Package ‘imputeTestbench’
July 5, 2019

Type Package
Title Test Bench for the Comparison of Imputation Methods
Date 2019-07-05
Maintainer Marcus W. Beck <mbafs2012@gmail.com>
Version 3.0.3
Description Provides a test bench for the comparison of missing data imputation methods in uni-variate time series. Imputation methods are compared using different error metrics. Proposed imputation methods and alternative error metrics can be used.
Imports dplyr, forecast, ggplot2, imputeTS, reshape2, stats, tidyr, zoo
BugReports https://github.com/neerajdhanraj/imputeTestbench/issues
License CC0
LazyData TRUE
RoxygenNote 6.1.1
Suggests knitr, rmarkdown, magrittr
NeedsCompilation no
Author Neeraj Bokde [aut],
Marcus W. Beck [cre, aut]
Repository CRAN
Date/Publication 2019-07-05 18:10:14 UTC

R topics documented:

impute_errors ......................................................... 2
mae ................................................................. 4
mape ............................................................... 5
plot_errors ......................................................... 5
plot_impute ........................................................ 6
print.errprof ......................................................... 8
rmse ............................................................... 8
sample_dat ........................................................ 9
impute_errors

Function working as testbench for comparison of imputing models

Description

Function working as testbench for comparison of imputing models

Usage

impute_errors(dataIn, smps = "mcar", methods = c("na.approx", 
"na.interp", "na_interpolation", "na.locf", "na_mean"),
methodPath = NULL, errorParameter = "rmse", errorPath = NULL,
blck = 50, blckper = TRUE, missPercentFrom = 10,
missPercentTo = 90, interval = 10, repetition = 10,
addl_arg = NULL)

Arguments

dataIn input ts for testing

smps chr string indicating sampling type for generating missing data, see details

methods chr string of imputation methods to use, one to many. A user-supplied function can be included if methodPath is used, see details.

methodPath chr string of location of script containing one or more functions for the proposed imputation method(s)

errorParameter chr string indicating which error type to use, acceptable values are "rmse" (default), "mae", or "mape". Alternatively, a user-supplied function can be passed if errorPath is used, see details.

errorPath chr string of location of script containing one or more error functions for evaluating imputations

blck numeric indicating block sizes as a percentage of the sample size for the missing data, applies only if smps = 'mar'

blckper logical indicating if the value passed to blck is a percentage of the sample size for missing data, otherwise blck indicates number of observations

missPercentFrom numeric from which percent of missing values to be considered

missPercentTo numeric for up to what percent missing values are to be considered

interval numeric for interval between consecutive missPercent values

repetition numeric for repetitions to be done for each missPercent value

addl_arg arguments passed to other imputation methods as a list of lists, see details.
Details

The default methods for `impute_errors` are `na.approx`, `na.interp`, `na_interpolation`, `na_locf`, and `na_mean`. See the help file for each for additional documentation. Additional arguments for the imputation functions are passed as a list of lists to the `addl_arg` argument, where the list contains one to many elements that are named by the methods. The elements of the master list are lists with arguments for the relevant methods. See the examples.

A user-supplied function can also be passed to `methods` as an additional imputation method. A character string indicating the path of the function must also be supplied to `methodpath`. The path must point to a function where the first argument is the time series to impute.

An alternative error function can also be passed to `errorparameter` if `errorPath` is not `NULL`. The function specified in `errorPath` must have two arguments where the first is a vector for the observed time series and the second is a vector for the predicted time series.

The `smps` argument indicates the type of sampling for generating missing data. Options are `smps = 'mcar'` for missing completely at random and `smps = 'mar'` for missing at random. Additional information about the sampling method is described in `sample_dat`. The relevant arguments for `smps = 'mar'` are `blck` and `blckper` which greatly affect the sampling method.

Infinite comparisons are removed with a warning if `errorParameter = 'mape'`. This occurs if any of the observed values in the original time series are zero. Error estimates for such datasets are evaluated only for non-zero observations.

Value

Returns an error comparison for imputation methods as an `errprof` object. This object is structured as a list where the first two elements are named `Parameter` and `MissingPercent` that describe the error metric used to assess the imputation methods and the intervals of missing observations as percentages, respectively. The remaining elements are named as the chr strings in `methods` of the original function call. Each remaining element contains a numeric vector of the average error at each missing percent of observations. The `errprof` object also includes an attribute named `errall` as an additional list that contains all of the error estimates for every imputation method and repetition.

