall A, Ratanamahatana C, Keogh E, Lonardi S, Janacek G (2006) A bit level representation for time series data mining with shape based similarity. *Data Mining and Knowledge Discovery* 13(1):11-40

Laurinec P, and Lucka M (2018) Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. *Data Mining and Knowledge Discovery*. Springer. DOI: 10.1007/s10618-018-0598-2

**See Also**

[trending](#)

**Examples**

```
clipping(rnorm(50))
```

---

`coefComp`*Functions for linear regression model coefficients extraction*

---

**Description**

The functions computes regression coefficients from a linear model.

**Usage**

```
lmCoef(X, Y)
```

```
r1mCoef(X, Y)
```

```
l1Coef(X, Y)
```

**Arguments**

X                    the model (design) matrix of independent variables  
Y                    the vector of dependent variable (time series)

**Value**

The numeric vector of regression coefficients

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[lm](#), [r1m](#), [rq](#)

**Examples**

```
design_matrix <- matrix(rnorm(10), ncol = 2)  
lmCoef(design_matrix, rnorm(5))  
  
r1mCoef(design_matrix, rnorm(5))  
  
l1Coef(design_matrix, rnorm(5))
```

---

denorm_atan	<i>Arctangent denormalisation</i>
-------------	-----------------------------------

---

**Description**

The denorm\_atan denormalises time series from Arctangent function.

**Usage**

```
denorm_atan(x)
```

**Arguments**

x                    the numeric vector (time series)

**Value**

the numeric vector of denormalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[denorm\\_z](#), [denorm\\_min\\_max](#)

**Examples**

```
denorm_atan(runif(50))
```

---

denorm_boxcox	<i>Two-parameter Box-Cox denormalisation</i>
---------------	--

---

**Description**

The denorm\_boxcox denormalises time series by two-parameter Box-Cox method.

**Usage**

```
denorm_boxcox(x, lambda = 0.1, gamma = 0)
```

**Arguments**

x	the numeric vector (time series) to be denormalised
lambda	the numeric value - power transformation parameter (default is 0.1)
gamma	the non-negative numeric value - parameter for holding the time series positive (offset) (default is 0)

**Value**

the numeric vector of denormalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[denorm\\_z](#), [denorm\\_min\\_max](#), [denorm\\_atan](#)

**Examples**

```
denorm_boxcox(runif(50))
```

---

denorm_min_max	<i>Min-Max denormalisation</i>
----------------	--------------------------------

---

**Description**

The denorm\_min\_max denormalises time series by min-max method.

**Usage**

```
denorm_min_max(x, min, max)
```

**Arguments**

x	the numeric vector (time series)
min	the minimum value
max	the maximal value

**Value**

the numeric vector of denormalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

## References

Laurinec P, Lucká M (2018) Clustering-based forecasting method for individual consumers electricity load using time series representations. *Open Comput Sci*, 8(1):38–50, DOI: 10.1515/comp-2018-0006

## See Also

[norm\\_min\\_max](#), [norm\\_min\\_max\\_list](#)

## Examples

```
# Normalise values and save normalisation parameters:
norm_res <- norm_min_max_list(rnorm(50, 5, 2))
# Denormalise new data with previous computed parameters:
denorm_min_max(rnorm(50, 4, 2), min = norm_res$min, max = norm_res$max)
```

---

denorm\_yj

*Yeo-Johnson denormalisation*

---

## Description

The denorm\_yj denormalises time series by Yeo-Johnson method

## Usage

```
denorm_yj(x, lambda = 0.1)
```

## Arguments

x                    the numeric vector (time series) to be denormalised  
lambda                the numeric value - power transformation parameter (default is 0.1)

## Value

the numeric vector of denormalised values

## Author(s)

Peter Laurinec, <tsreprpackage@gmail.com>

## See Also

[denorm\\_z](#), [denorm\\_min\\_max](#), [denorm\\_boxcox](#)

## Examples

```
denorm_yj(runif(50))
```

---

denorm\_z                      *Z-score denormalisation*

---

### Description

The denorm\_z denormalises time series by z-score method.

### Usage

```
denorm_z(x, mean, sd)
```

### Arguments

x	the numeric vector (time series)
mean	the mean value
sd	the standard deviation value

### Value

the numeric vector of denormalised values

### Author(s)

Peter Laurinec, <tsreprpackage@gmail.com>

### References

Laurinec P, Lucká M (2018) Clustering-based forecasting method for individual consumers electricity load using time series representations. *Open Comput Sci*, 8(1):38–50, DOI: 10.1515/comp-2018-0006

### See Also

[norm\\_z](#), [norm\\_z\\_list](#)

### Examples

```
# Normalise values and save normalisation parameters:
norm_res <- norm_z_list(rnorm(50, 5, 2))
# Denormalise new data with previous computed parameters:
denorm_z(rnorm(50, 4, 2), mean = norm_res$mean, sd = norm_res$sd)
```



---

elec_load	<i>2 weeks of electricity load data from 50 consumers.</i>
-----------	--

---

**Description**

A dataset containing the electricity consumption time series from 50 consumers of the length of 2 weeks. Every day is 48 measurements (half-hourly data). Each row represents one consumers time series.

