imputeTS: Time Series Missing Value Imputation in R

by Steffen Moritz and Thomas Bartz-Beielstein

Abstract The **imputeTS** package specializes on univariate time series imputation. It offers multiple state-of-the-art imputation algorithm implementations along with plotting functions for time series missing data statistics. While imputation in general is a well-known problem and widely covered by R packages, finding packages able to fill missing values in univariate time series is more complicated. The reason for this lies in the fact, that most imputation algorithms rely on inter-attribute correlations, while univariate time series imputation instead needs to employ time dependencies. This paper provides an introduction to the **imputeTS** package and its provided algorithms and tools. Furthermore, it gives a short overview about univariate time series imputation in R.

Introduction

In almost every domain from industry (Billinton et al., 1996) to biology (Bar-Joseph et al., 2003), finance (Taylor, 2007) up to social science (Gottman, 1981) different time series data are measured. While the recorded datasets itself may be different, one common problem are missing values. Many analysis methods require missing values to be replaced with reasonable values up-front. In statistics this process of replacing missing values is called *imputation*.

Time series imputation thereby is a special sub-field in the imputation research area. Most popular techniques like Multiple Imputation (Rubin, 1987), Expectation-Maximization (Dempster et al., 1977), Nearest Neighbor (Vacek and Ashikaga, 1980) and Hot Deck (Ford, 1983) rely on inter-attribute correlations to estimate values for the missing data. Since univariate time series do not possess more than one attribute, these algorithms cannot be applied directly. Effective univariate time series imputation algorithms instead need to employ the inter-time correlations.

On CRAN there are several packages solving the problem of imputation of multivariate data. Most popular and mature (among others) are **AMELIA** (Honaker et al., 2011), **mice** (van Buuren and Groothuis-Oudshoorn, 2011), **VIM** (Kowarik and Templ, 2016) and **missMDA** (Josse and Husson, 2016). However, since these packages are designed for multivariate data imputation only they do not work for univariate time series.

At the moment **imputeTS** (Moritz, 2016a) is the only package on CRAN that is solely dedicated to univariate time series imputation and includes multiple algorithms. Nevertheless, there are some other packages that include imputation functions as addition to their core package functionality. Most noteworthy being **zoo** (Zeileis and Grothendieck, 2005) and **forecast** (Hyndman, 2016). Both packages offer also some advanced time series imputation functions. The packages **spacetime** (Pebesma, 2012), **timeSeries** (Rmetrics Core Team et al., 2015) and **xts** (Ryan and Ulrich, 2014) should also be mentioned, since they contain some very simple but quick time series imputation methods. For a broader overview about available time series imputation packages in R see also (Moritz et al., 2015). In this technical report we evaluate the performance of several univariate imputation functions in R on different time series.

This paper is structured as follows: Section Overview **imputeTS** package gives an overview, about all features and functions included in the **imputeTS** package. This is followed by Usage examples of the different provided functions. The paper ends with a Conclusions section.

Overview imputeTS package

The **imputeTS** package can be found on CRAN and is an easy to use package that offers several utilities for *'univariate, equi-spaced, numeric time series'*.

Univariate means there is just one attribute that is observed over time. Which leads to a sequence of single observations o_1, o_2, o_3, ... o_n at successive points t_1, t_2, t_3, ... t_n in time. Equi-spaced means, that time increments between successive data points are equal $|t_1 - t_2| = |t_2 - t_3| = ... = |t_{n-1} - t_n|$. Numeric means that the observations are measurable quantities that can be described as a number.

In the first part of this section, a general overview about all available functions and datasets is given.

This is followed by more detailed overviews about the three areas covered by the package: 'Plots & Statistics', 'Imputation' and 'Datasets'. Information about how to apply these functions and tools can be found later in the Usage examples section.

General overview

As can be seen in Table 1, beyond several imputation algorithm implementations the package also includes plotting functions and datasets. The imputation algorithms can be divided into rather simple but fast approaches like mean imputation and more advanced algorithms that need more computation time like kalman smoothing on a structural model.

Simple Imputation	Imputation	Plots & Statistics	Datasets
na.locf	na.interpolation	plotNA.distribution	tsAirgap
na.mean	na.kalman	plotNA.distributionBar	tsAirgapComplete
na.random	na.ma	plotNA.gapsize	tsHeating
na.replace	na.seadec	plotNA.imputations	tsHeatingComplete
na.remove	na.seasplit	statsNA	tsNH4
			tsNH4Complete

Table 1: General Overview imputeTS package

As a whole, the package aims to support the user in the complete process of replacing missing values in time series. This process starts with analyzing the distribution of the missing values using the `statsNA` function and the plots of `plotNA.distribution`, `plotNA.distributionBar`, `plotNA.gapsize`. In the next step the actual imputation can take place with one of the several algorithm options. Finally, the imputation results can be visualized with the `plotNA.imputations` function. Additionally, the package contains three datasets, each in a version with and without missing values, that can be used to test imputation algorithms.