See Also

`sample_dat`

Examples

```r
# Not run:
# default options
aa <- impute_errors(dataIn = nottem)
aa
plot_errors(aa)

# change the simulation for missing obs
aa <- impute_errors(dataIn = nottem, smps = 'mar')
aa
plot_errors(aa)

# use one interpolation method, increase repetitions
```
```r
aa <- impute_errors(dataIn = nottem, methods = 'na.interp', repetition = 100)
plot_errors(aa)

# change the error metric
aa <- impute_errors(dataIn = nottem, errorParameter = 'mae')
plot_errors(aa)

# passing additional arguments to imputation methods
impute_errors(dataIn = nottem, addl_arg = list(na_mean = list(option = 'mode')))  
## End(Not run)
```

---

**mae**  
*Mean Absolute Error Calculation*

**Description**

takes difference between Original data and Predicted data as input

**Usage**

```r
mae(obs, pred)
```

**Arguments**

- **obs** numeric vector of original data
- **pred** numeric vector of predicted data

**Value**

`maeVal` as Mean Absolute Error

**Examples**

```r
## Generate 100 random numbers within some limits
x <- sample(1:7, 100, replace = TRUE)
y <- sample(1:4, 100, replace = TRUE)
z <- mae(x, y)
z
```
**mape**

*Mean Absolute Percent Error Calculation*

**Description**

takes difference between Original data and Predicted data as input

**Usage**

```r
mape(obs, pred)
```

**Arguments**

- `obs` numeric vector of original data
- `pred` numeric vector of predicted data

**Value**

`mapeVal` as Mean Absolute Error

**Examples**

```r
## Generate 100 random numbers within some limits
x <- sample(1:7, 100, replace = TRUE)
y <- sample(1:4, 100, replace = TRUE)
z <- mape(x, y)
z
```

**plot_errors**

*Function to plot the Error Comparison*

**Description**

Function to plot the Error Comparison

**Usage**

```r
plot_errors(dataIn, plotType = c("boxplot"))
```

**S3 method for class 'errprof'**

```r
plot_errors(dataIn, plotType = c("boxplot"))
```

**Arguments**

- `dataIn` an errprof object returned from `impute_errors`
- `plotType` chr string indicating plot type, accepted values are "boxplot", "bar", or "line"
Value

A ggplot object that can be further modified. The entire range of errors are shown if plotType = "boxplot", otherwise the averages are shown if plotType = "bar" or "line".

Examples

aa <- impute_errors(dataIn = nottem)

# default plot
plot_errors(aa)
## Not run:
# bar plot of averages at each repetition
plot_errors(aa, plotType = 'bar')

# line plot of averages at each repetition
plot_errors(aa, plotType = 'line')

# change the plot aesthetics
library(ggplot2)
p <- plot_errors(aa)
p + scale_fill_brewer(palette = 'Paired', guide_legend(title = 'Default'))
p + theme(legend.position = 'top')
p + theme_minimal()
p + ggtitle('Distribution of error for imputed values')
p + scale_y_continuous('RMSE')

## End(Not run)

plot_impute  
Plot imputations

Description

Plot imputations for data from multiple methods

Usage

plot_impute(dataIn, smps = "mcar", methods = c("na.approx", "na.interp", "na_interpolation", "na.locf", "na_mean"), methodPath = NULL, blk = 50, blkper = TRUE, missPercent = 50, showmiss = FALSE, addl_arg = NULL)

Arguments

dataIn  input ts for testing
smps    chr string indicating sampling type for generating missing data, see details
plot_impute

methods  chr string of imputation methods to use, one to many. A user-supplied function can be included if methodPath is used.
methodPath  chr string of location of script containing one or more functions for the proposed imputation method(s)
blck  numeric indicating block sizes as a percentage of the sample size for the missing data, applies only if smps = 'mar'
blckper  logical indicating if the value passed to blck is a percentage of the sample size for missing data, otherwise blck indicates number of observations
missPercent  numeric for percent of missing values to be considered
showmiss  logical if removed values missing from the complete dataset are plotted
addl_arg  arguments passed to other imputation methods as a list of lists, see details.

Details

See the documentation for impute_errors for an explanation of the arguments.