**Usage**

```
elec_load
```

**Format**

A data frame with 50 rows and 672 variables.

**Source**

Anonymized.

---

fast_stat	<i>Fast statistic functions (helpers)</i>
-----------	---

---

**Description**

Fast statistic functions (helpers) for representations computation.

**Usage**

```
maxC(x)
```

```
minC(x)
```

```
meanC(x)
```

```
sumC(x)
```

```
medianC(x)
```

**Arguments**

x                    the numeric vector

**Value**

the numeric value

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
maxC(rnorm(50))
```

```
minC(rnorm(50))
```

```
meanC(rnorm(50))
```

```
sumC(rnorm(50))
```

```
medianC(rnorm(50))
```

---

maape

*MAAPE*

---

**Description**

the maape computes MAAPE (Mean Arctangent Absolute Percentage Error) of a forecast.

**Usage**

```
maape(x, y)
```

**Arguments**

x                   the numeric vector of real values

y                   the numeric vector of forecasted values

**Value**

the numeric value in %

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Sungil Kim, Heeyoung Kim (2016) A new metric of absolute percentage error for intermittent demand forecasts, *International Journal of Forecasting* 32(3):669-679

**Examples**

```
maape(runif(50), runif(50))
```

---

mae	<i>MAE</i>
-----	------------

---

**Description**

The mae computes MAE (Mean Absolute Error) of a forecast.

**Usage**

```
mae(x, y)
```

**Arguments**

x	the numeric vector of real values
y	the numeric vector of forecasted values

**Value**

the numeric value

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
mae(runif(50), runif(50))
```

---

mape	<i>MAPE</i>
------	-------------

---

**Description**

the mape computes MAPE (Mean Absolute Percentage Error) of a forecast.

**Usage**

```
mape(x, y)
```

**Arguments**

x	the numeric vector of real values
y	the numeric vector of forecasted values

**Value**

the numeric value in %

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
mase(runif(50), runif(50))
```

---

mase

*MASE*

---

**Description**

The mase computes MASE (Mean Absolute Scaled Error) of a forecast.

**Usage**

```
mase(real, forecast, naive)
```

**Arguments**

real	the numeric vector of real values
forecast	the numeric vector of forecasted values
naive	the numeric vector of naive forecast

**Value**

the numeric value

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
mase(rnorm(50), rnorm(50), rnorm(50))
```

---

mdae	<i>MdAE</i>
------	-------------

---

**Description**

The mdae computes MdAE (Median Absolute Error) of a forecast.

**Usage**

```
mdae(x, y)
```

**Arguments**

x	the numeric vector of real values
y	the numeric vector of forecasted values

**Value**

the numeric value

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
mdae(runif(50), runif(50))
```

---

mse	<i>MSE</i>
-----	------------

---

**Description**

The mse computes MSE (Mean Squared Error) of a forecast.

**Usage**

```
mse(x, y)
```

**Arguments**

x	the numeric vector of real values
y	the numeric vector of forecasted values

**Value**

the numeric value

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
mse(runif(50), runif(50))
```

---

norm\_atan

*Arctangent normalisation*

---

**Description**

The norm\_atan normalises time series by Arctangent to max (-1,1) range.

**Usage**

```
norm_atan(x)
```

**Arguments**

x                    the numeric vector (time series)

**Value**

the numeric vector of normalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[norm\\_z](#), [norm\\_min\\_max](#)

**Examples**

```
norm_atan(rnorm(50))
```

---

norm_boxcox	<i>Two-parameter Box-Cox normalisation</i>
-------------	--

---

**Description**

The norm\_boxcox normalises time series by two-parameter Box-Cox normalisation.

**Usage**

```
norm_boxcox(x, lambda = 0.1, gamma = 0)
```

**Arguments**

x	the numeric vector (time series)
lambda	the numeric value - power transformation parameter (default is 0.1)
gamma	the non-negative numeric value - parameter for holding the time series positive (offset) (default is 0)

**Value**

the numeric vector of normalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[norm\\_z](#), [norm\\_min\\_max](#), [norm\\_atan](#)

**Examples**

```
norm_boxcox(runif(50))
```

---

norm_min_max	<i>Min-Max normalisation</i>
--------------	------------------------------

---

**Description**

The norm\_min\_max normalises time series by min-max method.

**Usage**

```
norm_min_max(x)
```

**Arguments**

x                    the numeric vector (time series)

**Value**

the numeric vector of normalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[norm\\_z](#)

**Examples**

```
norm_min_max(rnorm(50))
```

---

norm\_min\_max\_list            *Min-Max normalization list*

---

**Description**

The `norm_min_max_list` normalises time series by min-max method and returns normalization parameters (min and max).

**Usage**

```
norm_min_max_list(x)
```

**Arguments**

x                    the numeric vector (time series)

**Value**

the list composed of:

**norm\_values** the numeric vector of normalised values of time series

**min** the min value

**max** the max value

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>



**See Also**[norm\\_z\\_list](#)**Examples**

```
norm_min_max_list(rnorm(50))
```

---

norm\_min\_max\_params     *Min-Max normalisation with parameters*

---

**Description**

The `norm_min_max_params` normalises time series by min-max method with defined parameters.