Plots & Statistics functions

An overview about the available plots and statistics functions can be found in Table 2. To get a good impression what the plots look like section Usage examples is recommended.

Function	Description
plotNA.distribution	Visualize Distribution of Missing Values
plotNA.distributionBar	Visualize Distribution of Missing Values (Barplot)
plotNA.gapsize	Visualize Distribution of NA gap sizes
plotNA.imputations	Visualize Imputed Values
statsNA	Print Statistics about the Missing Data

Table 2: Overview Plots & Statistics

The `statsNA` function calculates several missing data statistics of the input data. This includes overall percentage of missing values, absolute amount of missing values, amount of missing value in different sections of the data, longest series of consecutive NAs and occurrence of consecutive NAs. The `plotNA.distribution` function visualizes the distribution of NAs in a time series. This is done using a standard time series plot, in which areas with missing data are colored red. This enables the user to see at first sight where in the series most of the missing values are located. The `plotNA.distributionBar` function provides the same insights to users, but is designed for very large time series. This is necessary for time series with 1000 and more observations, where it is not possible to plot each observation as a single point. The `plotNA.gapsize` function provides information about consecutive NAs by showing the most common NA gap sizes in the time series. The `plotNA.imputations` function is designated for visual inspection of the results after applying an imputation algorithm. Therefore, newly imputed observations are shown in a different color than the rest of the series.

Imputation functions

An overview about all available imputation algorithms can be found in Table 3. Even if these functions are really easy applicable, some examples can be found later in section Usage examples. More detailed information about the theoretical background of the algorithms can be found in the **imputeTS** manual (Moritz, 2016b).

Function	Option	Description
na.interpolation		
	linear	Imputation by Linear Interpolation
	spline	Imputation by Spline Interpolation
	stine	Imputation by Stineman Interpolation
na.kalman		
	StructTS	Imputation by Structural Model & Kalman Smoothing
	auto.arima	Imputation by ARIMA State Space Representation & Kalman Sm.
na.locf		
	locf	Imputation by Last Observation Carried Forward
	nocb	Imputation by Next Observation Carried Backward
na.ma		
	simple	Missing Value Imputation by Simple Moving Average
	linear	Missing Value Imputation by Linear Weighted Moving Average
	exponential	Missing Value Imputation by Exponential Weighted Moving Average
na.mean		
	mean	MissingValue Imputation by Mean Value
	median	Missing Value Imputation by Median Value
	mode	Missing Value Imputation by Mode Value
na.random		Missing Value Imputation by Random Sample
na.replace		Replace Missing Values by a Defined Value
na.seadec		Seasonally Decomposed Missing Value Imputation
na.seasplit		Seasonally Splitted Missing Value Imputation
na.remove		Remove Missing Values

Table 3: Overview Imputation Algorithms

For convenience similar algorithms are available under one function name as parameter option. For example linear, spline and stineman interpolation are all included in the `na.interpolation` function. The `na.mean`, `na.locf`, `na.replace`, `na.random` functions are all simple and fast. In comparison, `na.interpolation`, `na.kalman`, `na.ma`, `na.seasplit`, `na.seadec` are more advanced algorithms that need more computation time. The `na.remove` function is a special case, since it only deletes all missing values. Thus, it is not really an imputation function. It should be handled with care since removing observations may corrupt the time information of the series. The `na.seasplit` and `na.seadec` functions are as well exceptions. These perform seasonal split / decomposition operations as a preprocessing step. For the imputation itself, one out of the other imputation algorithms can be used (which one can be set as option). Looking at all available imputation methods, no single overall best method can be pointed out. Imputation performance is always very dependent on the characteristics of the input time series. Even imputation with mean values can sometimes be an appropriate method. For time series with a strong seasonality usually `na.kalman` and `na.seadec` / `na.seasplit` perform best. In general, for most time series one algorithm out of `na.kalman`, `na.interpolation` and `na.seadec` will yield the best results. Meanwhile, `na.random`, `na.mean`, `na.locf` will be at the lower end accuracy wise for the majority of input time series.

Datasets

As can be seen in Table 4, all three datasets are available in a version with missing data and in a complete version. The provided time series are designated as benchmark datasets for univariate time series imputation. They shall enable users to quickly compare and test imputation algorithms. Without these datasets the process of testing time series imputation algorithms would require to manually delete certain observations. The benchmark data simplifies this: imputation algorithms can directly be applied to the dataset versions with missing values, which then can be compared to

the complete dataset versions afterwards. Since the time series are specified, researchers can use these to compare their algorithms against each other.

