Package 'tsrobprep'

July 13, 2021

Title Robust Preprocessing of Time Series Data

Version 0.3.1

Date 2021-07-13

Description Methods for handling the missing values outliers are introduced in this package. The recognized missing values and outliers are replaced using a model-based approach. The model may consist of both autoregressive components and external regressors. The methods work robust and efficient, and they are fully tunable. The primary motivation for writing the package was preprocessing of the energy systems data, e.g. power plant production time series, but the package could be used with any time series data.

Depends R (>= 3.2.0)

License MIT + file LICENSE

Encoding UTF-8

Imports glmnet, MASS, Matrix, mclust, quantreg, splines, textTinyR, zoo

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Michał Narajewski [aut, cre] (<https://orcid.org/0000-0002-3115-0162>), Jens Kley-Holsteg [aut], Florian Ziel [aut] (<https://orcid.org/0000-0002-2974-2660>)

Maintainer Michał Narajewski <michal.narajewski@uni-due.de>

Repository CRAN

Date/Publication 2021-07-13 16:50:01 UTC

R topics documented:

`auto_data_cleaning` *Perform automatic data cleaning of time series data*

Description

Returns a matrix or a list of matrices with imputed missing values and outliers. The function automatizes the usage of functions model_missing_data, detect_outliers and impute_modelled_data. The function is designed for numerical data only.

Usage

```
auto_data_cleaning(
  data,
  S,
  tau = NULL,
  no.of.last.indices.to.fix = S[1],
  indices.to.fix = NULL,
  model.missing.pars = list(),
  detect.outliers.pars = list()
)
```

Arguments

`data`	an input vector, matrix or data frame of dimension nobs x nvars containing missing values; each column is a variable.
`S`	a number or vector describing the seasonalities (S_1, ..., S_K) in the data, e.g. c(24, 168) if the data consists of 24 observations per day and there is a weekly seasonality in the data.
`tau`	the quantile(s) of the missing values to be estimated in the quantile regression. Tau accepts all values in (0,1). If NULL, then the weighted lasso regression is performed.
`no.of.last.indices.to.fix`	a number of observations in the tail of the data to be fixed, by default set to S.
`indices.to.fix`	indices of the data to be fixed. If NULL, then it is calculated based on the no.of.last.indices.to.fix parameter. Otherwise, the no.of.last.indices.to.fix parameter is ignored.
`model.missing.pars`	named list containing additional arguments for the model_missing_data function.
`detect.outliers.pars`	named list containing additional arguments for the detect_outliers function.

Details

The function calls model_missing_data to clean the data from missing values, detect_outliers to detect outliers, removes them and finally applies again model_missing_data function. For details see the functions' respective help sections.

Value

A list which contains a matrix or a list of matrices with imputed missing values or outliers, the indices of the data that were modelled, and the given quantile values.

See Also

model_missing_data, detect_outliers, impute_modelled_data

Examples

```
## Not run:
autoclean <- auto_data_cleaning(
  data = GBload[,-1], S = c(48, 7*48),
  no.of.last.indices.to.fix = dim(GBload)[1],
  model.missing.pars = list(consider.as.missing = 0, min.val = 0)
)
autoclean$replaced.indices

## End(Not run)
```

---

detect_outliers *Detects unreliable outliers in univariate time series data based on model-based clustering*

Description

This function applies finite mixture modelling to compute the probability of each observation being outliying data in an univariate time series. By utilizing the Mclust package the data is assigned in G clusters whereof one is modelled as an outlier cluster. The clustering process is based on features, which are modelled to differentiate normal from outlying observation.Beside computing the probability of each observation being outlying data also the specific cause in terms of the responsible feature/ feature combination can be provided.

Usage

```
detect_outliers(
  data,
  S,
  proba = 0.5,
  share = NULL,
  repetitions = 10,
  decomp = T,
```

```
  PComp = F,
  detection.parameter = 1,
  out.par = 2,
  max.cluster = 9,
  G = NULL,
  modelName = "VVV",
  feat.inf = F,
  ext.val = 1,
  ...
)
```