**Usage**

```
norm_min_max_params(x, min, max)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
<code>min</code>	the numeric value
<code>max</code>	the numeric value

**Value**

the numeric vector of normalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**[norm\\_z\\_params](#)**Examples**

```
norm_min_max_params(rnorm(50), 0, 1)
```

---

norm_yj	<i>Yeo-Johnson normalisation</i>
---------	----------------------------------

---

**Description**

The `norm_yj` normalises time series by Yeo-Johnson normalisation.

**Usage**

```
norm_yj(x, lambda = 0.1)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
<code>lambda</code>	the numeric value - power transformation parameter (default is 0.1)

**Value**

the numeric vector of normalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[norm\\_z](#), [norm\\_min\\_max](#), [norm\\_boxcox](#)

**Examples**

```
norm_yj(runif(50))
```

---

norm_z	<i>Z-score normalisation</i>
--------	------------------------------

---

**Description**

The `norm_z` normalises time series by z-score.

**Usage**

```
norm_z(x)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
----------------	----------------------------------

**Value**

the numeric vector of normalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[norm\\_min\\_max](#)

**Examples**

```
norm_z(runif(50))
```

---

<code>norm_z_list</code>	<i>Z-score normalization list</i>
--------------------------	-----------------------------------

---

**Description**

The `norm_z_list` normalizes time series by z-score and returns normalization parameters (mean and standard deviation).

**Usage**

```
norm_z_list(x)
```

**Arguments**

`x` the numeric vector (time series)

**Value**

the list composed of:

**norm\_values** the numeric vector of normalised values of time series

**mean** the mean value

**sd** the standard deviation

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[norm\\_min\\_max\\_list](#)

**Examples**

```
norm_z_list(runif(50))
```

---

norm\_z\_params

*Z-score normalisation with parameters*

---

**Description**

The norm\_z\_params normalises time series by z-score with defined mean and standard deviation.

**Usage**

```
norm_z_params(x, mean, sd)
```

**Arguments**

x	the numeric vector (time series)
mean	the numeric value
sd	the numeric value - standard deviation

**Value**

the numeric vector of normalised values

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[norm\\_min\\_max\\_params](#)

**Examples**

```
norm_z_params(runif(50), 0.5, 1)
```

---

repr_dct	<i>DCT representation</i>
----------	---------------------------

---

**Description**

The `repr_dct` computes DCT (Discrete Cosine Transform) representation from a time series.

**Usage**

```
repr_dct(x, coef = 10)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
<code>coef</code>	the number of coefficients to extract from DCT

**Details**

The length of the final time series representation is equal to set `coef` parameter.

**Value**

the numeric vector of DCT coefficients

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[repr\\_dft](#), [repr\\_dwt](#), [dtt](#)

**Examples**

```
repr_dct(rnorm(50), coef = 4)
```

---

repr_dft	<i>DFT representation by FFT</i>
----------	----------------------------------

---

**Description**

The `repr_dft` computes DFT (Discrete Fourier Transform) representation from a time series by FFT (Fast Fourier Transform).

**Usage**

```
repr_dft(x, coef = 10)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
<code>coef</code>	the number of coefficients to extract from FFT

**Details**

The length of the final time series representation is equal to set `coef` parameter.

**Value**

the numeric vector of DFT coefficients

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[repr\\_dwt](#), [repr\\_dct](#), [fft](#)

**Examples**

```
repr_dft(rnorm(50), coef = 4)
```

---

repr_dwt	<i>DWT representation</i>
----------	---------------------------

---

## Description

The `repr_dwt` computes DWT (Discrete Wavelet Transform) representation (coefficients) from a time series.

## Usage

```
repr_dwt(x, level = 4, filter = "d4")
```

## Arguments

<code>x</code>	the numeric vector (time series)
<code>level</code>	the level of DWT transformation (default is 4)
<code>filter</code>	the filter name (default is "d6"). Can be: "haar", "d4", "d6", ..., "d20", "la8", "la10", ..., "la20", "bl14", "bl18", "bl20", "c6", "c12", ..., "c30". See more info at <a href="#">wt.filter</a> .

## Details

This function extracts DWT coefficients. You can use various wavelet filters, see all of them here [wt.filter](#). The number of extracted coefficients depends on the `level` selected. The final representation has length equal to  $\text{floor}(n / 2^{\text{level}})$ , where  $n$  is a length of original time series.

## Value

the numeric vector of DWT coefficients

## Author(s)

Peter Laurinec, <tsreprpackage@gmail.com>

## References

Laurinec P, Lucka M (2016) Comparison of representations of time series for clustering smart meter data. In: Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2016, pp 458-463

## See Also

[repr\\_dft](#), [repr\\_dct](#), [dwt](#)

**Examples**

```
# Interpretation: DWT with Daubechies filter of length 4 and
# 3rd level of DWT coefficients extracted.
repr_dwt(rnorm(50), filter = "d4", level = 3)
```

repr\_exp

*Exponential smoothing seasonal coefficients as representation***Description**

The repr\_exp computes exponential smoothing seasonal coefficients.

**Usage**