Reached RMSE or MAPE values on these datasets are easily understandable results to quote and compare against. Nevertheless, comparing algorithms using these fixed datasets can only be a first indicator of how well algorithms perform in general. Especially for the very short `tsAirgap` series (with just 13 NA values) random lucky guesses can considerably influence the results. A complete benchmark would include: 'Different missing data percentages', 'Different datasets', 'Different random seeds for missing data simulation'.

Overall there is a relatively small time series provided in `tsAirgap`, a medium one in `tsNH4` and a large time series in `tsHeating`. The `tsHeating` and `tsNH4` are both sensor data, while `tsAirgap` is count data.

Dataset	Description
tsAirgap	Time series of monthly airline passengers (with NAs)
tsAirgapComplete	Time series of monthly airline passengers (complete)
tsHeating	Time series of a heating systems' supply temperature (with NAs)
tsHeatingComplete	Time series of a heating systems' supply temperature (complete)
tsNH4	Time series of NH4 concentration in a waste-water system (with NAs)
tsNH4Complete	Time series of NH4 concentration in a waste-water system (complete)

Table 4: Overview Datasets

tsAirgap
The `tsAirgap` time series has 144 rows and the incomplete version includes 14 NA values. It represents the monthly totals of international airline passengers from 1949 to 1960. The time series originates from Box et al. (2015) and is a commonly used example in time series analysis literature. Originally known as 'AirPassengers' or 'airpass' this version is renamed to 'tsAirgap' in order improve differentiation from the complete series (gap signifies that NAs were introduced). The characteristics (strong trend, strong seasonal behavior) make the `tsAirgap` series a great example for time series imputation.

As already mentioned in order to use this series for comparing imputation algorithm results, there are two time series provided. One series without missing values (`tsAirgapComplete`), which can be used as ground truth. Another series with NAs, on which the imputation algorithms can be applied (`tsAirgap`). While the missing data for `tsNH4` and `tsHeating` were each introduced according to patterns observed in very similar time series from the same source, the missing observations in `tsAirgap` were created based on general missing data patterns.

tsNH4
The `tsNH4` time series has 4552 rows and the incomplete version includes 883 NA values. It represents the NH4 concentration in a waste-water system measured from 30.11.2010 - 16:10 to 01.01.2011 - 6:40 in 10 minute steps. The time series is derived from the dataset of the Genetic and Evolutionary Computation Conference (GECCO) Industrial Challenge 2014 [1].

As already mentioned in order to use this series for comparing imputation algorithm results, there are two time series provided. One series without missing values (`tsNH4Complete`), which can be used as ground truth. Another series with NAs (`tsNH4`), on which the imputation algorithms can be applied. The pattern for the NA occurrence was derived from the same series / sensors, but from an earlier time interval. Thus, it is a very realistic missing data pattern. Beware, since the time series has a lot of observations, some of the more complex algorithms like `na.kalman` will need some time till they are finished.

tsHeating
The `tsHeating` time series has 606837 rows and the incomplete version includes 57391 NA values. It represents a heating systems' supply temperature measured from 18.11.2013 - 05:12:00 to 13.01.2015 - 15:08:00 in 1 minute steps. The time series originates from the GECCO Industrial Challenge 2015 [2]. This was a challenge about 'Recovering missing information in heating system operating data'. Goal was to impute missing values in heating system sensor data as accurate as possible.

[1] http://www.spotseven.de/gecco-challenge/gecco-challenge-2014/
[2] http://www.spotseven.de/gecco-challenge/gecco-challenge-2015/

As already mentioned in order to use this series for comparing imputation algorithm results, there are two time series provided. One series without missing values (`tsHeatingComplete`), which can be used as ground truth. Another series with NAs (`tsHeating`), on which the imputation algorithms can be applied. The NAs thereby were inserted according to patterns found in similar time series. According to patterns found / occurring in other heating systems. Beware, since it is a very large time series, some of the more complex algorithms like `na.kalman` may need up to several days to complete on standard hardware.

Usage examples

To start working with the **imputeTS** package, install either the stable version from CRAN or the development version from GitHub (https://github.com/SteffenMoritz/imputeTS). The stable version from CRAN is hereby recommended.