Value

A ggplot object showing the imputed data for each method. Red points are labelled as 'imputed' and blue points are labelled as 'retained' from the original data set. Missing data that were removed can be added to the plot as open circles if showmiss = TRUE. See the examples for modifying the plot.

Examples

# default
plot_impute(dataIn = nottem)

# change missing percent total
plot_impute(dataIn = nottem, missPercent = 10)

# show missing values
plot_impute(dataIn = nottem, showmiss = TRUE)

# use mar sampling
plot_impute(dataIn = nottem, smps = 'mar')

# change the plot aesthetics
## Not run:
library(ggplot2)
p <- plot_impute(dataIn = nottem, smps = 'mar')
p + scale_colour_manual(values = c('black', 'grey'))
p + theme_minimal()
p + ggtitle('Imputation examples with different methods')
p + scale_y_continuous('Temp at Nottingham Castle (F)')

## End(Not run)
print.errprof  

Print method for errprof

Description

Print method for errprof class

Usage

```r
## S3 method for class 'errprof'
print(x, ...)
```

Arguments

- `x`: input errprof object
- `...`: arguments passed to or from other methods

Value

list output for the errprof object

rmse  

Root Mean Square Error Calculation

Description

takes difference between Original data and Predicted data as input

Usage

```r
rmse(obs, pred)
```

Arguments

- `obs`: numeric vector of original data
- `pred`: numeric vector of predicted data

Value

rmseVal as Root Mean Square Error

Examples

```r
## Generate 100 random numbers within some limits
x <- sample(1:7, 100, replace = TRUE)
y <- sample(1:4, 100, replace = TRUE)
z <- rmse(x, y)
z
```
Description

Sample time series using completely at random (MCAR) or at random (MAR)

Usage

```r
sample_dat(datin, smps = "mcar", repetition = 10, b = 10,
           blck = 50, blckper = TRUE, plot = FALSE)
```

Arguments

- `datin`: input numeric vector
- `smps`: chr string of sampling type to use, options are "mcar" or "mar"
- `repetition`: numeric for repetitions to be done for each missPercent value
- `b`: numeric indicating the total amount of missing data as a percentage to remove from the complete time series
- `blck`: numeric indicating block sizes as a proportion of the sample size for the missing data
- `blckper`: logical indicating if the value passed to `blck` is a proportion of missPer, i.e., blocks are to be sized as a percentage of the total size of the missing data
- `plot`: logical indicating if a plot is returned showing the sampled data, plots only the first repetition

Value

Input data with NA values for the sampled observations if `plot = FALSE`, otherwise a plot showing the missing observations over the complete dataset.

The missing data if `smps = "mar"` are based on random sampling by blocks. The start location of each block is random and overlapping blocks are not counted uniquely for the required sample size given by `b`. Final blocks are truncated to ensure the correct value of `b` is returned. Blocks are fixed at 1 if the proportion is too small, in which case "mcar" should be used. Block sizes are also truncated to the required sample size if the input value is too large if `blckper = FALSE`. For the latter case, this is the same as setting `blck = 1` and `blckper = TRUE`.

For all cases, the first and last observation will never be removed to allow comparability of interpolation schemes. This is especially relevant for cases when `b` is large and `smps = "mar"` is used. For example, `method = na.approx` will have `rmse = 0` for a dataset where the removed block includes the last n observations. This result could provide misleading information in comparing methods.
Examples

```r
a <- rnorm(1000)

# default sampling
sample_dat(a)

# use mar sampling
sample_dat(a, smps = 'mar')

# show a plot of one repetition
sample_dat(a, plot = TRUE)

# show a plot of one repetition, mar sampling
sample_dat(a, smps = 'mar', plot = TRUE)

# change plot aesthetics
library(ggplot2)
p <- sample_dat(a, plot = TRUE)
p + scale_colour_manual(values = c('black', 'grey'))
p + theme_minimal()
p + ggtitle('Example of simulating missing data')
```
Index

ggplot, 7

impute_errors, 2, 5, 7

mae, 4
mape, 5

na.approx, 3
na.interp, 3
na.locf, 3
na_interpolation, 3
na_mean, 3

plot_errors, 5
plot_impute, 6
print.errprof, 8

rmse, 8

sample_dat, 3, 9

ts, 2, 6