```
repr_exp(x, freq, alpha = TRUE, gamma = TRUE)
```

**Arguments**

x	the numeric vector (time series)
freq	the frequency of the time series
alpha	the smoothing factor (default is TRUE - automatic determination of smoothing factor), or number between 0 to 1
gamma	the seasonal smoothing factor (default is TRUE - automatic determination of seasonal smoothing factor), or number between 0 to 1

**Details**

This function extracts exponential smoothing seasonal coefficients and uses them as time series representation. You can set smoothing factors (alpha, gamma) manually, but recommended is automatic method (set to TRUE). The trend component is not included in computations.

**Value**

the numeric vector of seasonal coefficients

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Laurinec P, Lucka M (2016) Comparison of representations of time series for clustering smart meter data. In: Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2016, pp 458-463

Laurinec P, Loderer M, Vrablcova P, Lucka M, Rozinajova V, Ezzeddine AB (2016) Adaptive time series forecasting of energy consumption using optimized cluster analysis. In: Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on, IEEE, pp 398-405



**See Also**

[repr\\_lm](#), [repr\\_gam](#), [repr\\_seas\\_profile](#), [HoltWinters](#)

**Examples**

```
repr_exp(rnorm(96), freq = 24)
```

---

repr_feaclip	<i>FeaClip representation of time series</i>
--------------	--

---

**Description**

The `repr_feaclip` computes representation of time series based on feature extraction from bit-level (clipped) representation.

**Usage**

```
repr_feaclip(x)
```

**Arguments**

`x` the numeric vector (time series)

**Details**

FeaClip is method of time series representation based on feature extraction from run lengths (RLE) of bit-level (clipped) representation. It extracts 8 key features from clipped representation.

There are as follows:

$$\begin{aligned} repr = \{ &max_1 - max.fromrunlengthsofones, \\ &sum_1 - sumofrunlengthsofones, \\ &max_0 - max.fromrunlengthsofzeros, \\ &crossings - lengthofRLEencoding - 1, \\ &f_0 - numberoffirstzeros, \\ &l_0 - numberoflastzeros, \\ &f_1 - numberoffirstones, \\ &l_1 - numberoflastones\}. \end{aligned}$$

**Value**

the numeric vector of length 8

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Laurinec P, and Lucka M (2018) Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. *Data Mining and Knowledge Discovery*. Springer. DOI: 10.1007/s10618-018-0598-2

**See Also**

[repr\\_featrend](#), [repr\\_feacliptrend](#)

**Examples**

```
repr_feaclip(rnorm(50))
```

---

repr_feacliptrend	<i>FeaClipTrend</i> representation of time series
-------------------	---

---

**Description**

The `repr_feacliptrend` computes representation of time series based on feature extraction from bit-level representations (clipping and trending).

**Usage**

```
repr_feacliptrend(x, func, pieces = 2L, order = 4L)
```

**Arguments**

x	the numeric vector (time series)
func	the aggregation function for FeaTrend procedure (sumC or maxC)
pieces	the number of parts of time series to split
order	the order of simple moving average

**Details**

`FeaClipTrend` combines `FeaClip` and `FeaTrend` representation methods. See documentation of these two methods (check `See Also` section).

**Value**

the numeric vector of frequencies of features

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Laurinec P, and Lucka M (2018) Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. Data Mining and Knowledge Discovery. Springer. DOI: 10.1007/s10618-018-0598-2

**See Also**

[repr\\_featrend](#), [repr\\_feaclip](#)

**Examples**

```
repr_feacliptrend(rnorm(50), maxC, 2, 4)
```

---

repr_featrend	<i>FeaTrend representation of time series</i>
---------------	---

---

**Description**

The `repr_featrend` computes representation of time series based on feature extraction from bit-level (trending) representation.

**Usage**

```
repr_featrend(x, func, pieces = 2L, order = 4L)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
<code>func</code>	the function of aggregation, can be <code>sumC</code> or <code>maxC</code> or similar aggregation function
<code>pieces</code>	the number of parts of time series to split (default to 2)
<code>order</code>	the order of simple moving average (default to 4)

**Details**

FeaTrend is method of time series representation based on feature extraction from run lengths (RLE) of bit-level (trending) representation. It extracts number of features from trending representation based on number of pieces defined. From every piece, 2 features are extracted. You can define what feature will be extracted, recommended functions are `max` and `sum`. For example if `max` is selected, then maximum value of run lengths of ones and zeros are extracted.

**Value**

the numeric vector of the length pieces

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[repr\\_feaclip](#), [repr\\_feacliptrend](#)

**Examples**

```
# default settings
repr_featrend(rnorm(50), maxC)

# compute FeaTrend for 4 pieces and make more smoothed ts by order = 8
repr_featrend(rnorm(50), sumC, 4, 8)
```

---

repr\_gam

*GAM regression coefficients as representation*


---

**Description**

The repr\_gam computes seasonal GAM regression coefficients. Additional exogenous variables can be also added.