Imputation algorithms

All imputation algorithms are used the same way. Input has to be either a numeric time series or a numeric vector. As output, a version of the input data with all missing values replaced by imputed values is returned. Here is a small example, to show how to use the imputation algorithms. (all imputation functions start with na.'algorithm name')

For this we first need to create an example input series with missing data.

```
# Create a short example time series with missing values
 x <- ts(c(1, 2, 3, 4, 5, 6, 7, 8, NA, NA, 11, 12))
```

On this time series we can apply different imputation algorithms. We start with using `na.mean`, which substitutes the NAs with mean values.

```
# Impute the missing values with na.mean
 na.mean(x)
```

```
[1]  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  5.9  5.9  11.0  12.0
```

Most of the functions also have additional options that provide further algorithms (of the same algorithm category). In the example below it can be seen that `na.mean` can also be called with `option="median"`, which substitutes the NAs with median values.

```
# Impute the missing values with na.mean using option median
 na.mean(x, option="median")
```

```
[1]  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  5.5  5.5  11.0  12.0
```

While `na.interpolation` and all other imputation functions are used the same way, the results produced may be different. As can be seen below, for this series linear interpolation gives more reasonable results.

```
# Impute the missing values with na.interpolation
na.interpolation(x)
```

```
[1]  1  2  3  4  5  6  7  8  9  10  11  12
```

For longer and more complex time series (with trend and seasonality) than in this example it is always a good idea to try `na.kalman` and `na.seadec`, since these functions very often produce the best results. These functions are called the same easy way as all other imputation functions.

Here is a usage example for the `na.kalman` function applied on the `tsAirgap` (described in 2.2.4) time series. As can be seen in Figure 1, `na.kalman` provides really good results for this series, which contains a strong seasonality and a strong trend.

```
# Impute the missing values with na.kalman
# (tsAirgap is an example time series provided by the imputeTS package)
imp <- na.kalman(tsAirgap)

#Code for visualization
plotNA.imputations(tsAirgap, x.imp, tsAirgapComplete)
```

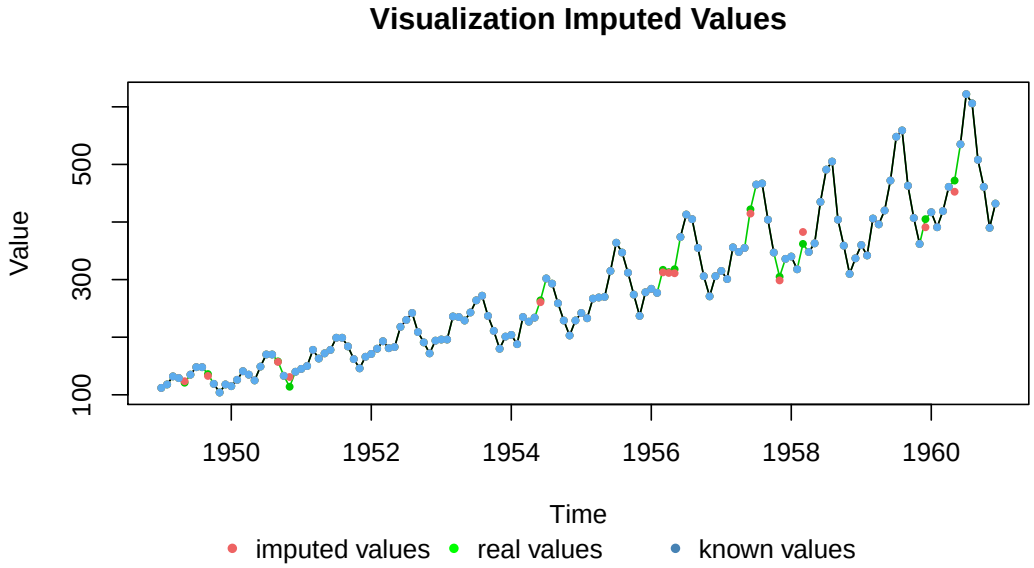

Figure 1: Results of imputation with na.kalman compared to real values

plotNA.distribution

This function visualizes the distribution of missing values within a time series. Therefore, the time series is plotted and whenever a value is NA the background is colored differently. This gives a nice overview, where in the time series most of the missing values occur. An example usage of the function can be seen below (for the plot see Figure 2).

```
# Example Code 'plotNA.distribution'
# (tsAirgap is an example time series provided by the imputeTS package)

# Visualize the missing values in this time series
plotNA.distribution(tsAirgap)
```