Arguments

data	an one dimensional matrix or data frame without missing data; each row is an observation.
S	vector with numeric values for each seasonality present in data.
proba	denotes the threshold from which on an observation is considered as being outlying data. By default is set to 0.5 (ranging from 0 to 1). Number of outliers increases with decrease of proba threshold.
share	controlls the size of the subsample used for estimation. By default set to pmin(2*round(length(data)^(sqrt(length(data))/length(data) (ranging from 0 to 1). In combination with the repetitions parameter the robustness and computational time of the method can be controlled.
repetitions	denotes the number of repetitions to repeat the clustering. By default set to 10. Allows to control the robustness and computational time of the method.
decomp	allows to perform seasonal decomposition on the original time series as preprocessing step before feature modelling. By default set to TRUE.
PComp	allows to use the principal components of the modelled feature matrix. By default set to FALSE.
detection.parameter	denotes a parameter to regulate the detection sensitivity. By default set to 1. It is assumed that the outlier cluster follows a (multivariate) Gaussian distribution parameterized by sample mean and a blown up sample covariance matrix of the feature space. The covariance matrix is blown up by detection.parameter * (2 * log(length(data)))^2. By increase the more extrem outliers are detected.
out.par	controls the number of artifially produced outliers to allow cluster formation of oultier cluster. By default out.par ist set to 2. By increase it is assumed that share of outliers in data increases. A priori it is assumed that out.par * ceiling(sqrt(nrow(data.original))) number of observations are outlying observations.
max.cluster	a single numeric value controlling the maximum number of allowed clusters. By default set to 9.
G	denotes the optimal number of clusters limited by the max.cluster paramter. By default G is set to NULL and is automatically calculated based on the BIC.
modelName	denotes the geometric features of the covariance matrix. i.e. "EII", "VII", "EEI", "EVI", "VEI", "VVI", etc.. By default modelName is set to "VVV". The help file for mclustModelNames describes the available models. Choice of modelName influences the fit to the data as well as the computational time.

feat.inf	logical value indicating whether influential features/ feature combinations should be computed. By default set to FALSE.
ext.val	denotes the number of observations for each side of an identified outlier, which should also be treated as outliyng data. By default set to 1.
...	additional arguments for the Mclust function.

Details

The detection of outliers is addressed by model based clustering based on parameterized finite Gaussian mixture models. For cluster estimation the Mclust function is applied. Models are estimated by the EM algorithm initialized by hierarchical model-based agglomerative clustering. The optimal model is selected according to BIC. The following features based on the introduced data are used in the clustering process:

org.series denotes the scaled and potantially decomposed original time series.

seasonality denotes determenistic seasonalities based on S.

gradient denotes the summation of the two sided gradient of the org.series.

abs.gradient denotes the summation of the absolute two sided gradient of org.series.

rel.gradient denotes the summation of the two sided absolute gradient of the org.series with sign based on left sided gradient in relation to the rolling mean absolut deviation based on most relevant seasonality S.

abs.seas.grad denotes the summation of the absolute two sided seasonal gradient of org.series based on seasonalties S.

In case PComp = TRUE, the features correspond to the principal components of the introduced feature space.

Value

a list containing the following elements:

data	numeric vector containing the original data.
outlier.pos	a vector indicating the position of each outlier and the corresponding neighboorhood controled by ext.val.
outlier.pos.raw	a vector indicating the position of each outlier.
outlier.probs	a vector containing all probabilities for each observation being outlying data.
Repetitions	provides a list for each repetition containing the estimated model, the outlier cluster, the probabilities for each observation belonging to the estimated clusters, the outlier position, the influence of each feature/ feature combination on the identified outyling data, and the corresponding probabilities after shift to the feature mean of each considered outlier, as well as the applied subset of the extended feature matrix for estimation (including artificially introduced outliers).
features	a matrix containg the feature matrix. Each column is a feature.
inf.feature.combinations	a list containg the features/ feature comibinations, which caused assignment to outlier cluster.

`feature.inf.tab`
: a matrix containing all possible feature combinations.

`PC`
: an object of class "princomp" containing the principal component analysis of the feature matrix.