**Usage**

```
repr_gam(x, freq = NULL, xreg = NULL)
```

**Arguments**

x	the numeric vector (time series)
freq	the frequency of the time series. Can be vector of two frequencies (seasonalities) or just an integer of one frequency.
xreg	the numeric vector or the data.frame with additional exogenous regressors

**Details**

This model-based representation method extracts regression coefficients from a GAM (Generalized Additive Model). The extraction of seasonal regression coefficients is automatic. The maximum number of seasonalities is 2 so it is possible to compute representation for double-seasonal time series. The first set seasonality (frequency) is main, so for example if we have hourly time series (`freq = c(24, 24*7)`), the number of extracted daily seasonal coefficients is 24 and the number of weekly seasonal coefficients is 7, because the length of second seasonality representation is always `freq_1 / freq_2`. The smooth function for seasonal variables is set to cubic regression spline. There is also possibility to add another independent variables (`xreg`).

**Value**

the numeric vector of GAM regression coefficients

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Laurinec P, Lucka M (2016) Comparison of representations of time series for clustering smart meter data. In: Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2016, pp 458-463

Laurinec P, Loderer M, Vrablcova P, Lucka M, Rozinajova V, Ezzeddine AB (2016) Adaptive time series forecasting of energy consumption using optimized cluster analysis. In: Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on, IEEE, pp 398-405

Laurinec P, Lucká M (2018) Clustering-based forecasting method for individual consumers electricity load using time series representations. Open Comput Sci, 8(1):38–50, DOI: 10.1515/comp-2018-0006

**See Also**

[repr\\_lm](#), [repr\\_exp](#), [gam](#)

**Examples**

```
repr_gam(rnorm(96), freq = 24)
```

---

repr_list	<i>Computation of list of representations list of time series with different lengths</i>
-----------	--

---

**Description**

The repr\_list computes list of representations from list of time series

**Usage**

```
repr_list(  
  x,  
  func = NULL,  
  args = NULL,  
  normalise = FALSE,  
  func_norm = norm_z,  
  windowing = FALSE,  
  win_size = NULL  
)
```

**Arguments**

x	the list of time series, where time series can have different lengths
func	the function that computes representation
args	the list of additional (or required) parameters of func (function that computes representation)
normalise	normalise (scale) time series before representations computation? (default is FALSE)
func_norm	the normalisation function (default is norm_z)
windowing	perform windowing? (default is FALSE)
win_size	the size of the window

**Details**

This function computes representation to an every member of a list of time series (that can have different lengths) and returns list of time series representations. It can be combined with windowing (see [repr\\_windowing](#)) and normalisation of time series.

**Value**

the numeric list of representations of time series

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[repr\\_windowing](#), [repr\\_matrix](#)

**Examples**

```
# Create random list of time series with different lengths
list_ts <- list(rnorm(sample(8:12, 1)), rnorm(sample(8:12, 1)), rnorm(sample(8:12, 1)))
repr_list(list_ts, func = repr_sma,
  args = list(order = 3))

# return normalised representations, and normalise time series by min-max normalisation
repr_list(list_ts, func = repr_sma,
  args = list(order = 3), normalise = TRUE, func_norm = norm_min_max)
```

---

repr_lm	<i>Regression coefficients from linear model as representation</i>
---------	--

---

### Description

The `repr_lm` computes seasonal regression coefficients from a linear model. Additional exogenous variables can be also added.

### Usage

```
repr_lm(x, freq = NULL, method = "lm", xreg = NULL)
```

### Arguments

<code>x</code>	the numeric vector (time series)
<code>freq</code>	the frequency of the time series. Can be vector of two frequencies (seasonalities) or just an integer of one frequency.
<code>method</code>	the linear regression method to use. It can be "lm", "rlm" or "l1".
<code>xreg</code>	the data.frame with additional exogenous regressors or the single numeric vector

### Details

This model-based representation method extracts regression coefficients from a linear model. The extraction of seasonal regression coefficients is automatic. The maximum number of seasonalities is 2 so it is possible to compute representation for double-seasonal time series. The first set seasonality (frequency) is main, so for example if we have hourly time series (`freq = c(24, 24*7)`), the number of extracted daily seasonal coefficients is 24 and the number of weekly seasonal coefficients is 7, because the length of second seasonality representation is always `freq_1 / freq_2`. There is also possibility to add another independent variables (`xreg`).

You have three possibilities for selection of a linear model method.

- "lm" is classical OLS regression.
- "rlm" is robust linear model using psi huber function and is implemented in MASS package.
- "l1" is L1 quantile regression model (also robust linear regression method) implemented in package quantreg.