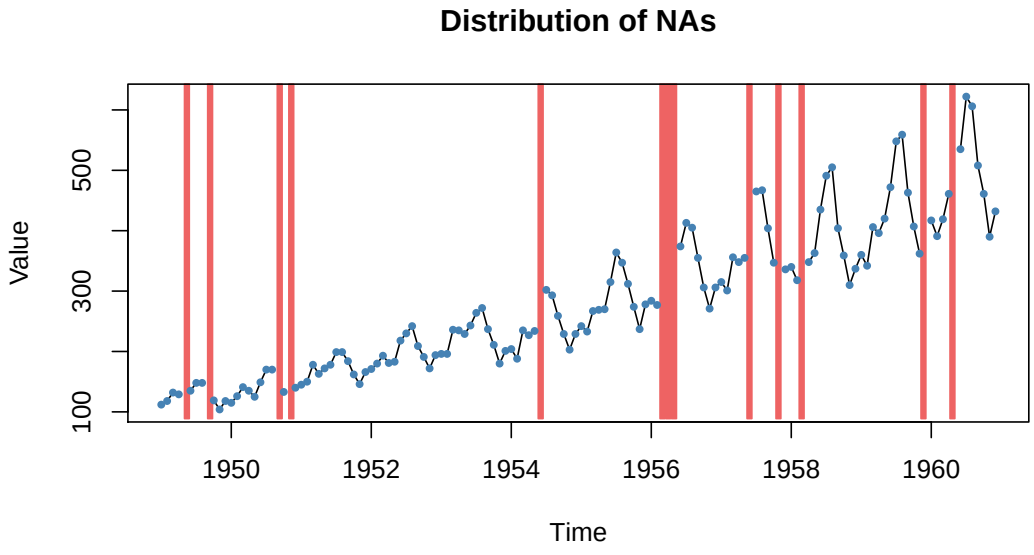

Figure 2: Example for plotNA.distribution

As can be seen in Figure 2, in areas with missing data the background is colored red. The whole plot is pretty much self-explanatory. The plotting function itself needs no further configuration

parameters, nevertheless it allows passing through of plot parameters (via ...).

plotNA.distributionBar

This function also visualizes the distribution of missing values within a time series. This is done as a barplot, which is especially useful if the time series would otherwise be too large to be plotted. Multiple observations for time intervals are grouped together and represented as bars. For these intervals, information about the amount of missing values are shown. An example usage of the function can be seen below (for the plot see Figure 3).

```
# Example Code 'plotNA.distributionBar'
# (tsHeating is an example time series provided by the imputeTS package)

# Visualize the missing values in this time series
plotNA.distributionBar(tsHeating, breaks = 20)
```

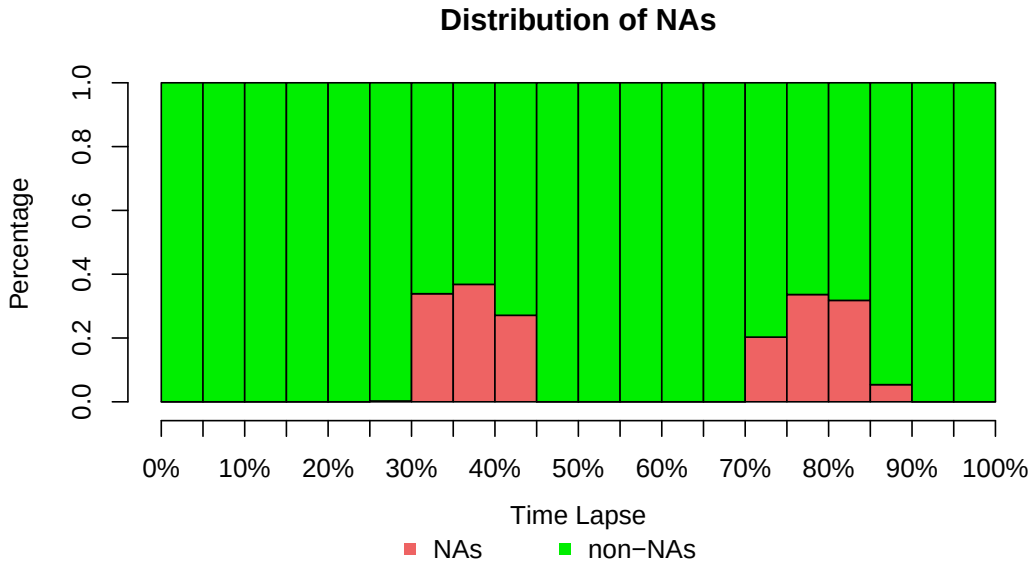

Figure 3: Example for plotNA.distributionBar

As can be seen in the x-axis of Figure 3, the `tsHeating` series is with over 600.000 observations a very large time series. While the missing values in the `tsAirgap` series (144 observations) can be visualized with `plotNA.distribution` like in Figure 2, this would for sure not work out for `tsHeating`. There just isn't enough space for 600.000 single consecutive observations/points in the plotting area. The `plotNA.distributionBar` function solves this problem. Multiple observations are grouped together in intervals. The 'breaks' parameter in the example defines that there should be 20 intervals used. This means every interval in Figure 3 represents approximately 30.000 observations. The first five intervals are completely green, which means there are no missing values present. This means from observation 1 up to observation 150.000 there are no missing values in the data. In the middle and at the end of the series there are several intervals each having around 40% of missing data. This means in these intervals 12.000 out of 30.000 observation are NA. All in all, the plot is able to give a nice but rough overview about the NA distribution in very large time series.