See Also

`model_missing_data`, impute_modelled_data, auto_data_cleaning

Examples

```
## Not run:
set.seed(1)
id <- 14000:17000
# Replace missing values
modelmd <- model_missing_data(data = GBload[id, -1], tau = 0.5,
                              S = c(48, 336), indices.to.fix = seq_len(nrow(GBload[id, ])),
                                 consider.as.missing = 0, min.val = 0)
# Impute missing values
data.imputed <- impute_modelled_data(modelmd)

#Detect outliers
system.time(
  o.ident <- detect_outliers(data = data.imputed, S = c(48, 336))
)

# Plot of identified outliers in time series
outlier.vector <- rep(F,length(data.imputed))
outlier.vector[o.ident$outlier.pos] <- T
plot(data.imputed, type = "o", col=1 + 1 * outlier.vector,
     pch = 1 + 18 * outlier.vector)

# table of identified raw outliers and corresponding probs being outlying data
df <- data.frame(o.ident$outlier.pos.raw,unlist(o.ident$outlier.probs)[o.ident$outlier.pos.raw])
colnames(df) <- c("Outlier position", "Probability of being outlying data")
df

# Plot of feature matrix
plot.ts(o.ident$features, type = "o",
        col = 1 + outlier.vector,
        pch = 1 + 1 * outlier.vector)

# table of outliers and corresponding features/ feature combinations,
# which caused assignment to outlier cluster
# Detect outliers with feat.int = T
set.seed(1)
system.time(
  o.ident <- detect_outliers(data = data.imputed, S = c(48, 336), feat.inf = T)
)
feature.imp <- unlist(lapply(o.ident$inf.feature.combinations,
                        function(x) paste(o.ident$feature.inf.tab[x], collapse = " | ")))
```

```
df <- data.frame(o.ident$outlier.pos.raw,o.ident$outlier.probs[o.ident$outlier.pos.raw],
                 feature.imp[as.numeric(names(feature.imp)) %in% o.ident$outlier.pos.raw])
colnames(df) <- c("Outlier position", "Probability being outlying data", "Responsible features")
View(df)

## End(Not run)
```

`GBload` *The electricity actual total load in Great Britain in year 2018*

Description

A dataset containing the electricity actual total load (MW) in Great Britain in year 2018 presented in half-hour interval. Each data point regards 30 minutes of electricity load starting at given time. The data consists of both missing values and outliers.

Usage

```
GBload
```

Format

A data frame with 17520 rows and 2 variables:

Date date indicating the delivery beginning of the electricity

Load actual electricity load in MW ...

Source

https://transparency.entsoe.eu/

`impute_modelled_data` *Impute modelled missing time series data*

Description

Returns a matrix or a list of matrices with imputed missing values or outliers. As argument the function requires an object of class "tsrobprep" and the quantiles to be imputed.

Usage

```
impute_modelled_data(object, tau = NULL)
```

Arguments

object	an object of class "tsrobprep" that is an output of function model_missing_data.
tau	the quantile(s) of the missing values to be imputed. tau should be a subset of the quantile values present in the "tsrobprep" object. By default all quantiles present in the object are used.

Value

A matrix or a list of matrices with imputed missing values or outliers.

See Also

model_missing_data, detect_outliers, auto_data_cleaning

Examples

```
## Not run:
model.miss <- model_missing_data(
    data = GBload[,-1], S = c(48,7*48),
    no.of.last.indices.to.fix = dim(GBload)[1], consider.as.missing = 0,
    min.val = 0
)
model.miss$estimated.models
model.miss$replaced.indices
new.GBload <- impute_modelled_data(model.miss)

## End(Not run)
```

---

model_missing_data *Model missing time series data*

Description

Returns an object of class "tsrobprep" which contains the original data and the modelled missing values to be imputed. The function model_missing_data models missing values in a time series data using a robust time series decomposition with the weighted lasso or the quantile regression. The model uses autoregression on the time series as explanatory variables as well as the provided external variables. The function is designed for numerical data only.

Usage

```
model_missing_data(
  data,
  S,
  tau = NULL,
  no.of.last.indices.to.fix = S[1],
  indices.to.fix = NULL,
  replace.recursively = TRUE,
```

```
  p = NULL,
  mirror = FALSE,
  lags = NULL,
  extreg = NULL,
  n.best.extreg = NULL,
  use.data.as.ext = FALSE,
  lag.externals = FALSE,
  consider.as.missing = NULL,
  whole.period.missing.only = FALSE,
  debias = FALSE,
  min.val = -Inf,
  max.val = Inf,
  Cor_thres = 0.5,
  digits = 3,
  ICpen = "BIC",
  decompose.pars = list(),
  ...
)
```

Arguments

data	an input vector, matrix or data frame of dimension nobs x nvars containing missing values; each column is a variable.
S	a number or vector describing the seasonalities (S_1, ..., S_K) in the data, e.g. c(24, 168) if the data consists of 24 observations per day and there is a weekly seasonality in the data.
tau	the quantile(s) of the missing values to be estimated in the quantile regression. Tau accepts all values in (0,1). If NULL, then the weighted lasso regression is performed.
no.of.last.indices.to.fix	a number of observations in the tail of the data to be fixed, by default set to first element of S.
indices.to.fix	indices of the data to be fixed. If NULL, then it is calculated based on the no.of.last.indices.to.fix parameter. Otherwise, the no.of.last.indices.to.fix parameter is ignored.
replace.recursively	if TRUE then the algorithm uses replaced values to model the remaining missings.
p	a number or vector of length(S) = K indicating the order of a K-seasonal autoregressive process to be estimated. If NULL, chosen data-based.
mirror	if TRUE then autoregressive lags up to order p are not only added to the seasonalities but also subtracted.
lags	a numeric vector with the lags to use in the autoregression. Negative values are accepted and then also the "future" observations are used for modelling. If not NULL, p and mirror are ignored.
extreg	a vector, matrix or data frame of data containing external regressors; each column is a variable.