### Value

the numeric vector of regression coefficients

### Author(s)

Peter Laurinec, <tsreprpackage@gmail.com>

## References

Laurinec P, Lucka M (2016) Comparison of representations of time series for clustering smart meter data. In: Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2016, pp 458-463

Laurinec P, Loderer M, Vrablcova P, Lucka M, Rozinajova V, Ezzeddine AB (2016) Adaptive time series forecasting of energy consumption using optimized cluster analysis. In: Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on, IEEE, pp 398-405

Laurinec P, Lucká M (2018) Clustering-based forecasting method for individual consumers electricity load using time series representations. Open Comput Sci, 8(1):38–50, DOI: 10.1515/comp-2018-0006

## See Also

[repr\\_gam](#), [repr\\_exp](#)

## Examples

```
# Extracts 24 seasonal regression coefficients from the time series by linear model
repr_lm(rnorm(96), freq = 24, method = "lm")
```

```
# Try also robust linear models ("rlm" and "l1")
repr_lm(rnorm(96), freq = 24, method = "rlm")
repr_lm(rnorm(96), freq = 24, method = "l1")
```

---

repr\_matrix

*Computation of matrix of representations from matrix of time series*

---

## Description

The `repr_matrix` computes matrix of representations from matrix of time series

## Usage

```
repr_matrix(  
  x,  
  func = NULL,  
  args = NULL,  
  normalise = FALSE,  
  func_norm = norm_z,  
  windowing = FALSE,  
  win_size = NULL  
)
```



**Arguments**

x	the matrix, data.frame or data.table of time series, where time series are in rows of the table
func	the function that computes representation
args	the list of additional (or required) parameters of func (function that computes representation)
normalise	normalise (scale) time series before representations computation? (default is FALSE)
func_norm	the normalisation function (default is norm_z)
windowing	perform windowing? (default is FALSE)
win_size	the size of the window

**Details**

This function computes representation to an every row of a matrix of time series and returns matrix of time series representations. It can be combined with windowing (see [repr\\_windowing](#)) and normalisation of time series.

**Value**

the numeric matrix of representations of time series

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[repr\\_windowing](#), [repr\\_list](#)

**Examples**

```
# Create random matrix of time series
mat_ts <- matrix(rnorm(100), ncol = 10)
repr_matrix(mat_ts, func = repr_paa,
  args = list(q = 5, func = meanC))

# return normalised representations, and normalise time series by min-max normalisation
repr_matrix(mat_ts, func = repr_paa,
  args = list(q = 2, func = meanC), normalise = TRUE, func_norm = norm_min_max)

# with windowing
repr_matrix(mat_ts, func = repr_feaclip, windowing = TRUE, win_size = 5)
```

---

`repr_paa`*PAA - Piecewise Aggregate Approximation*

---

**Description**

The `repr_paa` computes PAA representation from a vector.

**Usage**

```
repr_paa(x, q, func)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
<code>q</code>	the integer of the length of the "piece"
<code>func</code>	the aggregation function. Can be <code>meanC</code> , <code>medianC</code> , <code>sumC</code> , <code>minC</code> or <code>maxC</code> or similar aggregation function

**Details**

PAA with possibility to use arbitrary aggregation function. The original method uses average as aggregation function.

**Value**

the numeric vector

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Keogh E, Chakrabarti K, Pazzani M, Mehrotra Sh (2001) Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems* 3(3):263-286

**See Also**

[repr\\_dwt](#), [repr\\_dft](#), [repr\\_dct](#), [repr\\_sma](#)

**Examples**

```
repr_paa(rnorm(11), 2, meanC)
```

---

repr_pip	<i>PIP representation</i>
----------	---------------------------

---

**Description**

The repr\_pip computes PIP (Perceptually Important Points) representation from a time series.

**Usage**

```
repr_pip(x, times = 10, return = "points")
```

**Arguments**

x	the numeric vector (time series)
times	the number of important points to extract (default 10)
return	what to return? Can be important points ("points"), places of important points in a vector ("places") or "both" (data.frame).

**Value**

the values based on the argument return (see above)

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Fu TC, Chung FL, Luk R, and Ng CM (2008) Representing financial time series based on data point importance. Engineering Applications of Artificial Intelligence, 21(2):277-300

**Examples**

```
repr_pip(rnorm(100), times = 12, return = "both")
```

---

repr_pla	<i>PLA representation</i>
----------	---------------------------

---

**Description**

The repr\_pla computes PLA (Piecewise Linear Approximation) representation from a time series.

**Usage**

```
repr_pla(x, times = 10, return = "points")
```

**Arguments**

x	the numeric vector (time series)
times	the number of important points to extract (default 10)
return	what to return? Can be "points" (segments), places of points (segments) in a vector ("places") or "both" (data.frame).

**Value**

the values based on the argument return (see above)

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Zhu Y, Wu D, Li Sh (2007) A Piecewise Linear Representation Method of Time Series Based on Feature Points. Knowledge-Based Intelligent Information and Engineering Systems 4693:1066-1072

**Examples**

```
repr_pla(rnorm(100), times = 12, return = "both")
```

---

`repr_sax`*SAX - Symbolic Aggregate Approximation*

---

**Description**

The `repr_sax` creates SAX symbols for a univariate time series.

**Usage**

```
repr_sax(x, q = 2, a = 6, eps = 0.01)
```

**Arguments**

<code>x</code>	the numeric vector (time series)
<code>q</code>	the integer of the length of the "piece" in PAA
<code>a</code>	the integer of the alphabet size
<code>eps</code>	is the minimum threshold for variance in <code>x</code> and should be a numeric value. If <code>x</code> has a smaller variance than <code>eps</code> , it will be represented as a word using the middle alphabet.

**Value**

the character vector of SAX representation

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery - DMKD'03

**See Also**

[repr\\_paa](#), [repr\\_pla](#)

**Examples**