plotNA.gapsize

This plotting function can be used to visualize how often different NA gaps (NAs in a row) occur in a time series. The function shows this information as a ranking. This ranking can be ordered by

total NAs gap sizes account for (number occurrence gap size * gap length) or just by the number of occurrences of gap sizes. In the end the results can be read like this: In time series x, 3 NAs in a row occur most often with 20 occurrences, 6 NAs in a row occur 2nd most with 5 occurrences, 2 NAs in a row occur 3rd most with 3 occurrences. An example usage of the function can be seen below(for the plot see Figure 4).

```
# Example Code 'plotNA.gapsize'
# (tsNH4 is an example time series provided by the imputeTS package)

# Visualize the top gap sizes / NAs in a row
plotNA.gapsize(tsNH4)
```

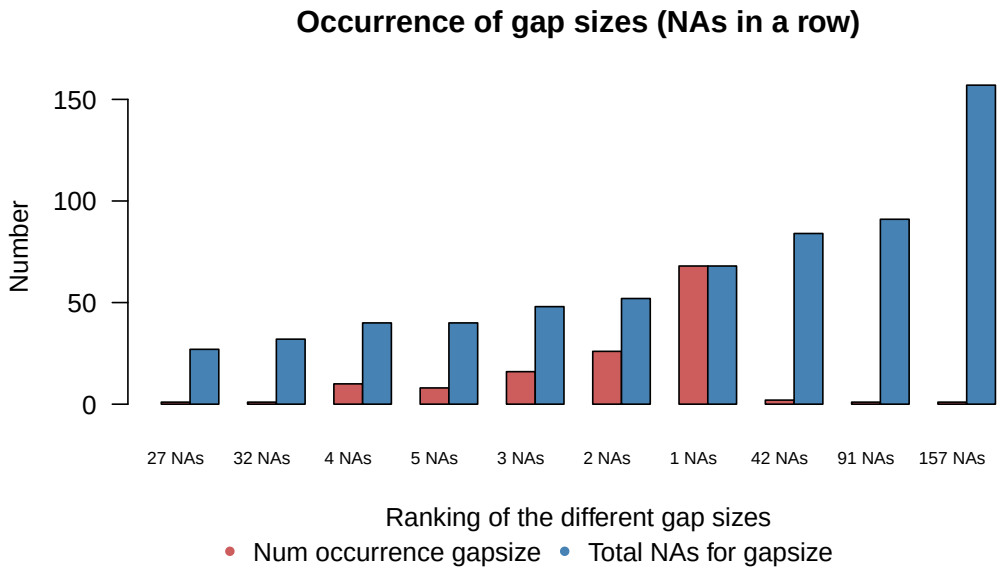

Figure 4: Example for printNA.gapsize

The example plot (Figure 4) reads the following: In the time series `tsNH4` gap size 157 occurs just 1 time, but makes up for most NAs of all gap sizes (157 NAs). A gap size of 91 (91 NAs in a row) also occurs just once, but makes up for 2nd most NAs (91 NAs). A gap size of 42 occurs two times in the time series, which leads to 3rd most overall (84 NAs). A gap size of one (no other NAs before or behind the NA) occurs 68 times, which makes this 4th in overall NAs (68 NAs).

plotNA.imputations

This plot can be used, to visualize the imputed values for a time series. Therefore, the imputed values (filled NA gaps) are shown in a different color than the other values. The function is used as below and Figure 5 shows the output.

```
# Example Code 'plotNA.imputations'
# (tsAirgap is an example time series provided by the imputeTS package)

# Step 1: Perform imputation for x using na.mean
tsAirgap.imp <- na.mean(tsAirgap)

# Step 2: Visualize the imputed values in the time series
plotNA.imputations(tsAirgap, tsAirgap.imp)
```

The visual inspection of Figure 5 indicates, that the imputed values (red) do not fit very well in the `tsAirgap` series. This is caused by `na.mean` being used for imputation of a series with a strong trend. The plotting function enables users to quickly detect such problems in the imputation results. If the ground truth is known for the imputed values, this information can also be added to the plot. The plotting function itself needs no further configuration parameters. Nevertheless, it allows passing through of plot parameters (via ...).