`n.best.extreg` a numeric value specifying the maximal number of considered best correlated external regressors (selected in decreasing order). If NULL, then all variables in extreg are used for modelling.

`use.data.as.ext` logical specifying whether to use the remaining variables in the data as external regressors or not.

`lag.externals` logical specifying whether to lag the external regressors or not. If TRUE, then the algorithm uses the lags specified in parameter lags.

`consider.as.missing` a vector of numerical values which are considered as missing in the data.

`whole.period.missing.only` if FALSE, then all observations which correspond to the values of consider.as.missing are treated as missings. If TRUE, then only consecutive observations of specified length are considered (length is defined by first element of S).

`debias` if TRUE, the recursive replacement is additionally debiased.

`min.val` a single value or a vector of length nvars providing the minimum possible value of each variable in the data. If a single value, then it applies to all variables. By default set to -Inf.

`max.val` a single value or a vector of length nvars providing the maximum possible value of each variable in the data. If a single value, then it applies to all variables. By default set to Inf.

`Cor_thres` a single value providing the correlation threshold from which external regressors are considered in the quantile regression.

`digits` integer indicating the number of decimal places allowed in the data, by default set to 3.

`ICpen` is the information criterion penalty for lambda choice in the glmnet algorithm. It can be a string: "BIC", "HQC" or "AIC", or a fixed number.

`decompose.pars` named list containing additional arguments for the robust_decompose function.

`...` additional arguments for the glmnet or rq.fit.fnb algorithms.

Details

The function uses robust time series decomposition with weighted lasso or quantile regression in order to model missing values and prepare it for imputation. In this purpose the robust_decompose function together with the glmnet are used in case of mean regression, i.e. tau = NULL. In case of quantile regression, i.e. tau != NULL the robust_decompose function is used together with the rq.fit.fnb function. The modelled values can be imputed using impute_modelled_data function.

Value

An object of class "tsrobprep" which contains the original data, the indices of the data that were modelled, the given quantile values, a list of sparse matrices with the modelled data to be imputed and a list of the numbers of models estimated for every variable.

See Also

robust_decompose, impute_modelled_data, detect_outliers, auto_data_cleaning

Examples

```
## Not run:
model.miss <- model_missing_data(
    data = GBload[,-1], S = c(48,7*48),
    no.of.last.indices.to.fix = dim(GBload)[1], consider.as.missing = 0,
    min.val = 0
)
model.miss$estimated.models
model.miss$replaced.indices
new.GBload <- impute_modelled_data(model.miss)

## End(Not run)
```

`robust_decompose` *Robust time series seasonal decomposition*

Description

Decompose a time series into trend, level and potentially multiple seasonal components including all interactions. The function allows for missings.

Usage

```
robust_decompose(
  x,
  S,
  wsize = max(2 * max(S), 25),
  use.trend = TRUE,
  K = 4,
  ICpen = "BIC",
  extreg = NULL,
  use.autoregressive = NULL
)
```

Arguments

`x`	a time series.
`S`	a number or vector describing the seasonalities (S_1, ..., S_K) in the data, e.g. c(24, 168) if the data consists of 24 observations per day and there is a weekly seasonality in the data.
`wsize`	is filter/rolling med size
`use.trend`	if TRUE, uses standard decomposition. If FALSE, uses no trend component.
`K`	a sigma (standard deviation) bound. The observations that exceed sigma*K become reduced weight in the regression.
`ICpen`	is the information criterion penalty, e.g. string "BIC", "HQC" or "AIC", or a fixed number.

`extreg` a vector, matrix or data frame of data containing external regressors; each column is a variable.

`use.autoregressive` if TRUE, removes the autoregression from the series. If NULL, it is derived data based.

Value

A list which contains a vector of fitted values, a vector of weights given to the original time series, and a matrix of components of the decomposition.

Examples

```
## Not run:
GBload.decomposed <- robust_decompose(GBload[,-1], S = c(48,7*48))
head(GBload.decomposed$components)

## End(Not run)
```

Index