```
x <- rnorm(48)
repr_sax(x, q = 4, a = 5)
```

---

repr\_seas\_profile      *Mean seasonal profile of time series*

---

### Description

The repr\_seas\_profile computes mean seasonal profile representation from a time series.

### Usage

```
repr_seas_profile(x, freq, func)
```

### Arguments

x	the numeric vector (time series)
freq	the integer of the length of the season
func	the aggregation function. Can be meanC or medianC or similar aggregation function.

### Details

This function computes mean seasonal profile representation for a seasonal time series. The length of representation is length of set seasonality (frequency) of a time series. Aggregation function is arbitrary (best choice is for you maybe mean or median).

### Value

the numeric vector

### Author(s)

Peter Laurinec, <tsreprpackage@gmail.com>

### References

- Laurinec P, Lucka M (2016) Comparison of representations of time series for clustering smart meter data. In: Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2016, pp 458-463
- Laurinec P, Loderer M, Vrablecova P, Lucka M, Rozinajova V, Ezzeddine AB (2016) Adaptive time series forecasting of energy consumption using optimized cluster analysis. In: Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on, IEEE, pp 398-405
- Laurinec P, Lucká M (2018) Clustering-based forecasting method for individual consumers electricity load using time series representations. Open Comput Sci, 8(1):38–50, DOI: 10.1515/comp-2018-0006

### See Also

[repr\\_lm](#), [repr\\_gam](#), [repr\\_exp](#)

**Examples**

```
repr_seas_profile(rnorm(48*10), 48, meanC)
```

---

repr_sma	<i>Simple Moving Average representation</i>
----------	---

---

**Description**

The repr\_sma computes Simple Moving Average (SMA) from a time series.

**Usage**

```
repr_sma(x, order)
```

**Arguments**

x	the numeric vector (time series)
order	the order of simple moving average

**Value**

the numeric vector of smoothed values of the length = length(x) - order + 1

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
repr_sma(rnorm(50), 4)
```

---

repr_windowing	<i>Windowing of time series</i>
----------------	---------------------------------

---

**Description**

The repr\_windowing computes representations from windows of a vector.

**Usage**

```
repr_windowing(x, win_size, func = NULL, args = NULL)
```

**Arguments**

x	the numeric vector (time series)
win_size	the length of the window
func	the function for representation computation. For example repr_feaclip or repr_trend.
args	the list of additional arguments to the func (representation computation function). The args list must be named.

**Details**

This function applies specified representation method (function) to every non-overlapping window (subsequence, piece) of a time series.

**Value**

the numeric vector

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**References**

Laurinec P, and Lucka M (2018) Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. Data Mining and Knowledge Discovery. Springer. DOI: 10.1007/s10618-018-0598-2

**See Also**

[repr\\_paa](#), [repr\\_matrix](#)

**Examples**

```
# func without arguments
repr_windowing(rnorm(48), win_size = 24, func = repr_feaclip)

# func with arguments
repr_windowing(rnorm(48), win_size = 24, func = repr_featrend,
  args = list(func = maxC, order = 2, pieces = 2))
```



---

`rleC`*RLE (Run Length Encoding) written in C++*

---

**Description**

The `rleC` computes RLE from bit-level (clipping or trending representation) vector.

**Usage**

```
rleC(x)
```

**Arguments**

`x` the integer vector (from `clipping` or `trending`)

**Value**

the list of values and counts of zeros and ones

**Examples**

```
# clipping
clipped <- clipping(rnorm(50))
rleC(clipped)
# trending
trended <- trending(rnorm(50))
rleC(trended)
```

---

`rmse`*RMSE*

---

**Description**

The `rmse` computes RMSE (Root Mean Squared Error) of a forecast.

**Usage**

```
rmse(x, y)
```

**Arguments**

`x` the numeric vector of real values  
`y` the numeric vector of forecasted values

**Value**

the numeric value

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
rmse(runif(50), runif(50))
```

---

smape

*sMAPE*

---

**Description**

The smape computes sMAPE (Symmetric Mean Absolute Percentage Error) of a forecast.

**Usage**

```
smape(x, y)
```

**Arguments**

x                   the numeric vector of real values  
y                   the numeric vector of forecasted values

**Value**

the numeric value in %

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**Examples**

```
smape(runif(50), runif(50))
```

---

trending	<i>Creates bit-level (trending) representation from a vector</i>
----------	--

---

**Description**

The trending Computes bit-level (trending) representation from a vector.

**Usage**

```
trending(x)
```

**Arguments**

x                    the numeric vector (time series)

**Details**

Trending transforms time series to bit-level representation.

It is defined as follows:

$$repr_t = \begin{cases} 1 & \text{if } x_t - x_{t+1} < 0, \\ 0 & \text{otherwise,} \end{cases}$$

where  $x_t$  is a value of a time series.

**Value**

the integer vector of zeros and ones

**Author(s)**

Peter Laurinec, <tsreprpackage@gmail.com>

**See Also**

[clipping](#)

**Examples**