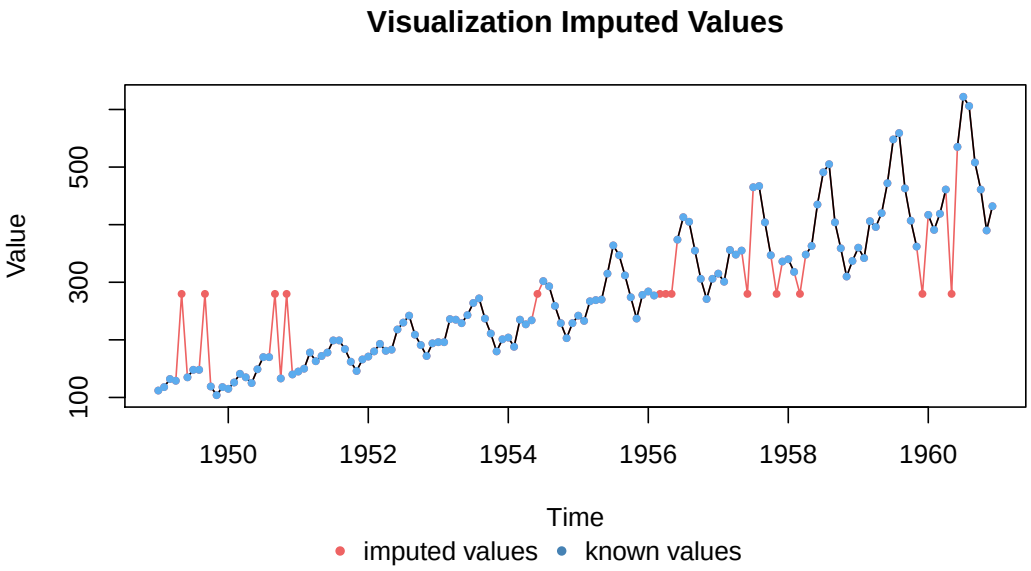

Figure 5: Example for printNA.imputations

statsNA

The `statsNA` function prints summary stats about the distribution of missing values in univariate time series. Here is a short explanation about the information it gives:

- Length of time series
  Number of observations in the time series (including NAs)
- Number of Missing Values
  Number of missing values in the time series
- Percentage of Missing Values Percentage of missing values in the time series
- Stats for Bins
  Number/percentage of missing values for the split into bins
- Longest NA gap
  Longest series of consecutive missing values (NAs in a row) in the time series
- Most frequent gap size
  Most frequent occurring series of missing values in the time series
- Gap size accounting for most NAs
  he series of consecutive missing values that accounts for most missing values overall in the time series
- Overview NA series
  Overview about how often each series of consecutive missing values occurs. Series occurring 0 times are skipped

The function is used as below and Figure 6 shows the output.

```
# Example Code 'statsNA'
# (tsNH4 is an example time series provided by the imputeTS package)

# Print stats about the missing data
statsNA(tsNH4)
```

```
> statsNA(x)
[1] "Length of time series:"
[1] 4552
[1] "-------------------------"
[1] "Number of Missing Values:"
[1] 883
[1] "-------------------------"
[1] "Percentage of Missing Values:"
[1] "19.4%"
[1] "-------------------------"
[1] "Stats for Bins"
[1] "  Bin 1 (1138 values from 1 to 1138) :      233 NAs (20.5%)"
[1] "  Bin 2 (1138 values from 1139 to 2276) :      433 NAs (38%)"
[1] "  Bin 3 (1138 values from 2277 to 3414) :      135 NAs (11.9%)"
[1] "  Bin 4 (1138 values from 3415 to 4552) :      82 NAs (7.21%)"
[1] "-------------------------"
[1] "Longest NA gap (series of consecutive NAs)"
[1] "157 in a row"
[1] "-------------------------"
[1] "Most frequent gap size (series of consecutive NA series)"
[1] "1 NA in a row (occuring 68 times)"
[1] "-------------------------"
[1] "Gap size accounting for most NAs"
[1] "157 NA in a row (occuring 1 times, making up for overall 157 NAs)"
[1] "-------------------------"
[1] "Overview NA series"
[1] "  1 NA in a row: 68 times"
[1] "  2 NA in a row: 26 times"
[1] "  3 NA in a row: 16 times"
[1] "  4 NA in a row: 10 times"
[1] "  5 NA in a row: 8 times"
[1] "  6 NA in a row: 4 times"
[1] "  7 NA in a row: 2 times"
```

Figure 6: Excerpt of statsNA output

Datasets

Using the datasets is self-explanatory, after the package is loaded they are directly available and usable under their name. No call of data() is needed. For every dataset there is always a complete version (without NAs) and an incomplete version (containing NAs) available.

```
# Example Code to use tsAirgap dataset
library("imputeTS")
tsAirgap
```

```
> tsAirgap
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129  NA 135 148 148  NA 119 104 118
1950 115 126 141 135 125 149 170 170  NA 133  NA 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234  NA 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277  NA  NA  NA 374 413 405 355 306 271 306
1957 315 301 356 348 355  NA 465 467 404 347  NA 336
1958 340 318  NA 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362  NA
1960 417 391 419 461  NA 535 622 606 508 461 390 432
```

Figure 7: Example tsAirgap time series

Conclusions

Missing data is a very common problem for all kinds of data. However, in case of univariate time series most standard algorithms and existing functions within R packages cannot be applied.
This paper presented the **imputeTS** package that provides a collection of algorithms and tools especially tailored to this task. Using example time series, we illustrated the ease of use and the advantages of the provided functions. Simple algorithms as well as more complicated ones can be applied in the same simple and user-friendly manner.