```
trending(rnorm(50))
```

---

TSrepr

*TSrepr package*

---

## Description

Package contains methods for time series representations computation. Representation methods of time series are for dimensionality and noise reduction, emphasizing of main characteristics of time series data and speed up of consequent usage of machine learning methods.

## Details

Package: TSrepr  
Type: Package  
Date: 2018-01-26 - Inf  
License: GPL-3

The following functions for time series representations are included in the package:

- [repr\\_paa](#) - Piecewise Aggregate Approximation (PAA)
- [repr\\_dwt](#) - Discrete Wavelet Transform (DWT)
- [repr\\_dft](#) - Discrete Fourier Transform (DFT)
- [repr\\_dct](#) - Discrete Cosine Transform (DCT)
- [repr\\_sma](#) - Simple Moving Average (SMA)
- [repr\\_pip](#) - Perceptually Important Points (PIP)
- [repr\\_sax](#) - Symbolic Aggregate Approximation (SAX)
- [repr\\_pla](#) - Piecewise Linear Approximation (PLA)
- [repr\\_seas\\_profile](#) - Mean seasonal profile
- [repr\\_lm](#) - Model-based seasonal representations based on linear model (lm, rlm, l1)
- [repr\\_gam](#) - Model-based seasonal representations based on generalized additive model (GAM)
- [repr\\_exp](#) - Exponential smoothing seasonal coefficients
- [repr\\_feaclip](#) - Feature extraction from clipping representation (FeaClip)
- [repr\\_featrend](#) - Feature extraction from trending representation (FeaTrend)
- [repr\\_feacliptrend](#) - Feature extraction from clipping and trending representation (FeaClip-Trend)

There are also implemented additional useful functions as:

- [repr\\_windowing](#) - applies above mentioned representations to every window of a time series
- [repr\\_matrix](#) - applies above mentioned representations to every row of a matrix of time series
- [repr\\_list](#) - applies above mentioned representations to every member of a list of time series

- [norm\\_z](#), [norm\\_min\\_max](#), [norm\\_boxcox](#), [norm\\_yj](#), [norm\\_atan](#) - normalisation functions
- [norm\\_z\\_params](#), [norm\\_min\\_max\\_params](#) - normalisation functions with defined parameters
- [norm\\_z\\_list](#), [norm\\_min\\_max\\_list](#) - normalisation functions with output also of scaling parameters
- [denorm\\_z](#), [denorm\\_min\\_max](#), [denorm\\_boxcox](#), [denorm\\_yj](#), [denorm\\_atan](#) - denormalisation functions

**Author(s)**

Peter Laurinec

Maintainer: Peter Laurinec <tsreprpackage@gmail.com>

# Index

- \* **datasets**
  - elec\_load, 9
- clipping, 3, 43
- coefComp, 4
- denorm\_atan, 5, 6, 45
- denorm\_boxcox, 5, 7, 45
- denorm\_min\_max, 5, 6, 6, 7, 45
- denorm\_yj, 7, 45
- denorm\_z, 5–7, 8, 45
- dtf, 21
- dwt, 23
- elec\_load, 9
- fast\_stat, 9
- fft, 22
- gam, 29
- HoltWinters, 25
- l1Coef (coefComp), 4
- lm, 4
- lmCoef (coefComp), 4
- maape, 10
- mae, 11
- mape, 11
- mase, 12
- maxC (fast\_stat), 9
- mdae, 13
- meanC (fast\_stat), 9
- medianC (fast\_stat), 9
- minC (fast\_stat), 9
- mse, 13
- norm\_atan, 14, 15, 45
- norm\_boxcox, 15, 18, 45
- norm\_min\_max, 7, 14, 15, 15, 18, 19, 45
- norm\_min\_max\_list, 7, 16, 19, 45
- norm\_min\_max\_params, 17, 20, 45
- norm\_yj, 18, 45
- norm\_z, 8, 14–16, 18, 18, 45
- norm\_z\_list, 8, 17, 19, 45
- norm\_z\_params, 17, 20, 45
- repr\_dct, 21, 22, 23, 34, 44
- repr\_dft, 21, 22, 23, 34, 44
- repr\_dwt, 21, 22, 23, 34, 44
- repr\_exp, 24, 29, 32, 38, 44
- repr\_feaclip, 25, 27, 28, 44
- repr\_feacliptrend, 26, 26, 28, 44
- repr\_featrend, 26, 27, 27, 44
- repr\_gam, 25, 28, 32, 38, 44
- repr\_list, 29, 33, 44
- repr\_lm, 25, 29, 31, 38, 44
- repr\_matrix, 30, 32, 40, 44
- repr\_paa, 34, 37, 40, 44
- repr\_pip, 35, 44
- repr\_pla, 36, 37, 44
- repr\_sax, 37, 44
- repr\_seas\_profile, 25, 38, 44
- repr\_sma, 34, 39, 44
- repr\_windowing, 30, 33, 39, 44
- rleC, 41
- rlm, 4
- rlmCoef (coefComp), 4
- rmse, 41
- rq, 4
- smape, 42
- sumC (fast\_stat), 9
- trending, 3, 43
- TSrepr, 44
- wt.filter, 23