The functionality provided makes the **imputeTS** package a good choice for preprocessing of time series ahead of further analysis steps that require complete absence of missing values.

Future research and development plans for forthcoming versions of the package include adding additional time series algorithm options to choose from.

Acknowledgment

Parts of this work have been developed in the project '*IMProvT: Intelligente Messverfahren zur Prozessoptimierung von Trinkwasserbereitstellung und -verteilung*' (reference number: 03ET1387A). Kindly supported by the Federal Ministry of Economic Affairs and Energy of the Federal Republic of Germany.

Supported by:

on the basis of a decision
by the German Bundestag

Bibliography

Z. Bar-Joseph, G. K. Gerber, D. K. Gifford, T. S. Jaakkola, and I. Simon. Continuous representations of time-series gene expression data. *Journal of Computational Biology*, 10(3-4):341–356, 2003. [p1]

R. Billinton, H. Chen, and R. Ghajar. Time-series models for reliability evaluation of power systems including wind energy. *Microelectronics Reliability*, 36(9):1253–1261, 1996. [p1]

G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015. [p4]

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. [p1]

B. L. Ford. An Overview of Hot-Deck Procedures. *Incomplete data in sample surveys*, 2(Part IV): 185–207, 1983. [p1]

J. M. Gottman. *Time-series analysis: A comprehensive introduction for social scientists*, volume 400. Cambridge University Press Cambridge, 1981. [p1]

J. Honaker, G. King, and M. Blackwell. Amelia II: A Program for Missing Data. *Journal of Statistical Software*, 45(7):1–47, 2011. URL http://www.jstatsoft.org/v45/i07/. [p1]

R. J. Hyndman. *forecast: Forecasting functions for time series and linear models*, 2016. URL http://github.com/robjhyndman/forecast. R package version 7.3. [p1]

J. Josse and F. Husson. missMDA: A Package for Handling Missing Values in Multivariate Data Analysis. *Journal of Statistical Software*, 70(1):1–31, 2016. doi: 10.18637/jss.v070.i01. [p1]

A. Kowarik and M. Templ. Imputation with the R package VIM. *Journal of Statistical Software*, 74 (7):1–16, 2016. doi: 10.18637/jss.v074.i07. [p1]

S. Moritz. *imputeTS: Time Series Missing Value Imputation*, 2016a. URL http://CRAN.R-project.org/package=imputeTS. R package version 1.7. [p1]

S. Moritz. *package imputeTS*, 2016b. URL http://cran.r-project.org/web/packages/imputeTS/imputeTS.pdf. R package version 1.7. [p3]

S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer, and J. Stork. Comparison of different Methods for Univariate Time Series Imputation in R. *ArXiv e-prints*, Oct. 2015. [p1]

E. Pebesma. spacetime: Spatio-Temporal Data in R. *Journal of Statistical Software*, 51(7):1–30, 2012. URL http://www.jstatsoft.org/v51/i07/. [p1]

Rmetrics Core Team, D. Wuertz, T. Setz, and Y. Chalabi. *timeSeries: Rmetrics - Financial Time Series Objects*, 2015. URL https://CRAN.R-project.org/package=timeSeries. R package version 3022.101.2. [p1]

D. B. Rubin. *Multiple imputation for nonresponse in surveys*. New York: Wiley, 1987. [p1]

J. A. Ryan and J. M. Ulrich. *xts: eXtensible Time Series*, 2014. URL https://CRAN.R-project.org/package=xts. R package version 0.9-7. [p1]

S. J. Taylor. *Modelling financial time series (second edition)*. World Scientific Publishing, 2007. [p1]

P. Vacek and T. Ashikaga. An examination of the nearest neighbor rule for imputing missing values. *Proc. Statist. Computing Sect., Amer. Statist. Ass*, pages 326–331, 1980. [p1]

S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011. URL http://www.jstatsoft.org/v45/i03/. [p1]

A. Zeileis and G. Grothendieck. zoo: S3 Infrastructure for Regular and Irregular Time Series. *Journal of Statistical Software*, 14(6):1–27, 2005. URL http://www.jstatsoft.org/v14/i06/. [p1]

Steffen Moritz
Cologne University of Applied Sciences
Cologne, Germany
steffen.moritz10@gmail.com

Thomas Bartz-Beielstein
Cologne University of Applied Sciences
Cologne, Germany
bartz.beielstein@fh-koeln.de